IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 12, 2020, accepted May 25, 2020, date of publication June 4, 2020, date of current version July 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2999891

Path Tracing Denoising Based on SURE Adaptive Sampling and Neural Network

QIWEI XING AND CHUNYI CHEN
School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China

Corresponding author: Chunyi Chen (chenchunyi@hotmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant U19A2063, in part by the Jilin Provincial Science & Technology Development Program of China under Grant 20170101005JC, Grant 20180519012JH, and Grant 20190302113GX, in part by the 13th Five-year Science & Technology Research Program of Jilin Provincial Department of Education under Grant JJKH20200792KJ, and in part by the Jilin provincial industrial innovation funds under Grant 2016C091.

ABSTRACT A novel reconstruction algorithm is presented to address the noise artifacts of path tracing. SURE (Stein's unbiased risk estimator) is adopted to estimate the noise level per pixel that guides adaptive sampling process. Modified MLPs (multilayer perceptron) network is used to predict the optimal reconstruction parameters. In sampling stage, coarse samples are firstly generated. Then each noise level is estimated with SURE. Additional samples are distributed to the pixels with high noise level. Next, we extract a few features from the results of adaptive sampling used for the subsequent reconstruction stage. In reconstruction stage, modified MLPs network is adopted to model a complex relationship between extracted features and optimal reconstruction parameters. An anisotropic filter is used to reconstruct the final images with the parameters predicted by neural networks. Compared to the state-of-the-art methods, experiment results demonstrate that our algorithm performs better than other methods in numerical error and visual image quality.

INDEX TERMS Adaptive sampling, SURE estimator, MLPs network, path tracing, denoising.

I. INTRODUCTION

Path tracing algorithm can generate realistic images with lots of visual effects [1]–[4]. But this method relies on stochastic point samples of an intricate integrand, they often suffer from noise. This motivates path tracing denoising approaches, which broadly fall into two categories. One is Sample-based techniques which almost operate in pixel space. (e.g. Bako *et al.* [5], Bitterli *et al.* [6]). Because the error per pixel is different from each other in scenes, the samples number needed is also different. with large error, such as the object boundary regions and texture detail areas, need more samples to prevent the loss of the details. On the country, regions with small error acquire less samples to accelerate the rendering. Unfortunately, at high sampling rate, the adaptive sampling methods are not as efficient as the other methods. The other one makes use of the traditional image processing methods such as filtering approaches (e.g. Rousselle [7], [8], Sen and Darabi [9], Li *et al.* [10]). However, the major disadvantage is that the images would be blurred while denoising under low sampling rate.

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

In order to combine the advantages of two kind of approaches, we propose a novel algorithm that leverages the power of neural networks in the following key manners compared to previous denoising methods:

- Rather than predicting the pixel color, we utilize the neural network to predict the optimal reconstruction parameters. Then we can make use of a cross-bilateral filters to reconstruct the final noise-free images with these predicted parameters. And the quality of final images is higher.
- Compared with those methods which used the neural network to predict the kernel size of the filter, we introduce a set of additional features extracted from the path tracing rendering process, such as the mean and deviation of color in the neighborhood around the pixel, which is computed on each color component, the world position of the hit-points. And we set them as the input of neural network. The more the extracted features, the more accurate relationship between the filter parameters and the features. In other words, we can get the more optimal filter parameters easily.
- To prevent the spike pixels after reconstruction, we use the statistical methods to handle the spike pixels.

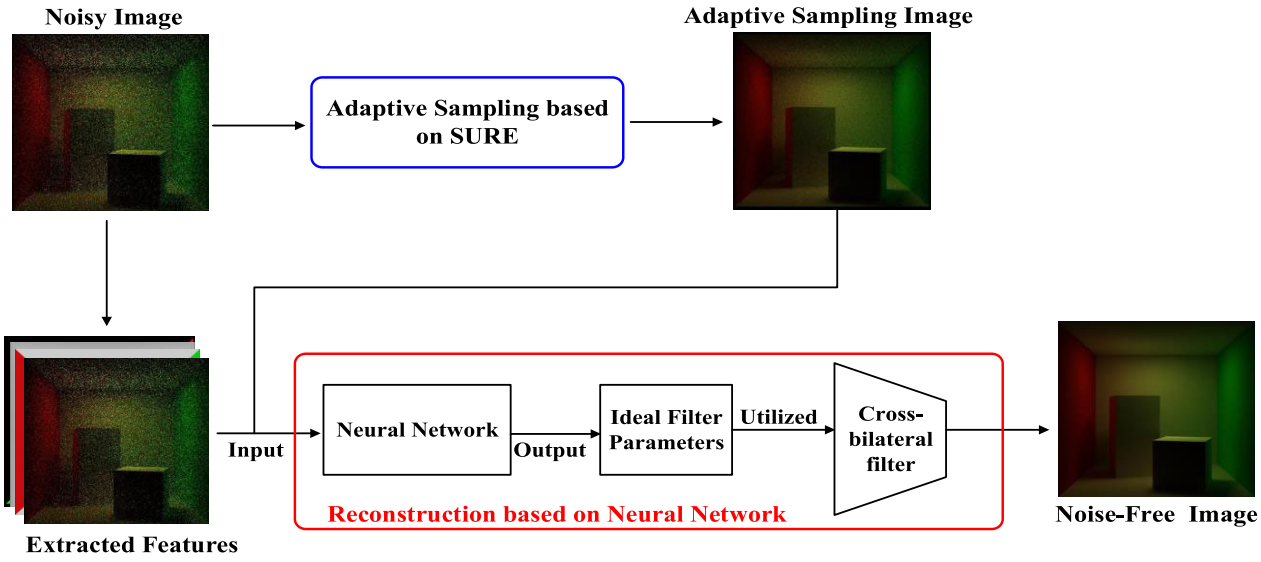

FIGURE 1. The framework of our major algorithm. As shown in fig, there are two major parts of our algorithm. The one is Adaptive Sampling based on SURE. And the other one is Reconstruction based on Neural Network. Input is the noisy images rendered by path tracing with low samples per pixel (spp). Firstly, we make use of the sampling process to deal with the input noisy image. At the same time, we extract a few features from input. Then we can get a better result than input. Secondly, we utilize the extracted features and the result of adaptive sampling as the input of neural network. Then with the neural network, we can get the ideal filter parameters. Finally, we use the cross-bilateral filter with these predicted parameters to reconstruct the final noisy-free images.

Meanwhile, to prevent the visual abrupt change, we converted the color space.

Overall, our major algorithm framework is shown in Fig.1.

We trained our network and extensively tested our algorithm on a variety of scenes, including various lighting phenomena such as distribution effects and diffuse and specular global illumination. Our method significantly reduces the error comparing to other methods, especially under the low sampling rate (e.g. 8 spp or less).

II. RELATED WORK

A. GENERAL IMAGE DENOISING

Image denoising is a classical yet still active topic in low level vision since it is an indispensable step in many practical applications. Over the past few decades, various models have been exploited for modeling image priors, including nonlocal self-similarity(NSS) models [11]–[14], sparse models [14]–[16], gradient models [17]–[19] and Markov random field(MRF) models [20]–[22]. In particular, the NSS models are popular in state-of-the-art methods such as BM3D [12], LSSC [14], NCSR [16] and WNNM [23]. However, most of the methods can hardly achieve high performance on denoising path tracing rendering, just because the noise of path tracing is with randomness. Thus, the natural image denoising methods are seldom used in denoising path tracing rendering.

B. DENOISING FOR PATH TRACING RENDERING

Zwicker *et al.* [4] present a recent survey of denoising techniques for Monte Carlo rendering. A priori methods characterize the structure of the integrand to derive analytical sampling rates and filters [2], [24]–[26]. They are often limited to a specific combination of light transport scenarios, such as motion blur, global illumination. To overcome these limitations, a lot of improved methods were proposed to denoise path tracing rendering.

A fruitful line of research adapted well-known image-processing filters [27], [28]. Unlike typical neural image denoising methods, path tracing rendering noise has strong variations across pixels. These methods often set the parameters of the filter according to a statistical estimate of the error obtained from the noisy images. The most successful techniques employ auxiliary features such as depth, normal or albedo. These extra features have been used to drive anisotropic diffusion or cross-bilateral filters [8], [10]. However, the reconstruction images with these methods are always fuzzy in low sampling rate.

Methods that work directly on the raw radiance samples often first reconstruct the radiance function using existing samples, then integrate the reconstructed function to generate images. McCool [29] use the nearest neighbors of each sample for reconstruction, which suffers from the curse of dimensionality when the integrand dimension is more than five. Sen and Darabi [9]'s method stands between the pixel-based and sample-based approach. However, they still rely on heuristics based on statistical aggregates. Hachisuka *et al.* [30] use 1D color histograms to match pixels that should be denoised jointly. However, their filter still operates on pixels and the histogram parameterization needs to be carefully set beforehand to cover the expected dynamic range. Delbracio *et al.* [31] extend adaptive manifolds to work directly with individual samples. They focus on limited global illumination and depth-of-field effects, but they do not handle motion blur.

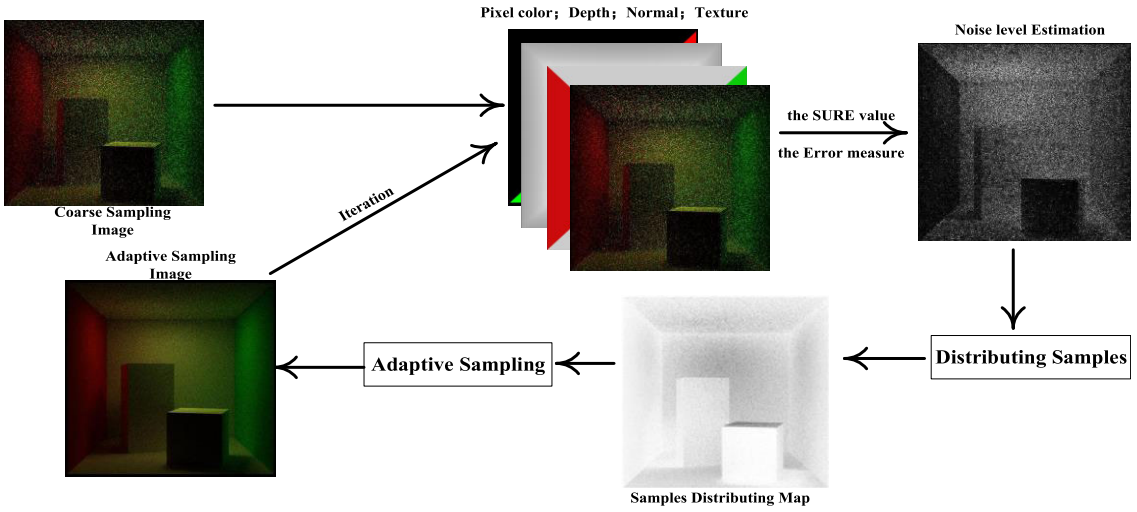

FIGURE 2. Our adaptive sampling method based on SURE.

C. NEURAL NETWORK FOR NOISY IMAGE RECONSTRUCTION

As a pioneer, Bauszat *et al.* [32] used machine learning to estimate the weights of filter automatically. Kalantari *et al.* [33] proposed the methods which combined the Gaussian denoiser and the residual learning of deep CNN and they obtained a better denoising results. Zhang *et al.* [34] improved the structure of networks. They introduced the graph-convolutional layers in order to exploit both local and non-local similarities and their architecture outperformed better than other CNN. However, most of their methods were not suitable for denoising the images generated by path tracing.

As for denoising the path tracing results, Bako *et al.* [5] used convolutional neural networks to infer not just the filter weights but also the form of a more complex filter kernel itself. Valsesia *et al.* [35] proposed a strategy which was based on the noisy data only. But the denoise process needed multiple images with different sampling rate, which took much more time. Lehtinen *et al.* [36] utilized a recurrent neural network to force the temporal coherence, enabling interactive denoising for real-time applications. Bako *et al.* [5] and Chaitanya *et al.* [37] extended the kernel prediction strategy in the work [6] by also considering temporal coherence at multiple scales. In concurrent works, Vogel *et al.* [38] attempted reconstruction for gradient-domain rendering, while Kettunen *et al.* [39] utilized raw samples as high-order statistics and a novel splatting approach to achieve better results with larger computational cost and storage space though.

We adopt the sampling strategy and parameters-predicting reconstruction with two key differences. The one is that we optimal the samples distribution, instead of unnecessary samples, so that, our number of samples is lower than other methods. Especially, compared the reconstruction method [40] based on SURE adaptive sampling, our method does not rely on sampling process to denoise the images. So, our method improve the efficiency a lot. The other one is that our method builds the relationship between the extracted features from the rendering and reconstruction parameters, instead of the features and pixel colors. We also adopt the optimization strategy of handling the spike pixels to obtain final images of better quality.

III. ADAPTIVE SAMPLE AND RECONSTRUCTION BASED ON NEURAL NETWORK

A. ADAPTIVE SAMPLING BASED ON SURE

In this paper, we utilize an adaptive sampling method based on SURE (Fig. 2).

As shown in Fig.2, our method includes multi-steps iterative adaptive sampling processes. The general steps are as follows:

Step 1. Coarse sampling. Generate the coarse image with a small number of samples per pixel by path tracing algorithm.

Step 2. Extracting features. Extract the features from coarse image such as the shading normal, depth, texture, visibility *et al.*

Step 3. Estimating the noise level. Calculate the variance in each feature space according to the feature extracted in Step 2. Then with the variance value, estimate the noise level per pixel.

Step 4. Distributing samples adaptively. According to the estimated noise level and the threshold predefined, increase and distribute corresponding number of samples per pixel. Then generate the adaptive sampling image by path tracing algorithm.

Step 5. Termination condition determination. If the noise levels of all pixels meet the termination conditions, adaptive sampling process ends. Otherwise, iteratively repeats the *Step 2* to *Step 5*.

Next, we will focus on how to use SURE to estimate the noise level.

1) SURE AND FILTER WEIGHT

In adaptive sampling process, it is necessary to assess the accuracy rate of the estimated value. Here, we utilize the SURE to evaluate the error between the estimated values and reference values.

The formula of evaluating SURE [10] is as follow:

$$\mathrm{SURE}\left(\overline{P_i}\right) = \left\|\overline{P_i} - P_i\right\|^2 + 2\sigma_i^2 \frac{\mathrm{d}\overline{P_i}}{\mathrm{d}P_i} - \sigma_i^2, \tag{1}$$

where $\overline{P_i}$ and P_i represent the reconstruction value and coarse sampling value, respectively. The derivative term could be computed as follows:

$$\frac{\mathrm{d}\overline{P_i}}{\mathrm{d}P_i} = \frac{1}{\sum_{j=1}^{n}\omega_{ij}} + \frac{1}{\sigma_c}\left(\frac{\sum_{j=1}^{n}\omega_{ij}P_i^2}{\sum_{j=1}^{n}\omega_{ij}} - \overline{P_i}^2\right), \tag{2}$$

where ω_{ij} represents the parameter of reconstruction method based on filter, and the parameter σ_c represents the standard deviation of samples in color space. Here we use the cross-bilateral filter. Therefore, we could get the filter weight parameters [11] as follows:

$$\omega_{ij} = \exp\left(-\frac{\left\|p_i - p_j\right\|^2}{2\sigma_p^2}\right)\exp\left(-\frac{\left\|c_i - c_j\right\|^2}{2\sigma_c^2}\right) \times \prod_{k=1}^{m}\exp\left(-\frac{\left\|f_{ik} - f_{jk}\right\|^2}{2\sigma_k^2\left(\sigma_{ik}^2 + \sigma_{jk}^2\right)}\right), \tag{3}$$

where $\sigma_p, \sigma_c, \sigma_k$ represent the standard deviation in the image space, color space and feature space, respectively. m represents the number of features extracted from coarse images. f_{ik} represents the sample mean of the k-th features and σ_{ik} represents the corresponding sample standard deviation. Here we utilize three kinds of features, shading normal, depth and texture.

2) ADAPTIVE SAMPLES DISTRIBUTION

Next, we utilize the SURE to estimate the noise level. For pixel i, the noise level estimator [10] can be computed by

$$L_i = \frac{\mathrm{SURE}\left(\overline{P_i}\right) + \sigma_i^2}{\overline{P_i}^2 + \delta}, \tag{4}$$

where, δ is a minimal value to prevent denominator zero. σ_i^2 represents the sample variance at pixel i. Then the noise level estimated can be used in the adaptive sampling process. We define four kinds of spatially varying samples number, which denoted as $\alpha = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ that goes from less to more. The choice threshold is accumulated as $\Phi = \{\Phi_1, \Phi_2, \Phi_3, \Phi_4\}$. Therefore, the samples number is simply chosen by considering the estimator L_i by

$$\begin{cases} if & 0 < L_i \le \Phi_1 & \alpha = \alpha_1 \\ else\ if & \Phi_1 < L_i \le \Phi_2 & \alpha = \alpha_2 \\ else\ if & \Phi_2 < L_i \le \Phi_3 & \alpha = \alpha_3 \\ else & & \alpha = \alpha_4 \end{cases}, \tag{5}$$

where α represents the number of increase samples per pixel. With α choice, the images are rendered by path tracing algorithm. Additionally, to prevent discontinuity(or ringing artifact) between two neighbor pixels, we set our varying of samples number as $\{1, 2, 4, 8\}$ and $\{1, \sqrt{2}, 2, 4\}$.

B. RECONSTRUCTION BASED ON NEURAL NETWORK

The goal of image reconstruction approaches is to take a noisy input image rendered with only a few samples and generate a noise-free image that is like the ground truth rendered with many samples. Expressed mathematically, the reconstruction image $\widehat{\boldsymbol{I}} = \left\{\widehat{I_r}, \widehat{I_g}, \widehat{I_b}\right\}$ at pixel i is computed as a weighted average of all pixels in a square neighborhood $N(i)$ (centered on pixel i) [11]

$$\widehat{\boldsymbol{I}}_i = \frac{\sum_{j\in N(i)}\omega_{i,j}\bar{\boldsymbol{I}}_j}{\sum_{j\in N(i)}\omega_{i,j}}, \tag{6}$$

where $\bar{\boldsymbol{I}}_j$ represents the noisy pixel color. $\omega_{i,j}$ is the weight between pixel i and its neighbor j as computed by the formula (3). Here, we modify this formula. Specifically, we replace the items $\frac{1}{\left(\sigma_{ik}^2+\sigma_{jk}^2\right)}$ in the formula with D_k, which represents the parameter of distance in feature domains. Then the modified formula is as follows:

$$\omega_{i,j} = \exp\left(-\frac{\left\|p_i - p_j\right\|^2}{2\sigma_p^2}\right)\exp\left(-\frac{\left\|c_i - c_j\right\|^2}{2\sigma_c^2}\right) \times \prod_{k=1}^{m}\left(-\frac{D_k\left\|f_{ik} - f_{jk}\right\|^2}{2\sigma_k^2}\right). \tag{7}$$

Thus, the quality of the reconstruction image depends on the parameters of filter σ_p, σ_c, σ_k. The challenge is how to find the optimal values of filter parameters to produce the highest-quality results. As the proposed theory [36], there exists a complex non-linear relationship between multi-dimensional features and optimal filter parameters. We use a modified MLPs network to fit the relationship. Our reconstruction framework is shown in Fig.3.

1) MULTI-DIMENSIONAL FEATURES

Multi-dimensional features are those directly output by the rendering system. We use seven kinds of features as the input of neural network, including pixel color, shading normal, texture values, direct illumination visibility, world position, the mean and standard deviation of pixel color in the neighborhood around pixels, and the mean and standard deviation of visibility in the neighborhood around pixels. These features were also used in reference in reference [8] and we found it to be useful in our framework. During rendering, for each pixel, we describe the world position in Cartesian coordinates (x, y, and z), color in RGB format, shading normal (i, j, and k), texture values in RGB format, and a single float value for the direct illumination visibility. Totally, there are twenty-five float values as the input of our neural network. Additionally, the mean and deviation of pixel color in the neighborhood

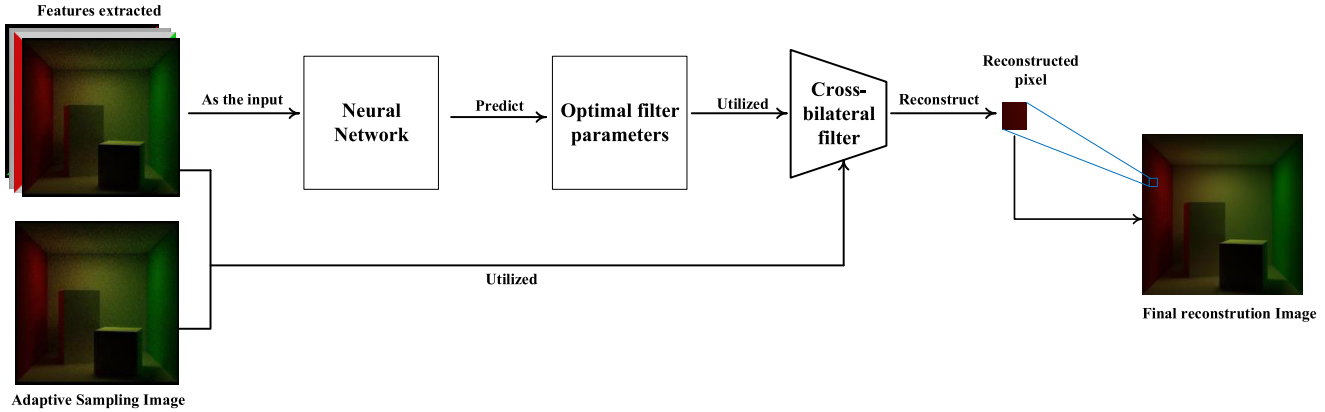

FIGURE 3. Our reconstruction framework. As shown in Figure, Our reconstruction approach takes the adaptive sampling results as origin images and extract multi-dimensional features including color, depth, shading normal, texture and position from the adaptive sampling results. The network takes these extracted features as input and outputs the ideal filter parameters. The filter makes use of these parameters and outputs the reconstructed result per pixel. When all pixels are reconstructed, we can obtain a final reconstruction image. Besides, our framework supports other derivable filters, such as the gaussian filter, cross non-local mean filter.

around the pixel are computed on each color component (six float values total).

2) OUR NEURAL NETWORK

As with any neural network, there are three crucial elements we should describe: (1) the structure of network, (2) an appropriate loss function to measure the distance between the reconstruction result and ground truth images, and (3) an optimization strategy to minimize the loss function.

a: THE STRUCTURE OF OUR NETWORK

We use a modified MLPs network since it is a simple, yet powerful system for discovering complex nonlinear relationships between its inputs and outputs. Moreover, MLPs are inherently parallel and can be efficiently implemented on a GPU. Our modified MLPs is shown in Fig.4. Illustrated as Fig.4, four layers are presented in the network. For clarity, we only show connections for one node at each layer. Each layer has several nodes fully connected by defined weights,

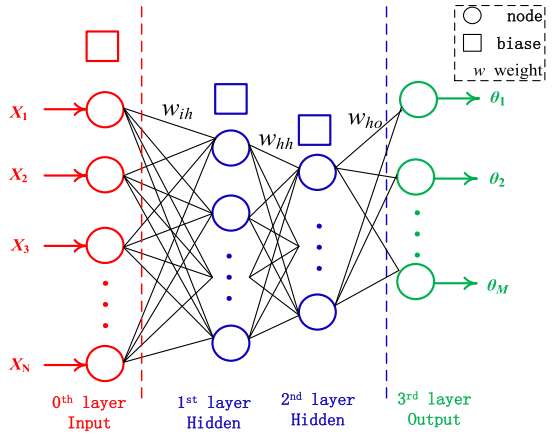

FIGURE 4. Our MLPs neural network per pixel.

where weight ω_{ij} represents connects node at layer i to node at layer j, and i, j are just the one of the three kinds of layers, input, hidden, or output. $X_1, X_2, \ldots, X_n$ represent the multi-dimensional features. $\theta_1, \theta_2, \ldots, \theta_M$ represent the predicted parameters of filter. As well, each layer has a bias item shown in Fig 4. Assume the output of last layer is c_t^{l-1}, the output of current layer can be computed by [32]

$$c_s^l = f^l\left(\sum_{t=1}^{n_{l-1}} \omega_{t,s}^l c_t^{l-1} + b_s^l\right), \tag{8}$$

where we present last layer as $l-1$, the current layer as l. $\omega_{t,s}^l$ represents the weight between two layers and b_s^l represents the bias item. Here, f^l is the activation function. In order to improve slow convergence and low-accuracy of general MLPs network, we replace the original sigmoid function used in hidden layers with the *relu-softsign* function. The expression of our activation function is

$$f(x) = \begin{cases} \dfrac{x}{1+|x|}, & x \le 0 \\ x, & x > 0 \end{cases}. \tag{9}$$

When x>0, the derivative of the function is a constant. When x≤0, the derivative of the function decreases slowly, and it can provide non-zero derivative for the non-positive number near zero. Therefore, the function can have higher accuracy than other functions and a faster convergence. As for the output layer, the *softplus* function is used as the activation function. Additionally, the activation function should be the non-linear function, otherwise the network would degenerate into a simple linear regression model. Compared with general MLPs network, we incorporate the filter into the training and testing process. The "backpropagate" of general MLPs network updates the weights through the results of the network. And our "backpropagate" is according to the results of the filter. So, this implies that the filter must be differentiable with respect to its filter parameters. Fortunately, our utilized cross-bilateral filter is differentiable.

Especially, the complexity of neural network depends on the number of layers and the nodes in each layer. Although the complex network can fit a complex relationship between the input and the output, the training process would be too hard to implement. Besides, the "over fitting" would occur much easier in complex models. When implementing neural networks, we set the number of nodes in input layer as twenty-five. Similarly, we set the number of nodes in hidden layers as ten. The number of nodes in output layers is seven.

b: OUR LOSS FUNCTION

Next, we describe our loss function, which is the metric to measure the results error between the reconstructed and ground truth. Compared with the widely used loss function, like MSE (mean squared error) and CEE (cross entropy error), we use a modified RelMSE(relative mean squared error) proposed by Rousselle *et al.* [41], which is more suitable for our application. Since the human visual system is more sensitive to color variations in darker regions of the image, we modify the RelMSE as follows:

$$E_i = \frac{n}{2} \sum_{q\in\{r,g,b\}} \frac{\left(\hat{c}_{i,q} - c_{i,q}\right)^2}{c_{i,q}^2 + \varepsilon}, \tag{10}$$

where n is the samples number. $\hat{c}_{i,q}$ and $c_{i,q}$ represent the pixel color in q^{th} color component of reconstruction image and the truth conference image at pixel i, separately. ε is a small number (10^{-3} in our implementation) to avoid division by zero. Here, the division by $c_{i,q}^2$ accounts for the perceptual effect by giving higher weight to the regions where the ground truth image is darker. As for the multiplication by $n/2$, the squared error decreases linearly with the samples number in path tracing rendering. The fewer samples, the greater errors in images. By multiplying the squared error by n, we cancel this inverse relationship and force all the images to have an equal contribution to the error regardless of their sampling rate. Furthermore, the error is divided by 2 to produce a simpler derivative [see Formula (12)].

c: OUR OPTIMIZATION STRATEGY TO TRAIN THE NETWORK

In fact, training network is a process of updating weights between two layers of network to minimize the loss function value. Therefore, we first need an optimization strategy for training network. Like most neural networks, we train the network through an iterative process called backpropagation. Our BP method include three steps: 1) the weights are randomly initialized to small values around zero (from -0.9 to 0.9 when implementing). Then compute the results of all nodes with the Formula (8) and this process can be implemented efficiently using a series of matrix multiplications. 2) The error between the computed and desired outputs is used to determine the effect of each weight on the output error. This requires taking the derivative of the error with respect to each weight. Besides, the activation functions also need to be differentiable. The two steps are performed for all the data used in training set and the error gradient of each weight is accumulated. 3) The weights are updated according to the accumulated error gradient. Carrying out all above steps iteratively until convergence. We called a single completed iteration of training as an *epoch*. And we need many epochs in whole training process to obtain a converged set of weights.

Since the backpropagation process need to compute the derivative of the loss function with respect to the weights between each two layers. With math symbol, the derivative can be expressed as $\partial E/\partial \omega_{t,s}^l$. Then using the chain rule, we can obtain a formula of computing the derivative [32] as follows:

$$\frac{\partial E_i}{\partial \omega_{t,s}^l} = \sum_{m=1}^{M} \left[\sum_{q\in\{r,g,b\}} \left[\frac{\partial E_{i,q}}{\partial \hat{c}_{i,q}} \frac{\partial \hat{c}_{i,q}}{\partial \theta_{m,i}} \right] \frac{\partial \theta_{m,i}}{\partial \omega_{t,s}^l} \right], \tag{11}$$

where M is the number of the filter parameters. The first item is the derivative of the loss function with respect to the reconstructed results and it can be computed rapidly as follows [32]:

$$\frac{\partial E_i}{\partial \hat{c}_{i,q}} = n \frac{\hat{c}_{i,q} - c_{i,q}}{c_{i,q}^2 + \varepsilon}. \tag{12}$$

Additionally, $\theta_{m,i}$ is the output of our network, which represents the optimal filter parameters. So, the middle item must be differential. Luckily, our filter is the differential one. Therefore, the middle item can be computed as follows [32]:

$$\frac{\partial \hat{c}_{i,q}}{\partial \theta_{m.i}} = \frac{\partial h_q\left(\bar{\mathbf{s}}_{N(i)}, \boldsymbol{\theta}_i\right)}{\partial \theta_{m,i}}, \tag{13}$$

where the $\bar{\mathbf{s}}_{N(i)}$ represents gather of the features of the neighborhood around pixel i. And h is our filter function. $\boldsymbol{\theta}_i$ represents the gather of M filter parameters.

3) OUR TRAINING DATA SET

In our training set, we used about thirty scenes with different distributed effects such as depth of field, motion blur, soft shadow, and global illumination. Additionally, for strengthening network training, we also used the Animation data set published by Disney [42]. Fig 5 shows some images of our training scenes. For each scene, we rendered the ground truth image with many samples (e.g., 4K, 8K, 16K or 32K) per pixel and trained all the images on the same network. Moreover, to avoid overfitting to one noise pattern, we rendered each scene five times at each sampling rate (totally 25 images for each scene). To train the network on all images efficiently, we performed mini-batch training [43], where

FIGURE 5. A subset of our training scenes.

each iteration of the backpropagation process is performed on a subset of the training set. Our subset consisted of one image at each sampling rate from all the training scenes. Furthermore, we alternated between the five noisy examples at each iteration.

4) HANDLING SPIKE PIXELS

Since path tracing rendered images could contain spike pixels, whose color is greater than their neighbors. Unfortunately, these pixels cannot be spread efficiently just with the filter reconstruction. Thus, we need to handle the spike pixels as a post-processing. In our implementation, we identify these spikes in the reconstructed result by calculating the mean and standard deviation of the colors for all the neighboring pixels in a block of size 5×5. If any color channel of the center pixel has values more than 2 standard deviations away from the block average, then it is labeled as a spike pixel. We remove the color of spike pixel and fill the pixel with the median block color. Besides, since the human eyes are sensitive to the abrupt change of color in RGB color space, we converted the pixel color in RGB space to the CLE-Lab space in the handling process to avoid the discomfort of human eyes. Although the handling process is not expressed as a differentiable one, it does not affect our training process without the handling process.

IV. RESULTS AND DISCCUSION

We implemented our algorithm in C++ and integrated it into PBRT2 [44]. All results were obtained on an Intel(R) Xeon(R) E5-1603 V4 2.8GHz machine with 32GB of memory and a NVIDIA Quadro K5200 GPU. We used CUDA for accelerating the reconstruction process.

Note that, we describe the advantages of our algorithm framework in three ways: (1) compared with other widely used reconstruction methods based on adaptive sampling and filter, (2) compared with the LBF algorithm, which also used the BP network for reconstruction,(3) compared with the widely used reconstruction methods based on Convolutional Neural Networks (CNN). To describe the quality of the image, we utilize RelMSE [41], SSIM [45] and PSNR [45].

A. COMPARED WITH THE GENERAL RECONSTRUCTI-ON METHODS BASED ON SAMPLING OR FILTERING

Fig.6 shows the comparison between our approach and state-of-the-art sampling methods on two scenes with different distributed effects. First, we examine the path traced *Sibenik* scene with global illumination [shown above Fig.6(a)]. Our average samples number is about 15 spp. And the other methods are at 16 spp. The images rendered by Low Discrepancy and Fuzzy suffer serious noise both in low frequency regions (red rectangle and green rectangle) and high

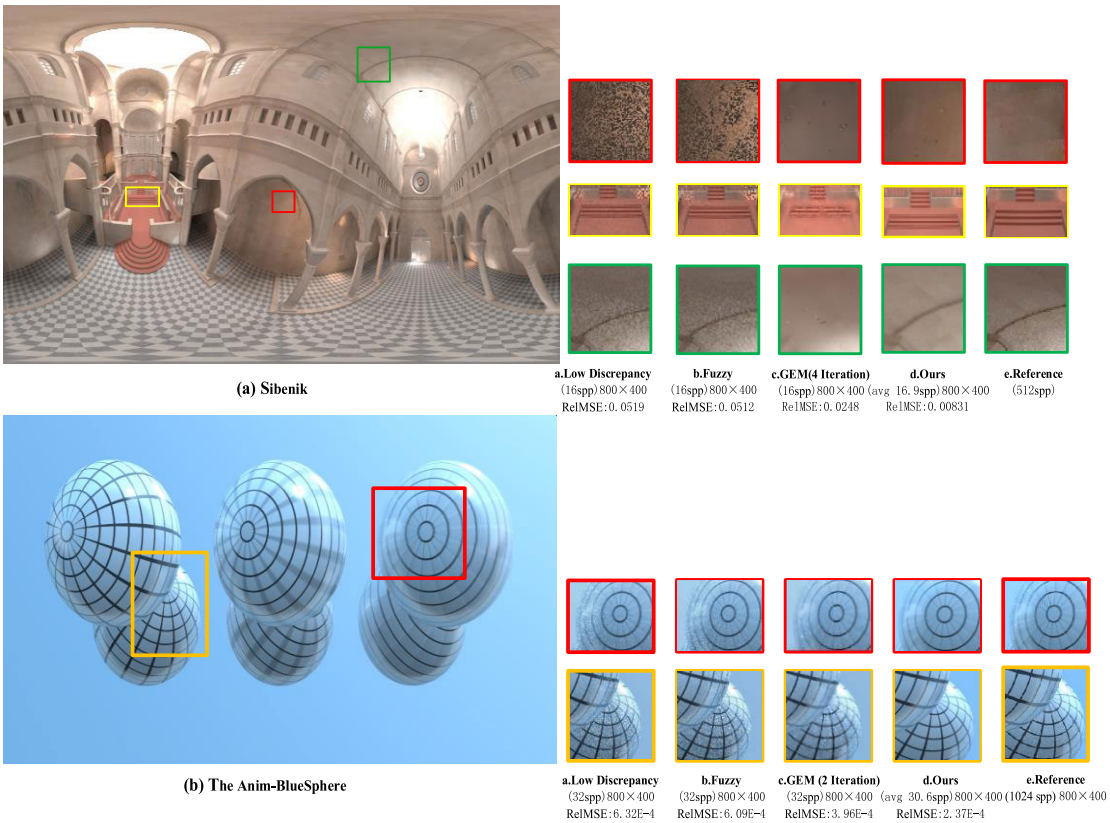

FIGURE 6. We compare our results with baseline methods based on sampling: Low Discrepancy [43], Fuzzy [46] and GEM [41] on a test set rendered scenes rendered in 16 spp or 32 spp. The sampling rate of ours is the closest value after averaging. And below the image, we calculate the RelMSE metric for measuring image quality.

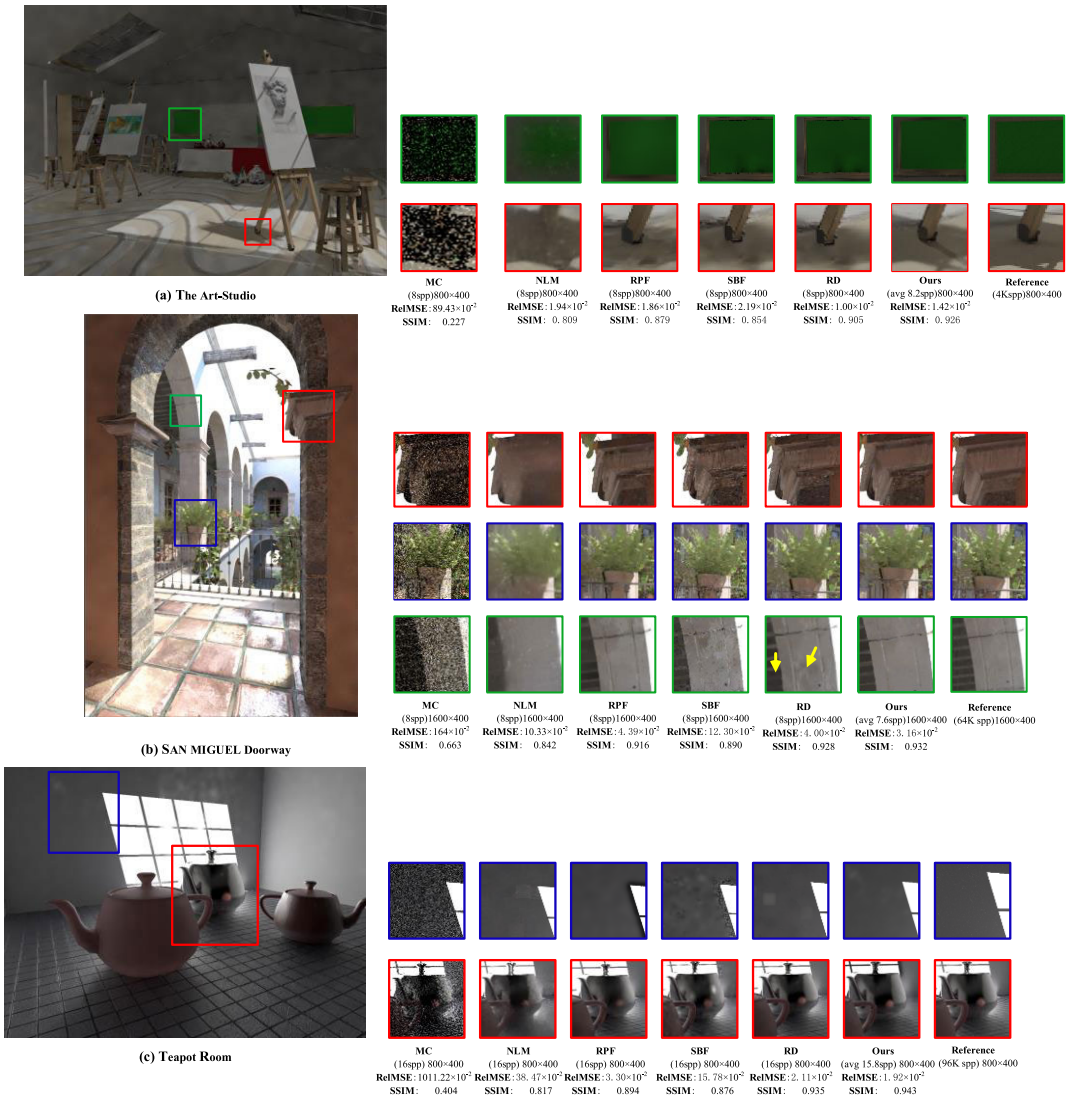

FIGURE 7. **We compare our results with baseline methods based on filter: NLM [7], RPF [9], SBF [10] and RD [8] on a test set rendered scenes rendered in 8spp or 32 spp. The sampling rate of ours is the closest value after averaging. And below the image, we calculate the value of RelMSE and SSIM for measuring image quality.**

frequency regions (yellow rectangle). Although the noise of level of GEM is as good as ours, it does not preserve the essential details, like the decorative design on the ceiling (green rectangle) and the image is more blur than ours. Our approach can not only preserve more details than others, but it also generates image of better quality (indicated under the image as RelMSE). The *Anim-Blue Sphere* scene is a challenging, path-traced scene containing three balls with mostly motion-blur [shown below Fig.6(b)]. This scene is rendered with 32 spp. Our average is about 31 spp. Three balls rotate with different speed increasing from left to right. The red rectangle shows that our method can sample the motive detail better than Low Discrepancy and F-divergences, whose image is noisy. Seen from the detail on the static ball (yellow rectangle), our method can remove the outlier spikes and generate a clearer image while GEM obviously causes an overly blurred one. Note that the RelMSE of ours is less than other methods.

Fig.7 shows the comparison between our approach and state-of-the-art reconstruction methods base on filter on three scenes with different distributed effects. For completeness, we perform approximately same sample comparisons to ensure fairness. First, the examined scene the *Art-Studio* [shown above Fig.7(a)] is a path-traced one which includes direct illumination and soft shadow. The NLM removes the noise in shadows, but over blurs the geometry of the easel-legs and the edge between the easel and shadows (red rectangle). Moreover, although RPF, SBF and RD use additional features, they often do not have appropriate filter weights, resulting in either over blurred the edge between the easel and shadows or residual noise on the texture of blackboard (green rectangle). Although our RelMSE is a little higher than RD, our result is smoother and with fewer artifacts than RD, which can be indicated from the SSIM. The *SAN MIGUEL Doorway* [shown in the middle of Fig.7(b)]

is a path-traced scene with severe noise at 8 spp. Again, NLM over blurs the texture of the plant (blue rectangle) on the marble column (red rectangle) and on the stone arches (green rectangle). Moreover, SBF and RPF produce results that are over blurred or contain residual noise, for example, on the marble column (red rectangle) and near the plant (blue rectangle). Although RD and our method preserve the texture on the stone arches (green rectangle), RD cannot properly remove the noise in the smooth regions (indicated by yellow arrows). Despite having a lowest RelMSE, we produce a relatively noise-free result that is better than the other methods both visually and in terms of SSIM. The *Teapot Room* scene [shown below Fig.7(c)] is a challenging, path-traced scene containing one glossy and two diffuse teapots with indirect illumination. Unfortunately, none of the other methods can effectively remove the strong indirect illumination noise on the back wall (blue rectangle) in low sampling rate. Note the ground truth image still has visible spikes at 96K spp, while we produce a relatively noise-free result. Seen from the SSIM and RelMSE, our image is the of higher quality than the other methods.

B. COMPARED WITH THE LBF ALGORITHM

Since LBF algorithm proposed in references [32] also uses MLPs network to reconstruct the image of MC noise. Our method modified the reconstruction method based on MLPs network. Here we compare our methods with LBF algorithm in some scenes. Although LBF algorithm can remove the noise of path tracing rendering, but it is not suitable for low sampling rate, since it does not distribute the samples optimally. We choose three test scenes for the comparison between our algorithm and the LBF methods in the low sampling rate. The results are shown in Fig 8.

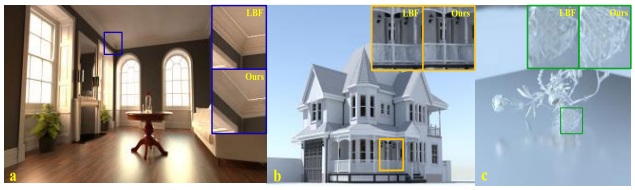

FIGURE 8. We compare our results with LBF algorithm on a test set rendered scenes. Among them, scene a is the Living Room rendered in 16spp. Scene b is the Vinctorian Style House. Scene c is Yeah Right. Scene b and Scene c are rendered in 8spp. The sampling rate of ours is the closest value after averaging.

From the results, the *Living Room* [shown in Fig.8(a)] is a path-traced scene that includes global illumination. There are totally three ambient sources and a glossy wood floor with high light. From the details on the ceiling (blue rectangle), the reconstruction image of LBF is noisy and aliasing. Our image is smoother, visually noise-free and not aliasing. The *Vinctorian Style House* [shown in Fig.8(b)] is a path-traced scene that contains many texture details. Seen from the details of the columns at the vestibule (yellow rectangle), although LBF can remove most noise, the result is much aliasing and over blurred. Our result is smoother and clearer at the part of details. *Yeah Right* [shown in Fig.8(c)] is a path-traced scene with mostly glossy specular reflection effects. The detail contrast among the LBF and ours shows that the result of LBF is much over blurred and of lower quality than ours.

C. COMPARED WITH THE RECONSTRUCTION METHODS BASED ON NEURAL NETWORK

To compare various path tracing denoising algorithms based on neural networks, we conduct experiments on noisy input images with different sample per pixel. Fig.9 shows some results and closeups from three typical scenes, including *Bathroom* with mirrors, *Pink Room* with colorful glasses under reflection and refraction situations and *White Room* that are relatively dim and make global illumination more difficult to denoise.

Overall, our work performs consistently on a par or better than the state-of-the-art methods in terms of both perceptual quality and quantitative metrics. NFOR, one of the best traditional offline filtering methods, suffers from splotchy-looking results and residual noise due to limited filter kernel size. Learning based methods (KPCN and RAE) generally obtain better results in low frequency areas but may produce over-smoothed ones with approximate colors for shading details. Our approach is satisfactory in both low-frequency and high-frequency areas and better detail preservation due to the sampling process.

D. DISCUSSION AND ANALYSIS

In our method, we adaptively distribute the samples per pixel and estimate the optimal filter parameters by training a modified network through direct minimization of the error between the reconstruction and ground truth images. There are two major differences between other methods and ours. First, as discussed in *Sec IV A*, the adaptive sampling based on SURE provides an optimal strategy of samples distribution, which can reduce the error caused by noise of samples and in some cases, a small error might result in obvious noisy pixels. Second, we apply a different activation function in hidden layers. Other methods mostly make use of the *ReLU* function. To demonstrate the advantages of our function, we compare the performance of different activation functions on the network of the same structure. *Table I* shows aggregate numerical performance of different activation functions. Since the sampling randomness of path tracing and the influence of different effects, the RelMSE and SSIM metric is fluctuated. If the change of the metric is less than 10^{-3}, we consider the algorithm is convergence. From the result, perceptually and quantitatively, our function outperforms the state-of-the-arts especially for small number of training steps.

Aggregate numerical performance of all activation functions over the dataset. Here, we compute the SSIM and the RelMSE (Here, we use the average of the dataset as the final calculation result). And all activation functions are used in our network of same structure.

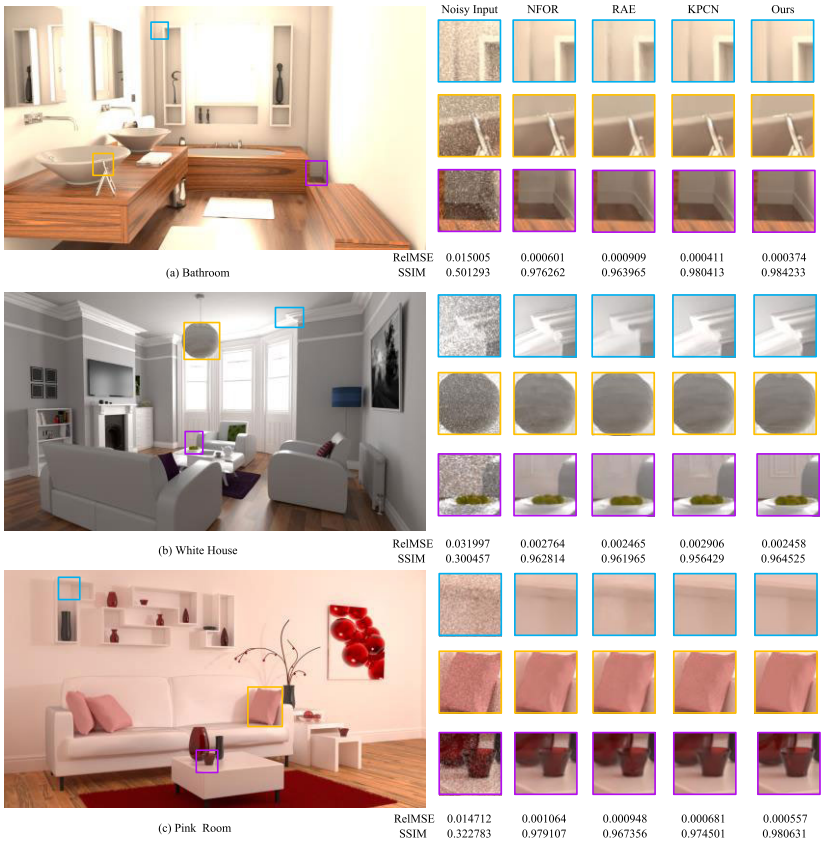

FIGURE 9. We compare our results with baseline methods: NFOR [6], RAE [38] and KPCN [5] on a test set rendered scenes rendered in 8 spp. The sampling rate of ours is the closest value after averaging. Below the images, we calculate the value of RelMSE and SSIM for measuring the quality of the image.

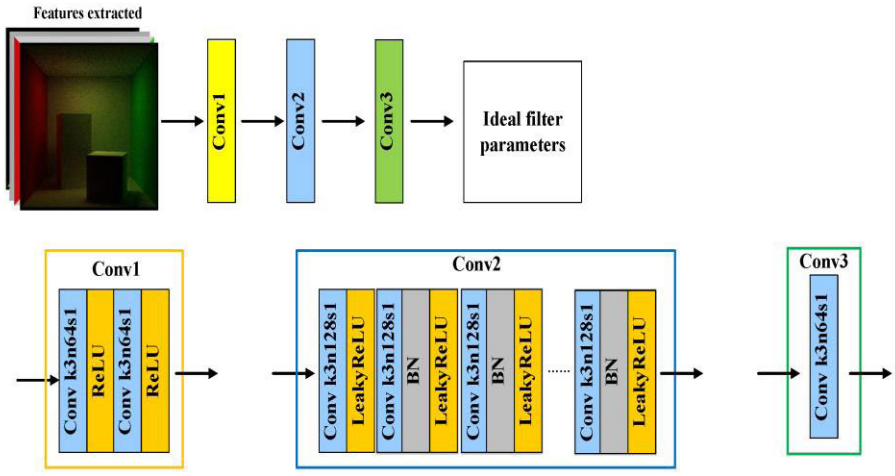

FIGURE 10. Overview of our utilized CNN. There are three different part in our CNN: Conv1, Conv 2 and Conv3. Interpretation of network layer annotations: e.g., k3n64s1 indicates that kernel size is 3, number of feature channels is 64 and stride is 1.

Of course, the MLPs network used in our algorithm is not the most optimal one. Inspired by the methods [5], we replace our network with a deep CNN. Fig.10 shows the structure of the deep CNN. We compare the results of the replaced CNN and the MLPs on different scenes including the Living Room, Vinctorian Style House, White House and Living Room.

Fig.11 shows the compared images with details. Fig.12 shows the quality assessment index of the scenes shown

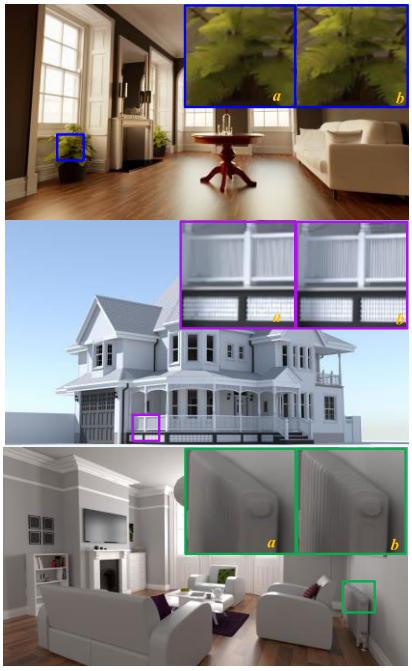

FIGURE 11. The comparison of rendered results between the CNN and MLPs. Shown above fig, the left details (marked as a) is reconstructed by the network with MLPs structure. And the right details (marked as b) is reconstructed by the network with CNN structure. From the comparison of details, we can find the texture of right are enhanced in comparison with the left.

in Fig.11, including the PSNR and SSIM. The results show that our experiment obtained significant improvement with the replaced CNN. This is largely due to that the CNN easily automatically models the relationship between the input auxiliary features of noisy image and the output filter parameters.

E. EXTENSION AND DISCUSSION

Extension. To further improve the accuracy of the network output, we could employ the GAN(generative adversarial network) to replace the CNN. Here we utilizes the structure of networks propose by Xin *et al.* [47] and divided input of our networks into 2 categories: diffuse part and specular part. We set the features including shading normal, world position, depth, texture of diffuse part and the texture of specular part as the input of the Encoder-Net and predict the features of diffuse and specular separately. Next, we can obtain the output of diffuse part and specular part. Finally, we merge the results of above two parts, which are our reconstruction images. The structure of the network is show as Figure 13.

We choose three scenes to test the extension algorithm, including the Livingroom shown in Fig.8(a), the White House shown in Fig.9(b) and the Vinctorian Style House shown in Fig.8(b). We compute the SSIM and PSNR of our methods with deep CNN and our extended method with Encoder-Net(E-Net). The results can be seen in Table 2. From the results, the extended method outperforms better.

TABLE 1. Different activation performance comparison.

Training steps	Functions	SSIM ($\times 10^{-2}$)	RelMSE ($\times 10^{-3}$)
50	*tanh*	70.14	0.374
	ReLU	72.29	0.337
	sigmoid	69.35	0.445
	ours	**74.48**	**0.363**
100	*tanh*	84.70	0.180
	ReLU	90.54	0.086
	sigmoid	90.31	0.094
	ours	**91.46**	**0.079**
150	*tanh*	94.14	0.067
	ReLU	95.82	0.046
	sigmoid	95.42	0.052
	ours	**96.67**	**0.033**
200	*tanh*	97.74	0.036
	ReLU	**98.28**	**0.027**
	sigmoid	98.05	0.031
	ours	98.24	0.029

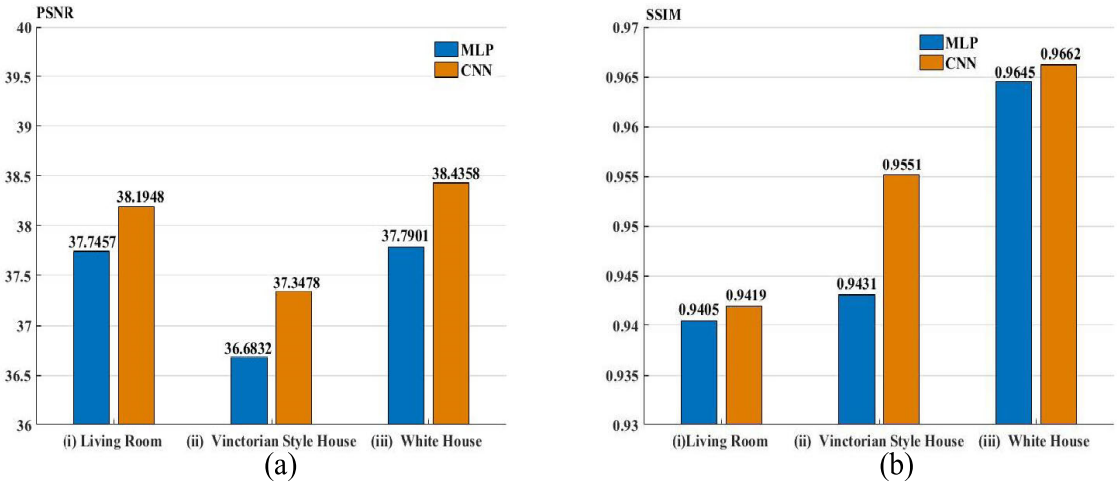

FIGURE 12. Performance comparison of PSNR and SSIM over three different scenes to between the MLPs structure and CNN structure. The subfigure (a) is the PSNR comparison results. And subfigure (b) is SSIM comparison results. In each subfigure, the horizontal axis represents different scenes, from left to right: (i) the Living Room scenes; (ii) the Vinctorian Style House scenes; (iii) White House scenes.

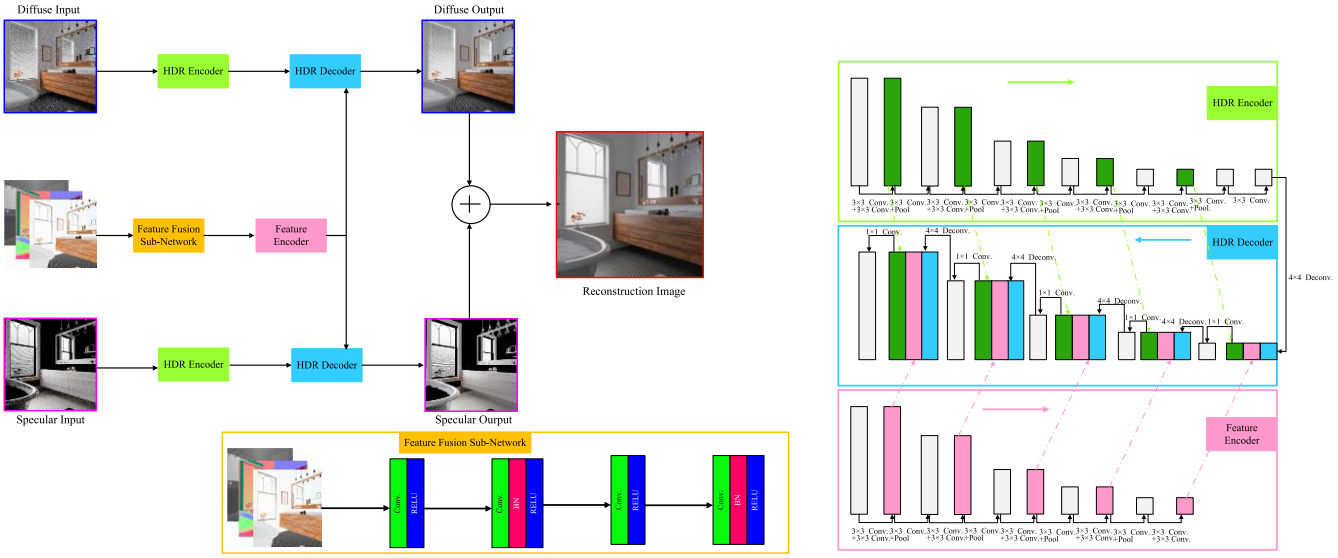

FIGURE 13. Overview of our utilized Encoder-Network. Conv. = convolutional layer, Pool = max pooling, and Deconv. = deconvolutional layer.

TABLE 2. Comparison between two methods with different network.

Scenes	SSIM	PSNR
Living Room	(CNN)0.9419	(CNN)38.1948
	(E-Net)**0.9477**	(E-Net)**38.3023**
White House	(CNN)0.9662	(CNN)38.4358
	(E-Net)**0.9714**	(E-Net)**38.5537**
Vinctorian Style House	(CNN)0.9551	(CNN)37.3478
	(E-Net)**0.9609**	(E-Net)**37.4551**

Limitation. CNN and the E-Net both limit the effectiveness of auxiliary features to early layers, especially when the size of the dataset is small. When denoising, the texture appears to be erroneously enhanced and the reflected illumination is weakened just as the Fig 14. The explanation would be that the diffuse and specular components have different noise patterns with different characteristics. Thus, we could not get the most optimal filter parameter with the CNN or E-Net due to its shared parameter of networks. From the mark of colored circle in Fig.14, there are some subtle differences, such as the texture on the ceiling, the specular lighting on the bottle. The same problem occurred with MLPs structure, such as in SAN MIGUEL scene shown in Fig.7, the texture on the marble columns of ours (green rectangle) is not as realistic as the ground truth image. Similarly, the noise on the back wall of the TEAPOT ROOM scene shown in Fig.7 and the blurred texture on the ceiling of the SIBENIK scene shown in Fig.6 are the problem of one kind. So that, the MLPs, CNN and E-Net structure are not the most optimal denoiser model.

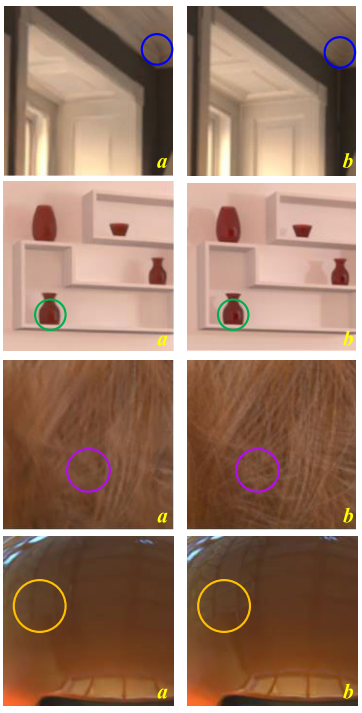

FIGURE 14. Our failure cases due to the limitations. Among them, the left details(marked by a) are rendered by our method and the right ones(marked by b) are reference images.

In the future, our work can be extended in several ways. First, the current network architecture is far from optimal. Network designs can be fine-tuned to simplify the model, and various strategies including activation function and model pruning can be explored to accelerate the convergence and improve the performance. Second, in addition to additive and multiplicative operations, more complex relationships can be exploited for better performance between the features and

the filter parameters. Finally, how to achieve comparable quality for path tracing with "light-weight" learning is worth exploring, as generating noise-free ground truth on a large scale is rather expensive [35].

V. CONCLUSION

We have presented a novel reconstruction approach to reduce noise in path tracing. In order to preserve more details, we use a metric based on SURE as the noise level estimator to guide iterative adaptive sampling stage. And for the purpose of modeling complex relationship between the ideal filter parameters and a set of features extracted from the rendering stage, we use a modified network as nonlinear regression model. We train our network on our own database including the Disney Animation Database and a set of scenes with a variety of distributed effects. With the predicted filter parameters, we reconstruct final images of different effects, including global illumination, motion blur, glossy reflections *et al.* Our reconstruction results show that our approach demonstrates visible improvement and the stronger applicability at low sampling rate over the state-of-the-art reconstruction methods.

REFERENCES

[1] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F. X. Silion, "Fourier depth of field," *ACM Trans. Graph.*, vol. 28, no. 2, pp. 18–30, May 2009.
[2] K. Egan, Y. Tseng, N. Holzschuch, F. Durand, and R. Ramamoorthi, "Frequency analysis and sheared reconstruction for rendering motion blur," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, New Orleans, FL, USA, 2009, p. 93.
[3] 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 *Proc. ACM SIGGRAPH Courses*, Los Angeles, CA, USA, 2015, p. 24.
[4] M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S.-E. Yoon, "Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering," *Comput. Graph. Forum*, vol. 34, no. 2, pp. 667–681, 2015.
[5] S. Bako, T. Vogels, B. Mcwilliams, M. Meyer, J. Novák, A. Harvill, P. Sen, T. Derose, and F. Rousselle, "Kernel-predicting convolutional networks for denoising Monte Carlo renderings," *ACM Trans. Graph.*, vol. 36, no. 4, p. 97, Jul. 2017.
[6] B. Bitterli, F. Rousselle, B. Moon, J. A. Iglesias-Guitián, D. Adler, K. Mitchell, W. Jarosz, and J. Novák, "Nonlinearly weighted first-order regression for denoising Monte Carlo renderings," in *Eurographics, Parque das Nações*. Brazil, 2016, pp. 107–117.
[7] F. Rousselle, C. Knaus, and M. Zwicker, "Adaptive rendering with non-local means filtering," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, Los Angeles, CA, USA, 2012, p. 195.
[8] F. Rousselle, M. Manzi, and M. Zwicker, "Robust denoising using feature and color information," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 121–130, Oct. 2013.
[9] P. Sen and S. Darabi, "On filtering the noise from the random parameters in Monte Carlo rendering," *ACM Trans. Graph.*, vol. 31, no. 3, p. 18, May 2012.
[10] T. Li, Y. Wu, and Y. Chuang, "SURE-based optimization for adaptive sampling and reconstruction," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, Los Angeles, CA, USA, 2012, p. 194.
[11] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, Jun. 2005, pp. 60–65.
[12] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.
[13] A. Buades, B. Coll, and J. Morel, "Nonlocal image and movie denoising," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2008, pp. 123–139.
[14] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *Proc. Int. Conf. Comput. Vis.*, Kyoto, Japan, 2009, pp. 2272–2279.
[15] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.
[16] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1620–1630, Apr. 2013.
[17] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D, Nonlinear Phenomena*, vol. 60, nos. 1–4, pp. 259–268, Nov. 1992.
[18] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Multiscale Model. Simul.*, vol. 4, no. 2, pp. 460–489, Jan. 2005.
[19] Y. Weiss and W. T. Freeman, "What makes a good model of natural images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Minneapolis, MN, USA, Jun. 2007, pp. 1–8.
[20] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black, "Efficient belief propagation with learned higher-order Markov random fields," in *Proc. Eur. Conf. Comput. Vis.*, Austria, 2006, pp. 269–282.
[21] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. London, U.K.: Springer-Verlag, 2009, pp. 448–456.
[22] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.*, vol. 82, no. 2, pp. 205–229, Jan. 2009.
[23] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vis.*, vol. 121, no. 2, pp. 183–208, Jan. 2017.
[24] L. Belcour, C. Soler, K. Subr, N. Holzschuch, and F. Durand, "5D covariance tracing for efficient defocus and motion blur," *ACM Trans. Graph.*, vol. 32, no. 3, p. 31, Jun. 2013.
[25] F. Durand, N. Holzschuch, C. Soler, E. Chan, and F. X. Silion, "A frequency analysis of light transport," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, USA, 2005, pp. 1115–1126.
[26] K. Egan, F. Hecht, F. Durand, and R. Ramamoorthi, "Frequency analysis and sheared filtering for shadow light fields of complex occluders," *ACM Trans. Graph.*, vol. 30, no. 2, pp. 1–13, Apr. 2011.
[27] N. K. Kalantari and P. Sen, "Removing the noise in Monte Carlo rendering with general image denoising algorithms," *Comput. Graph. Forum*, vol. 32, no. 2, pp. 93–102, May 2013.
[28] R. S. Overbeck, C. Donner, and R. Ramamoorthi, "Adaptive wavelet rendering," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, USA, 2009, p. 131.
[29] M. D. McCool, "Anisotropic diffusion for Monte Carlo noise reduction," *ACM Trans. Graph.*, vol. 18, no. 2, pp. 171–194, Apr. 1999.
[30] T. Hachisuka, W. Jarosz, R. P. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, H. W. Jensen, "Multidimensional adaptive sampling and reconstruction for ray tracing," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, USA, 2008, p. 33.
[31] M. Delbracio, P. Muse, A. Buades, J. Chauvier, N. Phelps, and J. Morel, "Boosting Monte Carlo rendering by ray histogram fusion," *ACM Trans. Graph.*, vol. 33, no. 1, p. 8, Jan. 2014.
[32] P. Bauszat, M. Eisemann, S. John, and M. Magnor, "Sample-based manifold filtering for interactive global illumination and depth of field," *Comput. Graph. Forum*, vol. 34, no. 1, pp. 265–276, Feb. 2015.
[33] N. K. Kalantari, S. Bako, and P. Sen, "A machine learning approach for filtering Monte Carlo noise," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, USA, 2015, p. 122.
[34] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[35] D. Valsesia, G. Fracastoro, and E. Magli, "Image denoising with graph-convolutional neural networks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taiwan, Sep. 2019, pp. 2399–2403.
[36] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning image restoration without clean data," in *Proc. Int. Conf. Mach. Learn.*, Sweden, 2018, pp. 2971–2980.
[37] 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 Trans. Graph.*, vol. 36, no. 4, p. 98, Jul. 2017.

[38] T. Vogel, F. Rousselle, B. Mcwilliams, G. Röthlin, A. Harvill, D. Adler, M. Meyer, and J. Novàk, "Denoising with kernel prediction and asymmetric loss functions," *ACM Trans. Graph.*, vol. 37, no. 4, p. 124, Aug. 2018.
[39] M. Kettunen, E. Harkonen, and J. Lehtinen, "Deep convolutional reconstruction for gradient-domain rendering," *ACM Trans. Graph.*, vol. 38, no. 4, p. 126, Jul. 2019.
[40] M. Gharbi, T. Li, M. Aittala, J. Lehtinen, and F. Durand, "Sample-based Monte Carlo denoising using a kernel-splatting network," *ACM Trans. Graph.*, vol. 38, no. 4, p. 125, Jul. 2019.
[41] H. Yuan, F. Wu, and C. Zheng, "Adaptive sampling guided by SURE and sample color histogram reconstruction," *J. Comput.-Aided Des. Comput. Graph.*, vol. 28, no. 4, pp. 533–539, Apr. 2016.
[42] F. Rousselle, C. Knaus, and M. Zwicker, "Adaptive sampling and reconstruction using greedy error minimization," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, Canada, 2011, p. 159.
[43] W. Disney. (Aug. 2018). *Moana Island Scene Data Set, Package Version v1.1-9*. [Online]. Available: https://www.technology.disneyanimation.com/islandscene
[44] J. Franklin, "The elements of statistical learning: Data mining, inference and prediction," *Math. Intelligencer*, vol. 27, no. 2, pp. 83–85, Mar. 2005.
[45] M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. San Mateo, CA, USA: Morgan Kaufmann, 2010, pp. 877–885.
[46] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[47] Q. Xu, D. Chen, H. Chen, and G. Wang, "Adaptive sampling based on fuzzy uncertainty," *J. Comput.-Aided Des. Comput. Graph.*, vol. 20, no. 6, pp. 689–699, Jun. 2008.
[48] 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," *J. Comput. Sci. Technol.*, vol. 34, no. 5, pp. 1123–1135, Sep. 2019.

QIWEI XING received the B.S. and M.S. degrees in computer science and information technology from Northeast Normal University. He is currently pursuing the Ph.D. degree in computer science and technology with the School of Computer Science and Technology. His research interests include computer graphics, 3D visualization, and virtual reality.

CHUNYI CHEN received the bachelor's and master's degrees in computer science and technology and the Ph.D. degree in physical electronics from the Changchun University of Science and Technology (CUST). He was a Postdoctoral Researcher with Jilin University, from January 2010 to July 2012, and a Visiting Scholar with Pennsylvania State University, from March 2013 to March 2014. He has been a Professor with CUST, since 2016. His research interests include virtual reality, realistic graphics rendering, information security theory, optical wireless communications, and so forth.