EPJ Web of Conferences **318**, 04010 (2025) https://doi.org/10.1051/epjconf/202531804010
AAPM-III 2025

On the modifications of one method for rendering global illumination using a generative adversarial neural network

Kirill Rubinov*, and *Alexander* Frolov

Russia National Research University "MPEI", Russia, Moscow

Abstract. Three variants for modifying one method for rendering 3D scenes using a generative adversarial network are proposed to generate realistic lighting in screen space. Experiments using software implementations of neural networks have shown that adding 25% resolution ray traced rendering to the generator input improves image quality when assessed by both the SSIM structural similarity index and the peak signal-to-noise ratio PSNR. Adding noise to the discriminator layers improves the quality of network performance in both respects, but not significantly. Passing the ray traced rendering to the hidden convolutional layers of the generator instead of the input leads to a decrease in quality in both metrics compared to other modifications, but avoids unwanted effects on the edges of objects.

1 Introduction

The main 3D rendering techniques used in modern systems are ray tracing and rasterization. Ray tracing utilizes a physical simulation of the light propagation in a scene and naturally produces realistic results, but at high resolution it requires large computational resources [1, 2]. Rasterization is based on creating a projection of scene objects onto a plane and using approximate methods to calculate lighting, which saves computing resources at the cost of reducing realism [1]. The main problem with rasterization is the visualization of global illumination, that is, the illumination of some objects by light reflected from other objects. One of the trends in the field of computer graphics is the development of methods to improve the quality of rasterization and bring it closer to the level achieved by the ray tracing method.

In recent years, neural networks have been actively used for image processing. In particular, neural network methods for visualizing three-dimensional scenes have been proposed. A method of training a generative adversarial network for approximating global illumination of a 3D scene is presented in [3]. The authors introduce a variant of GAN that learns a mapping from G-buffers (depth map, normal map, and diffuse map) and direct illumination to any global illumination solution. The method is evaluated through a comparison with both a state-of-the-art real-time GI technique (VXGI) and an offline rendering GI technique (path tracing). It is shown that the method produces effective GI

* Corresponding author: RubinovKA@mpei.ru

approximations and is also computationally cheaper than existing GI techniques. A similar technique is explored in [4] where the authors utilize the Pix2Pix architecture and specify the hyper-parameters and methodology used to mimic ray traced images from a set of input features. It is demonstrated that the GAN produces images with a quality comparable to the quality of the ray traced images, but is able to produce the image in a fraction of the time. In [5] a different approach is considered: the authors use two adversarial networks to generate ambient occlusion based on the rasterized maps and radiance based on the environment map.

Neural networks can also be used at the final stage of the rendering process to increase the resolution of the rendered image. This technique is utilized in NVIDIA DLSS [6]. It is also explored in a variety of works, such as [7].

2 Problem statement

To achieve this goal, we create three variants of generative adversarial networks, apply them to a number of test examples of images of typical sizes, and evaluate the results of their processing visually, as well as by similarity indicators PSNR — Peak Signal-Noise Ratio and SSIM — Structural Similarity Index Measure [8]. Based on the results of visual and quantitative comparison, we evaluate the quality of the proposed modification options and confirm that based on the available test images and a rendered image of the scene obtained by ray tracing in a reduced resolution IRT, as well as, possibly, adding noise layers to the discriminator, it is possible to improve the rendering quality according to generally accepted indicators and, thus, the quality of the modified method. Visualization of a scene obtained by ray tracing in a reduced resolution has not been used in works on global illumination visualization known to the authors.

3 Solution methods and assumptions

3.1 Generative adversarial network

To generate realistic images, a neural network architecture called Generative Adversarial Network (GAN) is used [9]. It consists of two parts: generator G and discriminator D, which are differentiable functions represented by multilayer perceptrons. Generator receives input data $z \sim p_z(z)$ and produces an image $G(z) \sim p_g(z)$ as output. The input can be a vector of random values, an image, or other data. The discriminator takes image x as input and outputs the probability $D(x)$ that x is a real image (that is, $x \sim p_{data}(x)$, where p_{data} is the unknown distribution of the real data).

During the training process, D receives as input an equal number of real images and images produced by the generator. The generator's task is to produce images that the discriminator cannot distinguish from real ones, while the discriminator, on the contrary, learns to recognize fake images. Thus, G and D compete with each other: D tries to maximize the probability of correctly recognizing all images, and G tries to minimize the probability of correctly recognizing generated images. To do this, the distribution of the output data of the generator p_g needs to be brought as close as possible to the unknown distribution of the real data p_{data}. Then, the problem is formulated as a game for two players with an evaluation function $V(G, D)$ [6]:

$$min_G \, max_D \, V\,(G, D) = E_{x \sim p_{data}(x)}[ln\, D\,(x)] + E_{x \sim p_z(z)}\big[ln\big(1 - D(G(z))\big)\big],$$

where, E is the mathematical expectation, $p_{data}(x)$ is the real data distribution and $p_z(z)$ is the distribution of the generator's input vector.

At each training iteration, the parameters of the discriminator D are updated first based on errors in recognizing m real and m images generated by G. Here, back propagation of the discriminator loss function is used [6]:

$$\frac{1}{m}\sum_{i=1}^{m}\left[ln\, D\,(x^{(i)}) + ln\left(1 - D\left(G(z^{(i)})\right)\right)\right].$$

After this, D with the updated parameters is used to calculate the loss function of the generator [7]:

$$\frac{1}{m}\sum_{i=1}^{m} ln\left(1 - D\left(G(z^{(i)})\right)\right).$$

In practice, instead of minimizing $ln\left(1 - D\left(G(z^{(i)})\right)\right)$, the problem of maximizing $ln\, D\left(G(z^{(i)})\right)$ can be solved — this helps to avoid undesirable behavior of the gradient, which may lead to a halt in the learning process [10], cited according to IEEE License Number 5910741407965, License date Nov 16, 2024.

3.2 Method of solving the problem

The modified generative adversarial network [3] uses four rasterized maps mentioned above to generate the final image with global illumination. The generator is represented by a convolutional network of the U-Net architecture [11] and produces image I_{GI} on output — a rendered image of the scene with global illumination; the discriminator is a convolutional network for binary classification. The network diagram is shown in Figure 1.

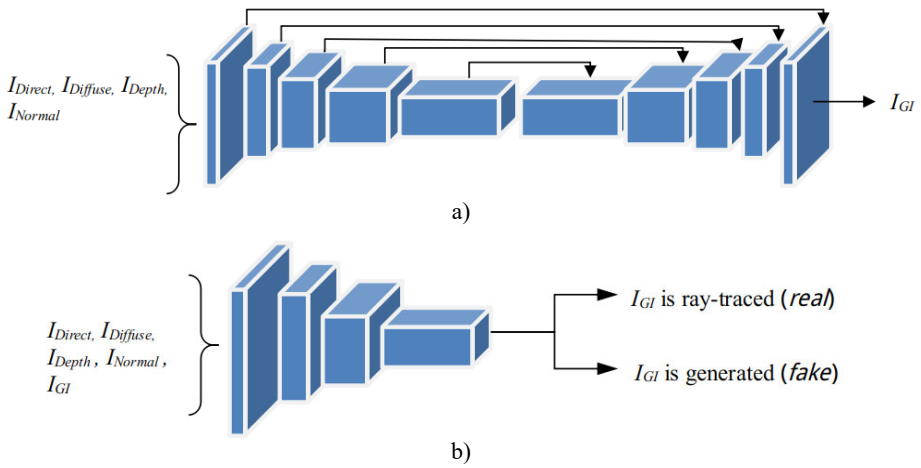

Fig. 1. Generative adversarial network for generating global illumination: a) generator, b) discriminator. I_{Depth} - depth map, $I_{Diffuse}$ - surface color map (diffuse map), I_{Direct} - direct illumination map, I_{Normal} - normal map.

The idea of the modifications of the method is that in the neural network, in addition to rasterized maps, also receives as input a fully rendered image I_{RT} of the scene obtained with ray tracing in a reduced resolution. At a low enough resolution, a ray traced image can be rendered in a short time. The work uses a resolution 4 times lower than the main resolution.

Figure 2 shows the proposed network modification options. $I_{SR} = F_{SR}(I_{RT})$, where F_{SR} is the interpolation method used when passing I_{RT} to the input layer.

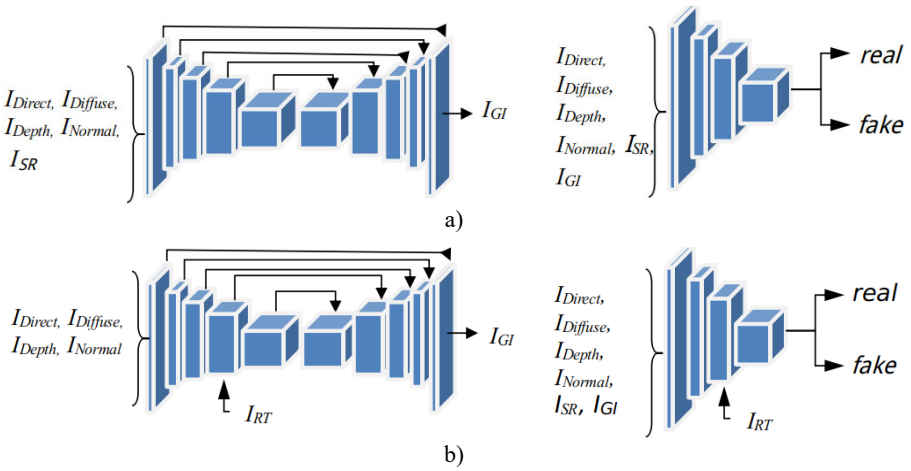

Fig. 2. Generator architecture variants: a) using interpolation, b) without using interpolation.

3.3 Software implementation

The software implementation of the methods was developed with Python using the PyTorch library [12]. Three versions of the neural network have been developed and trained, which are compared with the original model from [3]:

1. Model that accepts all data on the input layer (see Figure 2, a). Bicubic interpolation is used to bring the I_{RT} to the desired resolution.
2. A model similar to the model of the first option, in which a Gaussian noise layer is added after each of the convolutional layers of the discriminator. Adding noise to the discriminator is a well-known method for improving the quality of training of generative adversarial networks [13].
3. A model in which instead of passing I_{RT} to the input layer of the generator, it is passed directly to the hidden convolutional layer, where feature maps have the same resolution (see Figure 2, b). This eliminates the need to use interpolation methods.

The network training dataset consists of elements of the form (I_{Depth}, $I_{Diffuse}$, I_{Direct}, I_{Normal}, I_{RT}). All images in the set have the same resolution and correspond to the same angle and scene state.

To create the training dataset, the 3D graphics program Blender was used [14]. The GitHub page of [4] presents a Blender scene file containing a script for creating a training dataset using objects randomly placed in 3D space. The colors of objects and walls, as well as the angle and direction of lighting, are also determined randomly. For rasterization, the built-in EEVEE engine is used [15]. Ray tracing was performed using the installed AMD Radeon Prorender add-on [16]. Google Colab service [17] was used to train neural networks and conduct experiments.

To compare the quality of the images created using the trained neural networks, all network variants were tested on the same dataset. The examples of images of typical sizes ($2^n \times 2^{n+1}$, n = 8, 9, 10) obtained using different versions of the model are shown in Figures 3-5.

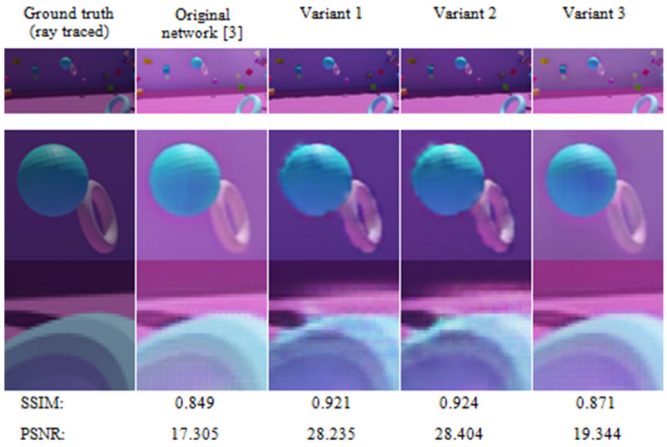

Fig. 3. Comparison of rendering methods. Top row — full images of size 256x512, middle and bottom rows — selected fragments. Index values are given for full images (top row).

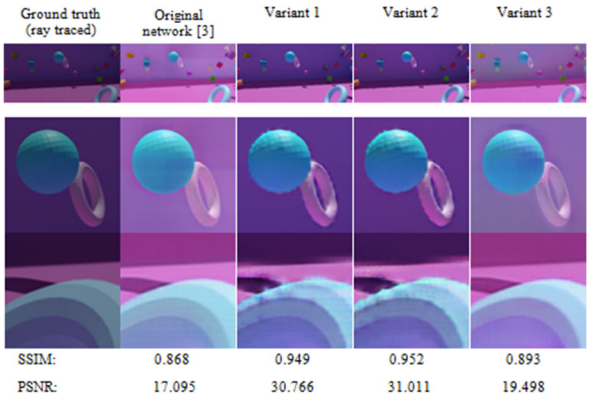

Fig. 4. Comparison of rendering methods. Top row — full images of size 512x1024, middle and bottom rows — selected fragments. Index values are given for full images (top row).

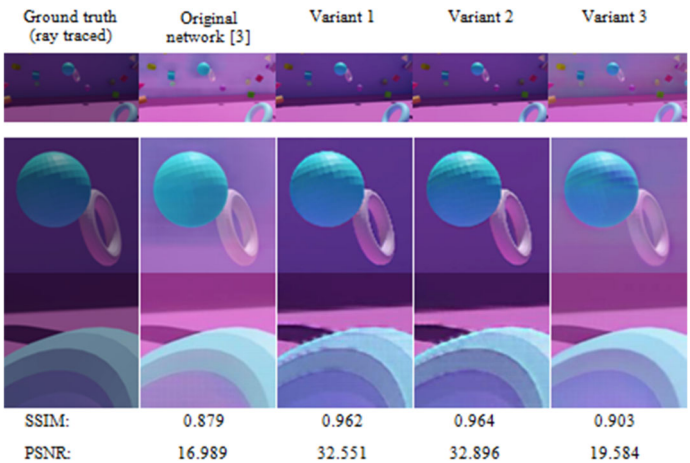

Fig. 5. Comparison of rendering methods. Top row — full images of size 1024x2048, middle and bottom rows — selected fragments. Index values are given for full images (top row) [10], cited according to IEEE License Number 5910741407965, License date Nov 16, 2024.

Having visually assessed the resulting images, we can formulate the following assumptions:

1. Adding ray traced low-resolution rendering to the input data (Variant 1) improves the overall quality of the resulting image. Even with a small training dataset, the colors are close enough to the real image, as the neural network can extract them directly from the low-resolution version. The edges of objects appear sharper, but there are some jagged edges caused by using an interpolated lower resolution image.
2. Adding Gaussian noise layers to the discriminator (Variant 2) partially smooths out the jagged edges of objects.
3. Transferring the ray traced image to the hidden layer (Variant 3) helps to avoid the appearance of the jagged edges. However, in this case, the colors of the image are not as close to ground truth as in 1, but closer than in the case of the original network.

4 Discussion

To quantitatively assess the quality of the results obtained and confirm their specified qualitative characteristics, two image similarity indicators are used: PSNR and SSIM [8]. Both metrics are provided by the Scikit-Image library [18].

The results of assessing the quality of the obtained images are given in Table 1. Based on them, conclusions can be drawn that are consistent with the corresponding assumptions.

1. Adding full visualization to the input data significantly improves the image quality (Variant 1: 0.921 / 28.235, 0.949 / 30.766, 0.962 / 32.551 instead of 0.849 / 17.305, 0.868 / 17.095, 0.879 / 16.989 for the original network).
2. A slight improvement compared to the first option is provided by adding layers with Gaussian noise to the discriminator (Variant 2: 0.924 / 28.404, 0.952 / 31.011, 0.964 / 32.896).
3. Transferring the ray traced image to hidden layers reduces the quality of the images, but it remains at a higher level than the original network (Variant 3: 0.871 / 19.344, 0.893 / 19.498, 0.903 / 19.584 instead of 0.849 / 17.305, 0.868 / 17.095, 0.879 / 16.989).

Table 1. SSIM/PSNR values.

Image resolution	256x512	512x1024	1024x2048
Original network	0.849 / 17.305	0.868 / 17.095	0.879 / 16.989
Variant 1	0.921 / 28.235	0.949 / 30.766	0.962 / 32.551
Variant 2	**0.924 / 28.404**	**0.952 / 31.011**	**0.964 / 32.896**
Variant 3	0.871 / 19.344	0.893 / 19.498	0.903 / 19.584

Additionally, let us consider the difference in the behavior of the PSNR indicator in the considered network options. For networks with full visualization transmitted to the input (Variant 1, 2), the indicator increases with increasing image size (Variant 1: 28.235, 30.766, 32.551; Variant 2: 28.404, 31.011, 32.896). For the final version of the network (Variant 3), it grows, but only slightly (by less than 0.8%): 19.344, 19.498, 19.584. However, for the original network, the values of the indicator decrease with increasing image size: 17.305, 17.095, 16.989. This suggests that the original network is better at handling lower resolution images.

5 Conclusion

In this paper we propose options for modifying the method of rendering three-dimensional scenes using a generative adversarial network to generate realistic lighting in screen space

based on rasterized maps, which consists of using additionally a ray traced scene rendering with 25 percent resolution and, possibly, adding layers of Gaussian noise in the discriminator. It is shown that such modifications allow one to improve the quality of visualization in terms of PSNR and SSIM [9], which is demonstrated by the results of applying the proposed modified method on typical test examples used in [4].

References

1. M. Pharr, W. Jakob, G. Humphreys, *Physically based rendering: From theory to implementation* (MIT Press, Cambridge, US, 2023).
2. L. Eversberg, J. Lambrecht, Sensors **21(23)**, 7901 (2021).
3. M.M. Thomas, A.G. Forbes, Deep illumination: Approximating dynamic global illumination with generative adversarial network. arXiv preprint arXiv:1710.09834v2, last accessed 11.11.2024
4. J. Harris-Dewey, K. Klein, International Journal of Electronics and Electrical Engineering **10(1)**, 1-6 (2022).
5. F. Abbas, M. Malah, M.C. Babahenini, Pattern Recognition Letters **166(C)**, 209-217 (2023). https://doi.org/10.1016/j.patrec.2022.12.007
6. A. Watson, Deep Learning Techniques for Super-Resolution in Video Games. https://arxiv.org/abs/2012.09810, last accessed 11.11.2024
7. L. Xiao, S. Nouri, M. Chapman, A. Fix, D. Lanman, A. Kaplanyan, ACM Transactions on Graphics **39(142)**, 1-12 (2020).
8. Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, IEEE Transactions on Image Processing **13(4)**, 600-612 (2004).
9. I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Communications of the ACM **63(11),** 139-144 (2020).
10. K. Rubinov, A. Frolov, A. Mamontov, *Educational Resources for Remote Comparative Study of Generative Adversarial Networks*, in Proceedings of the 7th International Conference on Information Technologies in Engineering Education (Inforino), Moscow, Russian Federation, pp. 1–5 (2024).
11. O. Ronneberger, P. Fischer, T. Brox, Lecture Notes in Computer Science **9351**, 234-241 (2015).
12. P. Yakubovskiy, *Segmentation_models_pytorch* (Documentation, 2024) https://readthedocs.org/projects/segmentation-modelspytorch/downloads/pdf/latest/ last accessed 11.11.2024
13. C.K. Sønderby, J. Caballero, L. Theis, W. Shi, F. Huszár, Amortised MAP Inference for Image Super-resolution. https://arxiv.org/abs/1610.04490, last accessed 11.11.2024
14. Home of the Blender project, https://www.blender.org/, last accessed 11.11.2024
15. I.A. Astuti, I.H. Purwanto, T. Hidayat, D.A. Satria, Haryoko, R. Purnama, *Comparison of Time, Size and Quality of 3D Object Rendering Using Render Engine Eevee and Cycles in Blender*, in Proceedings 5th Int. Conf. on Computer and Informatics Engineering, (IC2IE) (Jakarta, Indonesia: IEEE), pp. 54-59 (2022). https://doi.org/10.1109/IC2IE56416.2022.9970186
16. A. Yoshimura, S. Ikeda, T. Harada, Geometry and Texture Streaming Architecture/in Radeon™ ProRender. Advanced Micro Devices, Inc. Technical Report No. 22-10-daf5 (2022).

17. K. Profeta, Introduction to Google Colaboratory for Research. https://www.youtube.com/playlist?list=PL6QnpHKwdPYgKlgb9MjaGVlN9YPEWdGkJ/, last accessed 11.11.2024
18. E. Gouillart, 3.3. scikit-image: Image processing – Scientific Python Lectures. https://lectures.scientific-python.org/packages/scikit-image/index.html/, last accessed 11.11.2024