International Journal of Hybrid Information Technology
Vol.13, No.1 (2020), pp.1-6
http://dx.doi.org/10.21742/ijhit.2020.13.1.01

Implementation of Autoencoder Based Neural Network for Realtime Ray Tracing

HakHyun Lee[1], Chelwon Jo[2], and Kwang-Yeob Lee[3]

Dept. Computer Engineering, SeoKyeong Univ., 124, Seogyeong-ro, Seongbuk-gu, Seoul, Republic of Korea
[1]kyuns@kakao.com, [2]cheolgu94@skuniv.ac.kr, [3]kylee@skuniv.ac.kr

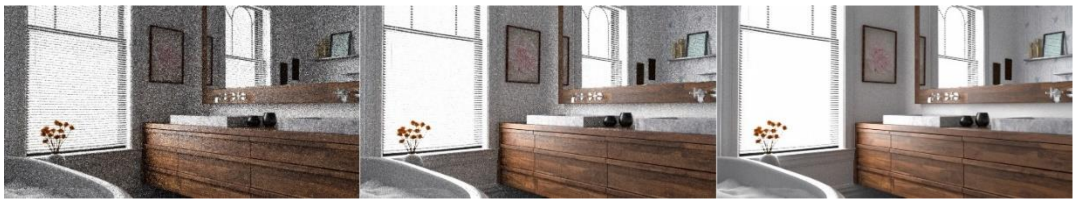
Abstract

This paper proposes a denoising neural network for real-time ray tracing. The ray-tracing method is applied in graphics to increase the reality and in particular, Monte Carlo Rendering is most effective. However, ray tracing that applies Monte Carlo Rendering has a steep rise in the number of calculations with the increase of the number of rays. Therefore, to solve this problem, various methods are being proposed to reduce the number of rays and to decrease the occurring noise. In this paper, an autoencoder-based neural network that can effectively remove noise while using a small number of rays was implemented. An autoencoder that uses a 1×1 convolution in creating the last feature map was proposed to significantly lower the amount of calculation. The proposed structure can handle an 8196 spp ray-tracing image in 20 seconds at 64 spp.

Keywords: *Ray tracing, Denoising, Monte Carlo rendering, Neural network, Autoencoder*

1. Introduction

Ray tracing is a computer graphic technique that recreates light effects by tracking the lights from the light source to create realistic computer graphics. However, since images are created by tracking each light from the light source, the amount of computation is massive. The method to solve this issue is a path tracing that tracks the light in reverse from the pixels of the image to the light source. Currently, path tracing means ray tracing. It is realistically impossible to create images by tracing all lights in the pixels. Therefore, a few lights are traced from the pixel, and the average of them is defined as the pixel value. This method is called Monte Carlo Rendering [1].



(a) 8spp (b) 64spp (c) 1024spp

Figure 1. Comparison of the quality of ray tracing image according to changes in spp

Article history:
Received (November 12, 2019), Review Result (December 24, 2019), Accepted (January 30, 2019)

When tracing a lot of lights per pixel, more realistic images are created. The unit for tracing light is called sample per pixel (spp). When spp decreases, the amount of calculation drops, but images with more noise are created. On the other hand, when spp increases, more realistic images are created, but the amount of calculation increases significantly [Figure 1].

The difference of calculation amount according to spp is as shown in [Table 1].

Table 1. Comparison of calculation amount

Resolution	8spp	128spp	1024spp	8196spp
720p	7,373	117,965	943,718	7,553,434
1080p	16,589	265,421	2,123,366	16,986,931
4K	70,779	1,132,462	9,059,697	72,477,573

(unit = 1000)

To identify the increase of calculation amount according to the number of rays, the image of [Figure 1] based on 4K, which is the highest grade of images, was measured. Assuming that one calculation is performed for one sample during ray tracing at 8196 spp with noise-free, approximately 72.4 billion calculations are required. Such amount of calculation is too high to handle in real-time. When rendering at 8 spp, the number of calculations decreases by 1,000 times, and therefore, the spp must be reduced to decrease the amount of calculation.

Table 2. Peak Signal Noise Ratio (PSNR) and Structural Similarity(SSIM) by spp variation [2][3]

	8spp	128spp	1024spp
PSNR	13.6834	20.1158	25.1086
SSIM	0.1839	0.3987	0.6321

However, as shown in [Table 2], when the number of spp decreases, noise included in the image relatively increases, and the quality of the image becomes poor. This paper proposes a method for using a small amount of spp to obtain the image quality similar to the images that use more spp to remove noise from the image. Furthermore, this paper applies a deep learning-based neural network with high efficiency for removing noise and proposes an optimized structure of the applied neural network.

2. Proposed neural network

2.1. Autoencoder

The proposed neural network was configured using the Unet based Autoencoder [4]. First, the image size is gradually decreased using the encoder while the size of the feature map is increased. At this time, the layer between the encoder and decoder having the most feature maps is called the bottleneck layer, and it has a total of 512 feature maps. Afterward, the size of the original image is restored, passing through the decoder. Noise is removed by the decoder and an image without noise is shown. Then, noise that was not removed and the checked patterns that may occur at the boundary of the filter are removed additionally. The proposed neural network structure is as follows [Figure 2].

2.2. 1×1 Convolution

1×1 convolution was proposed by M Lin et al. [5]. If this method is used when the size of the feature map decreases, the amount of calculation decreases significantly. Using this method, the neural network gains more time. In this paper, the amount of calculation was decreased when the feature map was decreased using this method in the decoder.

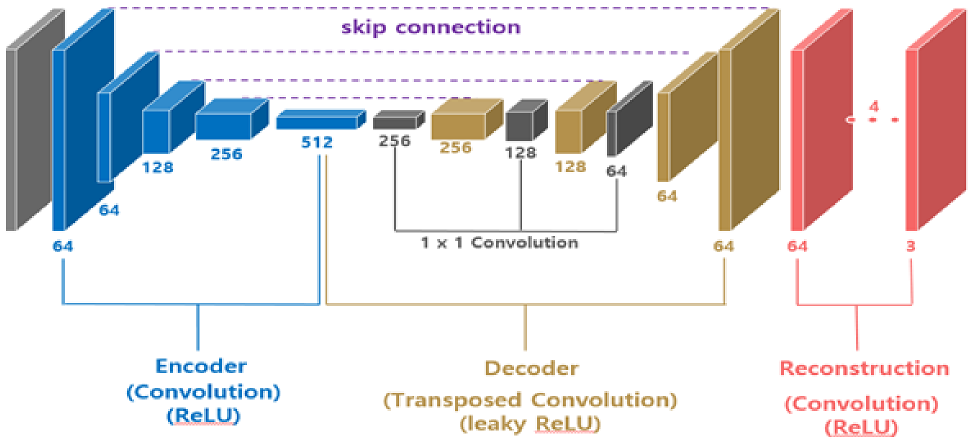


Figure 2. Denoising auto encoder

3. Implementation

This algorithm is comprised of preprocessing, encoding/decoding, and reconstruction processes. A noisy image is converted to allow the entry of neural networks through preprocessing and the features are extracted bypassing the encoding layer. The extracted features then pass the decoding layer again to return to the original image size. Lastly, the denoising image is generated by passing the reconstruction layer. During the training, the image is in small image units at a size of 64×64 extracted from random images in patch units. When learning is performed using the original image, it requires excessively large capacity memories, so it is divided into patch units and then the patches are gathered in batch units for use.

3.1. Preprocessing

Preprocessing allows output images from the renderer to be used with the input of the neural network. The image created by the renderer can output the color (R, G, B) as well as the values used for calculating the color value. This is applied for the use of neural network input. In general, Color 3ch (channel), Specular 3ch, Diffuse 3ch, Normal 3ch, Albedo 3ch, and Depth 2ch are generated and used. In addition, 2ch for Color, Specular, Diffuse, Normal, Albedo variance and 1ch for depth variance for a total of 22ch are used. At this time, when the values with large fluctuations are used as an input for neural networks, there is a possibility that the model will be divergence. Therefore, it is used by changing the standard deviation from the square root of variance. In addition, to detect the noise candidates and edges, the gradient of the pixel values is computed. The gradient is performed once in the x-direction and once in the y-direction. Therefore, Color 3ch, Specular 3ch, Normal 3ch, Albedo 3ch, and Depth 1ch for a total of 16ch times two for a total of 32 are generated. Lastly, using the median values with some of the noise removed, additional hints are given to the neural network. This is only applied in the color, specular, and diffuse where noise exists. Images that completed preprocessing will have a total of 63ch [Figure 3].

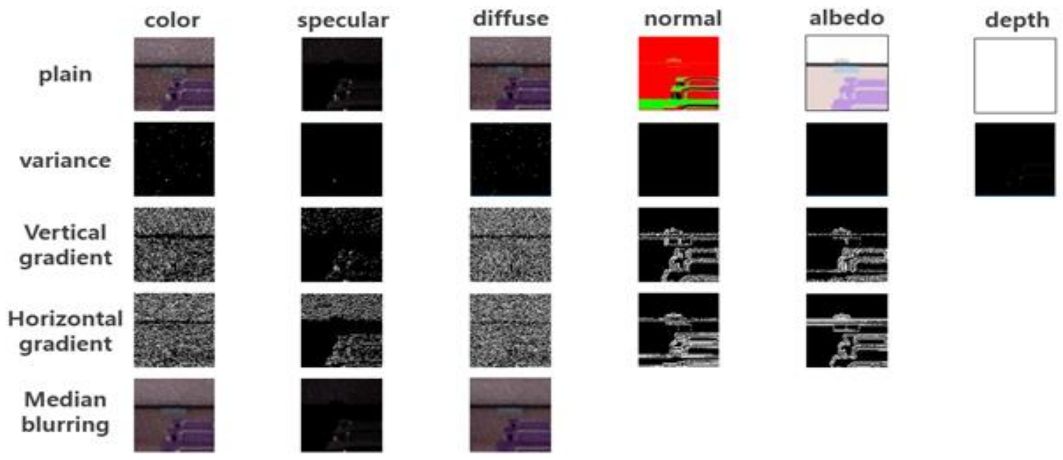


Figure 3. Preprocessed feature

3.2. Encoding/Decoding

Encoding is a work that reduces the size of the image and increases the number of feature maps. When an image has a size of 1024×1024, it undergoes encoding and the size is reduced significantly to 1024×1024, 512×512, 256×256, and 128×128. At the bottleneck layer between encoding and decoding, it has a total of 512 feature maps. Rectified Linear Unit (ReLU) [6] is used as the activation function. Information in each layer in encoding is saved and used as skip connection [7] in decoding. Decoding is a process that the features are reduced again and the original image is restored. Each layer is connected through the encoding layer and skip connection. Skip connection is added when restoring the original images of feature maps that were found bypassing the encoding. Through this, it finds the weighted values that create images similar to the original image more quickly during the training. Furthermore, it well presents the details of the actual image. The decoder's activation function uses Leaky Rectified Linear Unit (leaky ReLU) [8]. In addition, when restoring the original image, the feature map is reduced, and therefore, 1×1 convolution can be used. 1×1 convolution is first reduced before passing the filter, and therefore, the amount of calculation can be reduced by approximately 1/10.

3.3. Reconstruction

Reconstruction is a layer for post-processing the images that completed encoding/decoding. This is used to remove the noise that is not removed in the encoding/decoding layer and the check patterns that are created in the boundaries of the filter. Images that pass this layer are converted into a three-channel image comprised of only R, G, and B from the image comprised of feature maps.

4. Experiment result

4.1. Rendering time test

The data set for the experiment is shown in the following [Figure 4].

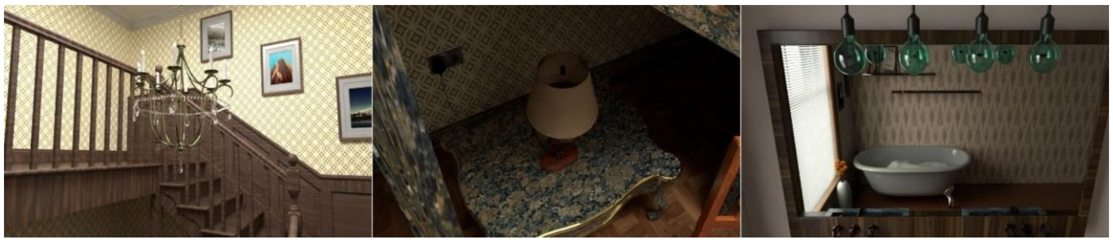


Figure 4. Experiment dataset

When displaying the same quality compared to noise for the video in [Figure 1], 8196 spp and 64 spp were compared. In 64 spp, the autoencoder neural network proposed in this paper was applied. The experiment environment was executed in AMD's 2990 wx (32core) and tungsten renderers, and the results are as shown in Table 3. When comparing the rendering time, the renderer of this paper showed a shorter execution time by 240 times.

Table 2. Rendering time comparison

	64 spp	8196 spp
Rendering Time	20sec	1hour 20min 48sec

4.2. Inference time test

To measure the effects of the 1×1 convolution-based autoencoder proposed in this paper, the inference time was measured in the experiment image. Results, as shown in [Table 4], were found in the aforementioned measurement environment. When using 1×1 convolution, it is evident that processing time was decreased by 30%.

Table 3. Inference time average in 64 scenes

	without 1×1	with 1×1
Inference Time	0.1930sec	0.1534sec

5. Conclusion

The ray-tracing renderer configured in this paper can remove noise through deep learning and use small amounts of rays to create high-quality graphic images. By reducing the number of rays to improve the processing speed to configure ray-tracing graphics in real-time, it is possible to configure high-quality content in mobile devices. In particular, the autoencoder in the 1×1 convolution structure can be applied in various neural networks, and therefore contribute to the studies regarding mobile neural networks.

References

[1] VEACH, Eric, GUIBAS, and Leonidas J., "Optimally combining sampling techniques for Monte Carlo rendering," Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM, pp.419-428, **(1995)** DOI: 10.1145/218380.218498

[2] Wang Z., Bovik A. C., Sheikh H. R., and Simoncelli E. P., "Image quality assessment: from error visibility to structural similarity," IEEE transactions on image processing, vol.13, no.4, pp.600-612, **(2004)** DOI: 10.1109/TIP.2003.819861

[3] Hore A. and Ziou D., “Image quality metrics: PSNR vs. SSIM,” 2010 20th International Conference on Pattern Recognition, IEEE, pp.2366-2369, **(2010)** DOI: 10.1109/ICPR.2010.579

[4] Vincent P., Larochelle H., Lajoie I., Bengio Y., and Manzagol P. A., “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” Journal of machine learning research, vol.11, pp.3371-3408, **(2010)** DOI: 10.1016/j.mechatronics.2010.09.004

[5] Veach E., and Guibas L. J., “Optimally combining sampling techniques for Monte Carlo rendering,” Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp.419-428, **(1995)** DOI: 10.1145/218380.218498

[6] Nair V., and Hinton G. E., “Rectified linear units improve restricted Boltzmann machines,” Proceedings of the 27th international conference on machine learning (ICML-10), pp.807-814, **(2010)**

[7] Mao,X. J., Shen C., and Yang Y. B., “Image restoration using convolutional auto-encoders with symmetric skip connections,” arXiv preprint arXiv:1606.08921, (2016)

[8] Maas A. L., Hannun A. Y., and Ng A. Y., “Rectifier nonlinearities improve neural network acoustic models,” Proc. Icml, vol.30, no.1, p.3, **(2013)**

Authors

HakHyun Lee
2014.2 ~ present: Seokyung Univ. Dept. Computer Engineering Bachelor's course

Chelwon Jo
2018: BS degree in Seokyung Univ. Dept. Computer Engineering BS degree
2018.3 ~ present: Seokyung Univ. Dept. Computer Engineering. Master's course

KwangYeob Lee
1985: BS degree in Electronics Engineering, Sogang University
1987: MS degree in Electronics Engineering, Yonsei University.
1994: Ph.D. degree in Electronics Engineering, Yonsei University.
1989 ~ 1995.2: Senior Researcher, Hyundai Electronics Inc.
1995.3 ~ present: Professor, Dept. of Computer Engineering, Seokyeong University