J Inf Process Syst, Vol.17, No.4, pp.737~753, August 2021
https://doi.org/10.3745/JIPS.02.0162

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS

Evaluation of Artificial Intelligence-Based Denoising Methods for Global Illumination

Soroor Malekmohammadi Faradounbeh* and SeongKi Kim**

Abstract
As the demand for high-quality rendering for mixed reality, videogame, and simulation has increased, global illumination has been actively researched. Monte Carlo path tracing can realize global illumination and produce photorealistic scenes that include critical effects such as color bleeding, caustics, multiple light, and shadows. If the sampling rate is insufficient, however, the rendered results have a large amount of noise. The most successful approach to eliminating or reducing Monte Carlo noise uses a feature-based filter. It exploits the scene characteristics such as a position within a world coordinate and a shading normal. In general, the techniques are based on the denoised pixel or sample and are computationally expensive. However, the main challenge for all of them is to find the appropriate weights for every feature while preserving the details of the scene. In this paper, we compare the recent algorithms for removing Monte Carlo noise in terms of their performance and quality. We also describe their advantages and disadvantages. As far as we know, this study is the first in the world to compare the artificial intelligence-based denoising methods for Monte Carlo rendering.

Keywords
Denoising, Filtering, Global Illumination, Monte Carlo Noise, Noise Removal

1. Introduction

Rendering generates a final image that can see the virtual objects with the material in a 3D scene. We can have any imaginable objects, apply light sources that we would never be able to purchase, or change the time of day, or even the planet that we are on. The rendering algorithm receives descriptive input that includes information about the properties of their materials, set of light sources, location of objects, and camera that represents the eye in the virtual world. The result is an image that captures the scene as if the user were viewing it in the real world from that particular camera location. Local illumination considers only the direct lights from the light sources, and it has been widely used due to its high performance. However, the demand for high-quality rendering has increased for mixed reality, videogame, and simulation, and global illumination can satisfy this need. Global illumination simulates rays from light sources and models how a ray bounces from one surface onto another (indirect light) as well as from a light source onto an object (direct light), and it can create more detailed and realistic lighting.

Calculating the effects by all rays for photorealistic results is definite and formulated using rendering

Manuscript received September 17, 2020; first revision December 8, 2020; accepted December 29, 2020.
Corresponding Author: SeongKi Kim (skkim9226@gmail.com)
* Dept. of Computer Engineering, Keimyung University, Daegu, Korea (soroormalekmohamadi@gmail.com)
** Division of SW Convergence, Sangmyung University, Seoul, Korea (skkim9226@gmail.com)

equations [1]. To simulate significant light transport effects with global illumination in films, animations, videogames, and other industrial fields, many researchers and practitioners have used ray tracing-based methods. When complex scenes with reflection models are rendered, Monte Carlo (MC) methods are practical [2] and can be used to realize global illumination. Path tracing based on MC integration [1,2], or particle density estimation such as photon mapping [3,4] has been widely used. Due to the density estimation step, photon mapping is a biased but consistent method [5]. Owing to the robustness, generality, flexibility, and simplicity, path tracing has been widely used for realistic rendering and rendering equations [5,6]. In this method, random samples are selected from the integration domain, and the integral is approximated based on the statistical mean of the evaluated integrand of each sample. Therefore, evaluating any complex multidimensional integral becomes a simple predictive value problem.

As an effective technique for photorealistic images, MC path tracing can produce critical effects such as area lighting, motion blur, and depth of field [1]. It has been used to generate realistic images in videogames and films, with high-quality renderings produced from a variety of 3D models. To model all types of reflections and refractions in random rays, hybrid methods can be used to exploit the advantageous properties of the radiosity and ray tracing. In other words, these algorithms are similar to the auction system since both determine an output from many inputs [7].

However, the results by path tracing are noisy, and tracing a lot of rays is required [2] for a reasonable quality, which makes it slow. For example, hours (or even days) can be required for just a single image. Moreover, path tracing suffers from high variance. While tracing the rays, the variance in the MC estimator decreases linearly as the number of samples increases [7]. Therefore, a larger number of rays finally leads to the reliable estimation of the integral, but the high cost of tracing such rays results in high computational costs. In contrast, a small number of samples can be quickly evaluated, although the inaccurate estimation of the true value of the integral appears as noise in the final images [3]. This means that the rendered images will have a high noise level if they have only a few samples.

Owing to the importance of fast photorealistic rendering with less noise, this problem has been subject to extensive studies for the past three decades, and many researchers have suggested methods. For example, a denoising filter [4] can be used. These filters choose samples per pixel from the neighboring pixels and analyze them [8] to remove noises with a small number of samples. To denoise a frame within a movie, the frame difference method detects a moving object using the difference between several consecutive frames, but this method is only suitable for specific scenes that are sensitive to ambient noise [9,10]. The recent nonlocal means (NL-means) filters [11,12] were effective and successful denoising methods that evaluate every pixel as the weighted average of the neighboring pixels. Besides, data-adaptive transforms such as independent component analysis (ICA) and principal component analysis (PCA), non-data adaptive transform, and BM3D [13] as one of the extensions of the NL-means approach, convolutional neural network (CNN)-based denoising methods, MLP models, and deep learning-based approach [14] have been suggested. Furthermore, owing to the popularity of the MC method, there has been renewed interest in the filtering of general MC noise as well as powerful algorithms for such purposes [15-20]. In this study, we described and evaluated these recent researches.

For the comparison and evaluation of these methods, the scenes are created as datasets including SPONZA, Amusement Park, or a product of the physically based ray tracer (PBRT) renderer system. In each of these scenes, a wide range of distributed effects with different numbers of triangles and all types of light is used. For example, Amusement Park includes 22.9 million triangles and 3.4 million emissive triangles (all light comes from emissive and environment maps) wherein most of the emissive triangles move within every frame.

This paper has the following contributions. First, this paper reviews and compares sample-based and pixel-based algorithms for MC noise. Second, successful AI-based filtering methods for denoising an MC rendered image are described, and studies on both pixel- and sample-based methods are detailed for the first time in the world. Third, this paper introduces the commonly used scenes and datasets to evaluate different methods.

The rest of this paper is organized as follows: Section 2 describes the proposed algorithms; Section 3 introduces two common datasets in this field and summarizes the advantages and disadvantages of the algorithms in terms of their performance and rendering aspects; finally, Section 4 concludes our study.

2. Overview of Noise Removal Filters

This section describes the developed algorithms for denoising in a rendered scene. In general, the noise removal filters can be categorized as shown in Fig. 1.

MC methods have noise in many cases because they rely on random point samples of a complex integrand. This has spurred the development of denoising filters, which can be categorized into two techniques as shown in Fig. 1. Sample-based techniques track each sample [21], whereas pixel-based methods remove noise on the rendered image directly [17,22]. The goal of every filtering approach is to minimize the difference between the filter and ground truth images.

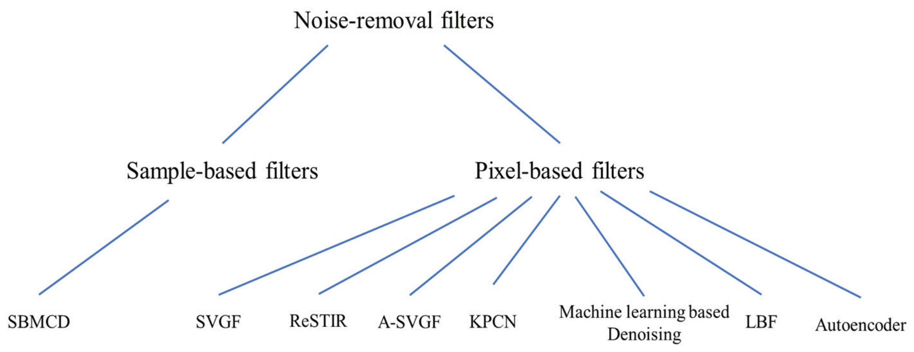

Fig. 1. Taxonomy of denoising methods.

2.1 Pixel-based Filters

Pixel-based filters aggregate all samples in a single per-pixel average. Although they are efficient in terms of storage and unaffected by the complexity of the scene, flat images prevent artists from using deep-compositing techniques [23]. Pixel-based denoising methods operate on pixels rather than samples, so they can efficiently produce a high-quality denoised result [24].

2.1.1 Common features

The first step is to render a noisy image with a few samples, and next is to apply the filter for the denoised results. Fig. 2 shows these processes.

Preprocessing
Filtering
Postprocessing

Fig. 2. Overall processing of denoising.

As the first stage, operations such as normalization or gradient extraction are performed. The second one is to reduce or eliminate the noise, and the last step involves performing exponential transform or reconstruction. A filter computes the weighted average of all neighborhood pixels to generate the filtered pixel [18]. For example, if we want to have color c at single pixel i, based on a common filtering approach, the algorithm calculates the weighted average of all neighborhood pixels $N(i)$ around pixel i:

$$\hat{c}_i = \frac{\sum_{j\in N(i)} d_{ij}\bar{c}_j}{\sum_{j\in N(i)} d_{ij}} \tag{1}$$

where d_{ij} is the weight between pixels i and j and is defined by the filter, and $\bar{c}_j$ is a color at pixel j.

Weight d_{ij} defines a type of filter. For example, in a cross bilateral filter, this weight is given by this equation, which is a multiplication of several terms that compute a Gaussian distance between two pixels in terms of the color of the screen pixel and several additional features, including the positions, normal shading, and texture.

All methods have a specific filter that takes a set of the same features in a neighborhood around every pixel [25]. It also takes a set of filter parameters that vary per pixel to output the filtered pixel [18].

2.1.2 Kernel-predicting convolutional networks (KPCN)

Developed by Bako et al. [17], KPCN uses a CNN that has emerged in recent years as an epidemic in machine learning, particularly on the problem of image denoising [26-28]. The learning-based techniques are effective for denoising MC rendering methods [29,30].

This approach used a deep CNN, but the layers of this network are not fully connected to avoid the danger of overfitting and increase the performance during the training phase. Rectified Linear Unit (ReLU) activations ($f^l(a) = max(0, \alpha)$) are used by all layers except the last layer, which uses the identify function ($f^l(a) = a$) [13].

CNNs can learn nonlinear functions of the input features (it is significant to achieve better outputs) by combining many of these layers with activation functions together [28]. Each layer within a CNN is applied to multiple spatial kernels with learnable weights. These are naturally suited for the denoising task and used before traditional noise removal [26].

Fig. 3 illustrates the general structure of this method. Inputs are divided into specular and diffuse light and are processed by independent pipelines. In general, CNN predicts a different smoothing filter kernel for every pixel [31].

KPCN yields good denoising performance for a relatively small number of instances. More complex denoising functions can increase flexibility, reduce modeling bias, and prevent overfitting. Using these functions and ensuring that there are sufficient samples to estimate can lead to the generation of a generic model that can generalize the denoising result to images that are not used during training.

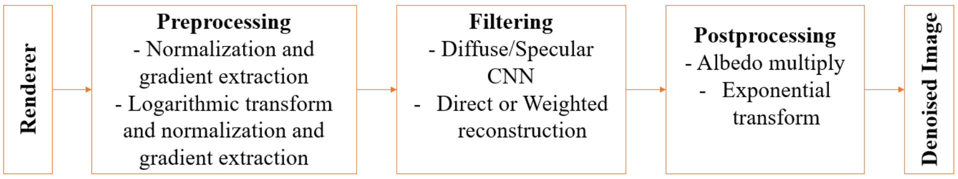
Renderer
Preprocessing
- Normalization and gradient extraction
- Logarithmic transform and normalization and gradient extraction
Filtering
- Diffuse/Specular CNN
- Direct or Weighted reconstruction
Postprocessing
- Albedo multiply
- Exponential transform
Denoised Image

Fig. 3. The general structure of KCPN method [13].

2.1.3 Learning-based filter (LBF)

LBF improves PBRT2 [26], and noisy input data have a complex relationship with the optimal filter parameters. In addition, this method uses features generated by the rendering system. The neighboring noisy samples are used to compute the secondary features, which serve as inputs to the neural network.

This method uses a multilayer perceptron neural network and combines the network with a matching filter during both training and testing. This trained network can generate filtered images on a wide range of distributed effects such as depth of field, motion blur, area lighting, glossy reflections, and global illumination [18]. The most important challenge is to counteract the distributed effects, and filter adjustment is required for all features while maintaining the details as much as possible [15]. This method is shown in Fig. 4.

This method can be extended to perform animated sequences by applying a filtering process in the spatiotemporal volumes.

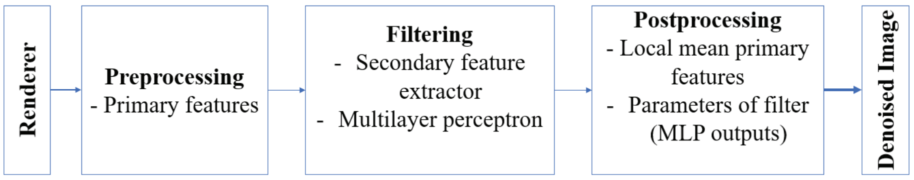

Fig. 4. The LBF method structure uses a multi-layer perceptron neural network and combines it directly with a matching filter during training and testing [18].

2.1.4 Reservoir-based spatiotemporal importance resampling (ReSTIR)

This is an MC approach used in direct lighting based on a generalization of the resampled importance sampling, which allows unbiased spatial samples, As an algorithm used for all direct light, this algorithm is based on the resampled important sampling [32]. Resampling generates equally weighted samples for importance sampling. The importance of resampling parameters lies in the reduction of near-optimal variance [32]. Unlike post-processing methods, this method such as most denoising methods can trace additional samples as well as the criteria that determine where this method has failed and allocate additional samples to such areas. ReSTIR interactively samples direct lighting from thousands or millions of dynamic emissive triangles and supports millions of polygonal lights (even off-screen, and lights can move with the environment) in real time with shadows [33]. Through such filtering technique, denoising need not remain a post-process but is performed once rendering is effectively completed; in addition, denoising is moved into the core of the renderer, filtering the probability density functions rather than colors. Fig. 5 illustrates the steps of the ReSTIR algorithm.

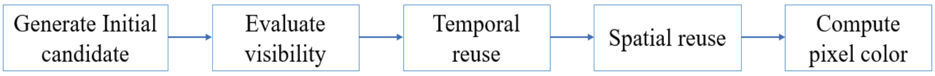

Fig. 5. Steps of ReSTIR algorithm.

As shown in Fig. 5, with this method, *m* independent per-pixel light candidates are first generated and resampled. Next, the selected samples from this step are tested for their visibility, and occluded samples are discarded. The selected samples in each pixel reservoir are then combined with the prior frame's output, determined using back-projection. For the leverage information from a pixel's neighbors, *n* rounds of spatial reuse are applied. Finally, the image is shaded, and the final reservoirs are forwarded to the next frame.

The main data structure is composed of image buffers, which makes this method fast, simple, and efficient in terms of memory. However, the usage of this method is limited to tasks conducted on the first vertex of the camera path. Therefore, it cannot be used easily for direct light or global illumination beyond the first hit.

A wide range of prior approaches has addressed light sampling and sample reuse in rendering or has developed mathematical tools related to this work.

2.1.5 Spatiotemporal variance-guided filtering (SVGF)

This algorithm separates the direct and indirect effects by light sources and denoises them independently; in addition, it estimates the amount of noise in different regions within the image to provide more useful information for denoising routines. This means that the history buffers (from a prior frame reconstruction) and normal, albedo, depth, motion vectors, and mesh ID rasterization are required. No neural network or learning algorithms are used in this technique. In general, this is a reconstruction method that can generate a stable temporal sequence of images from global illumination in any path-per-pixel [19].

The reconstruction filter is shown in Fig. 6. It uses temporal accumulation to determine the integrated color/moments and variance estimation to obtain the filtered color [19].

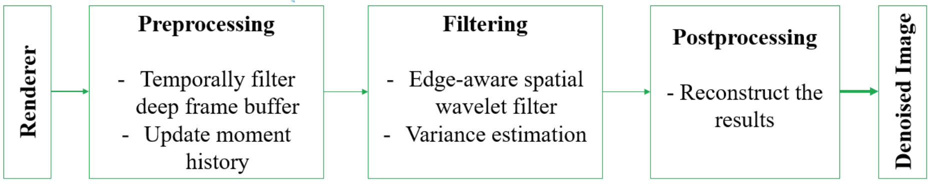

Fig. 6. The general structure of SVGF [19].

2.1.6 Adaptive SVGF (A-SVGF)

Adaptive SVGF is a newer technique that removes problems such as flicker by using an adaptive temporal filter. This method estimates a gradient for real-time adaptive temporal filtering, which runs on modern graphics hardware at speeds of 2 ms at 1080 pixels, and it can be included by deferred renderers [34]. The SVGF method [19] introduces temporal blur, and the light sequence continues to be illuminated when the light source is turned off and glossy objects exist (Fig. 7). The difference between A-SVGF and SVGF is that a simple 3×3 box filter is used by the À-trous wavelet, reducing the quality but boasting of significant speed. This algorithm estimates, reconstructs sparse temporal gradients that are used to adjust time coefficient α (accumulation element) per pixel. The temporal gradient is measured to control temporal accumulation adaptively, which depends on the shading function of the previous frame and current frame [19]. In general, reliable historical data are rejected and replaced with constant temporal weight α using the weight of each adaptive pixel calculated from the temporal gradients. To compute the per-pixel weight, the resulting dense gradient field is used [34]. This algorithm is based on the resampling of important samples.

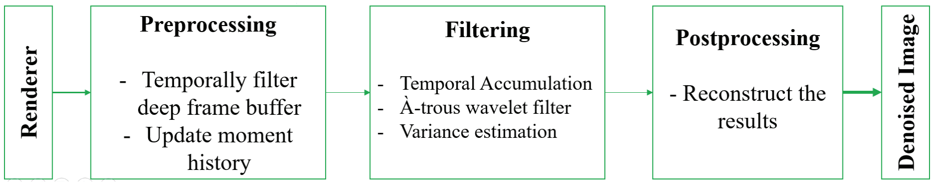

Fig. 7. An overview of A-SVGF method.

Compared to previous studies, performance is significantly increased and temporal stability including is improved when reconstructing the final image. This temporal filter can be integrated into the deferred renderer and is 5%–47% better matched with reference images compared to previous crossover filters. In addition, this algorithm produces temporarily stable outcomes 10 times faster and requires 10 ms (± 15%) if it runs on modern graphics hardware with resolution of 1920×1080 [19].

2.1.7 Machine learning-based denoising

Recently, machine learning has been applied to denoising. After starting as a solution to the problem of classifying noisy text [35,36], the problem domain was expanded to include more computer graphics focused on problems such as denoising within the path-traced images [17,37-40], real-time denoising for ray tracing [15,41], general image reconstruction, and upscaling images while maintaining the details through super-resolution. However, the difficulty in using these lies in their computational cost. Without dedicated hardware or tradeoff in quality and performance, it is difficult to achieve real-time denoising with high quality through machine learning.

Thus, research has been conducted in the area of algorithmic techniques to denoise images in real time using spherical harmonics to encode low-frequency data, guided filters for blurring while avoiding sharp changes in the normal or albedo information, and spatiotemporal reprojection for reusing data that are not view-dependent, such as global illumination or shadows.

Nowadays, MC-based rendering is widespread in visual effects production and animation [42]. The latest method used in feature film production is machine learning denoising, which follows the approach by Vogels et al. [38]. However, they modified the neural network architecture to increase the performance.

The denoiser separately denoises three parts: (1) color output, (2) diffusion divided by the rendered surface color, and (3) specular reflection. It then multiplies the denoised diffusion and adds the multiplication result to the specular part to obtain the final denoised color. The alpha is also denoised in an additional pass. In general, the neural network-based solutions can reduce the time [43].

2.1.8 Autoencoder

An autoencoder is one of the types of neural networks with an encoder and a decoder phase in the input and output section, respectively [44]. The basic architecture of Autoencoder shown in Fig. 8. An autoencoder is trained to correct the imperfect input and copy it to output. In fact, the autoencoder attempts to learn how to undo its input corruption. These phases let the network operation continuously decrease and increase. Moreover, if the hidden layer of the encoder is non-linear, it will differently behave from PCA, and it will be able to capture multi-modal aspects of the input distribution. Generally, due to the high-frequency signals of PCA filtering, the details of an image can be lost when reconstructing an image [45]. Due to the limited capacity of the neural network, however, it can lead to compromised image quality compared to offline techniques. On the other hand, an exact value for the rule strength has been reached, and the optimal solution is achieved by training the same as an activation function in a finite period at the fuzzy-based techniques, with no complicated mathematical models required [46]. Fuzzy approaches have already been applied in many areas such as filtering and image improvement, and they have the potential to combine with other techniques for denoising global illumination. Generally, fuzzy-based filtration is regarded as a nonlinear filtering technique [47]. Fuzzy systems can also be used to reduce the uncertainty in stacks of autoencoders [48].

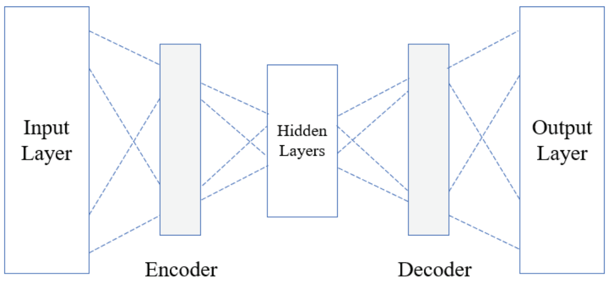

Fig. 8. Overview of autoencoder structure.

2.2 Sample-based Filters

One of the other subsets of denoising approaches is sample-based, meaning they work on individual samples and theoretically allow the production of suitable inputs for deep-compositing pipelines as a result. However, none of the denoising techniques are general enough to be used in production. Light field reconstruction techniques [49,50] handle only a subset of light transport cases, and generic methods [51,52] scale poorly owing to the high sampling rates needed for high-end production; the recent approach by Bauszat et al. [53] handles only indirect illumination and depth of field, relying solely on geometric buffers to guide the filter, so it is prone to incompatibility with high sampling rates.

2.2.1 Sample-based Monte Carlo denoising (SBMCD)

A convolutional network can learn to denoise MC renderings directly from the samples. However, both variance and covariance can be estimated and calculated with samples [54]. It is challenging to learn the mapping between samples and images because the samples should be treated in a permutation-invariant manner, and the samples' order is arbitrary.

If the images are too noisy, however, post-processing methods are often unable to recover clean and sharp images. Usually, hundreds of samples per pixel are required on average for a good-quality image (It will be a tedious, time-consuming process). Compared to previous methods, the sample-based kernel-splatting denoising (SBKSD) [21] produces cleaner outputs with an extremely small number of samples.

Based on the kernel-predicting architecture, the individual samples are splatted on adjacent pixels (as shown in Fig. 9). Splatting is a natural solution for situations such as depth-of-field, motion blur, and many light transmission paths, where it is easier to predict which pixels a sample contributes to, instead of a collection method that should specify for each pixel which samples (or surrounding pixels) are relevant. By using individual sample embeddings and context features per pixel, this architecture is not fixed for replacing each pixel of samples.

Compared to previous recent approaches, this method handles severe noise of images with low samples count (e.g., 4 SPP [samples per pixel]) and provides higher-quality results both numerically and visually.

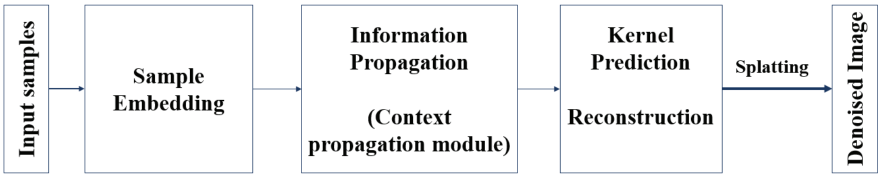

Fig. 9. Architecture of sampled-based kernel-splatting denoising method [20]. Dividing the extraction of per-sample feature and sharing spatial information in alternating steps (scattering or splatting) are a more natural operation for a sample-based configuration [21].

3. Datasets and Scenes

In general, for the evaluation and testing of denoising methods for MC rendering, rendered scenes such as SPONZA or renderers like PBRT or other specific renderers are used to create validation data and scenes.

3.1 SPONZA

The SPONZA Atrium model was created by Marko Dabrovic and has been widely used as a lighting test [55]. The SPONZA sample scene is a sample that can be used by developers and researchers to test lighting for their games, setups, and new methods. Fig. 10 shows the scenes.

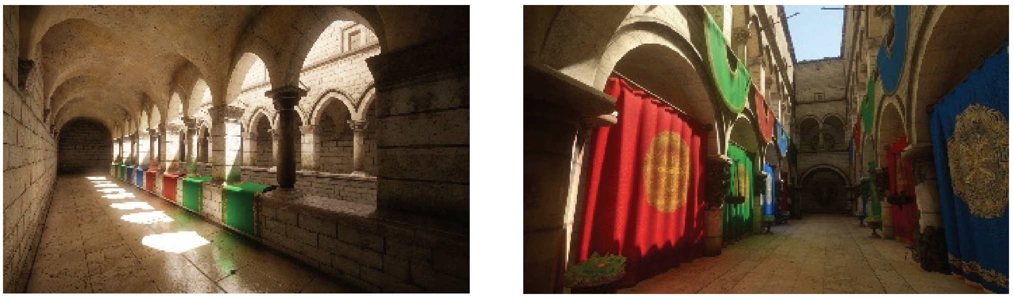

Fig. 10. SPONZA Atrium scenes.

3.2 Classroom Scene

The Classroom scene has similar SPONZA scene features and different lighting setup. The classroom scene has directional light source, sky illumination, and rich set of textures, thin geometric shapes, and layered materials [56].

3.3 Dungeon Scene

The dungeon scene is illuminated by using multiple static area light sources [19]. This scene has many challenging situations such as moving shadows, differently shaded regions with sharp boundaries, rapidly changing direct/indirect illumination, and direct/indirect glossy reflection.

3.4 Physically Based Ray Tracer

Physics-based rendering is a widely adopted practical roadmap with more accurately simulating materials and lights and is used for most physically based shading and lighting systems such as film production. These scenes are based on the ray-tracing algorithm and implemented and designed with three main goals: physically based, illustrative, and complete [34]. Fig. 11 shows some of the scenes produced by PBRT and which have been used in most filter denoising studies.

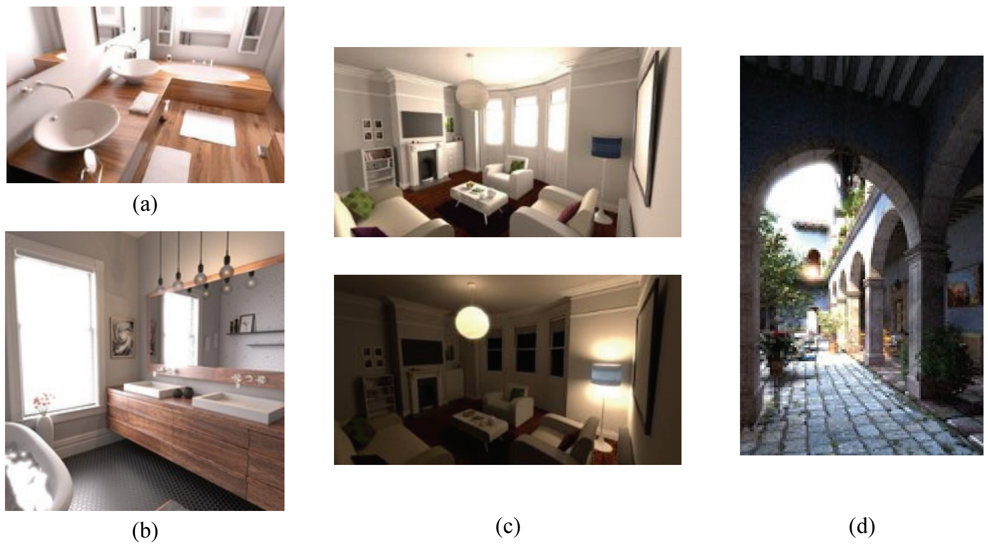

Fig. 11. Some scenes are available for use with PBRT. (a) Modern-bathroom model includes depth of field and soft indirect lighting. The overall appearance of the scene is significantly affected by the indirect illumination, due to the existence of mirrors and bright white walls. (b) Another modern-bathroom scene. Visible noise-free rendering is challenging: specular light transmission for efficient render by large mirrors and light sources in a very small area surrounded by glass lamps is not too easy. (c) Interior scene with two illumination configurations. The day type is primarily illuminated by light coming through the windows from outdoors, whereas the night version is illuminated by the two lights in the scene. (d) A sophisticated model inspired by a hotel in Mexico (San Miguel de Allende) [34].

4. Evaluation

4.1 Metrics

To measure the noise removal effects by method, we used the root-mean-square error (R-MSE) and structural similarity index (SSIM) [57] error metrics that compared to a 4,096 SPP image as a reference because these metrics are widely used for the noise removal filters. We also measured the weight relative mean square error (Rel-MSE) [58] relative to brightness. Rel-MSE is calculated by dividing the estimated MSE by the square value of the scale selected with a small constant to avoid over-emphasis in very dark image areas.

Table 1 shows the equation and the parameters for the measurement.

Table 1. Equations for the performance measurement

Method	Equation	Parameters
MSE	$\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(x(i,j)-y(i,j))^2$	$x(i,j)$, $y(i,j)$: original and denoised images i,j: pixel position in $M \times N$ image.
SSIM	$\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{({\mu_x}^2+{\mu_y}^2+c_1)({\sigma_x}^2+{\sigma_y}^2+c_2)}$	$\mu_x,\mu_y; {\sigma_x}^2, {\sigma_y}^2; \sigma_{xy}$: the average values of x, y c_1, c_2: two parameters that are used to stabilize the division with a weak denominator

4.2 Results

Due to the different conditions of each scene (test set), we provide the evaluation result on SPONZA in Table 2 and on PBRT renderer in Table 3. In Table 2, SPP is the number of light rays existing in every pixel, and we can obtain a more converged image with more SPPs.

A-SVGF and neural bilateral grid denoiser [56] have better results than LBF and SVGF. However, the KPCN method with 4 SPPs generates much better result than the A-SVGF method with a few modifications because it benefits from deep learning and neural networks. Table 3 compares their performance with the PBRT renderer.

In this comparison, SBMCD as a sample-based method shows better results with 4 SPPs compared to two pixel-based methods (LBF and deep Monte Carlo rendering [23]). Based on these results, it is expected that better results with few SPPs can be obtained by improving the sample-based methods.

Table 2. Noise removal effects of denoising algorithms on the SPONZA test set

Filter methods	SSIM	R-MSE	Rel-MSE	SPP
KPCN	-	-	~0.006	4
SVGF	0.4315	0.0753	-	1
LBF	0.7770	0.0320	-	1
A-SVGF	0.9227	0.0227	-	1
An autoencoder-based method (neural bilateral grid denoiser)	0.9270	-	-	1

Table 3. Noise removal effects of denoising algorithms on PBRT renderer's test set

Filter methods	DSSIM (1-SSIM)	R-MSE	SPP
SBMCD	0.0685	0.0482	4
LBF	0.0869	1.5814	4
A machine learning-based method (deep Monte Carlo rendering)	0.1294	1.0867	4

Figs. 12 show some results of the methods on datasets, and they are taken while moving the camera. All comparative methods record and maintain outstanding structures and provide acceptable and noise-free results.

Some examples of the results that apply pixel-based methods, such as KPCN and LBF, as well as sample-based methods (SBMCD) are shown in Fig. 12. All of these methods have been able to provide noise-free results with different inputs; and as can be seen in the picture, the SBMCD method is very accurate, and the results are similar to the ground-truth image.

We also measured the required time for removing noises to compare their efficiency, and the results are presented in Table 4. These denoisers were evaluated on the system with Intel i7-6900K CPU and NVIDIA Titan XP GPU. The elapsed time (seconds) is considered to process a 1024×1024 image based on the PBRT dataset.

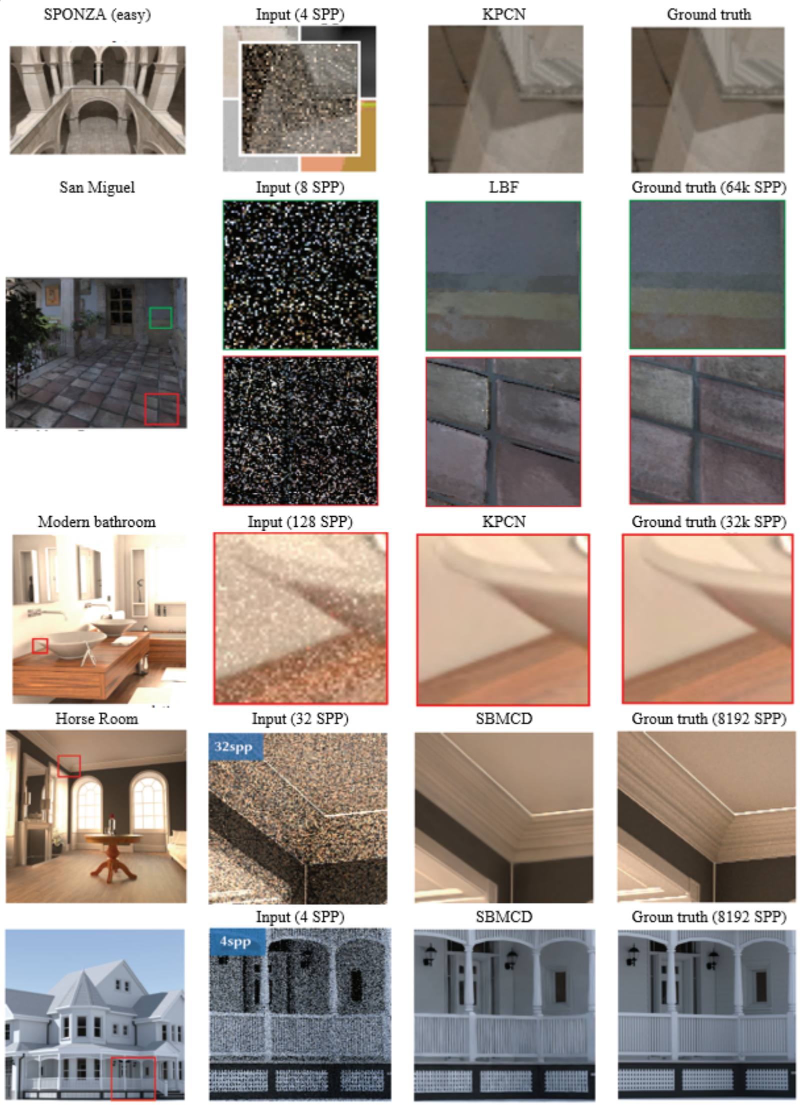

Fig. 12. Comparison between sample-based and pixel-based methods [17,21,31].

Table 4. Elapsed time of denoising algorithms on PBRT renderer's test set

Filter methods	Type	4 SPP	8 SPP	16SPP	32 SPP	64 SPP	128 SPP
LBF	Pixel-based and machine learning	10.4	-	-	-	-	-
KPCN	Pixel-based	14.6	-	-	-	-	-
SBMCD	Sample-based	6.0	10.1	18.9	35.9	67.0	156.5

The elapsed time of pixel-based methods is constant because all of them should check all pixels and calculate the average regardless of the status of a pixel, but the sample-based method increases linearly with the number of samples. However, the use of machine learning acts as an optimizer and takes less time than others.

Other methods such as A-SVGF and SVGF can be executed in real time with the benefit of machine learning, and the neural bilateral grid denoiser as an autoencoder approach can also remove the noises in real time. The elapsed time of real-time denoising algorithms is shown in Table 5. The scenes are animated with different camera flythroughs and were rendered at 60 FPS (frame per second). To check whether these approaches are applicable to interactive scenarios, the time was measured by the breakdown of the frame time. The common filters cannot handle this kind of animation due to the global effect on shading, including issues of temporal blur and stability.

Table 5. Elapsed time of real-time denoising algorithms

Filter methods	Type	Time (ms)	Scene	Device
A-SVGF	Pixel-based and machine learning	<2	Dungeon	NVIDIA Titan XP
SVGF	Pixel-based and machine learning	4.400	Classroom	NVIDIA Titan XP
Neural bilateral grid denoiser	Pixel-based and autoencoder	9.407	Classroom	NVIDIA RTX 2080

5. Conclusion

Recently, there have been numerous studies on denoising algorithms. Rendering fully converged, noise-free images is often too expensive, and much effort has been made to improve the image quality produced from this renderer, especially to obtain high-quality results with fewer samples, which is critical for high-performance/realistic rendering. In this study, we compared some of these denoising algorithms to show their potential and balance between quality and performance. As previously mentioned, the LBF, KPCN, SVGF, A-SVGF, and ReSTIR algorithms introduced for denoising are pixel-based, and LBF, KPCN, and A-SVGF used the advantage of neural networks to improve the performance of their systems. Moreover, except the ReSTIR algorithm, the rest of the algorithms use a similar process and include post-processing steps. In addition to comparing and introducing the pixel-based denoising methods (as common methods), other methods such as sample-based methods (SBMCD) are also discussed. We also tried to show the potential of using machine learning methods (deep Monte Carlo rendering) and neural networks (neural bilateral grid denoiser) to improve the final quality.

This paper has limitations, however. Noise removal for augmented reality (AR) applications has not been explored in this study. This field can provide researchers with a range of opportunities and research topics. Furthermore, other methods of artificial intelligence, neural networks, and fuzzy logic can be used to remove noises within the resulting images. In the future, we plan to investigate these methods.

References

[1] J. T. Kajiya, "The rendering equation," in *Proceedings of the 13th Annual Conference on Computer Graphics And Interactive Techniques*, Dallas, TX, 1986, pp. 143-150.

[2] E. Veach, "Robust Monte Carlo methods for light transport simulation," Ph.D. dissertation, Stanford University, Stanford, CA, 1997.

[3] H. W. Jensen, "Global illumination using photon maps," in *Rendering Techniques '96.* Vienna, Austria: Springer, 1996, pp. 21-30.

[4] Z. Zeng, L. Wang, B. B. Wang, C. M. Kang, and Y. N. Xu, "Denoising stochastic progressive photon mapping renderings using a multi-residual network," *Journal of Computer Science and Technology*, vol. 35, pp. 506-521, 2020.

[5] K. Vardis, "Efficient illumination algorithms for global illumination in interactive and real-time rendering," Ph.D. dissertation, Athens University of Economics & Business, Greece, 2016.

[6] T. Chen, J. Shi, J. Yang, and G. Li, "Enhancing network cluster synchronization capability based on artificial immune algorithm," *Human-centric Computing and Information Sciences*, vol. 9, article no. 3, 2019. https://doi.org/10.1186/s13673-019-0164-y

[7] P. Dutre, K. Bala, and P. Bekaert, *Advanced Global Illumination*. Boca Raton, FL: AK Peters/CRC Press, 2006.

[8] M. Mara, M. McGuire, B. Bitterli, and W. Jarosz, "An efficient denoising algorithm for global illumination," in *Proceedings of High Performance Graphics*, Los Angeles, CA, 2017.

[9] Y. Zhang, H. Wang, and X. Fan, "Algorithm for detection of fire smoke in a video based on wavelet energy slope fitting," *Journal of Information Processing Systems*, vol. 16, no. 3, pp. 557-571, 2020.

[10] R. Wan, L. Ding, N. Xiong, W. Shu, and L. Yang, "Dynamic dual threshold cooperative spectrum sensing for cognitive radio under noise power uncertainty," *Human-centric Computing and Information Sciences*, vol. 9, article no. 22, 2019. https://doi.org/10.1186/s13673-019-0181-x

[11] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, 2005, pp. 60-65.

[12] J. Lv and X. Luo, "Image denoising via fast and fuzzy non-local means algorithm," *Journal of Information Processing Systems*, vol. 15, no. 5, pp. 1108-1118, 2019.

[13] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080-2095, 2007.

[14] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, article no. 7, 2019. https://doi.org/10.1186/s42492-019-0016-7

[15] N. K. Kalantari, S. Bako, and P. Sen, "A machine learning approach for filtering Monte Carlo noise," *ACM Transactions on Graphics*, vo. 34, no. 4, article no. 122, 2015. https://doi.org/10.1145/2766977

[16] N. K. Kalantari, "Utilizing machine learning for filtering general Monte Carlo noise," Ph.D. dissertation, University of California, Santa Barbara, CA, 2015.

[17] S. Bako, T. Vogels, B. McWilliams, M. Meyer, J. Novak, A. Harvill, P. Sen, T. Derose, and F. Rousselle, "Kernel-predicting convolutional networks for denoising Monte Carlo renderings," *ACM Transactions on Graphics*, vol. 36, no. 4, article no. 97, 2017. https://doi.org/10.1145/3072959.3073708

[18] C. Schied, A. Kaplanyan, C. Wyman, A. Patney, C. R. A. Chaitanya, J. Burgess, S. Liu, C. Dachsbacher, A. Lefohn, and M. Salvi, "Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination," in *Proceedings of High Performance Graphics*, Los Angeles, CA, 2017, pp. 1-12.

[19] C. Schied, C. Peters, and C. Dachsbacher, "Gradient estimation for real-time adaptive temporal filtering," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 1, no. 2, article no. 24, 2018. https://doi.org/10.1145/3233301

[20] Y. Liu, C. Zheng, Q. Zheng, and H. Yuan, "Removing Monte Carlo noise using a Sobel operator and a guided image filter," *The Visual Computer*, vol. 34, no. 4, pp. 589-601, 2018.

[21] M. Gharbi, T. M. Li, M. Aittala, J. Lehtinen, and F. Durand, "Sample-based Monte Carlo denoising using a kernel-splatting network," *ACM Transactions on Graphics*, vol. 38, no. 4, article no. 125, 2019. https://doi.org/10.1145/3306346.3322954

[22] B. Bitterli, F. Rousselle, B. Moon, J. A. Iglesias-Guitian, D. Adler, K. Mitchell, W. Jarosz, and J. Novak, "Nonlinearly weighted first-order regression for denoising Monte Carlo renderings," *Computer Graphics Forum*, vol. 35, no. 4, pp. 107-117, 2016.

[23] D. Vicini, D. Adler, J. Novak, F. Rousselle, and B. Burley, "Denoising deep Monte Carlo renderings," *Computer Graphics Forum*, vol. 38, no. 1, pp. 316-327, 2019.

[24] H. Park, B. Moon, S. Kim, and S., E. Yoon, "P-RPF: pixel-based random parameter filtering for Monte Carlo rendering," in *Proceedings of 2013 International Conference on Computer-Aided Design and Computer Graphics*, Guangzhou, China, 2013, pp. 123-130.

[25] Z. Wang, X. Huang, and F. Huang, "A new image enhancement algorithm based on bidirectional diffusion," *Journal of Information Processing Systems*, vol. 16, no. 1, pp. 49-60, 2020.

[26] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 341-349, 2012.

[27] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, Vigo Galicia, Spain, 2016, pp. 5-10.

[28] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142-3155, 2017.

[29] K. M. Wong and T. T. Wong, "Deep residual learning for denoising Monte Carlo renderings," *Computational Visual Media*, vol. 5, no. 3, pp. 239-255, 2019.

[30] X. Yang, D. Wang, W. Hu, L. J. Zhao, B. C. Yin, Q. Zhang, X. P. Wei, and H. Fu, "DEMC: a deep dual-encoder network for denoising Monte Carlo rendering," *Journal of Computer Science and Technology*, vol. 34, no. 5, pp. 1123-1135, 2019.

[31] M. Kettunen, E. Harkonen, and J. Lehtinen, "Deep convolutional reconstruction for gradient-domain rendering," *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 126, 2019. https://doi.org/10.1145/3306346.3323038

[32] J. Talbot, D. Cline, and P. K. Egbert, "Importance resampling for global illumination," in *Proceedings of the Eurographics Symposium on Rendering Techniques*, Konstanz, Germany, 2005, pp. 139-146.

[33] B. Bitterli, C. Wyman, M. Pharr, P. Shirley, A. Lefohn, and W. Jarosz, "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting," *ACM Transactions on Graphics*, vol. 39, no. 4, article no. 148, 2020. https://doi.org/10.1145/3386569.3392481

[34] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Cambridge, MA: Morgan Kaufmann, 2014.

[35] P. Gallinari, Y. Lecun, S. Thiria, and F. F. Soulie, "Distributed associative memories: a comparison," in *Proceedings of COGNITIVA,* Paris, La Villette, 1987.

[36] D. G. Mixon and S. Villar, "Sunlayer: stable denoising with generative networks," 2018 [Online]. Available: https://arxiv.org/abs/1803.09319.

[37] N. K. Kalantari and P. Sen, "Removing the noise in Monte Carlo rendering with general image denoising algorithms," *Computer Graphics Forum*, vol. 32, no. 2pt1, pp. 93-102, 2013.

[38] T. Vogels, F. Rousselle, B. McWilliams, G. Rothlin, A. Harvill, D. Adler, M. Meyer, and J. Novak, "Denoising with kernel prediction and asymmetric loss functions," *ACM Transactions on Graphics*, vol. 37, no. 4, article no. 124, 2018. https://doi.org/10.1145/3197517.3201388

[39] H. Dahlberg, D. Adler, and J. Newlin, "Machine-learning denoising in feature film production," in *Proceedings of ACM SIGGRAPH 2019 Talks*, Los Angeles, CA, 2019.

[40] S. Laine, T. Karras, J. Lehtinen, and T. Aila, "High-quality self-supervised deep image denoising," *Advances in Neural Information Processing Systems*, vol. 32, pp. 6970-6980, 2019.

[41] C. R. A. Chaitanya, A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila, "Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder," *ACM Transactions on Graphics*, vol. 36, no. 4, article no. 98, 2017. https://doi.org/10.1145/3072959.3073601

[42] A. Keller, L. Fascione, M. Fajardo, I. Georgiev, P. Christensen, J. Hanika, C. Eisenacher, and G. Nichols, "The path tracing revolution in the movie industry," in *Proceedings of ACM SIGGRAPH 2015 Courses*, Los Angeles, CA, 2015.

[43] A. Alsaiari, R. Rustagi, M. M. Thomas, and A. G. Forbes, "Image denoising using a generative adversarial network," in *Proceedings of 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, Kahului, HI, 2019, pp. 126-132.

[44] S. Agarwal, A. Agarwal, and M. Deshmukh, "Denoising images with varying noises using autoencoders," in *Computer Vision and Image Processing*. Singapore: Springer, 2019, pp. 3-14.

[45] H. Dai and L. Shao, "Pointae: point auto-encoder for 3d statistical shape and texture modelling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, Korea, 2019, pp. 5410-5419.

[46] K. P. Kumar, "Fuzzy-based machine learning algorithm for intelligent systems," in *Data Management, Analytics and Innovation*. Singapore: Springer, 2019, pp. 321-339

[47] N. Chauhan and B. J. Choi, "Denoising approaches using fuzzy logic and convolutional autoencoders for human brain MRI image," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 19, no. 3, pp. 135-139, 2019.

[48] B. Costa and J. Jain, "Fuzzy deep stack of autoencoders for dealing with data uncertainty," in *Proceedings of 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, New Orleans, LA, 2019, pp. 1-6.

[49] J. Lehtinen, T. Aila, J. Chen, S. Laine, and F. Durand, "Temporal light field reconstruction for rendering distribution effects," in *Proceedings of ACM SIGGRAPH 2011 papers*, Vancouver, Canada, 2011, pp. 1-12.

[50] J. Lehtinen, T. Aila, S. Laine, and F. Durand, "Reconstructing the indirect light field for global illumination," *ACM Transactions on Graphics*, vol. 31, no. 4, article no. 51, 2012. https://doi.org/10.1145/2185520.2185547

[51] T, Hachisuka, W. Jarosz, R. P. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, and H. W. Jensen, "Multidimensional adaptive sampling and reconstruction for ray tracing," in *Proceedings of ACM SIGGRAPH 2008 papers*, Los Angeles, CA, 2008, pp. 1-10.

[52] P. Sen and S. Darabi, "On filtering the noise from the random parameters in Monte Carlo rendering," *ACM Transactions on Graphics*, vol. 31, no. 3, article no. 18, 2012. https://doi.org/10.1145/2167076.2167083

[53] P. Bauszat, M. Eisemann, S. John, and M. Magnor, "Sample-based manifold filtering for interactive global illumination and depth of field," *Computer Graphics Forum*, vol. 34, no. 1, pp. 265-276, 2015.

[54] Q. Zhang, Y. Li, F. Al-Turjman, X. Zhou, and X. Yang, "Transient ischemic attack analysis through non-contact approaches," *Human-centric Computing and Information Sciences*, vol. 10, article no. 16, 2020. https://doi.org/10.1186/s13673-020-00223-z

[55] B. Cantrell and N. Yates, *Modeling the Environment: Techniques and Tools for the 3D Illustration of Dynamic Landscapes*. Hoboken, NJ: John Wiley & Sons, 2012.

[56] X. Meng, Q. Zheng, V. Varshney, G. Singh, and M. Zwicker, "Real-time Monte Carlo denoising with the neural bilateral grid," in *Proceedings of the 31st Eurographics Symposium on Rendering (EGSR): Digital Library Only Track*, London, UK, 2020, pp. 13-24.

[57] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.

[58] F. Rousselle, C. Knaus, and M. Zwicker, "Adaptive sampling and reconstruction using greedy error minimization," *ACM Transactions on Graphics*, vol. 30, no. 6, pp. 1-12, 2011.

Soroor Malekmohammadi Faradounbeh **https://orcid.org/0000-0003-4606-8847**

She received her B.S. degree (Software Engineering) from Islamic Azad University, Mobarakeh Branch in 2013, and M.S. degree in Artificial Intelligence from Islamic Azad University, Najafabad Branch in 2017. Since September 2020, she is with the Department of CSE from Keimyung University as a PhD candidate. She is currently a PhD. Her current research interests include graphics algorithm, intelligent computing, virtual/augmented reality, game algorithms, machine learning and neural networks.

SeongKi Kim **https://orcid.org/0000-0002-2664-3632**

He received his Ph.D. degree in CSE from Seoul National University in 2009. He has researched the GPU and the GPGPU at the Samsung Electronics from 2009 to 2014. He has also researched at the Ewha Womans University, Sangmyung University and Keimyung University from 2014 to 2020. Since March 2020, he is an assistant professor at the Sangmyung University. His current research interests include graphics algorithms, an algorithm optimization with the GPU, and game/virtual/augmented reality. He is a member of the ACM and the IEEE.