Hindawi
Advances in Multimedia
Volume 2022, Article ID 9169772, 9 pages
https://doi.org/10.1155/2022/9169772

Research Article

Monte Carlo Noise Reduction Algorithm Based on Deep Neural Network in Efficient Indoor Scene Rendering System

Xiwen Chen and Jianfei Shen

College of Visual Arts, Hunan Mass Media Vocational and Technical College, Changsha 410100, China

Correspondence should be addressed to Jianfei Shen; 19118804923c@sina.com

Received 26 March 2022; Revised 21 April 2022; Accepted 5 May 2022; Published 14 June 2022

Academic Editor: Qiangyi Li

Because of its flexibility and universality, Monte Carlo integral has become the preferred algorithm of most realistic image synthesis. However, the quality of rendered images is often affected by the estimated variance, which is mainly reflected in image noise visually. To reduce the variance, Monte Carlo rendering systems often require extensive sampling, which also causes a lot of time spent trying to render noiseless images. For this problem, we propose a Monte Carlo noise reduction algorithm based on deep neural networks and apply it to the efficient rendering of an indoor scene. The algorithm can reduce noise in real time. In order to solve the gradient disappearance problem of deep convolutional neural network, the residual structure of the network is added to the original convolutional network. The proposed algorithm can achieve better noise reduction quality in the real-time guarantee.

1. Introduction

In recent years, physics-based realistic picture synthesis technology has been widely used in games, animation, film and television special effects, and other industries. Rendering algorithm has become the mainstream picture synthesis technology because it can meet strict quality requirements, is easy to implement, and is robust. In a computer drawing image, you first need to use a computer model to represent a specific data structure and then convert the model into an image displayed on the screen. This conversion process is called rendering. Different rendering methods are used to generate images of different styles in different application scenarios, such as different styles of movies, or real-time and non-real-time rendering; in these different scenarios, the rendering techniques are different. Therefore, rendering is an important research field in computer graphics, among which realistic rendering is the research focus of the majority of researchers. Reality rendering technology requires generating images according to real-life scenes and light and shadow. Although in practical application, many rendering techniques use some approximate methods to simulate the real light and shadow performance, the ultimate goal of the realistic rendering technology is to generate photo-level images.

The generation of realistic images mainly includes two key steps, and the first step is an accurate representation of the scene model. Since the computer cannot use continuous values to describe the scene, it first needs to describe the scene accurately with discrete values. The second step is to use realistic rendering techniques. At present, the mainstream uses light projection algorithms, including raytracing technology, distributed raytracing, and Monte Carlo path-tracking algorithm. Creating a realistic image from the scene model requires computing a complex high-dimensional integration at each pixel of the image. The Monte Carlo path-tracking algorithm is an important global light rendering technology that can render very realistic images. Due to its simple algorithmic implementation and its ability to ensure consistency in rendering, this algorithm is widely used in movie special effects. Due to the great application prospect and the importance of this algorithm. The Monte Carlo rendering (MC) system estimates integration values by tracking light (sampling) in a multidimensional space. Although images can be rendered by a small number of samplings, the estimation results relative to the real value

have a large error; these errors in the rendering results are a large number of noise because the variance of Monte Carlo estimation method decreases linearly according to the number of sampling increase; so in order to get a more accurate estimation, results often need a lot of sampling. Computing heavy sampling requires a long rendering time, which limits the scope of Monte Carlo methods in the modern rendering industry. One solution is to put a small amount of light to the scene to quickly render a noise image and then use a filter as a posterior method to filter the noise image to obtain noise-free results. This method, also known as the image spatial noise reduction method, is the subject of extensive research in recent years [1]. Although a large number of image space Monte Carlo rendering noise reduction methods have been proposed, the best performance is the regression-based methods [2]. Higher-order regression improves certain rendering quality, but these quality improvements come at the cost of increasing complexity and with smaller returns because higher-order regression is prone to overfitting and introducing noise.

Based on this background, to avoid overfitting to noise, a deep neural-based Monte Carlo noise reduction method is proposed and applied to the rendering of efficient indoor scenes. The regression model was obtained with a set of low-sampled noisy images and corresponding high-sampled images as training data. This approach uses a deep neural network as a learning model and uses only a small number of scene images to generate training data [3]. The full text is divided into four chapters. Chapter 1 introduces the research background and research necessity and chapter arrangement; chapter 2 introduces the feasibility of deep Carlo algorithm and the Monte Carlo algorithm, proposes the Monte Carlo algorithm based on the deep neural network to efficient indoor scene rendering, analyzes the experimental results and errors, and draws the conclusion. Experiments show that the Monte Carlo noise reduction method based on deep neural network can complete the scene rendering and noise reduction work efficiently in the indoor scene rendering. The algorithm can reduce noise in real time. In order to solve the gradient disappearance problem of deep convolutional neural network, the residual structure of the network is added to the original convolutional network. The proposed algorithm can achieve better noise reduction quality in the real-time guarantee.

2. State of the Art

Fiber tracking is to calculate the light in 3D environment by tracking the reverse fiber path of the camera to the light source. The light path can reflect the light distribution and transmission rules in 3D space. Light in the real world is continuously distributed and needs to be simulated in a discrete state when drawn in 3D. Light can be equivalent to vector lines and propagate in three dimensions according to the law of the illumination model. The flowchart of the raytracing is shown in Figure 1.

Based on Maxwell's electromagnetism equation, Kajiya proposed the concept of rendering equation and proposed the relevant calculation method: optical path-tracking

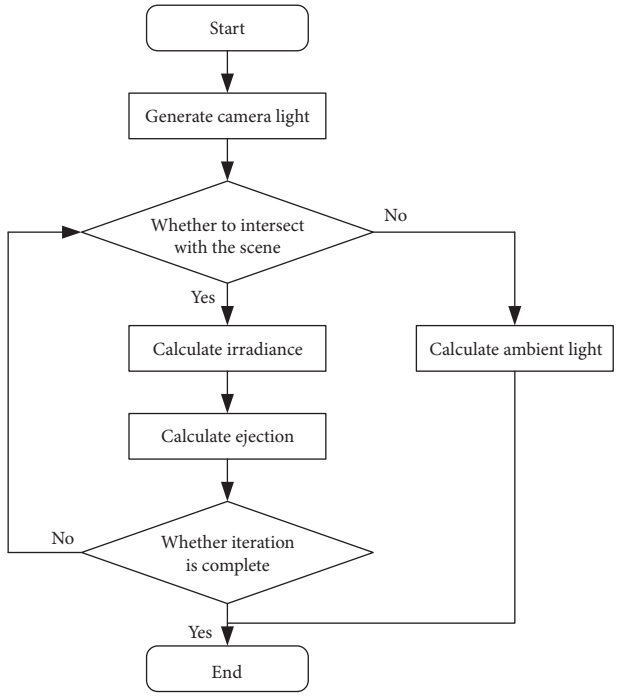


Figure 1: Raytracing flowchart.

method [4]. Based on the raytracing method, the rendering equation is gradually developed. In the same year, Robert Cook also proposed the distributed raytracing algorithm, which uses Monte Carlo's sampling method to calculate soft shadow effects, motion blur, and DOF for special effects [5]. Raytracing based on Monte Carlo methods has become the gold standard for rendering, and it generates realistic images in a physically correct way. But Monte Carlo raytracing has historically been slow, usually taking hours to days to generate a noiseless image. Thus, real-time Monte Carlo raytracing over 30 frames per second was considered impractical until recent breakthroughs in graphics hardware made it a reality. Today, real-time Monte Carlo raytracing has generated great interest not only in the video game industry, but also universally explored in interactive previews, dynamic lighting, and material editing for movies, animation, computer-aided design, and visualization. At the heart of real-time Monte Carlo raytracing is noise reduction or reconstruction technology. However, with such a low sampling rate and high-performance requirements, accurate image filtering methods are likely to fail [6].

Recently, deep learning methods have shown great advantages in noise reduction for unbiased Monte Carlo path tracing. The work of Kalantari et al. is the first work using neural networks for Monte Carlo rendering noise reduction [7]. Nikolentzos et al. introduce a novel kernel prediction network (KPCN) that uses convolutional neural networks to estimate locally weighted kernels to calculate each noise-reducing pixel from its neighbors. They decompose the diffuse and mirror reflection components, pretrain them separately on the two networks, and then fine-tune the full framework. KPCN shows great improvement over previous Monte Carlo noise reduction methods [8]. Further

improvements of this work by Vogels et al. combining KPCN with multiple task-specific modules and optimized assembly using asymmetric losses resulted in a more robust solution. They solve the residual low-frequency noise based on the multiscale architecture, and their single-frame noise reducers cannot directly solve this problem [9]. Li et al. use residual learning to achieve high-quality noise reduction for path-tracking rendering [10]. Their direct prediction model (RDP) directly predicts the corresponding noise-free color, rather than the kernel weight of each pixel. Xu et al. also reduced the MC rendering with a generative adversarial network (GAN) by directly predicting the final color [11]. This is the first Monte Carlo noise reduction method based on adversarial learning. And they also adapted a well-designed conditional-assisted feature modulation method to better utilize the rendered feature buffer. Similarly, Yang et al. have also introduced a dual-encoder network (DEMC) and a feature fusion subnetwork to handle feature buffers, respectively. They first fused the feature buffers and then learned the relationship between the noisy scene data and the ideal filter parameters with the multilayer perceptron neural network and used the learned model for the new scenarios [12]. Instead of learning from the first convolutional neural network, which can learn the noise-reducing Monte Carlo, rendering directly from samples is proposed by Glick [13].

3. Methodology

3.1. Neural Network-Based Monte Carlo Method. The Monte Carlo method is also known as the statistical simulation method. Using randomized trials to simulate the solution of the problem, it is statistically considered that a solution approximates the real solution [14]. The Monte Carlo method is defined as follows: $\int_a^b f(x)\mathrm{d}x$; the integral equation is calculated by the sampling method as follows:

$$F_N = \frac{b-a}{N}\sum_{i=1}^{N} f(X_i). \tag{1}$$

In the above equation, N points are sampled in the integral range, and the solution of the integral equation is estimated with the sampling value of this N point. When the sampling data N is long enough, the existing

$$E[F_N] = \int_a^b f(x)\mathrm{d}x. \tag{2}$$

The Monte Carlo method is a good choice for not using explicit equations $f(x)$ (e.g., solving rendering equations). At present, the Monte Carlo method is widely used in finance, economics, computational physics, and other fields. Meanwhile, the Monte Carlo method of the artificial neural network of not so popular [15]. By introducing the Monte Carlo method for the stochastic simulation of the input samples, we give the closest distribution model to realize the learning and training of the neural network. This method can avoid the BP neural network into the dilemma of local minimum, have good applications in the fields of power prediction, sunspot prediction, bioinformation recognition and other fields, and show better performance than BP neural network [16, 17]. With the improved computer performance, it can complete the training in a relatively short time. The introduction of Monte Carlo method is also an important direction in the field of machine learning. A schematic representation of the neural network-based Monte Carlo method is shown in Figure 2.

The neural network model in the Monte Carlo method, based on the deep neural network, is a typical fully connected artificial neural network, as shown in Figure 1(a), including the input layer, hidden layer, and output layer. Suppose the number of input layer nodes is N, the number of hidden layers is M, the output layer nodes are L, and the order input vector is x, $x = x1, x2, \ldots, x_N$, The output vectors are y, $y = y1, y2, \ldots, y_L$. The neural network is calculated in the following form.

$$\begin{aligned} h_i &= \omega_{ji} \cdot x_j + b_i, \\ o_j &= f_i(\beta_i \cdot h_i), \\ y_l &= v_{ik} \cdot o_i, \end{aligned} \tag{3}$$

where $\omega_{ji}(j = 1, 2, \ldots N, i = 1, 2, \ldots M)$ represents the weights from the input layer to the hidden layer and b_i represents the bias of the ith neural node. f_i represents the activation function of the first neural node. f_i represents the coefficients of the activation function, and this parameter is introduced to regulate the activation ability of neurons in the hidden layer. f_i represents the weights from the hidden layer to the output layer, o_i and b_i are two intermediate quantities, h_i is the output value of the input layer to the hidden layer input signal after calculation, and o_i is the output value of the neural node after the activation function activation.

For this artificial neural network, after a given training task, the network structure includes the input layer, the hidden layer, and the output layer. Furthermore, the selection of the activation function is usually determined during the process of network initialization. Although the activation function of each neural node can be adjusted during training, adjusting the activation function during training will destabilize the network, so the selection of activation function is usually determined during the initialization phase.

Second, the network parameters were initialized in a stochastic manner. For this network, there are some parameters to be adjusted, such as $\{\omega_{ji}, b_i, \beta_i, v_{ik}\}$, where ω_{ji} and v_{ik} are the two parameter matrices that affect and connect to each other, which is like two mirrors regulating the output of the network. In order to save the training time and avoid the network from producing large fluctuations, one of the parameter matrices is usually fixed, and another parameter matrix is adjusted to realize the training of the network. In the process of initialization, pairs ω_{ji} are usually randomly assigned in the interval [−1, 1], while pairs v_{ik} are fixed. After a random assignment v_{ik} of −1 or 1 to each term, the parameter matrix v_{ik} was not changed. Next, the parameters b_i and parameters β_i were randomly initialized within their appropriate intervals.

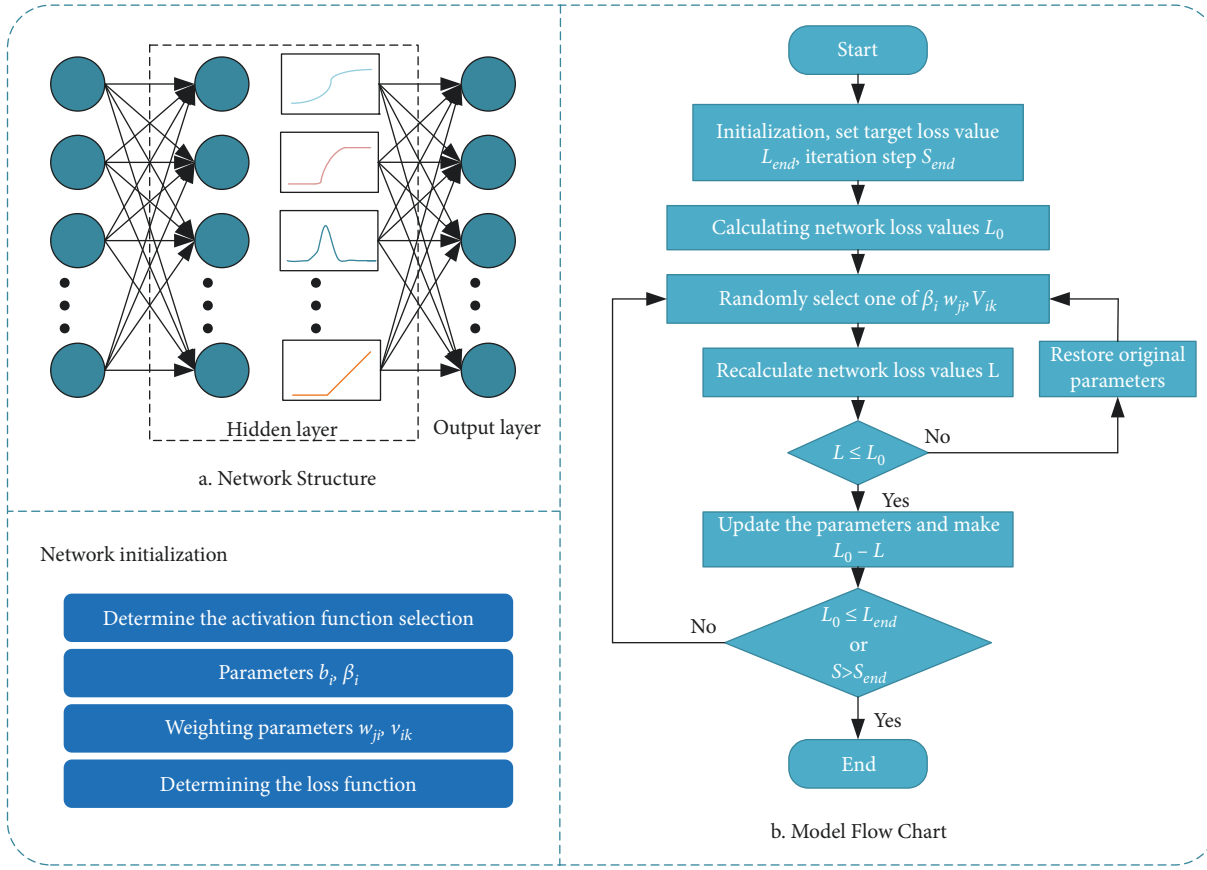


Figure 2: A Monte Carlo method based on neural networks.

Finally, define the appropriate loss function $L(y, y')$, where $y = y1, \ldots, y_k$. For the true output value, $y' = y1, \ldots, y_{k'}$ is the output value of the artificial neural network. The common loss function (Sum of Squares for Error, SSE) is defined as

$$\text{SSE} = \sum_{i=1}^{k} (y - y')^2. \tag{4}$$

For this artificial neural network model, the parameter adjustment was performed by optimizing the loss function values. Monte Carlo was used to randomly select the adjustment parameter and update the loss function by adjusting a small amount to the parameter if and only if the value of the loss function is reduced or remains constant. The reduction of the loss function value indicates that the network is moving in a better direction. Taking the value of the loss function unchanged and the parameter change is to keep the network in an update state and ensure that the network finds the optimal solution in the whole parameter space. The specific training process of the whole network is as follows:

(1) Network initialization: It including two aspects: determine the network structure, mainly the number of network nodes, the loss function, and determine the activation function; random parameter initialization: select the appropriate range; set the initial value for the parameters $\{\omega_{ji}, b_i, \beta_i, v_{ik}\}$. After initialization, the loss value of the network is calculated.

(2) Change to the parameters: Monte Carlo was used to randomly select one of the three parameters $\{\omega_{ji}, b_i, \beta_i\}$ to ensure a small amount in the set parameter range ε.

(3) Parameter update: After step (2), recalculate the network loss value L. If the value of L decreases or remains unchanged, update the loss value L and the parameters selected in step (2); otherwise, the parameters remain unchanged.

(4) Network training: Repeat steps (2) and (3) in an iterative manner until the termination condition for the network training is reached. That is, a certain number of training steps or the loss value reaches the set threshold. The flowchart of the three-layer artificial neural network model of Monte Carlo method is shown in Figure 3(b). It should be pointed out that after the whole network is initialized and the initial loss value is calculated, the subsequent network loss value is calculated in an incremental way as the adjustment of each iteration process is only a single neuron. Compared to the recomputation of the model, this way can reduce the computational consumption and very significantly improve the speed of network training.

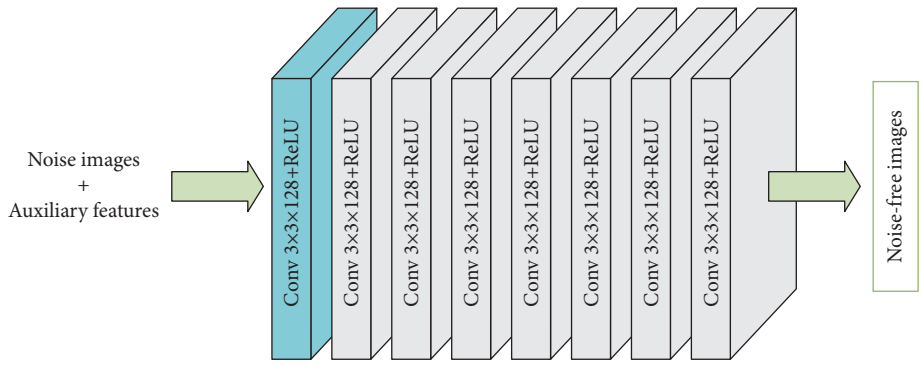


Figure 3: Deep neural network model design.

3.2. Depth Residual Noise Reduction Neural Network. In the rendering of indoor scenes, on the shiny surface of objects such as furniture and appliances, the variance seems to be of high frequency and relatively small noise. This is because a large variance may occur on the gloss surface if the mirror reflection range is small and not enough photons arrive and not well represent the incident light distribution. Deviation and variance bring multiscale noise into the same rendering result of random progressive photon maps. Therefore, an image space noise reduction method is needed to calculate the filtering color of pixels by modeling and estimating the filtering function g, where $\widehat{c_P}$ is the weighted sum of pixel colors q in the neighborhood $N(i)$ centered on pixel i:

$$\widehat{c_i} = \sum_{j \in N(i)} g\left(x_i, \theta_{i,j}\right) c_j. \tag{5}$$

For multiscale noise reduction in random progressive photon mapping rendering, a large-size filter is required to better remove large noise in low-frequency regions. But large filters are always unable to remove noise from high-frequency areas while ensuring not being excessively smooth, such as corners or cabinet gaps. The smaller filters do just the opposite. Multiple residual noise reduction networks model the function g of random progressive photon maps, an approach that can simultaneously benefit from size-size filters.

Using weighted reconstruction methods (e.g., function G direct noise reduction and random progressive photon mapping) always leads to conflicting noise reduction constraints. Assuming a set of photons contributes to a surface, it is difficult to decide whether to average their contributions, blur (whether they are biased estimates of diffuse surfaces) or remain sharp (whether they have some blurred focus). This paper solves this problem by splitting the rendering result, the final radiance of each pixel, into two components: the caustic component and the global component. These two parts represent the radiance estimation of two photons: pyrophoton and global photon. A photon is called a pyroscopic photon if passing through the delta bidirectional scattering distribution function to the diffuse surface or reaches the maximum depth and is always on the concentrated gloss surface, without any diffuse reflection; otherwise, it is called a global photon.

It is well known that increasing network depth improves model power. However, increasing the network depth is a challenge for the learning process due to the vanishing gradient problem [18]. Deep residual networks solve this problem by using hop connections [19]. These jump connections and residual blocks in the network allow the reuse of upstream features to establish multiscale similarity mapping capabilities and maintain some consistency with the upstream blocks [12]. Specifically, for noise reduction tasks, it is always necessary to design relatively deep networks to improve the model power, while maintaining some consistency with the input. This makes deep residual networks naturally suitable for noise reduction tasks. This multiple residual noise reduction network, proposed herein, consists of a novel multiple residual extractor and a kernel predictor with a weighted reconstruction. Using multiple residual blocks, we can significantly improve the training performance and better handle the multiscale noise while keeping the depth fixed.

The multiple-residue extractor extracts spatial features, which have nine multiple-residue blocks, as well as convolution, activation, and batch normalization layers. In this paper, two residual functions with different receptive fields (i.e., filter core size) are used in each residual block to better deal with multiscale noise in low- and high-frequency regions:

$$H(x) = \text{Concatenate}\left(F_1(x), F_2(x)\right) + x. \tag{6}$$

Here, $H(x)$ is the output of the multiple residual blocks, $F_1(x)$ and $F_2(x)$ is a different residual function, and x is the input of the block (jump connection). The purpose of hop connections is to build a unified map that allows for feature reuse.

3.3. A Monte Carlo Noise Reduction Model Based on a Deep Neural Network. Monte Carlo noise reduction goal is to get $\overline{C_P}$ as close as possible to the real pixel color value, that is, filtered pixel color value $\widehat{C_P}$; in order to obtain the estimate, usually for each pixel vector X_P block neighborhood operations to produce filter output pixel color p, given the noise reduction function $g(X_p, \theta)$, parameters θ represent noise reduction parameters; each pixel ideal noise reduction parameters can be expressed as

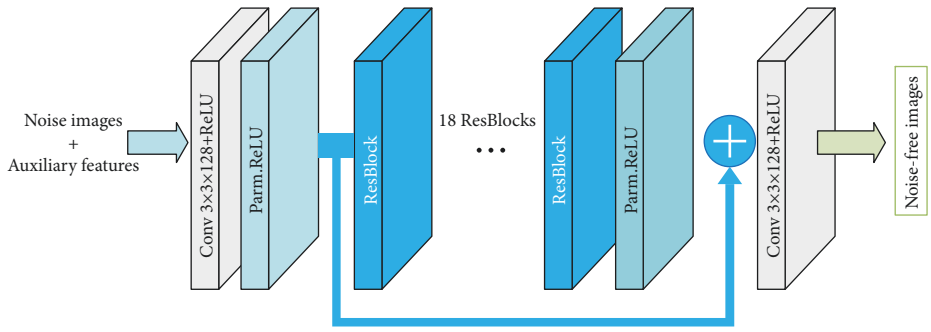


Figure 4: Design of a deep neural network model for an access residual block.

$$\hat{\theta} = \min_{\theta} l\left(\overline{C_p}, g\left(X_p, \theta\right)\right), \tag{7}$$

where g is a noise reduction model that needs to learn. $l(\overline{C}, \hat{C})$ For the loss function between the true color value and the noise-reduced color value, the filtered pixel color value is

$$\widehat{C_p} = g\left(X_p, \hat{\theta}_p\right). \tag{8}$$

The input vectors consist of the following:

$$X_p = \left\{C_p(r,g,b), z_i, n_i(x,y), a_i(r,g,b)\right\}. \tag{9}$$

Specifically, $C_p(r,g,b), z_i, n_i(x,y), a_i(r,g,b)$ represent pixel color value, depth value, perspective space normal, and albedo.

The network structure used in this article is shown in Figure 3. Deep neural networks use multiple convolutional kernels in each layer of the network, with trainable weights shared by image space. The convolutional structure is more suitable for noise reduction tasks compared with traditional image noise reduction methods. Furthermore, by connecting multiple layers together through activation functions, deep neural networks are capable of learning highly nonlinear functions, and such functions are important for obtaining high-quality output.

Based on back propagation method of gradient optimization neural network, because the back propagation for hidden layer gradient using the chain rule, gradient value will be a series of multiplications, leading to shallow hidden layer gradient that will show violent attenuation; this is the origin of the gradient problem, residual network through jumping connection effectively flows to the shallow network, and the shallow part of the network weight parameters can get very good training. In this paper, the residual block is added to the convolutional neural network proposed above, and the network depth effectively improves the final noise reduction effect. The residual network uses a total of 18 layers of residual blocks, and the structure is shown in Figure 4 [19, 20].

4. Result Analysis and Discussion

In order for the deep neural network to learn the complex relationship between input and output data and also to avoid overfitting occurrence, this experiment is required to obtain large amounts of data. At present, there is no public data set for rendering noise reduction experiment based on deep learning. Before the experiment, this paper requires a lot of time to prepare a data set with large data volume (renderer chooses Tungsten). The objects in the data set include a variety of objects, and there are various distributed effects (motion blur, focal dispersion, etc.). For the data collection here, we used a camera with different focal lengths, exposures, and resolutions to record indoor video, respectively, and then extract the video data from frames, and obtained about 20,000 pieces of data of different sizes. Figure 5 shows some of the training data in this data set.

In this paper, deep learning framework is TensorFlow, operating system environment is Windows, GPU acceleration is adopted, patch size is 128128, step length is 80, and finally 57k patch blocks were used for training. The network optimizer uses Adam, with a momentum value set to 0.9, and $L1$-loss as a loss function. Training uses a minimum batch quantity of 10, training iterations of 106, and the learning rate table is as follows:

(1) Learning rate of 0.01 was used for the first 1,000 iterations.

(2) Use the learning rate of 0.001 for the second 1,000 iterations.

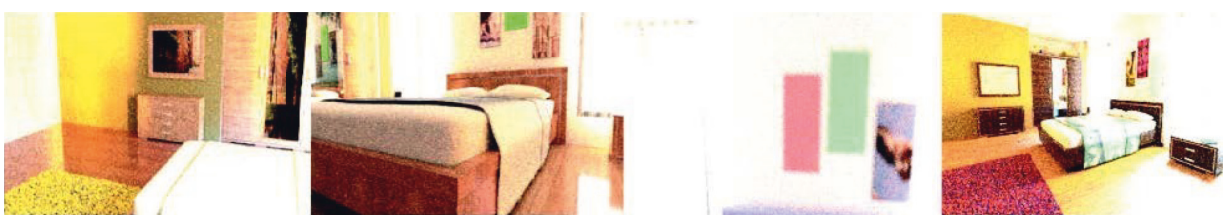
(3) The learning rate of 104 is used for the remaining iterations.

This network has 13847296 trainable parameters and takes 36 hours. The method presented here, together with NFOR, LBF, and KPCN [21], have been compared. For comparison, for NFOR using the author's open-source code, the author has added the source code to the Tungsten renderer; for KPCN, the author has already trained the model directly; and for the LBF method, the author has integrated the method into PBRT2. The main focus of this article is fast rendering high-quality images, such as game rendering, virtual reality, and prototype design; these applications' rendering speed is very important, but each pixel sampling (SPP) will reduce the rendering speed, so the noise images and feature images are 4SPP, real images (Ground Truth) with 32K SPP or higher sampling rendering, to ensure that no noise in human eye perception. To measure the performance of these methods, RelMSE (RelativeMSE) and SSIM (Structural Similarity Index) are used as the measure of noise reduction results. The value range of 0 to 1,

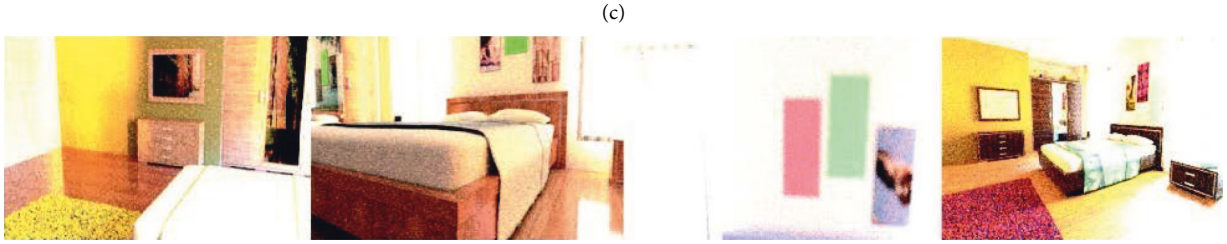
Figure 5: A partial example of the training data set.

(a)

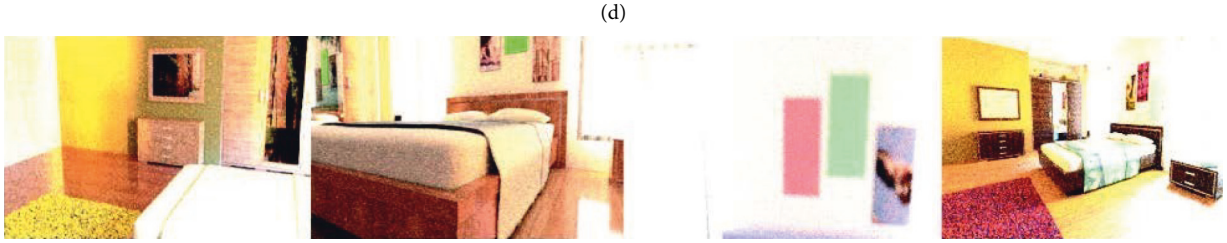
(b)



(c)



(d)

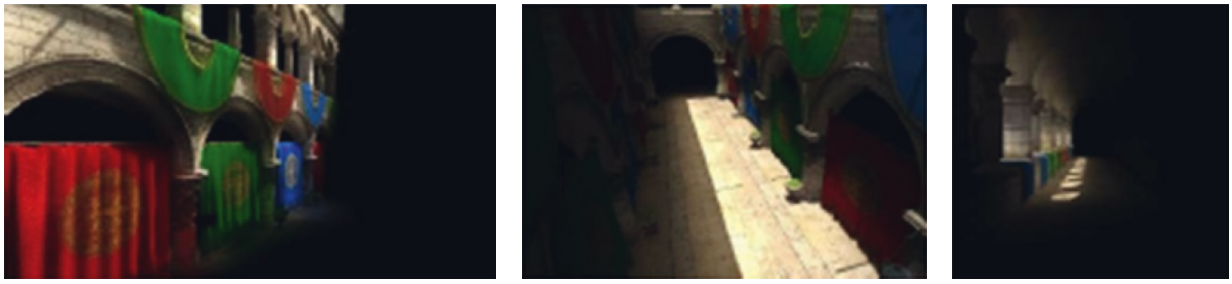


(e)

Figure 6: Noise reduction effect of Monte Carlo algorithm based on deep neural network. (a) Input SSIM = 0.99, RelMSE = 126.433. (b) NFORSSIM = 0.9985, RelMSE = 9.271. (c) KPCN SSIM = 0.9987, RelMSE = 5.578. (d) Ours SSIM = 0.9993, RelMSE = 3.228, (e) Reference dynamic change 0.0–36.34.

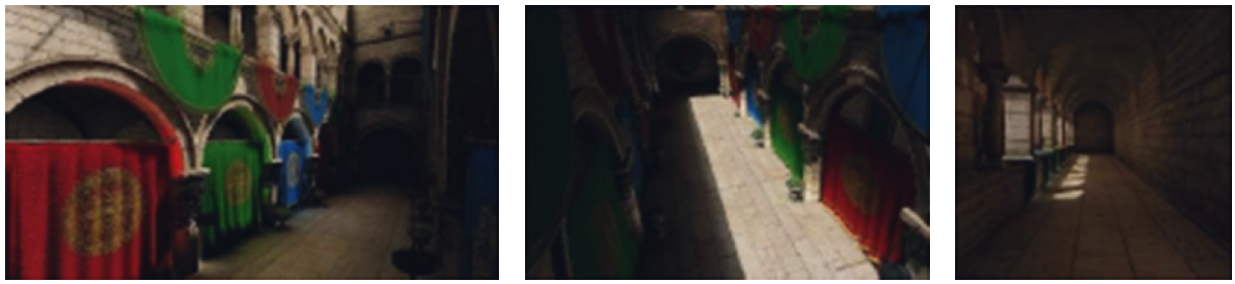
1 represents the perfect match with the real image. Figure 6 shows the noise reduction results, and the data shows the noise reduction time without the rendering time.

The rendering effect of noise reduction with the Monte Carlo algorithm using a deep neural network is shown in Figures 7 and 8. Figure 7 shows the rendering effect of the



(a) (b) (c)

Figure 7: Noise reduction effect of Monte Carlo algorithm based on deep neural network. (a) Perspectives 1. (b) Perspectives 2. (c) Perspectives 3.



(a) (b) (c)

Figure 8: Noise reduction effect of Monte Carlo algorithm based on deep neural network. (a) Perspectives 1. (b) Perspectives 2. (c) Perspectives 3.

ordinary Monte Carlo algorithm, and Figure 8 shows the noise reduction effect of the Monte Carlo algorithm based on the deep neural network. The resolution of the experimental scene is 1280×720, the single-frame time for noise reduction of the ordinary Monte Carlo algorithm is 29.6 ms, and the single-frame time of this algorithm is 26.2 ms. From Figure 5, 24-24 is the overall effect of the algorithm in this paper. Comparing Figures 6 and 7, it can be seen that the method used in this paper can obtain more details in the distant indirect light area, the color mixing of indirect light, and the global light effect such as soft shadow are also obvious than the ordinary Monte Carlo algorithm.

5. Conclusion

In recent years, physics-based realistic picture synthesis technology has been widely used in games, animation, film and television special effects, and other industries. Among them, Monte Carlo rendering algorithm has become the mainstream image synthesis technology because it can meet strict quality requirements, is easy to implement, and is robust. However, to produce high-quality, noiseless-free results, Monte Carlo rendering algorithms often need to set extremely high sampling rates, which means high computational cost and huge computational time.

For the second problem, this paper proposes a Monte Carlo noise reduction method based on deep neural network, introduces the deep neural network, and deepens the layer number of the deep noise reduction network model. Experiments show that the Monte Carlo noise reduction method based on deep neural network can complete the scene rendering and noise reduction work efficiently in the indoor scene rendering. In subsequent work, the algorithm needs to be applied to an outdoor complex environment to further verify the feasibility and applicability of the algorithm.

Data Availability

The labeled data set used to support the findings of this study is available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Hunan Mass Media Vocational and Technical College.

References

[1] P. Bauszat, M. Eisemann, and M. A. Magnor, "Guided Image Filtering for Interactive High-Quality Global Illumination," in *Proceedings of the Computer Graphics Forum, F*, pp. 1361–1368, Eurographics Association, Goslar, Germany, June 2011.

[2] B. Bitterli, F. Rousselle, B. Moon et al., "Nonlinearly Weighted First-Order Regression for Denoising Monte Carlo Renderings," *Computer Graphics Forum*, vol. 35, no. 4, pp. 107–117, 2016.

[3] T. Turki and Z. Wei, "A Noise-Filtering Approach for Cancer Drug Sensitivity Prediction," 2016, https://arxiv.org/abs/1612.00525.

[4] J. T. Kajiya, "The rendering equation," *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 143–150, 1986.

[5] I. Wald, C. Benthin, and P. Slusallek, "Distributed interactive ray tracing of dynamic scenes," in *Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics 2003*, pp. 20-21, Seattle, WA, USA, October 2003.

[6] B. Moon, J. A. Iglesias-Guitian, S. E. Yoon, and K. Mitchell, "Adaptive rendering with linear predictions," *ACM Transactions on Graphics*, vol. 35, no. 4CD, pp. 121–1, 2016.

[7] S. Kumar, X. Gao, I. Welch, and M. Mansoori, "A Machine Learning Based Web Spam Filtering Approach," in *Proceedings of the IEEE International Conference on Advanced Information Networking & Applications*, IEEE, Crans-Montana, Switzerland, March 2016.

[8] G. Nikolentzos, P. Meladianos, J. P. Tixier, S. Antoine, S. Konstantinos, and M Vazirgiannis, "Kernel Graph Convolutional Neural Networks," 2017, https://arxiv.org/abs/1710.10689.

[9] H. Fan, R. Wang, Y. Huo, and H. Bao, "Real-time Monte Carlo Denoising with Weight Sharing Kernel Prediction Network," 2022, https://arxiv.org/abs/2202.05977.

[10] Y. Li, W. Zheng, and Z. Zheng, "Deep Robust Reinforcement Learning for Practical Algorithmic Trading," *IEEE Access*, vol. 7, p. 1, 2019.

[11] B. Xu, J. Zhang, R. Wang et al., "Adversarial Monte Carlo denoising with conditioned auxiliary feature modulation," *ACM Transactions on Graphics*, vol. 38, no. 6, pp. 1–12, 2019.

[12] X. Yang, D. Wang, W. Hu et al., "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.

[13] N. Glick, "Sample-based classification procedures related to empiric distributions," *IEEE Transactions on Information Theory*, vol. 22, no. 4, pp. 454–461, 1976.

[14] J. B. Detemple, R. Garcia, and M. Rindisbacher, "A Monte Carlo method for optimal portfolios," *The Journal of Finance*, vol. 58, no. 1, pp. 401–446, 2003.

[15] H. Zhao, "General Vector Machine," 2016, https://arxiv.org/abs/1602.03950.

[16] H. Chen, *Supervised Learning Machine and its Application Based on Monte Carlo Method*, Lanzhou University, Lanzhou, China, 2015.

[17] B. Yong, Z. Xu, J. Shen, H. Chen, Y. Tian, and Q. Zhou, "Neural Network Model with Monte Carlo Algorithm for Electricity Demand Forecasting in Queensland," in *Proceedings of the Australasian Computer Science Week Multiconference*, Association for Computing Machinery, New York, NY, USA, January 2017.

[18] A. M. Schaefer, S. Udluft, and H.-G. Zimmermann, "Learning long-term dependencies with recurrent neural networks," *Neurocomputing*, vol. 71, no. 13-15, pp. 2481–2488, 2008.

[19] K. He, X. Zhang, S. Ren, and J. Sun, *Identity Mappings in Deep Residual Networks*, Springer, Cham, Switzerland, 2016.

[20] S. Wu, S. Zhong, and Y. Liu, "Deep Residual Learning for Image Steganalysis," *Multimedia Tools and Applications*, vol. 77, 2017.

[21] T. Vogels, F. Rousselle, B. Mcwilliams, M. Meyer, and J. Novak, *Denoising Monte Carlo Renderings Using Progressive Neural Networks*, Google patents, California, CA, USA, 2021.