Denoising Monte Carlo Ray Tracing using Deep Learning

Harry Chen
University of Toronto
harry7557558@gmail.com

Jeffrey Ming Han Li
University of Toronto
jeffreymh.li@mail.utoronto.ca

***Abstract*—We present a web-based tool that tackles a less-done task: rendering mathematically defines shapes by physically simulating light transport, and removing rendering noise using a deep learning model, both in real-time. The renderer leverages Monte Carlo path tracing, a physically based technique that simulates light transport by tracing light paths and accumulating weighted radiance estimates. The denoising model is a U-Net with residual connections designed for both high quality and real-time inference. The model is trained on noisy-clean image pairs by minimizing a weighted sum of L1, L2, VGG perceptual, and adversarial loss functions. Combining rendering and denoising, we are able to achieve a frame rate up to 40fps at full screen resolution. This demonstrates the possibility of real-time rendering and deep learning model inference natively on the web, as well as creating a tool that allows educators and amateur artists to visualize mathematical surfaces. The tool can be found under the following URL: https://spirulae.github.io/implicit3-rt.**

I. Introduction

The synthesis of realistic images of virtual worlds is the primary driving force for the development of computer graphics and ray tracing techniques to demonstrate a 3D space using a 2D image.

The most common ray tracing technique for realistic visualization is Monte Carlo Path Tracing, which generates unbiased estimates of radiance by simulating ray bouncing in random directions using various Monte Carlo sampling techniques to reduce variance. However, Monte Carlo Path Tracing often comes with noise and high processing time for a clear image. Yet, introducing denoising models such as U-Net and ResNet allows Monte Carlo Path Tracing to operate effectively and produce clearer images.

Motivated by this problem, this article demonstrates denoising models comparison between ResNet, UNet, Residual UNet, GAN, and Improved GAN with auxiliary buffers implemented on a self-built 3D Math function plotter. With a pursuit of education purposes and enhancing people's accessibility to a ready-to-use 3D math plotter or ray tracing platform, the 3D Math Function plotter allows users to access ray tracing on multiple platforms(mobiles, computers, etc.) via a website.

A. Motivation

Image denoising is traditionally achieved through signal filtering techniques like the well-known Gaussian filtering, as well as improvements such as bilateral filtering, which are fast but often blur edges without removing noise completely. One of the earliest deep-learning works developed for CGI production-level denoising was KPCN. [Bako et al., 2017] However, its kernel prediction mechanism requires a substantial model size, which renders it impractical for real-time applications. An advancement in denoising is the introduction of the DEMC model [Yang et al., 2019] that leverages a U-net architecture for enhanced speed, as well as incorporating preprocessing as gamma transforms and integrating renderer outputs such as albedo and normal. [Alsaiari et al., 2019] employs a weighted sum of L1, VGG perceptual, noise reduction, and adversarial loss functions for model training, which show high generalizability of small models but face challenge of GAN color shifts. Intel Open Image Denoise (OIDN) [Áfra, 2024] is a popular software for real-time ray tracing denoising applications. While lacking elaborate features other than a plain U-Net, OIDN targets real-time speed. However, it has limited platform support due to its highly low-level inference implementation.

B. Problem Definition

The most common issue faced by Monte Carlo Path Tracing is the noise and high processing time for a clear image. Therefore, to tackle this problem, the objectives are to remove rendering noise without blurring edges and losing fine details, keeping the overall color of the images by using a small denoising model at high inference speed on the web. The denoising model is also expected to output a clean image of the same dimension by inputting a noisy image and including auxiliary buffers for better image quality.

II. Related Work

Until the late 2010s, Monte Carlo path tracing has largely been based on Veach's Ph.D. thesis [Veach, 1998] written in the late 90s, which generates unbiased estimates of radiance by simulating ray bouncing in random directions, with various sampling techniques to reduce variance. While techniques developed in the past years (notably [Ouyang et al., 2021]) demonstrate rapid convergence in extreme edge cases, classical Monte Carlo path tracing is suitable enough for our project. Regardless of the development of path tracing, physically rendering mathematically defined shapes in web browsers is a very less explored area in both engineering and academia. The closest work we find is shaders shared on Shadertoy [Beautypi (Inigo Quilez and Pol Jeremias),], a web-based platform that allows amateur users to create WebGL fragment

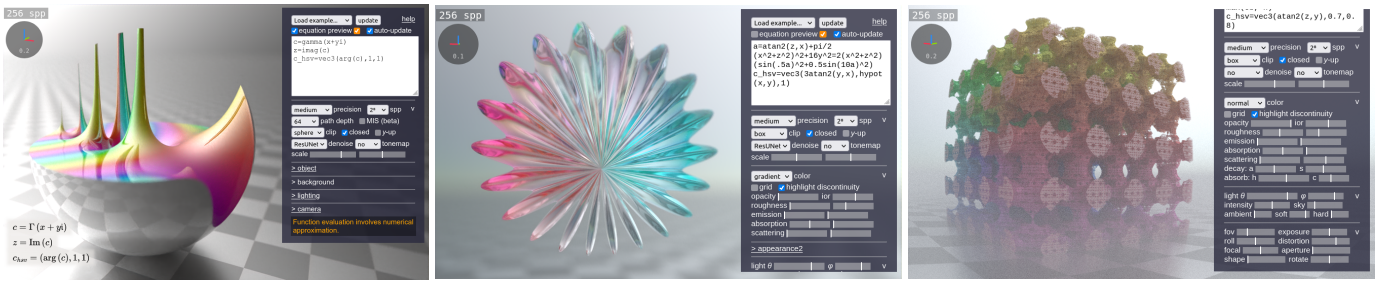

Fig. 1. Screenshots of our demo, showing various capabilities: Support for variable/function definition, complex numbers, and custom color; Simulating diffuse, glazed, metallic, and refractive surfaces; Simulating fog, camera out-of-focus blur, and sky lighting; Image denoising using our models. All these examples run within web browser and can be selected from the set of examples.

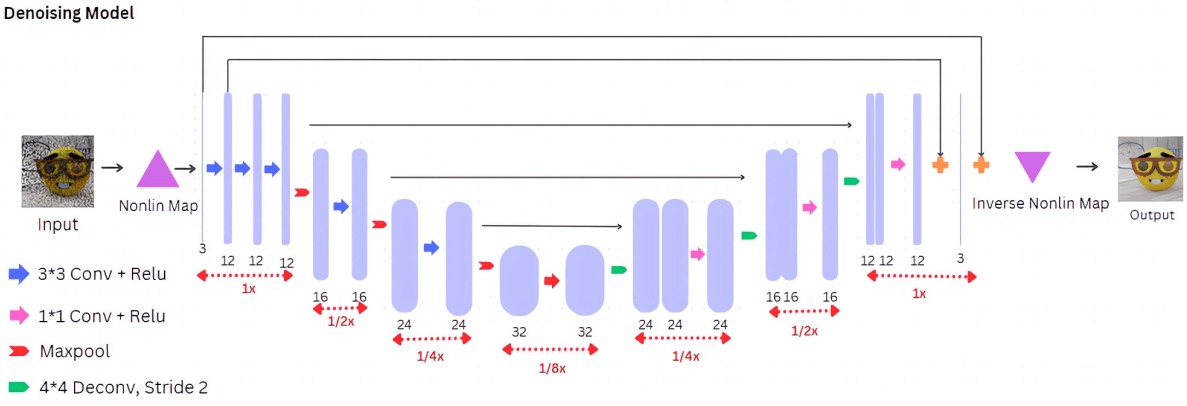

Fig. 2. Our Residual U-Net with GAN and Combined Loss

shaders, which makes path tracing possible but is unintuitive since users need to write code.

Image denoising is traditionally achieved through signal filtering techniques like the well-known Gaussian filtering, as well as improvements such as bilateral filtering, which are fast but often blur edges without removing noise completely. One of the earliest deep-learning works developed for CGI production level denoising was KPCN. [Bako et al., 2017] However, its kernel prediction mechanism requires substantial model size, which renders it impractical for real-time applications. An advancement in denoising is the introduction of the DEMC [Yang et al., 2019] model that leverages an U-net architecture for enhanced speed, as well as incorporating preprocessing like gamma transforms and integrating renderer outputs such as albedo and normal.

[Alsaiari et al., 2019] employs a weighted sum of L1, VGG perceptual, noise reduction, and adversarial loss functions for model training, which show high generalizability of small models but faces challenge of GAN color shifts. Intel Open Image Denoise (OIDN) [Áfra, 2024] is a popular software for real-time ray tracing denoising applications. While lacking elaborate features other than a plain U-Net, OIDN targets real-time speed. However, it has limited platform support due to its highly low-level inference implementation.

III. Methodology

Our software consists of a web user interface, where the equation input by the user is parsed and converted into shader source, with optimization using dynamic programming. Both shader source and path tracing code are compiled as a single WebGL fragment shader. With renderer parameters (such as random number seeds and parameters controlling surface appearance, lighting, and camera) passed to the shader as WebGL uniform variables, at each pixel, the shader computes an unbiased radiance estimate, which is averaged to produce a noisy image. The denoiser takes the noisy image and inferences the denoising model to produce a clean image, which is displayed on the screen.

The denoising model is mainly a U-Net. An U-Net is similar to an image encoder-decoder model, but with connections between encoder and decoder. The input image is first downscaled through convolutional and pooling layers with increasing depth, then upscaled through transposed convolutional layers, with the downscale part connected to the upscale part through concatenation to prevent loss of information. Finally, there are two residual connections before producing the output image. The model is trained by minimizing a weighted sum of L1, L2, VGG perceptual, and adversarial (GAN) loss functions [Alsaiari et al., 2019]. We also added an original loss function to correct color shift by matching the global mean of pixels before and after denoising.

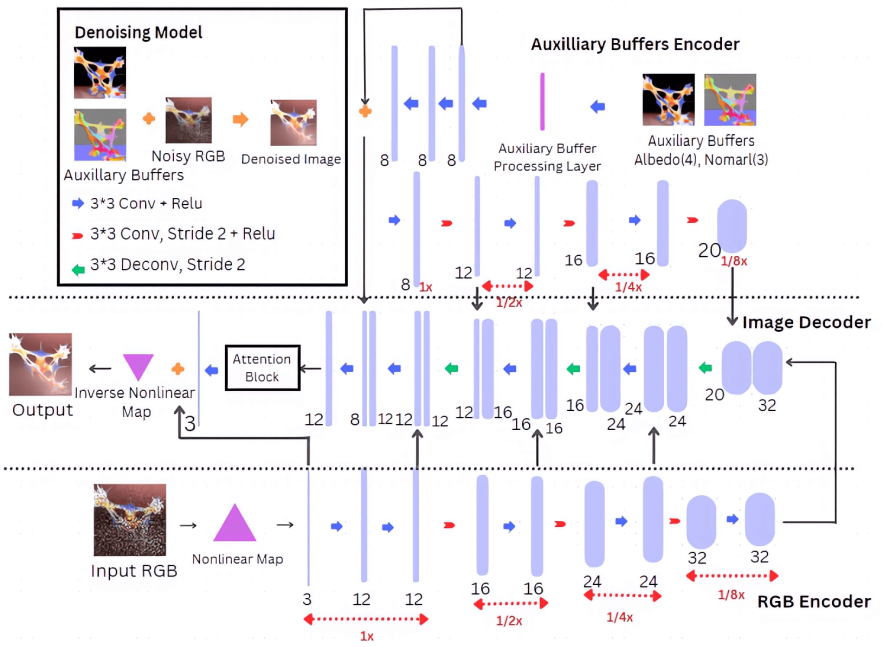

Fig. 3. Our Attention U-Net with Auxiliary Buffer Input

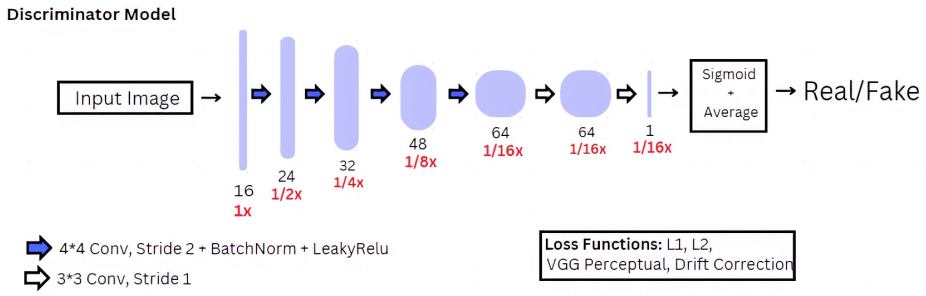

Fig. 4. Our discriminator, used for training of both models

Since our project focuses on inference speed rather than existing large models for offline rendering, it took experimentation for us to create a model to fit our needs. We initially started with a deep residual network (ResNet), which was proven too slow, so we moved to a U-Net that allows allocating more model parameters into encoder-decoder "bottleneck" to speed up inference with equal quality. After experiencing image blurring and color shift issues, we added residual connections to the U-Net (ResUNet), which empirically improved generalizability to rare colors and resulted in faster convergence in the early training stage. Introducing GAN significantly improved the model's ability to capture fine details but introduced additional color shifts, and therefore we introduced an additional loss function to correct it.

IV. Results and Discussion

As shown in Table 1, the quality of the noisy image is significantly improved by implementing denoising models. ResNet fails to remove noise completely. U-Net blurs feature the most for its stacked convolutional layers, and ResUNet creates clearer images by adding residual connections to the U-Net structure. Overall, ResUNet trained with the GAN model approaches the ground truth images the most when judged by human eyes.

We test the performance of the ResUNet + GAN model in our web-based path tracer, on a laptop equipped with a NVIDIA RTX 3070 GPU. The frame rate is obtained through WebGL timer query supported by Google Chrome. For the default "Red Flower" scene, at 1920×1080 resolution, it renders at 48fps without denoising and 17fps with denoising, while at 1024×768 resolution, the numbers are respectively 95fps and 42fps. For the "Fractal Sponge" scene, which involves complex geometry and indirect lighting, the scene renders at 12fps and 10fps without and with denoising, demonstrating a neglectable impact of denoising on rendering speed.

Image 1: Ground Truth

Image 2: Noisy

Image 3: ResNet

Image 4: U-Net

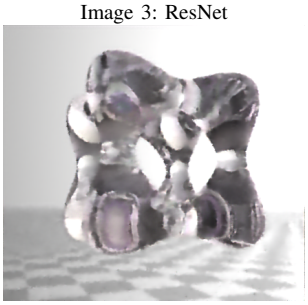

Image 5: ResUNet

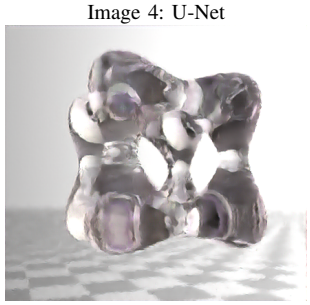

Image 6: ResUNet + GAN

TABLE I
COMPARISON OF SOME OF OUR MODELS

A. Ethical Considerations

Since the renderer is only capable of rendering mathematical shapes for hobbies and educational purpose, it is unlikely that it can cause unwanted ethical issues. The denoiser, however, can be used for a variety of denoising tasks not limited to mathematical shapes. When applied on images containing identifying information like faces and car plate numbers, it is important for the model to be accurate to avoid harm from misidentification.

B. Replication Package

For main development repository containing renderer code and inference engine can be found under the spirulae repository. Scripts that train denoising models can be found under the Graphics repository.

V. Conclusion

Through the creation of this tool, we demonstrate the possibility of real-time rendering and deep learning model inference natively on the web, which is something very less-done before. Released freely and accessibly, the tool allows educators and amateur artists to visualize mathematical surfaces with exceptionally high visual quality without need for expensive software.

VI. Future Work

The rendering parts of our project have been largely functional, but the denoising part still has a large room for improvement. On the engineering side, we need an implementation for optimized inference speed: the model is currently inference with WebGL and JavaScript written from scratch since third-party solutions we explored (onnx.js, WebDNN, ShaderNN, etc.) are either incompatible or lack an efficient interface with our WebGL renderer. On the academic side, we are still in the process of experimenting with more models for higher speed and performance, which may include auxiliary buffer fusion, squeeze-excitation modules, attention mechanisms, etc.

VII. Limitations

Currently, the demo runs very slow and crashes frequently on devices without a dedicated graphics card, which breaches our compatibility objective. The tool also struggles to handle auxiliary buffers in the presense of camera out-of-focus blur and fog, which makes it less usable to complex scenes.

VIII. Acknowledgements

We appereciate University of Toronto Machine Intelligence Student Team (UTMIST), and University of Toronto Engineering Society for supporting this research with workshops and funding. We also thank the CUCAI team for providing an opportunity for the showcasing our project.

References

[Áfra, 2024] Áfra, A. T. (2024). Intel® Open Image Denoise. https://www.openimagedenoise.org.

[Alsaiari et al., 2019] Alsaiari, A., Rustagi, R., Thomas, M. M., and Forbes, A. G. (2019). Image denoising using a generative adversarial network. In *Proceedings of the IEEE 2nd International Conference on Information and Computer Technologies*, pages 126–132.

[Bako et al., 2017] Bako, S., Vogels, T., Mcwilliams, B., Meyer, M., Novák, J., Harvill, A., Sen, P., Derose, T., and Rousselle, F. (2017). Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics*, 36(4):97:1–97:14.

[Beautypi (Inigo Quilez and Pol Jeremias),] Beautypi (Inigo Quilez and Pol Jeremias). Shadertoy. Very special thanks to Activision, Reinder Nijhoff, Patrick Labatut, Henrique Lorenzi, Otavio Good, Philip Wagner, Yanling He, Juan A. Martinez (stage7), Mari Miyashita, Nikochan, Sara Goepfert, Jose Manuel Perez (JosSs), Teresa, Sara (Gizma), Brett (AudEo Flow), Dave Hoskins, Osama Mahmood, Joan Perez, Kamran Saifullah, Chintu Solanki, Eduardo (@deb$_{s}ecurity$).

[Ouyang et al., 2021] Ouyang, Y., Liu, S., Kettunen, M., Pharr, M., and Pantaleoni, J. (2021). Restir gi: Path resampling for real-time path tracing. *Computer Graphics Forum (Proceedings of High Performance Graphics)*.

[Veach, 1998] Veach, E. (1998). *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 408 Panama Mall, Suite 217, Stanford, CA, United States. Adviser: Leonidas J. Guibas.

[Yang et al., 2019] Yang, X., Hu, W., Wang, D., Zhao, L., Yin, B., Zhang, Q., Wei, X., and Fu, H. (2019). Demc: A deep dual-encoder network for denoising monte carlo rendering. *Journal of Computer Science and Technology*, 34(5):1123–1135. Published in Journal of Computer Science and Technology. The final publication is available at http://link.springer.com/article/10.1007/s11390-019-1964-2.