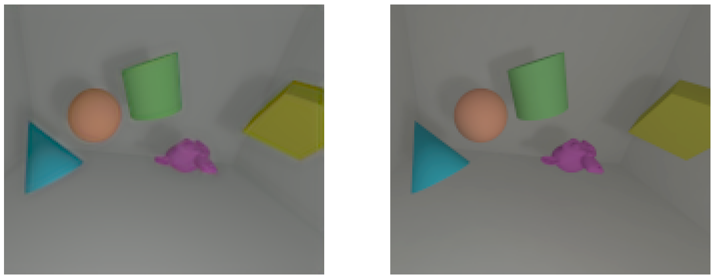
Fast Screen Space Global Illumination

IFT3150 - Summer 2019 Project

Bowen Peng
Université de Montréal
bo.wen.peng@umontreal.ca

Supervised by
Prof. **Pierre Poulin**
D.I.R.O., Université de Montréal
poulin@iro.umontreal.ca



One of the images is produced from a neural network, the other from path tracing.

Abstract

Rendering algorithms such as path tracing [1] are commonly used in practice to produce photorealistic images from a 3D scene. However, for complex scenes involving a lot of indirect lighting, path tracing is very computationally expensive and often produces a noisy image when the time budget is limited [2]. The current state of the art method for accelerating rendering time uses a deep convolutional neural network to denoise an image which has been sampled at fewer than 10 samples per pixel [3].

The method presented here removes the path tracing step entirely and only use a convolutional neural network to estimate in screen space a visually pleasing photorealistic image using information obtained from the g-buffer of a rasterization rendering engine. This method is similar to deferred shading, but instead of a human built shader, we use a machine learning algorithm that learns to reproduce global illumination effects seen in path tracing. This method aims to reproduce the quality of path traced images in real-time for conventional rendering pipelines, and can also be used in parallel with path tracing and denoising techniques as the image generated by the network can serve as an initial estimation which can be refined by more accurate methods.

1. Introduction

The main goal of global illumination (GI) in computer graphics is to produce photorealistic images by simulating physical light phenomenon in a 3D scene. Compared to direct illumination, three visual effects are the most prominent. In direct illumination, shadows are totally black due to the absence of light when there are no diffuse interreflections. Furthermore, shadow edges are often too sharp and well defined (e.g. when using shadow maps). Finally, diffuse objects do not reflect light onto other objects. These elements distract the viewer and makes it obvious that the image is not photo- realistic.

1.1. Visual Effects of GI

Global illumination by light simulation can be introduced to remedy these problems. By introducing indirect illumination, three effects are immediately evident for the viewer, as seen in Figure 1.

Shadows are not completely black. This is caused by light bouncing around. Any object that is not a perfect black body will reflect some light. (Figure 1.a.)

Shadow edges are soft, due to the non-zero size of the light source (unlike point-source lights commonly seen in traditional raster rendering), the presence of the penumbra and diffuse reflections. (Figure 1.b.)

Some objects might be brighter and their color might bleed onto other objects, due to the previously mentioned diffuse inter-reflections. Some shadows might take on a color tint for the same reasons. (Figure 1.c.)

Other subtle effects of light wave simulation, such as (but not restricted to) subsurface scattering, transmission, polari- zation, fluorescence, doppler effect, interference and chromatic aberration are less noticeable. One exception is caustics, where light is reflected and refracted by a curved object or fluid, creating patterns of light concentration. This effect is mostly noticeable underwater. However, it is important to note that

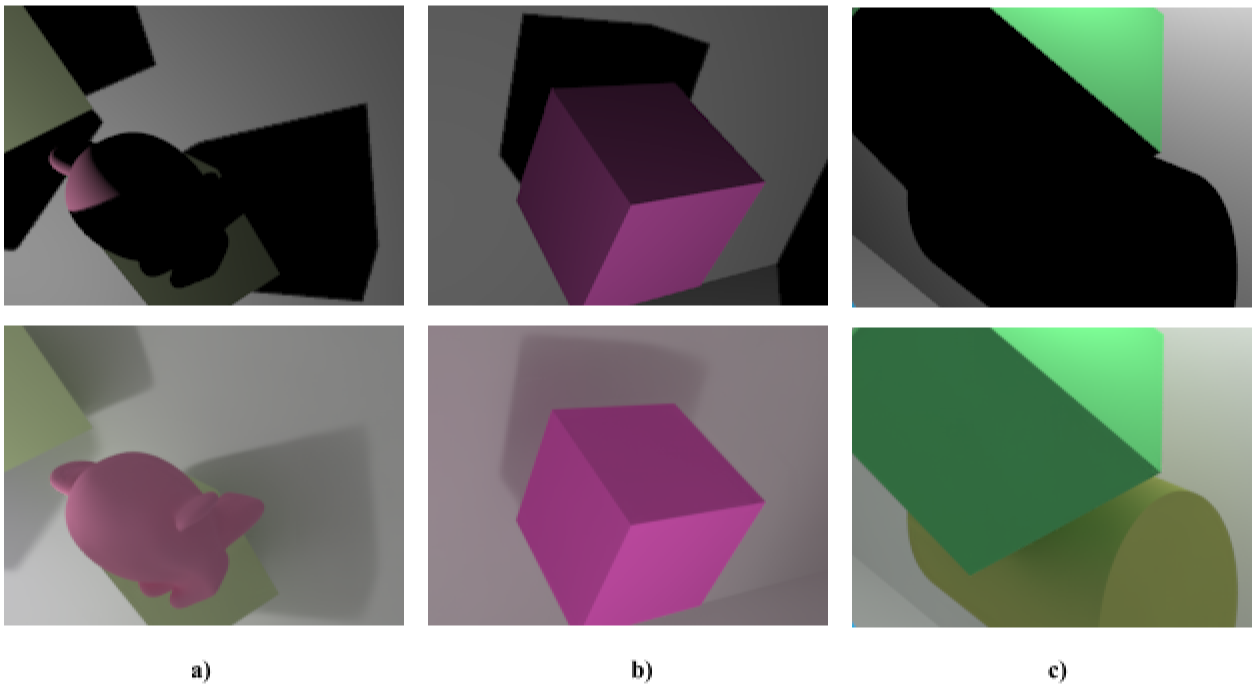


Figure 1: Effects of Global Illumination (bottom) versus Local Illumination (top).

these optical phenomena were also not taken into account in Kajiya's original path tracing rendering equation [1].

Due to the subtlety of other effects and the difficulty of simulating caustics, we will only tackle the three aforementioned effects of GI in this work.

1.2. GI Algorithms

The most common way of realizing GI is by using a path tracing algorithm. In [1], Kajiya defined a rendering equation and proposed a Monte Carlo algorithm, path tracing, as a numerical approximation. It consists of randomly launching rays of light in a scene and integrate the total luminance for each point on the surface of objects. Since it is a Monte Carlo algorithm and light paths are unpredictable, the algorithm might never converge to an acceptable solution within a reasonable amount of time, especially when the scene is complex and light sources sparse. A unconverged image is very noisy. Other algorithms for simulating GI and solving the rendering equation such as lightmaps, photon mapping [4] and radiosity [5] do exist, but they are much less popular in practice, as they come with many downsides. Some of these alternative algorithms are static (i.e. they pre-compute lighting before the actual rendering pass), and cannot handle dynamic objects.

Traditional methods on accelerating path tracing include bidirectional path tracing, adaptive/importance sampling [2], cone tracing, etc. However these methods are relatively complex to implement compared to conventional path tracing and do not always produce visually pleasing results. Some, such as adaptive sampling, tend to introduce bias in the resulting image.

Recently, denoising algorithms were seen to outperform all other methods at accelerating path tracing. Statistical methods [6] performed well until they were surpassed by machine learning techniques employing convolutional neural networks. Instead of modifying the path tracing algorithm itself for faster convergence, these methods attempt at recovering a visually pleasing image from a unconverged, noisy image. Denoising algorithms have their own issues, such as poor temporal consistency (later works mitigate the issue by using recurrent neural networks [3]), something very important for image sequences, interactive media and animations.

Others have looked for alternatives to denoising and have tried to predict GI feature and effects directly from scene data, without any path tracing. Most of these methods operate directly on screen space (on the pixels). While algorithms such as Screen Space Ambient Occlusion (SSAO) [11], Screen Space Directional Occlusion (SSDO) [12], Screen Space Ray Tracing [13], and Screen Space Reflections (SSR) [14] are very fast on GPUs, they only simulate a small subset of effects produced by GI.

Developments in image-to-image translation using convolutional neural networks have allowed researchers to solve most of these issues. Neural networks have been successfully trained to learn a mapping from scene data (diffuse map, normal map, depth map, direct illumination, etc.) to a path traced image.

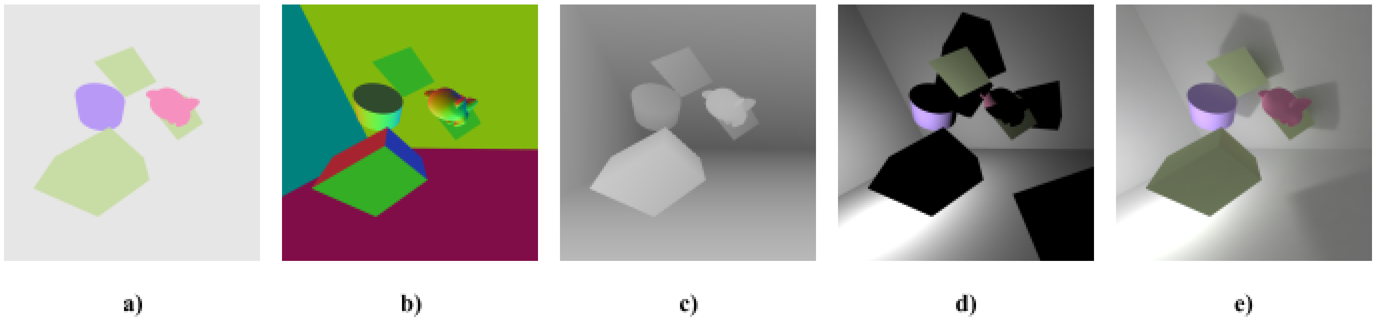


Figure 2: One example from the dataset, in order: **a)** Diffuse Color, **b)** Normal, **c)** Z-Buffer, **d)** Local Illumination and finally, **e)** Path Traced Image

In this work we will explore, improve and discuss the effectiveness, weaknesses and problems that might arise with learning to predict global illumination using scene data with convolutional neural networks.

2. Related Work

In [7], Nalbach et al. have demonstrated the effectiveness of convolutional neural networks at replicating most screen space shading effects, such as Ambient Occlusion, Depth of Field, etc. One of the screen space effects learned was indirect illumination. However, their work was not exhaustive as the neural network was trained to only reproduce a subset of GI effects. Their local illumination image was already path traced (without any light bounce), thus soft shadows and correct BRDF shading were already present. They were only interested in the diffuse light reflection and color bleed component of GI.

In [8], Thomas discusses more in depth about the network architecture needed for a good prediction of GI. They also propose techniques for accelerating voxel cone tracing. However, they also do not explore the possibility of complete image translation from a rasterized render to a path traced render.

As it was not attempted before, we will aim at creating a full image translation algorithm that takes a rasterized render without global illumination information as input and predict its corresponding noise-free path traced image.

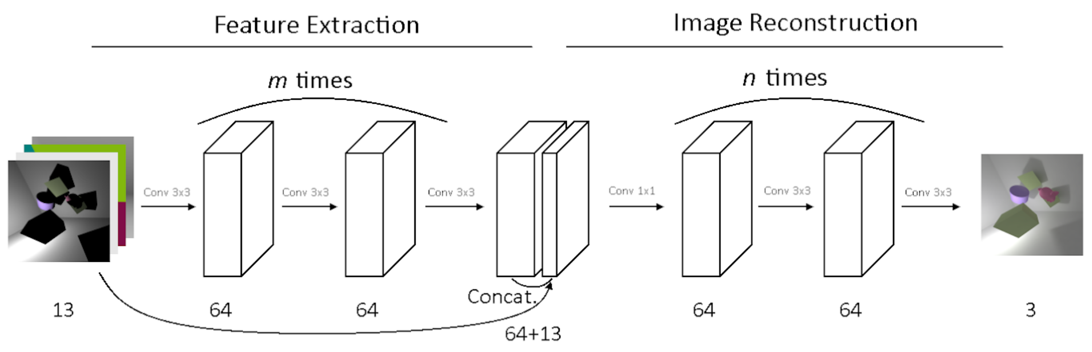
3. Proposed Method

For this problem of image translation, we used a fast and local convolutional network inspired by Dong et al. [9]. Our reasoning behind using a shallow network is that most of the three prominent perceptual visual effects found in GI and discussed earlier are local and affect mostly nearby pixels. Some shadows or color bleed might come from very far, but those edge cases do not justify using a very deep neural network that would be difficult to train and slow to use in practice. If accuracy is more important for the user, the actual simulation of light rays is necessary as it is impossible to accurately predict the total luminance on an object using only screen space information, especially without any prior information about the location of lights themselves.

3.1. Network Architecture

The proposed network consists of two parts. Feature extraction and image reconstruction. All intermediate convolutional layers use 64 filters of size 3×3×64, except the layer immediately preceding image reconstruction. That layer uses 64 filters of size 1×1×(64+13), and is equivalent to a element-wise linear regression operation on the features extracted by earlier layers concatenated with the input pixel buffers. The input convolutional layer uses 64 filters of size 3×3×13 and the last layer uses 3 filters of size 3×3×64. For non-linearity, all intermediate layers use LeakyReLU with alpha of 0.1. Zero padding is used to keep all feature maps at a constant spatial size. An overview of the network architecture is depicted in Figure 3. Our final network had m=3 and n=2.

Figure 3: Network Architecture



Thus, this network takes as input pixel buffers from a rasterization engine (diffuse buffer, z-buffer, normals, etc.) and predicts its corresponding image generated by a path tracing engine. One example is shown in Figure 2.

3.2. Training

For the objective function, we minimize the mean squared error $\frac{1}{2}||y - f(x)||^2$ over the training set. Training is carried out using mini-batch gradient descent (mini-batches of 32) with backpropagation (learning rate of 0.1). We also set the momentum parameter at 0.9 and L_2 weight decay with a factor of 0.001. For better stability during training, we clip the gradient at +/- 0.01.

3.3. Dataset Generation

We used a custom dataset generated from a python script in Blender. For each pair of images, random objects and lights are scattered around in a room, and the camera position and orientation is also randomized. The random objects are composed of cubes, cones, cylinders, spheres and the commonly used suzanne monkey. Then, using two different engines, (rasterizer and path tracer) we generate the rasterized g-buffer data with the path traced ground truth, as seen in Figure 2. For the rasterizer engine, only diffuse materials and direct illumination with hard shadow maps are used. Glossy materials were removed from the dataset after testing due to poor convergence of the neural network when trying to predict reflections. We suspect this might be caused by our objective function. Since it minimizes the mean squared error, the network will try to predict the average of all possible glossy reflections when objects in the reflections are not seen elsewhere in the scene. This will generate a very blurry "average" reflection.

4. Analysis

Our shallow network performs well for recovering most soft shadows and predicting indirect illumination as illustrated in Figure 5. However, our predictor fails in the rare case where

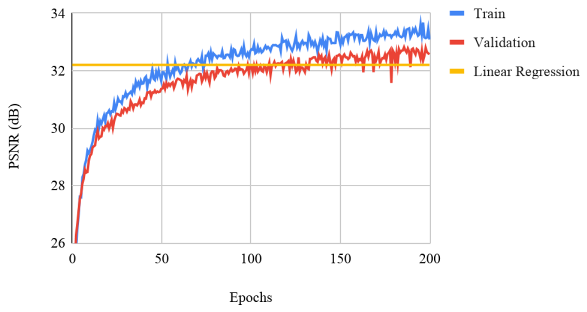


Figure 4: PSNR During Training

the shadows are extremely soft or the diffuse reflection is very strong. Since these cases are very rare in the training dataset, the network must have failed to learn. Furthermore, since the predictor does not know the positions of lights, it cannot always accurately guess how soft a shadow needs to be. Most of the images produced by the predictor have the same shadow softness, while it varies a lot in the ground truths. We also see some artifacts in the output, especially on object boundaries. This is due to the large learning rate used and early stopping of the training to prevent overfitting. Training for a longer time with a lower learning rate will reduce visible artifacts. Finally, since this is a screen space algorithm, it cannot predict effects not present on screen. When a coloured object is near a light, it should tint the scene with its color. However if this object is not present on screen, the predictor will fail to produce the effect, as seen in Figure 6.

Deeper networks are better at predicting GI as a larger receptive field helps them to "see" more of the scene. This in turn allows the predictor to learn the intricacies of indirect illumination with a larger view of the scene. However, since deeper networks are slower and harder to train, their disadvantages overshadow the benefits. A deep enough network will be so slow that other methods such as small sample path tracing w/ denoising will outperform it given the same demanded quality and time budget.

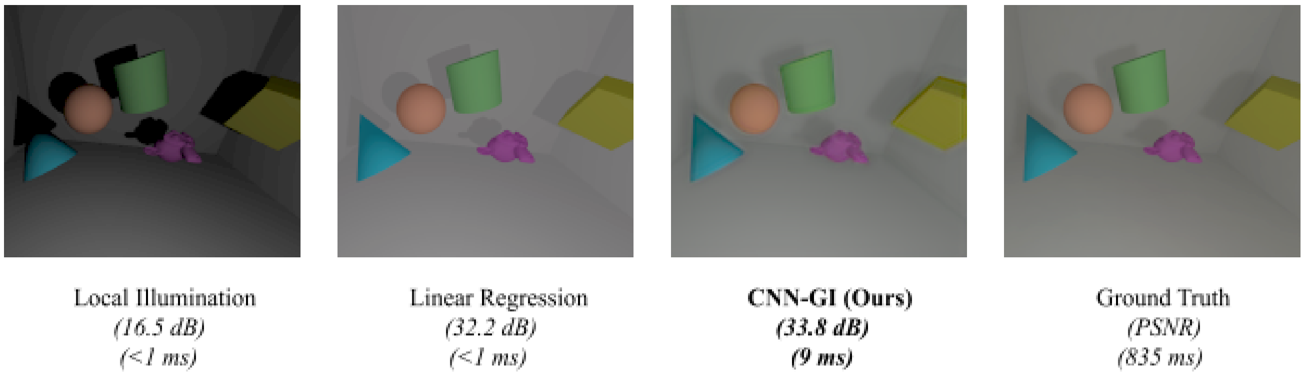
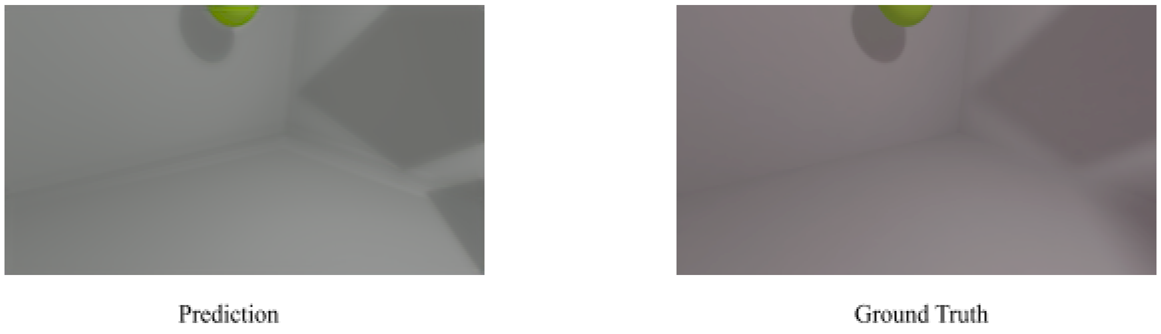


Figure 5: Results Comparison

Figure 6: Failure case, the red object tinting the scene is not visible in the camera.



5. Experimental Results

Using images of size 240×135, the neural network was trained for 300 epochs, with mini-batches of 32. The dataset used contained 1000 images, split between training (900) and validation (100) sets. The network used has a total of 238k parameters. The final PSNR obtained on the validation set is 33.8 dB. Figure 4 confirms the improvement of PSNR over time during training. No overfitting is visible as the validation loss does not increase. Visual inspection of the output confirms that the network successfully learned to produce effects of GI (Figure 5). The total training time on Google Colab using a NVIDIA Tesla T4 GPU was 6 hours. Inference time for our neural network architecture is 9ms, allowing real-time use, compared to 835ms for the path tracer.

For comparison, we have also tested simple linear regression (42 parameters) directly on the input buffers as a baseline for this problem. As expected, linear regression cannot predict soft shadows or inter-reflections, but it can add light to the dark shadows, giving an ambient lighting effect, but the resulting image looks flat (Figure 5). For completeness, the PSNR of local illumination is 16.5 dB and linear regression is 32.2 dB. However, it is important to keep in mind that PSNR is not a good and reliable indicator of visual quality or GI accuracy. Furthermore, linear regression has an analytical solution and does not suffer from problems such as local minima seen in neural networks.

More examples can be found in the Appendix at the end.

6. Future Work

We have merely demonstrated the possibility of predicting GI from rasterized images, further improvements are possible with more research. We have listed here a few ideas.

Generative Adversarial Networks [10] can be used to improve the visual quality of the predicted image. Using MSE as a loss is not optimal and often leads to blurry results. However, it should be noted that GANs are notoriously hard to train.

The predictor could be given a full 360 degree view of the scene. This will fix the failure case previously discussed.

Giving information about the position of lights will help the predictor produce more accurate soft shadows. Currently the predictor has a hard time determining how soft the shadows should be.

Different network architectures might be even better suited to this task. Some might even be currently unknown. The search space is endless!

7. Conclusion

In this work, we have presented one method of quickly generating visually pleasing images that are similar to the global illumination produced by path tracing algorithms. We train a convolutional neural networks to solve this task, and it performs relatively well. Since the network is shallow and fast, we believe this method can be applied to interactive 3D software for real time GI prediction. Furthermore, since 3D scenes might vary widely, each 3D scene could have a corresponding pre-trained neural network. This would allow real time dynamic global illumination without the performance penalty of path tracers.

References

[1] KAJIYA J.: The Rendering Equation, 1986

[2] KULLA C., FAJARDO M.: Importance Sampling Techniques for Path Tracing in Participating Media, 2012

[3] CHAITANYA C., KAPLANYAN A., ET AL.: Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder, 2017

[4] JENSEN H.: Global Illumination using Photon Maps, 1996

[5] GORAL C., TORRANCE K., ET AL.: Modeling the Interaction of Light Between Diffuse Surfaces, 1984

[6] LI T., WU Y., ET AL.: SURE-based Optimization for Adaptive Sampling and Reconstruction, 2012

[7] NALBACH O., ARABADZHIYSKA E., ET AL.: Deep Shading: Convolutional Neural Networks for Screen Space Shading, 2017

[8] THOMAS M., FORBES A.: Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Networks, 2018

[9] DONG C., LOY C., ET AL.: Accelerating the Super-Resolution Convolutional Neural Network, 2016

[10] GOODFELLOW I., POUGET-ABADIE J., ET AL.: Generative Adversarial Nets, 2014

[11] MITTRING M.: Finding Next Gen - CryEngine 2, 2007

[12] RITSCHEL T., GROSCH T., ET AL.: Approximating Dynamic Global Illumination in Image Space, 2009

[13] MCGUIRE M., MARA M.: Efficient GPU Screen-Space Ray Tracing, 2014

[14] JOHNSSON M.: Approximating ray traced reflections using screenspace data, 2012

Appendix

All examples ordered from top to bottom in the order:
Local Illumination, Linear Regression, CNN, Ground Truth