Demo Available

UNIVERSITY OF TORONTO

Denoising CGI Renderings using Deep Learning

Harry Chen, Jeffrey Li, Victor Deng, Daniel Chua, Justin Wu, Sijie Han,
Muhammad Ahsan Kaleem, Richard Huang, Rick Lin, Jack Chen

UTMIST

Why Path Tracing and Denoising

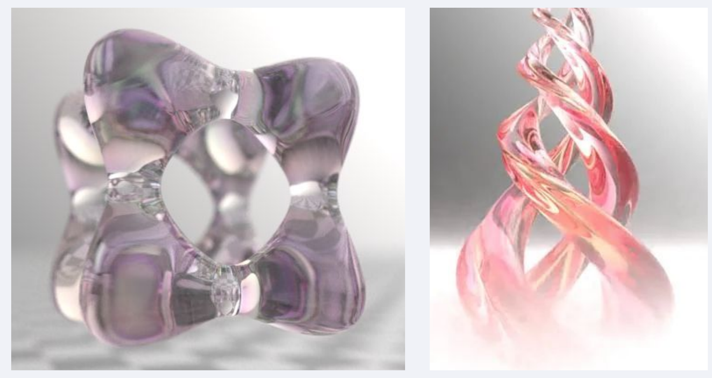
Path tracing, a physically-based rendering technique, can generate high-quality CGI images, but it produces unwanted noise before convergence. Recently, deep learning methods are used to remove noise, speed up rendering while improve image fidelity.

Path Tracing is a quality rendering technique that physically simulates light transport.

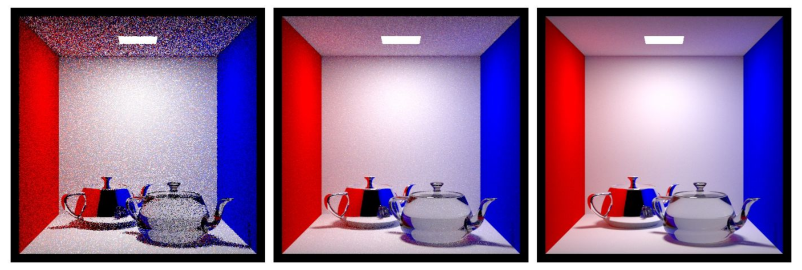
Path tracing naturally handles:

- 🧊 Glossy reflection and Refraction
- 🌓 Soft shadow and Ambient occlusion
- 🎥 Depth of field and Lens distortion
- Translucency and Fog/Smoke
- ... More!

However, the stochastic nature of path tracing produces noise across pixels, which reduces image fidelity. Reducing noise traditionally requires a large number of computationally expensive samples.

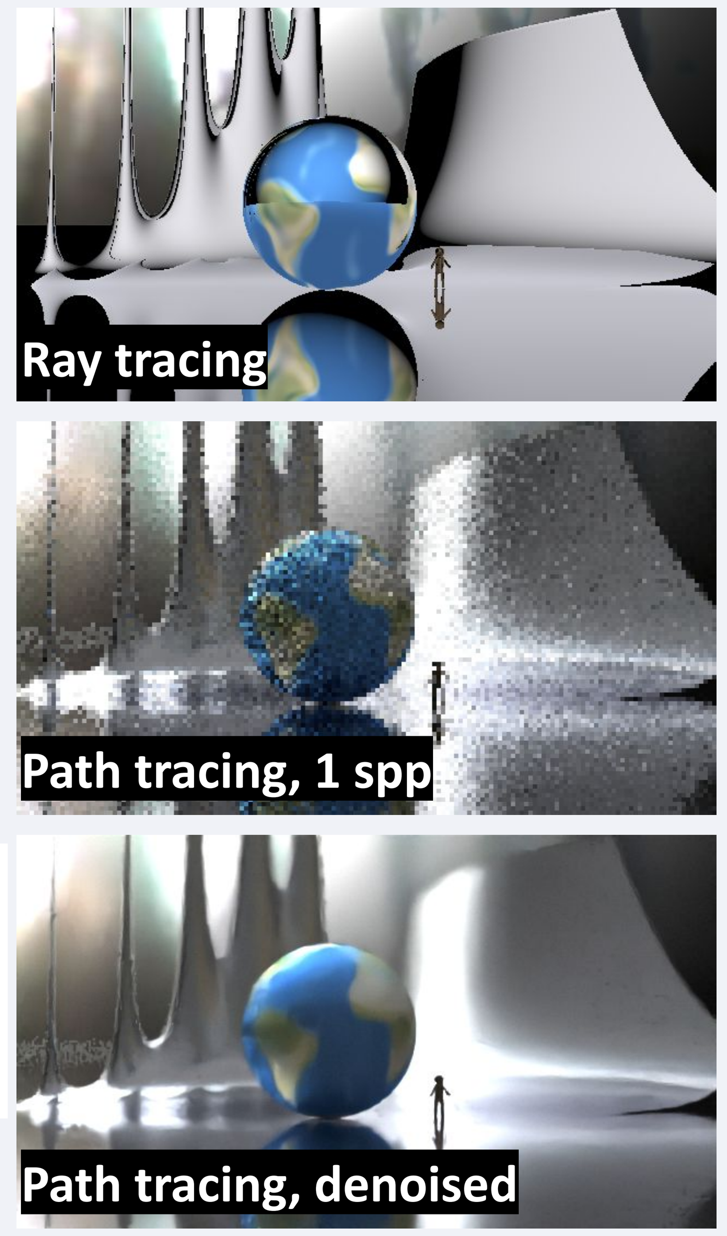


Examples of path-traced images in our rendering engine, demonstrating refraction, fog, and depth of field

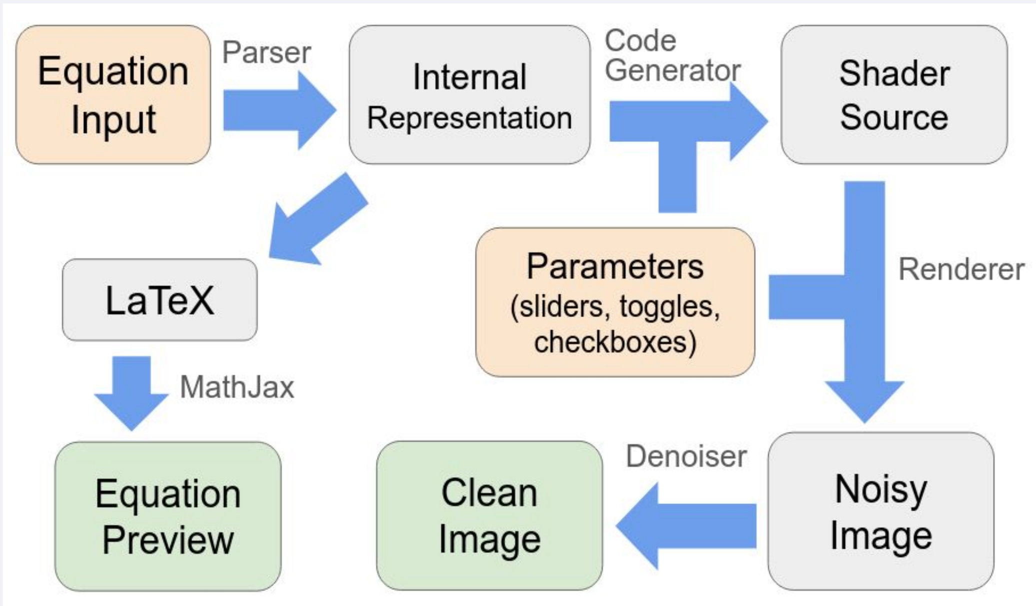


Path tracing with 1, 16, 256 samples per pixel (spp)

Christensen et al., "The Path to Path-Traced Movies," Oct. 2016.

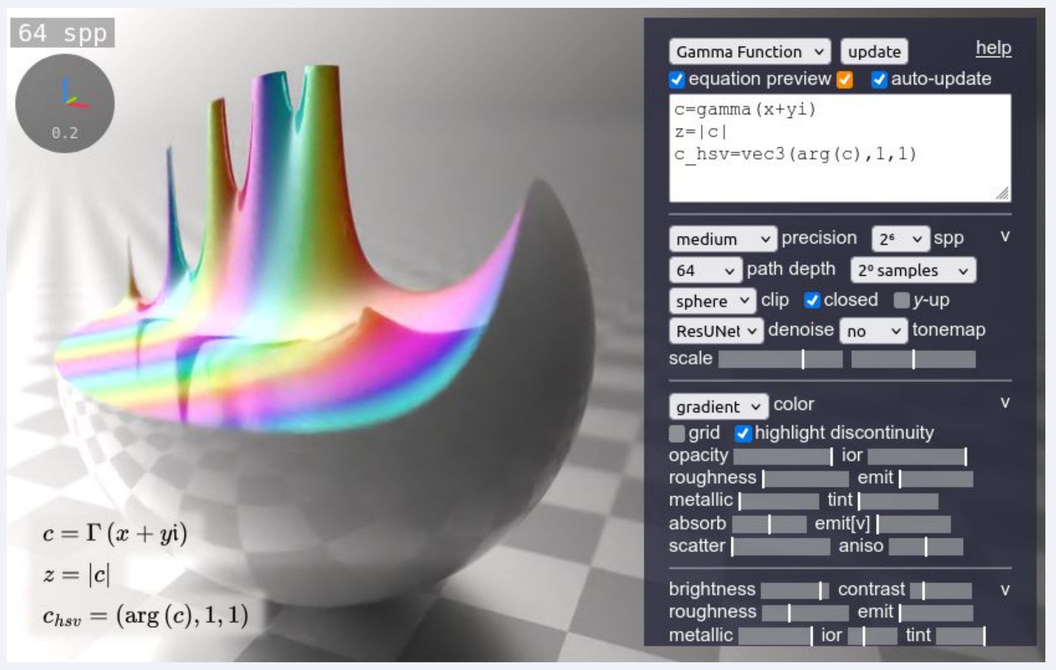


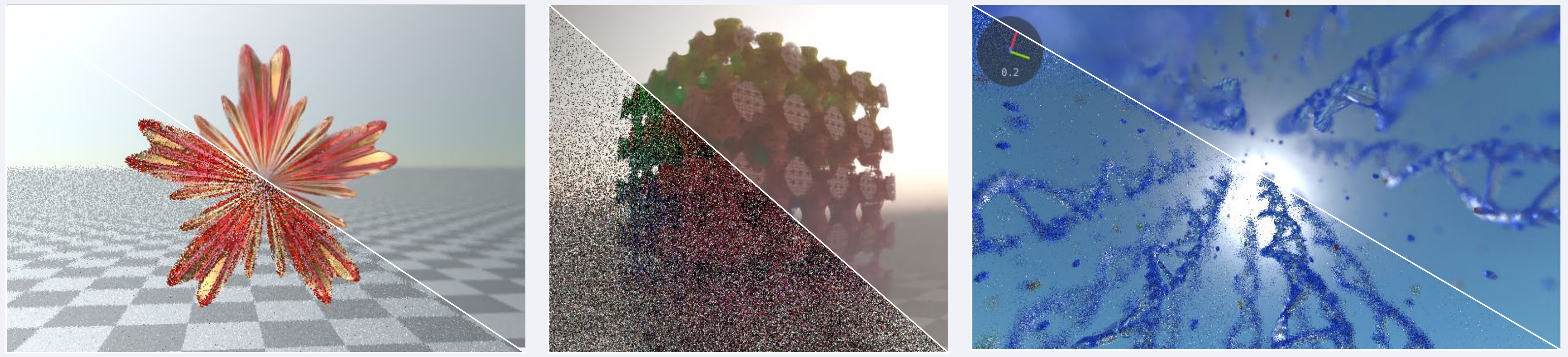
Implementation and Results



Our project involves developing a rendering engine along with the denoising model. We focus on the following objectives in our design and implementation:

- 🔥 Performance: We tried to achieve real-time speed. For speed we fully fused denoising into rendering pipeline. We minimized dependency in our code.
- 💎 Uniqueness: Our demo runs completely inside a web browser. We render mathematical shapes—the raymarching algorithm is capable of performing intersection for arbitrary implicit surfaces.
- 🤩 Impression: we aim to produce stunning visuals, and we made the demo compatible with desktop and mobile devices. We highlight path tracing features like refraction and depth of field.
- 💡 Educational: By rendering user-input mathematical equations, the demo demonstrates mathematical capability to produce visual art. We open-source our demo and provide documentation.



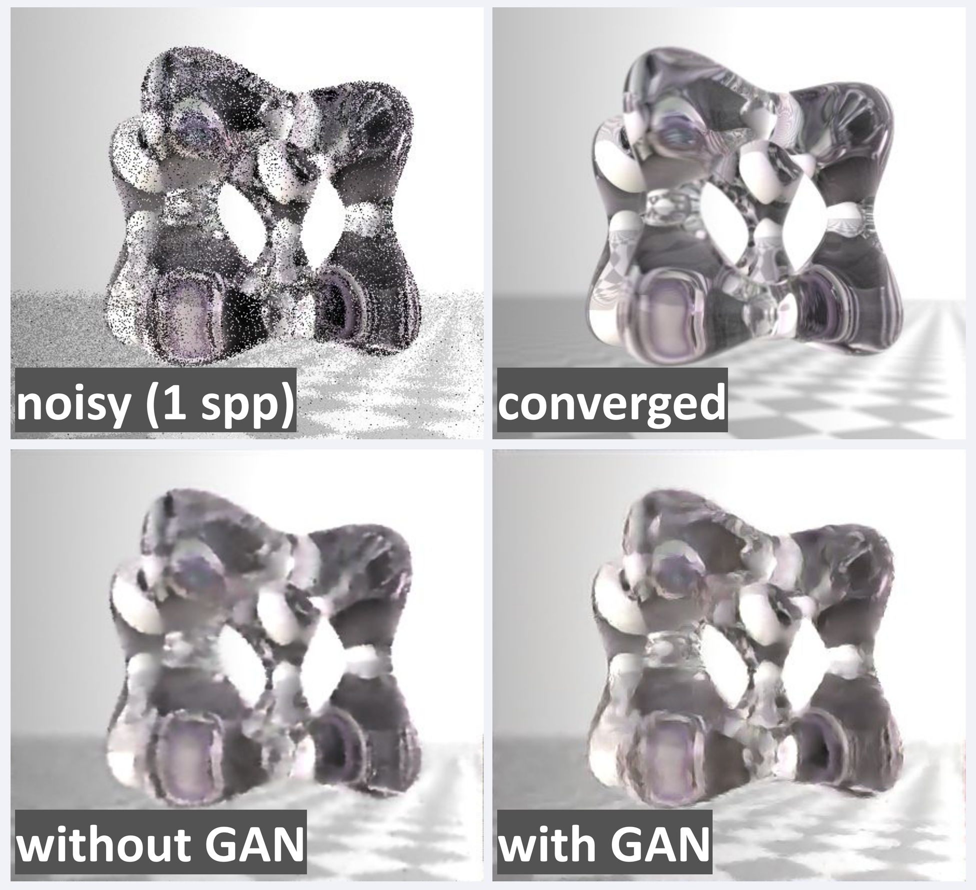


Quantitative Results (1024×768, NVIDIA RTX 3070, 1 spp)

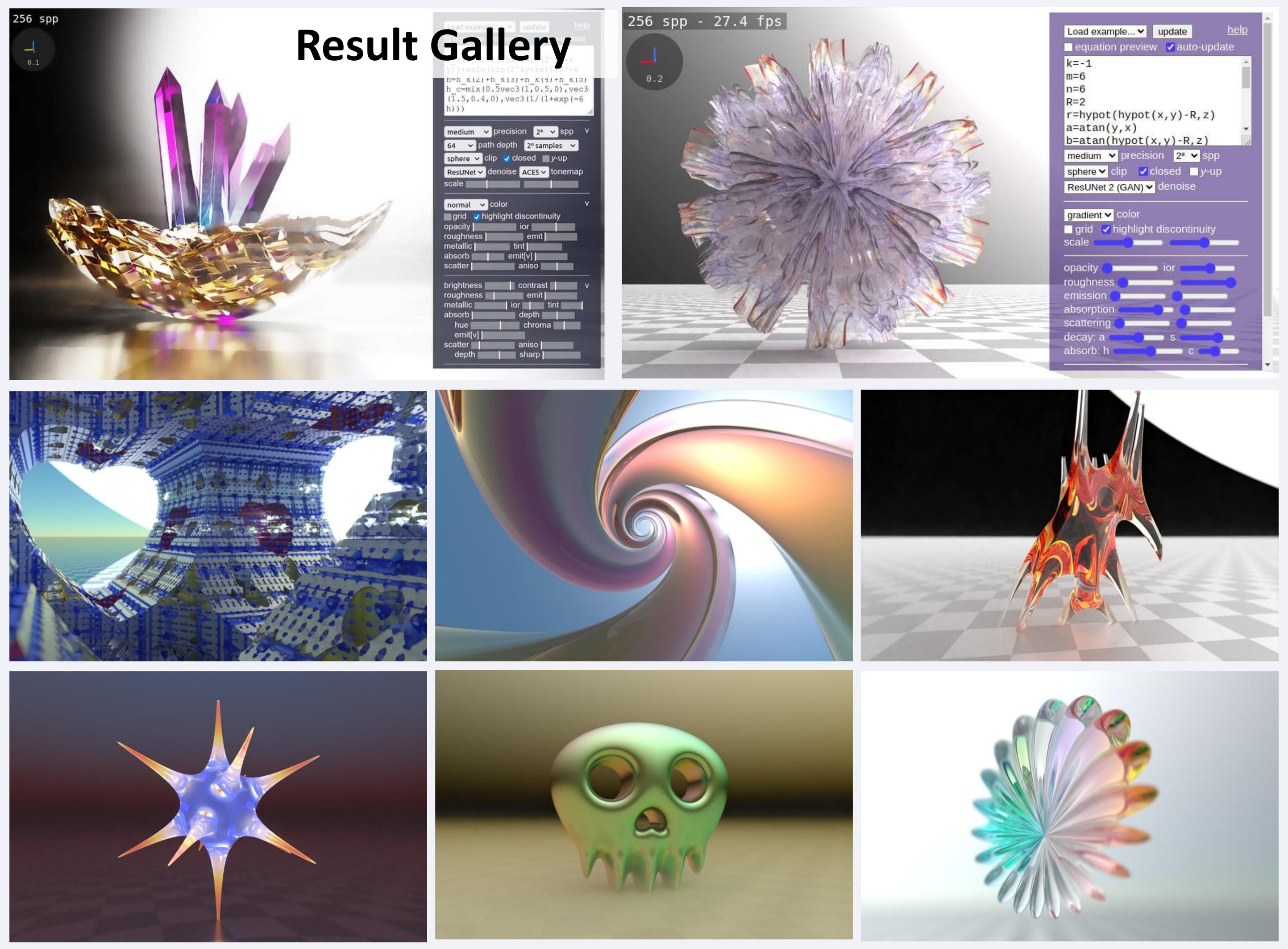
Our implementation runs at 42 fps for the flower scene with denoising, and 95 fps without. For the sponge scene, which involved complex geometry and indirect lighting, it runs 10 fps with denoising and 12 fps without, where the slowdown of denoising becomes minimal.

Config	FPS
Flower, denoise	42
Flower, no denoise	95
Sponge, denoise	10
Sponge, no denoise	12

Training with **adversarial loss (GAN)** allows recovering fine details



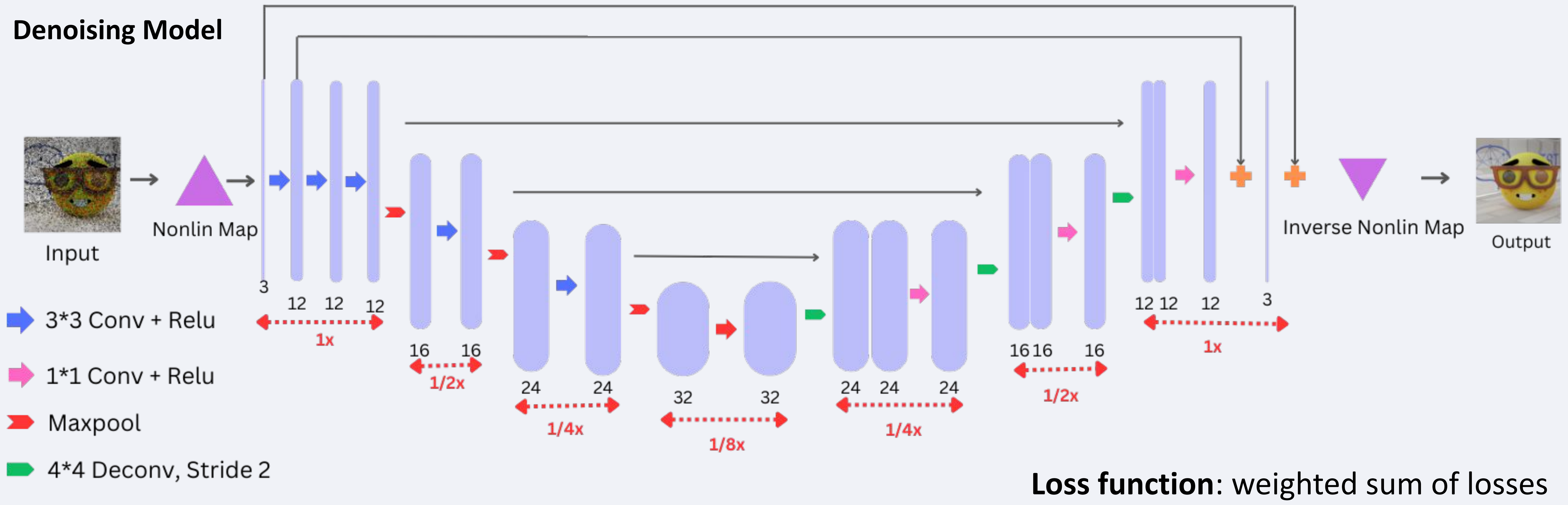
Result Gallery



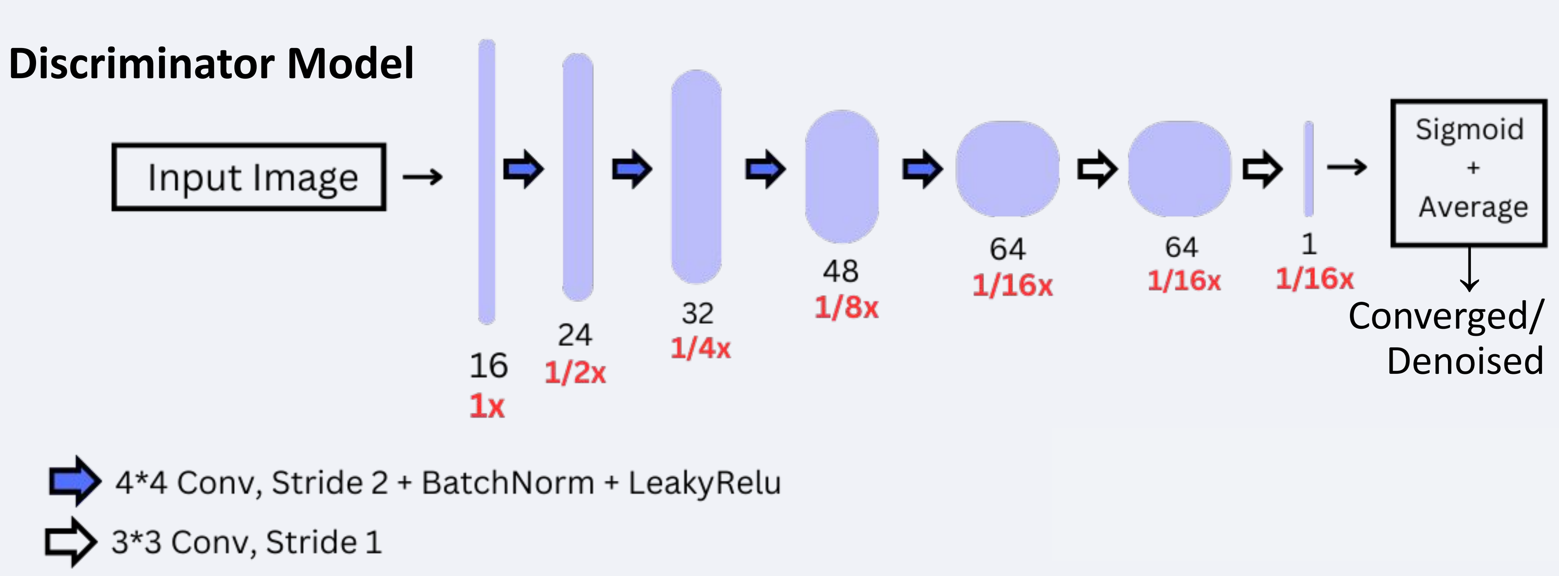
Model Architecture and Training

For the denoising task, we choose a U-Net CNN model with very few parameters that is suitable for real-time inference. A nonlinear mapping is applied to input and output of the model to handle high dynamic range pixels. The model is trained adversarially by trying to cheat a discriminator model, allowing capturing fine visual details.

Denoising Model



Discriminator Model



Loss function: weighted sum of losses

- **L1**: for accurate pixel color
- **L2**: prevents large pixel deviation
- **Perceptual**: evaluated with pre-trained VGG, mimics human visual perception
- **Adversarial**: evaluated using a discriminator, encourages fine details
- **Drift correction**: large loss between averaged pixel values, reduce color drift caused by adversarial loss

Training data: 129 scenes from our rendering engine

- Each scene contains independent renders with power-of-2 sample counts until convergence
- Image with arbitrary sample count can be synthesized with a weighted sum of renders