Deep Global Illumination with Shaders

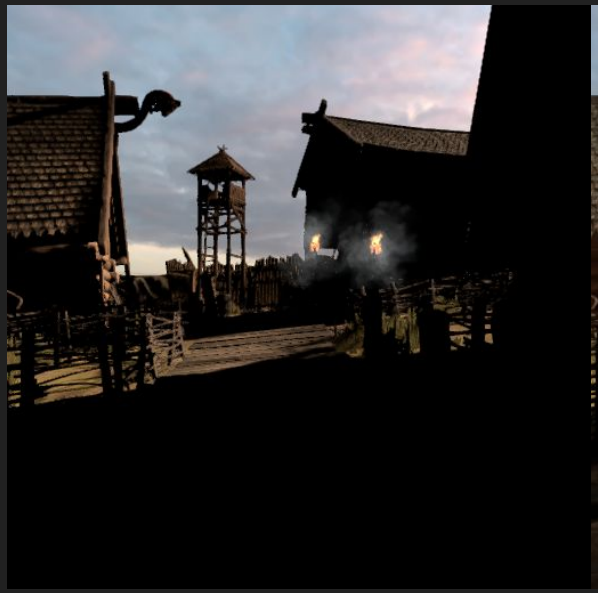
Samuel Hatin 301394799

Presentation Layout

- Introduction to global illumination techniques in games
- Examples of classical algorithms
- Neural network and global illumination
- Deep global illumination with shaders
- Issues with problem setting
- Conclusion

Goal

Use deep neural networks to approximate global illumination in real time for games

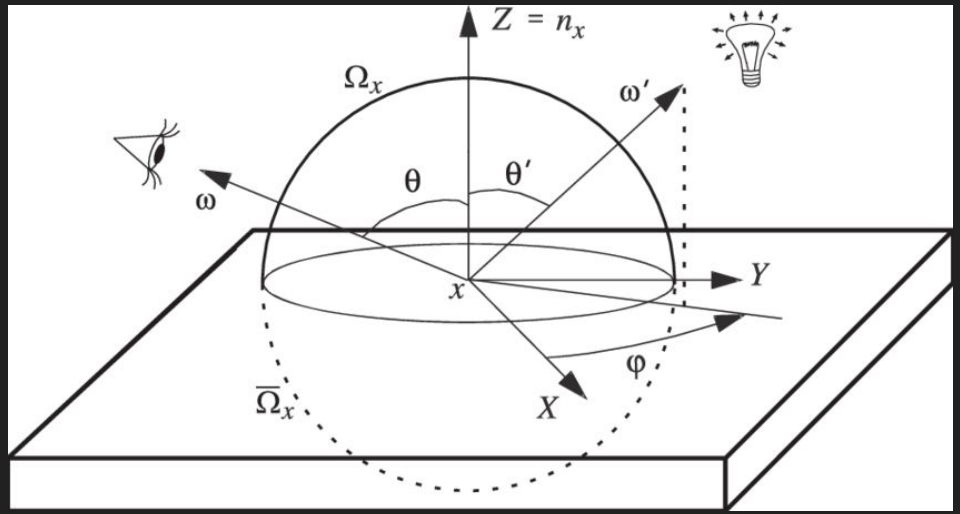


Direct lighting

Direct + global illumination

Global Illumination

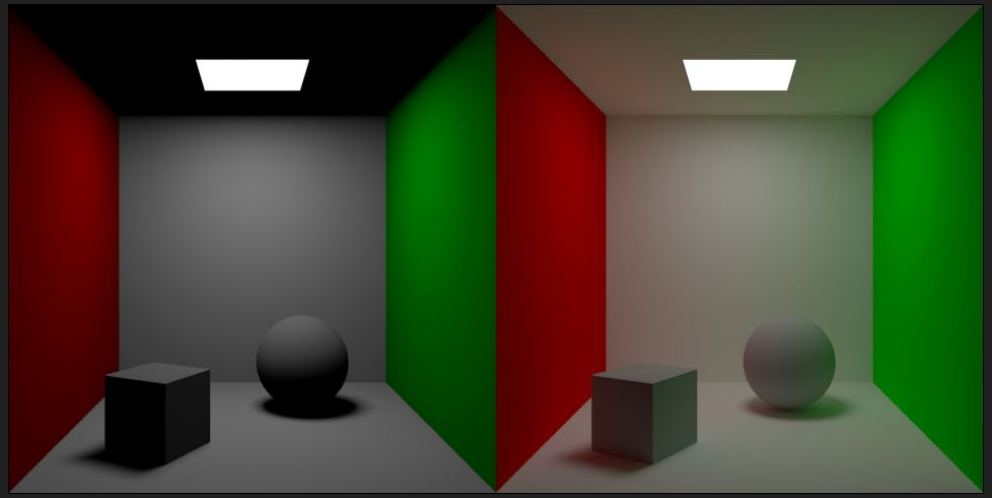
Rendering equation: compute output light at a point by taking into account all the incoming light



Global Illumination

Indirect light bounces are expensive to compute in real time

Good results require millions of rays



Classical Methods for Games

- Precompute static lighting

- Approximate dynamic lighting by using multiple static samples

Examples:

- Lightmaps
- Light probes
- Dynamic objects
- Ambient occlusion and PRT
- VXGI
- Real time ray tracing

Lightmapping

- Project static geometry on a texture map and store lighting information at each point
- Low frequency details only

Many games use lightmaps as they are fast and provide a good approximation of lighting.

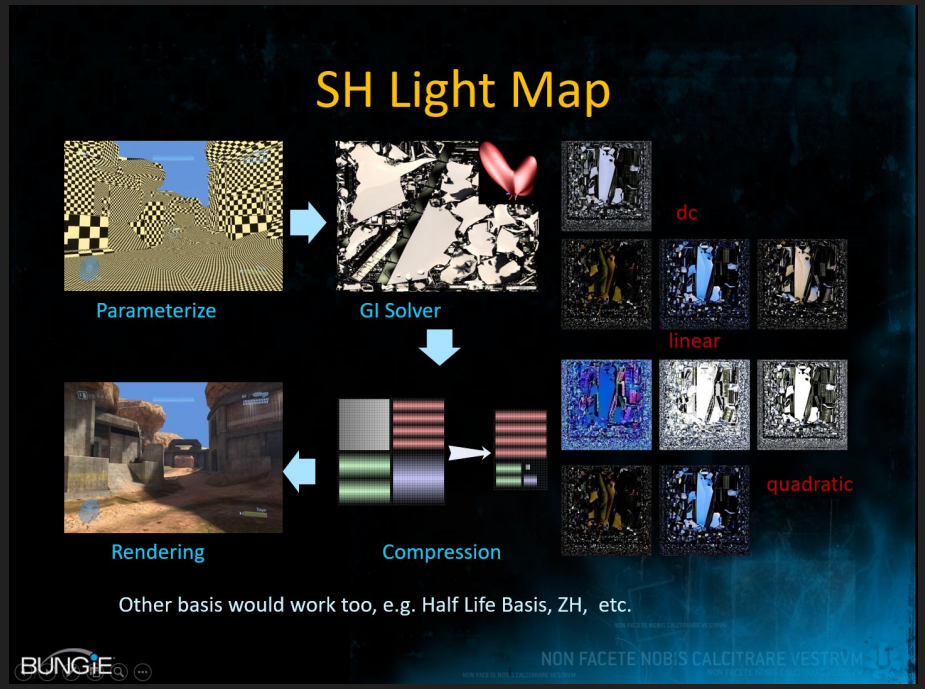
Light probes

- Capture static lighting at a point in space
- Low frequency details only
- Does not have to be on a surface

- Dense grids of light probes are used

Lightmap example

Lightmaps are precomputed by sending millions of rays into the scene and storing lighting information on a texture



Source: Lighting and Material of Halo 3 GDC 2008

Lighting information

How to store lighting information to be reusable at runtime?

- For each lightmap sample, need to store a function that represents the light color in different directions

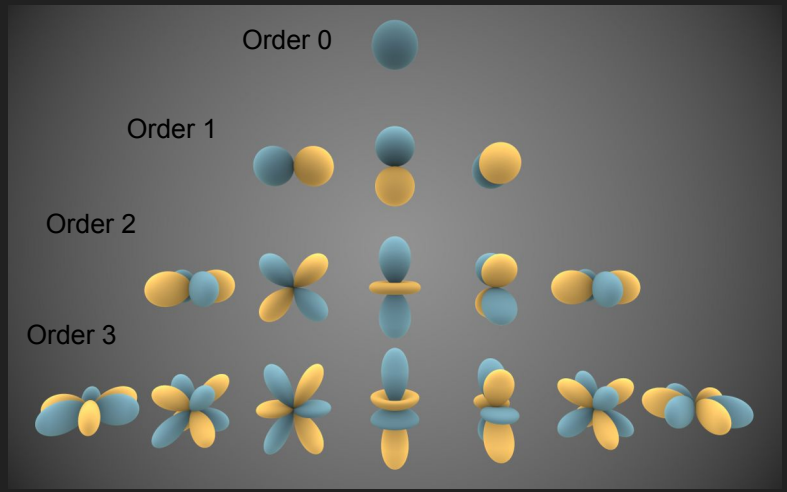
- Optional: store dominant light direction for shadows

Lighting information

Which function to use to represent lighting in all directions?

- Commonly used: Spherical harmonic basis functions of varying order.

  - Fast to evaluate on GPU, uses many dot products
  - Easy to combine with other techniques
  - Low frequency information only

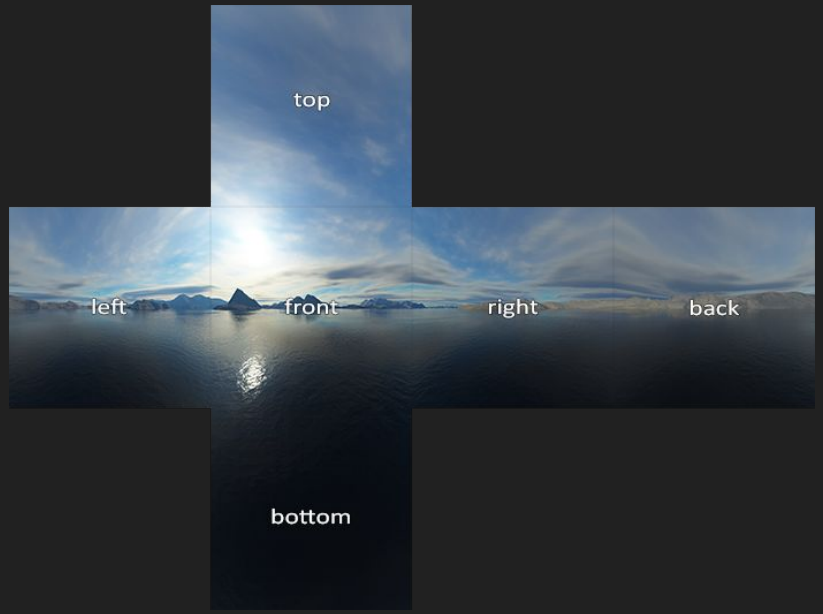


High frequency lighting information

Usually represented with cubemaps

- Quick and easy way to get high frequency detail in reflections

- Can become memory heavy and lack realism



What about dynamic objects?

All previous methods are for static objects.

Use interpolation between lightmap samples coming from the lightmap and light probes.

Cubemaps for different scene sections

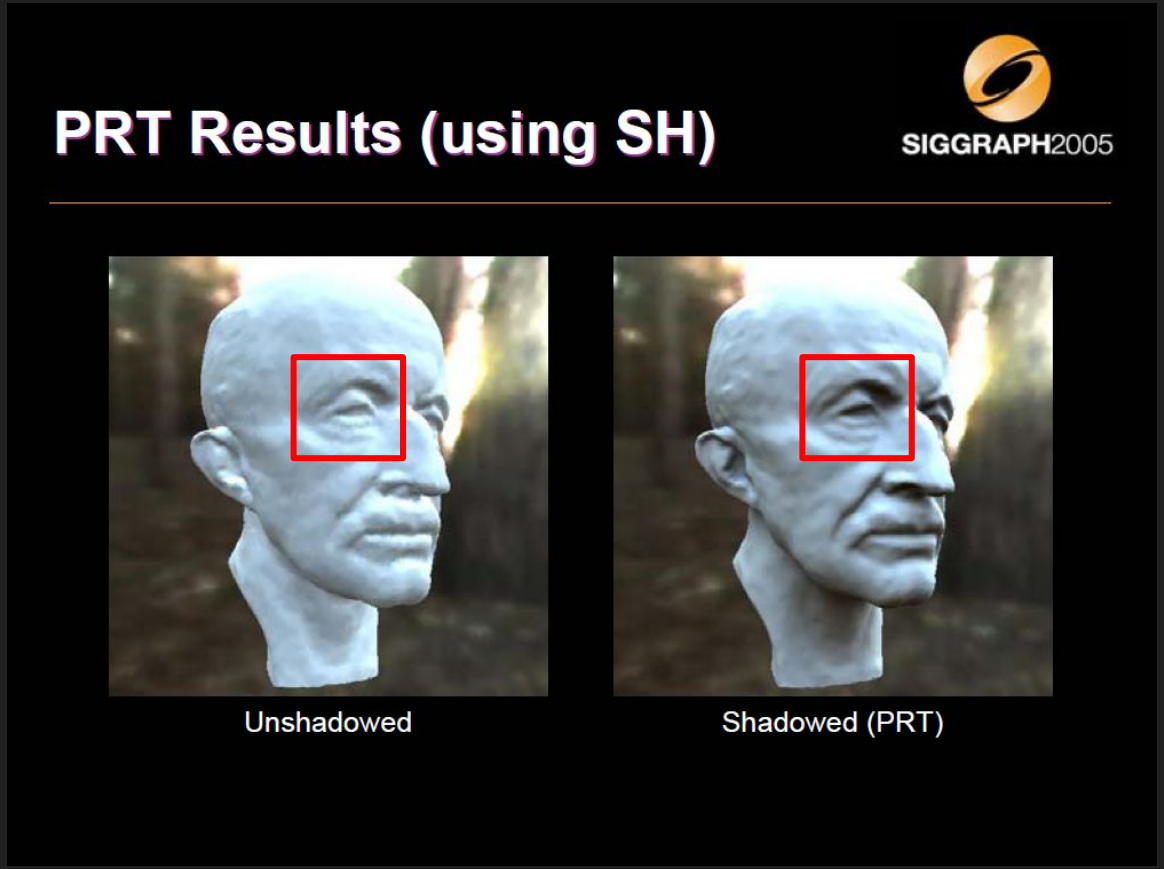
Sampling Lightmap for Dynamic Objects

Per object lighting

Objects can cast shadows on themselves. This is hard to compute at runtime. Solutions:

- Store “average” information as an ambient occlusion map
- Store radiance transfer at each vertex.

Precomputed Radiance Transfer Example (Sloan et al.)



Problems with classical techniques

- Static: effects such as daylight cycles, environment destruction and dynamic environments are hard to represent with static information

- Approximation: quality of lighting on dynamic objects depends on density of samples.

Voxel Based Global Illumination

- This recent technique attempts at solving the static nature of precomputed lighting by using cone tracing

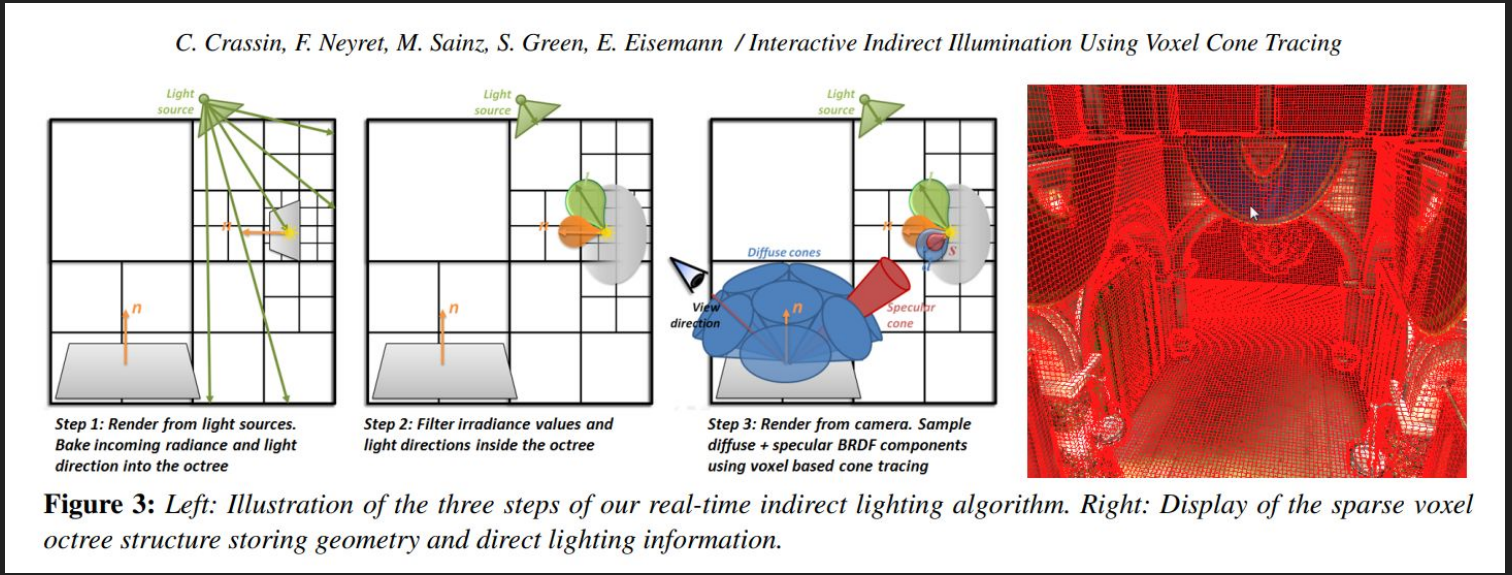


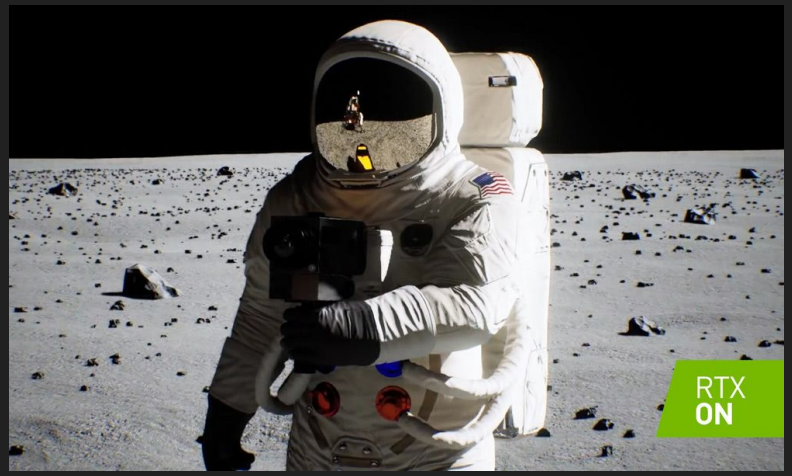
Figure 3: *Left: Illustration of the three steps of our real-time indirect lighting algorithm. Right: Display of the sparse voxel octree structure storing geometry and direct lighting information.*

Voxel Based Global Illumination Problems

- Memory heavy, need to store the per voxel lighting information
- Grid needs to be fine to obtain high quality results
- Light bleeding: thin surfaces may have the same voxel for 2 different light settings.

Real Time Ray Tracing

- Best global illumination results currently available
- Very costly to compute, requires high performance hardware



Using Machine Learning for GI

- Generative adversarial networks have been used to do image inpainting, style transfer and translation.

- Could we use them to apply global illumination to a real time renderer?

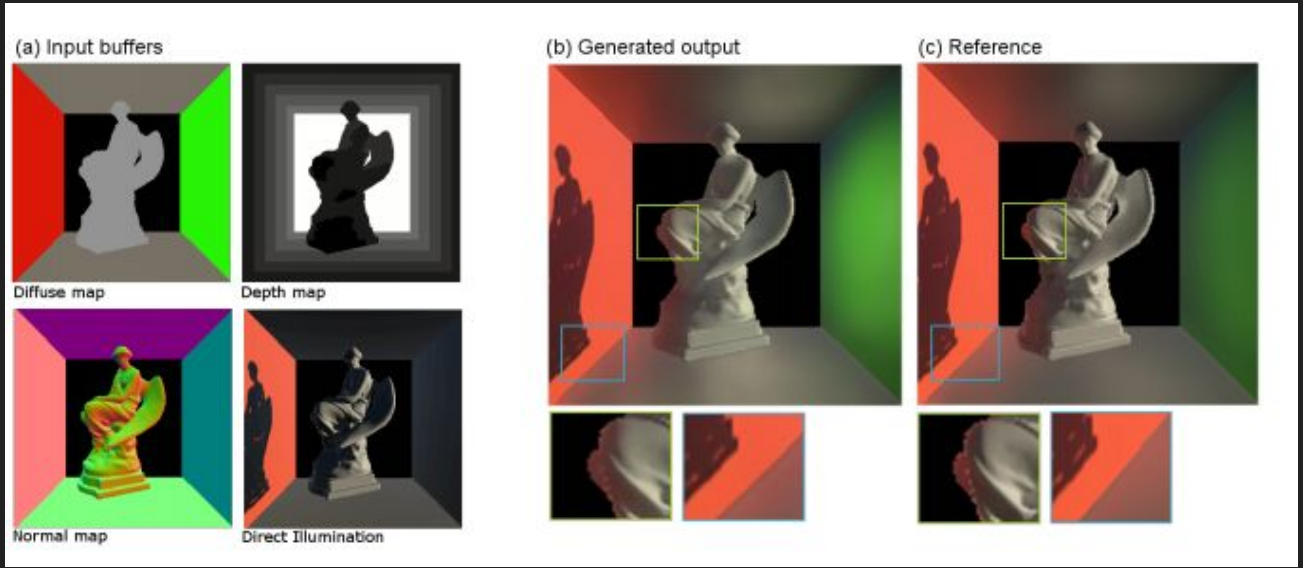
Problem Setting

The goal is to use as much information as possible that is available on the GPU

- Input: Geometry buffer as images (direct, diffuse, normal, depth and others)
- Output: final render ready to be displayed

Previous Work

- Approximating Dynamic Global Illumination with GANs : Manu Mathew Thomas, Angus G. Forbes 2017



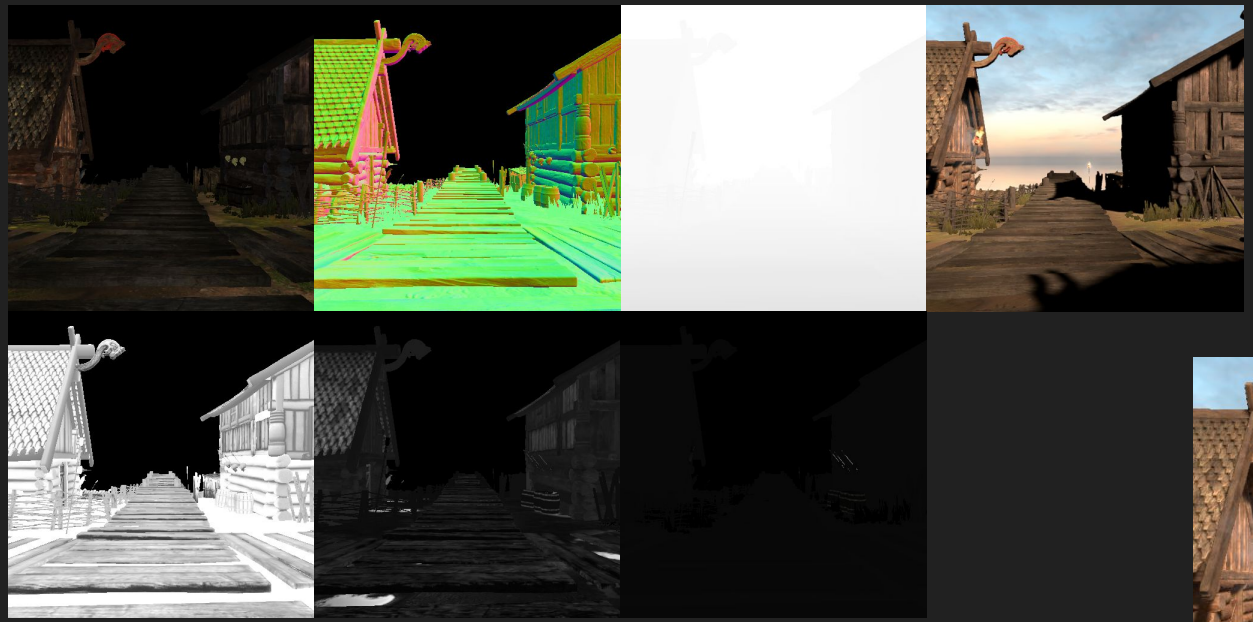
Previous Work

- Input: Diffuse, depth, normal and direct illumination
- Output: Final render
- According to the authors the network successfully learns a GI approximation
- Problems: No ambient occlusion, 1 network per scene

Constraints

- Why train a network for each scene? Training a network to generate GI for any scene would require a much larger amount of data, larger model and longer training time.
- Why use screen-space information? Readily available on the GPU in the form of images

Including Shaders in Approximated GI



Goals

- One set of weights for each scene
- Faster than VXGI or real time raytracing
- Physically accurate

Generating Data

- Unity Renderer and Viking sample scene
- G-buffer capture tool
- Voxel based global illumination plugin (1)
- Automatic capture from gameplay

1: https://github.com/sonicether/SEGI

Dataset

1000 buffer samples containing:

- Diffuse
- Depth
- Normal
- Specular
- Roughness
- Ambient Occlusion
- Direct
- VXGI ground truth

Original Network Structure (cGAN)

Discriminator:

- patchGAN
- Input: G-buffer + illuminated render

Generator:

- UNet variant (64 filters per convolution)
- Input: G-buffer

Training

- Modified the model to have 128 filters per convolution

- 80/20 train/test split
- 48 hours of training on P100 GPU
- Evaluation time : 4 ms on 256 x 256 images (60 fps uses 16 ms per frame)

Problems

- Network fails to learn high frequency detail
- Image size
- Evaluation speed

Fundamental Flaws in Problem Setting

There are too many things to do for this neural network:

- Incoming light approximation (incomplete information)
- Rendering
- Blending with directly lit image

Incoming light approximation

We give the neural network screen space information (normals, albedo, etc)

It needs to learn the parameters of all the light sources in the scene (world space)

Without knowing offscreen geometry and light positions this is very difficult

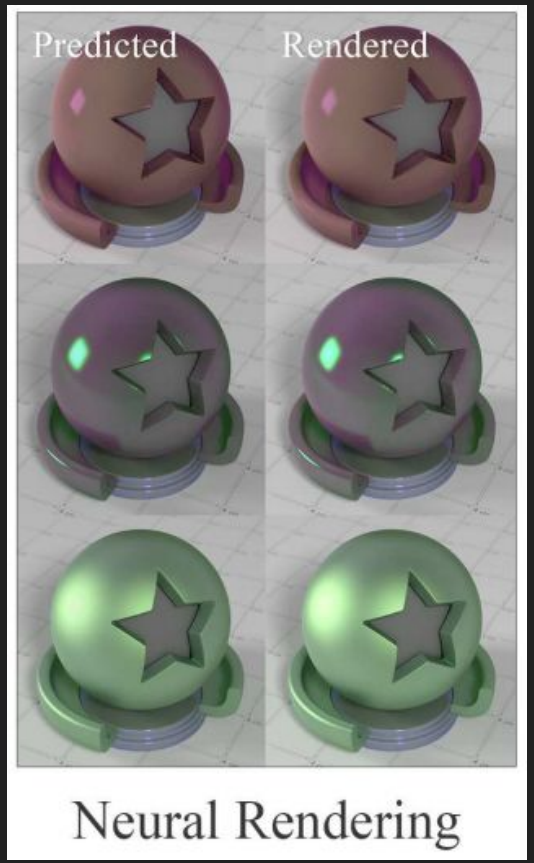
Rendering

Since we are using physically based rendering for the direct lighting, the network needs to learn how to do the same with regions that receive indirect light.

- Neural rendering is a complex problem by itself
- In fixed scene and lighting, it is possible to get a good renderer (Gaussian Material Synthesis)

Neural Rendering in Gaussian Material Synthesis

- Fixed light position
- Fixed scene geometry
- Fixed camera position



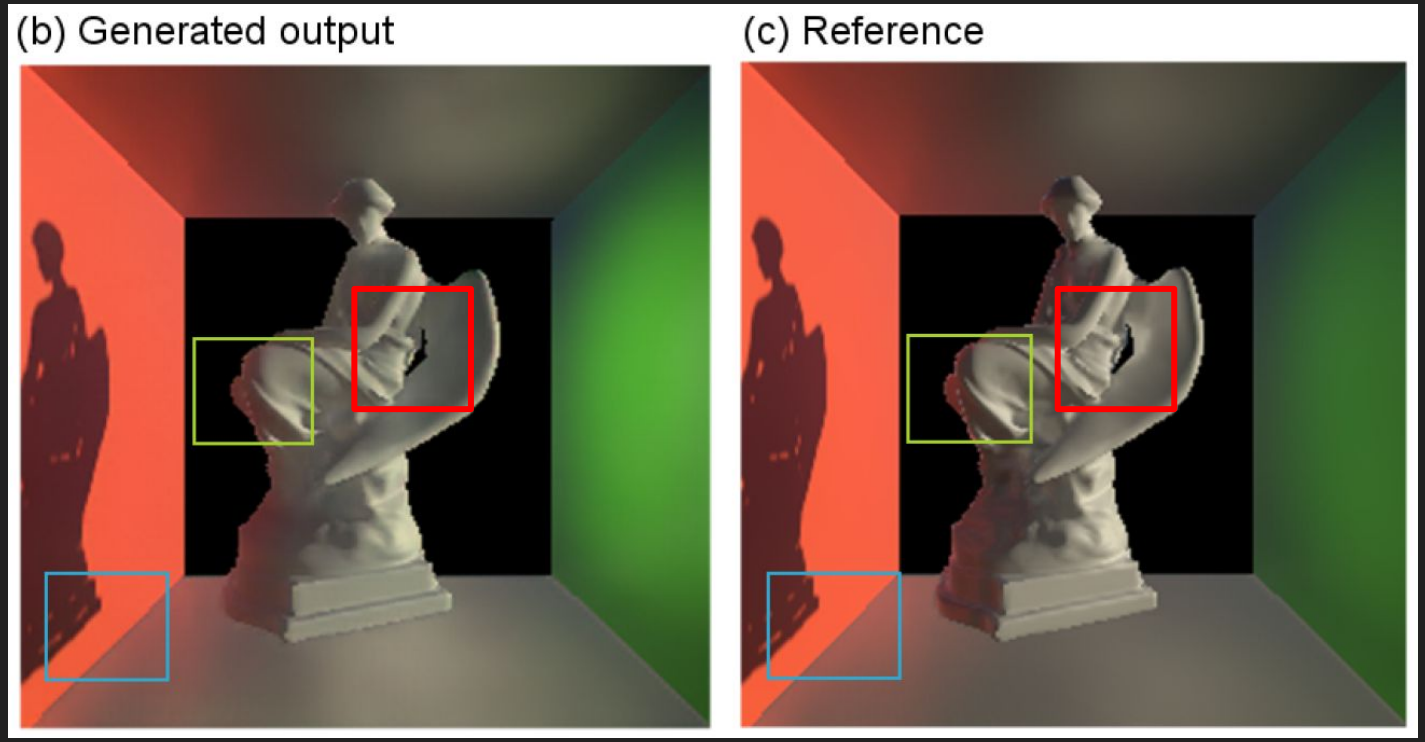
Problems with Deep Global Illumination

- The original paper on DGI only used diffuse colors and a simple shader.
- It failed to learn ambient occlusion.

Ambient occlusion should be present if the network learned how to do proper GI

The results look convincing because of the lack of high frequency details

Problems with Deep Global Illumination



Problems with Deep Global Illumination

Problems with Deep Global Illumination

Problems with Deep Global Illumination

Information Required for DGI

- World space geometry data
- Light source data
- Material model parameters for each surface

Network size would increase significantly and become too slow

Can Deep Learning be used in Rendering?

Neural networks can be used to speed up complex calculations

- Path Guiding Using Machine Learning (Jaroslav Křivánek Siggraph 2018)
- Deep Learning for Light Transport Simulation (Jan Novak Siggraph 2018)
- Neural Network Ambient Occlusion (Holden et al. Siggraph 2016)

Conclusion

Deep Learning is not well suited for replacing the lighting pipeline in games

It is expensive to train, slow to evaluate and inaccurate (lack of information)

Existing techniques such as static approximations, VXGI and real time ray tracing provide much more accurate results with less computations

Project Conclusion

- Original goal was unachievable and problem was ill-posed
- Learned a lot about real time rendering and GI techniques
- Learning experience on why selecting the right problem is important

References

- Deep Global Illumination (Thomas et al. 2018)
- Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments (Sloan et al. 2005)
- Stupid Spherical Harmonics Tricks (Peter-Pike Sloan 2008)
- Lighting and Material of Halo 3 (Hao Chen 2008)
- Gaussian Material Synthesis (Zsolnai-Feher et al. 2018)

Screen-Space Ambient Occlusion

Use depth map to estimate ambient occlusion