Reconstruction of Radiance Field with Neural network for real-time rendering

Ruohan Yang,

Alibaba Group, Shanghai,201199, China
e-mail: alencell@163.com

Abstract. Fast global illumination methods has been one of the most exiting area of rendering because it could solve many industrial problems of achieving balance between realistic rendering and stable frame rate. So this study employs a neural network to learn a radiance field from the distribution of a particular group of objects by reconstructing a 3D radiance field using spherical harmonics and neural networks. The network learned the estimated radiance distribution in space by voxel data and creating a hash grid containing information from spherical harmonics. This research also provided a new simplified light transition function that emulated the behavior of light in a specific scene, as global illumination could be broken down into a sequence of the light transport progress. It may be possible to see how the light changes throughout a 3D environment by computing the global GI using this multiple-ray bounce model.

Keywords: Deep learning, Rendering, Global illumination

1. Introduction

In the Graphics field, particularly in Game rendering, the former research abstracted the model that light obeys the classical Mechanics law when the object is far larger than the size of atoms, So the radiance distribution was determined in every point of a 3D space once the objects have been determined. If the field's status was given in the beginning, that means the status of time t could is calculated with only one possible outcome.

The intermediate light the objects are receiving could be viewed as a reflection from all of the other objects, all the way retrieving back to the emitter. This is also fundamental to Path tracing because the optical path is reversible. Considering all the objects reflect radiance just like a small emitter, The light transport will finally achieve a stable status following the energy conservation rule.

This can be viewed as a function that maps this 3D scene into a radiance distribution. The core of the research to simulate Global illumination lies in how to calculate this progress using a neural network which itself could be a universal function approximation.

The input of the neural network should be the attributes of the scene we want to render. To measure the irradiance at each point, the vertices' position and the normal, the light source, and the color should be considered.

The paper served the purpose of a methods of accelerating calculating the global illumination with a pretrained neural network, and take a step further of analysing the light transport process.Using spherical harmonics to represent the radiance field, the light transition function gives the neural network the possibility to perform this linear transform.

2. Related work

Realistic rendering is a very complex process involving calculating a point's irradiance with integration over the hemisphere, in this process, it must take into account the surface's characteristics [1], such as the reflection of other objects, its own shadow as well as shadows produced by other objects, and the properties of the surface with the BRDF equation:

$$L_o(p,\vec{w}_o) = L_e(p,\vec{w}_o) + \int_{\Omega} L_o(p',-\vec{w}_o) f(p,\vec{w}_i \rightarrow \vec{w}_o) V(p,p') \cos\theta_i d\vec{w}_i \tag{1}$$

$\mathrm{L_o}$ is the light intensity coming from direction $\mathrm{w_o}$, which is equal to the point's own emission intensity plus combining all light directions. $\mathrm{f(p,\vec{w}_i \rightarrow \vec{w}_o)}$ is the BRDF function that map the direction of light source to the direction of radiance departure.

There are lots of previous work was done in the rendering field. For offline rendering, Monte Carlo path tracing needs lots of sampling and thus can't be used in real-time. The equation will be discussed later.

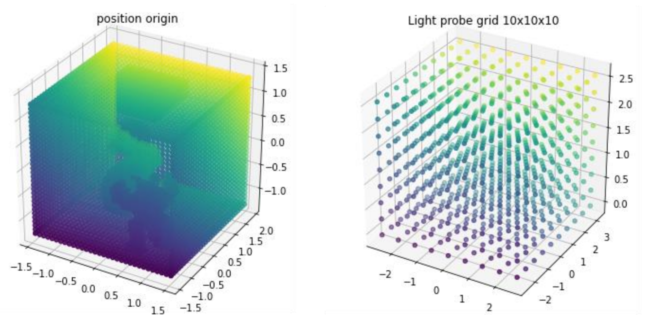
Nerf and Plenoxel only provide the methods to use ground truth images training the network to learn the radiance field and it couldn’t use in real-time when light changes or in any interactive scenario[2][3]. Spherical harmonics can approximate the light function over one single point and have been widely used in the game, but they couldn’t take the dynamic objects into precomputation when apply to the Game Engine [4][5]. Differential rendering was able to reverse the rendering process but for rasterization, it could now only achieve limited results [6]. Real-time Global illumination is still a promising field for exploration.

This paper has proposed a light transition function and converted it employing neural networks to learn the function, which can map the voxel scene to a light field Grid. Additionally, it aims to take a step forward of simulating light’s behavior as well.

3. Voxel Grid

The scene could be interpreted as a bunch of vertices attributes and faces, but can also be the voxel. The former has variable density distribution because the mesh vertices may not spread across space uniformly. And it also needs large memory to store. The latter could be an easy way to have an approximate representation of the scene and be controlled by the voxel grid size [7].

Besides, the voxel could be easily sorted and have a spatial closer distance, which could be used to encode its position (Figure 1 (a)&(b)).



(a) (b)

Figure 1 (a) voxel grid; (b)sparse light probes grid

In the dataset, each voxel has 10 attributes, they are

$$\{V^{N\times N\times N}\} = \{\boldsymbol{p},\boldsymbol{kd},\boldsymbol{e},\boldsymbol{id}\} = \{x_p, y_p, z_p, k_d r_p, k_d g_p, k_d b_p, er_p, eg_p, eb_p, Hashid\} \tag{2}$$

V represents the voxel collection and it’s each axis contains N voxels, p is the position of the voxel, $\mathrm{k_d}$ is the color, e is the emission color, and Hashid represents the Hash ID of voxel founded is in the hash grid which will mention later.

The first way to calculate normal is using the cross product of tangent and binormal:

$$\text{tangent} = \boldsymbol{d} = (d_x,0,d_z), \text{binormal} = \boldsymbol{b} = (0,d_y,d_z), \text{normal} = \frac{\boldsymbol{d}\times\boldsymbol{b}}{\|\boldsymbol{d}\times\boldsymbol{b}\|}, \tag{3}$$

In discrete form, x and y should be the surface voxels and Δh is the size of each voxel.
In terms of Hash id, it could be calculated using the later formula:

$$\text{Hashid} = \text{floor}\left(\frac{p_x - T_x}{\Delta h_x}\right) + \text{floor}\left(\frac{p_y - T_y}{\Delta h_y}\right) \cdot N_x + \text{floor}\left(\frac{p_z - T_z}{\Delta h_z}\right) \cdot N_y \cdot N_z \tag{4}$$

N is the entire resolution of the grid, for example, 32 for a 32×32×32 grid, and T is the grid's transform and equals the grid's outside bound's minimum position. The position correlation through the Hash Grid is encoded using a single, distinct code for each point's light information and voxels.

3.1. Light transition function

It is important to examine the fundamental measurement of light before proposing the equation. First, flux, which is the differential of energy over time, is the amount of energy that passes through a region in one unit of time [8]:

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t} \tag{5}$$

The radiant flux is measured in Watts, and Q is measured in Joules.

Another important concept is irradiance, which measures the average power density across an area and is represented by the symbol E. As a result, the equation should be $\mathrm{E} = \Phi/\mathrm{A}$. The irradiance at a point p is thus taking the limit of delta area A, and

$$E(p) = \frac{\mathrm{d}\Phi}{\mathrm{d}A} \tag{6}$$

Also, the radiance exitance from the point at a particular solid angle w, was defined as

$$L = \lim_{\Delta w \to 0} \frac{\Delta E_w(p)}{\Delta w} = \frac{\mathrm{d}\Phi}{\mathrm{d}w\mathrm{d}A_\perp} \tag{7}$$

For an entire sphere, the total solid angle is 4π , d for hemisphere is 2π.

3.2. Path tracing

The path tracing rendering ought to be introduced. The following is an example of the path-tracing equation:

$$L(x'{\rightarrow}x) = L_e(x'{\rightarrow}x) + \int_A f(x''{\rightarrow}x'{\rightarrow}x)L(x''{\rightarrow}x')G(x'{\rightarrow}x)\,dA(x'') \tag{8}$$

A is all of the surfaces in the scene, the radiance of x comes from x' is the sum of emission radiance of the x' itself and the total contribution of all of the surface exited radiance. G is the shadow term representing visibility.

This function is recursive so it could be expanded to:

$$\begin{aligned} L(x'{\rightarrow}x) = L_e(x'{\rightarrow}x) &+ \int_A L_e(x''{\rightarrow}x')f(x''{\rightarrow}x'{\rightarrow}x)G(x''{\rightarrow}x')\,dA(x'') \\ &+ \int_A\int_A L_e(x'''{\rightarrow}x'')f(x'''{\rightarrow}x''{\rightarrow}x')G(x'''{\rightarrow}x'') \\ &\times f(x''{\rightarrow}x'{\rightarrow}x)G(x'''{\rightarrow}x'')dA(x'')dA(x''') \end{aligned} \tag{9}$$

The Light transport path could be seen as the function of light transporting between the two points $\mathrm{x_n}$ and $\mathrm{x_{n-1}}$.Methodology

The light transition was similar to the LPV (Light Propagation Volumes) [9]. Each voxel's direct shading should be calculated in each step and stored in the voxel attributes. The voxels that have been illuminated should then take part in the second step of light transition using the results of the previous step as the intermediate light emitter. If the substance was thought to be uniformly diffuse, then the procedure should be as easy as just containing the multiply and add operator. The Markov chain is followed by the transition process, with the status at time t merely depending on $t-1$.

At the end of the transition, object shading was transmitted several times and obtained an equilibrium.

3.3. Voxel irradiance definition

The irradiance's total inside a voxel could be calculated by the definition in the radiometry [8]. Because the voxel is 3D and contains the total photons been hit inside a volume. This requires performing an integral operation across the height Δh. The total irradiance over a volume denoted as E_V could be seen as the sum of rathe diance of many small sliced areas overlapping each other. the formula is therefore clear:

$$E_V = E_A \Delta h \tag{10}$$

we denoted the irradiance for the area as $\mathrm{E_A}$.

As we also know,

$$E_V = \frac{\mathrm{d}\Phi}{\mathrm{d}V} \tag{11}$$

And this formula could be proved by the following steps:

Let's Φ_A describe the energy flux over an area, Φ_V over the volume, then:

$$\Phi_A = \int_0^A E(p)\mathrm{d}A \tag{12}$$

According to the definition, so similarly, we could write Φ_V ,

$$\Phi_V = \int_0^V E(p)\mathrm{d}V = \int_0^h \int_0^A E(p) \cdot \mathrm{d}A \cdot \mathrm{d}h \tag{13}$$

Insert equation (10) to this and change the $\mathrm{E(p)} \cdot \mathrm{dA}$ term, it obtains:

$$\Phi_V = \int_0^h \mathrm{d}\Phi_\mathrm{A}\, \mathrm{d}h \tag{14}$$

So we get:

$$\Phi_\mathrm{A} = \frac{\mathrm{d}\Phi_V}{\mathrm{d}h} \tag{15}$$

Take the derivative of both sides of the function respect to area A, it could be rewritten as

$$\frac{d\Phi_\mathrm{A}}{dA} dh = \frac{d\Phi_V}{dh\mathrm{d}A}, \tag{16}$$

Which is identical to:

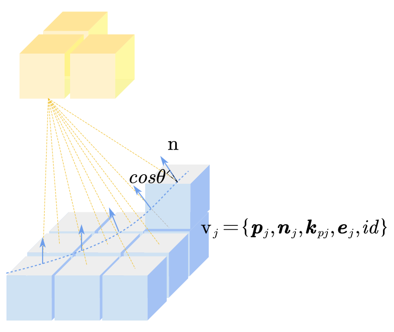
$$E_A dh = \frac{d\Phi_V}{dV} = E_V \tag{17}$$

It is the same just like the equation (13).

Similarly, we rewrite the sum of volume of radiance $\mathrm{L_V}$:

$$L_V = \frac{\Delta E_V}{\Delta w} = \frac{E_V}{4\pi} \tag{18}$$

Especially, when taking into account a diffuse material that consistently reflects light.


n
$cos\theta$
$\mathrm{v}_j = \{\boldsymbol{p}_j, \boldsymbol{n}_j, \boldsymbol{k}_{pj}, \boldsymbol{e}_j, id\}$

$$E_{p_i} = \sum_j^{N^3} \frac{E_{p_j}}{4\pi} |\boldsymbol{n} \cdot \boldsymbol{p}_j| \boldsymbol{k}_{pi}$$

Figure 2 the equation

3.4. Transition

To perform the transition, we apply the basic equation of integral of radiance over a hemisphere. The equation would be like this:

$$E(p,n) = \int_{\Omega} L(p,w)|\cos\theta|\, dw \tag{19}$$

(θ is the angle of the face normal with the light direction.)

So, the quantity of the irradiance in a voxel is:

$$E_V = \int_0^V L_V\, |\cos\theta| dV \cdot S(V) \tag{20}$$

($S(V)$term is the visuability, if needed it should precisely be calculated.)

If using a voxel grid, then the continuous function will be converted to a discreated function.

At the ith voxel, it is approximately the sum over all voxels' radiance exitance:

$$E_{p_i} \approx \sum_{j}^{N^3} L_j\, |\cos\theta| S(i,j) \tag{21}$$

Figure 2 illustrates the equation. To simplify the problem, shadow term was not considered when perform the transition.

The transition step is discussed as follow:

The irradiance at time 0 at voxel i is only the light voxel that emit photons. Then the distribution of the irradiance will be records on the grid:

$$E_{p_i}^0 = E_{p_i}^0 \tag{22}$$

At time 1, the light is transmitted to the surface of the object and being recorded in the grid again, using the irradiance calculated in the first step to giving the result of the second-ray:

$$E_{p_i}^{t=1} = \sum_{j}^{N^3} L_{p_j}\, |\cos\theta| = \sum_{j}^{N^3} L_{p_j} \boldsymbol{k}_{p_i} |\boldsymbol{n}_i \cdot \boldsymbol{p}_j| = \sum_{j}^{N^3} \frac{E_{p_j}^0}{4\pi} \boldsymbol{k}_{p_i} |\boldsymbol{n}_i \cdot \boldsymbol{p}_j| \tag{23}$$

This could evaluate how much irradiance the second ray has contributed. Theoretically, because the energy is continuously absorbed by the surface, the transition process is not seriously a probability transition, but it still is a Markov chain process and obeys the energy conservation rule. It could be written in the following form:

$$E_{p_i}^{t=1} = \sum_{j}^{N^3} E_{p_j}^0\, T(V_i,V_j) \tag{24}$$

We use T as the function of transition, which is the function of color, normal, positions of the voxel i and j,as well as of the probability that light could being transmitted from voxel i to j:

$$T(V_i,V_j) = \beta \boldsymbol{k}_{p_i} |n_x p_a + n_y p_b + n_z p_c| \cdot PDF(V_i,V_j), 0 \leqslant T \leqslant 1 \tag{25}$$

(For the considering other factors of transition process, a coefficient β should be defined to transform from voxel i to j.)

Thus, the time t result can be obtained from the previous calculation of irradiance:

$$E_{p_i}^{t+1} = \sum_{j}^{N^3} E_{p_j}^t\, T(V_i,V_j) = \sum_{j}^{N^3} \left(\sum_{k}^{N^3} E_{p_k}^{t-1} T(V_j,V_k) \right) T(V_i,V_j) = \cdots \ldots \tag{26}$$

And the transition Matrix takes the following form:

$$\{E^t\} = \begin{bmatrix} E^t_{p_1} \\ E^t_{p_2} \\ E^t_{p_3} \\ E^t_{p_4} \\ E^t_{p_5} \end{bmatrix}, \{\boldsymbol{k_d}\} = \begin{bmatrix} \boldsymbol{k}_{d1} \\ \boldsymbol{k}_{d2} \\ \boldsymbol{k}_{d3} \\ \boldsymbol{k}_{d4} \\ \boldsymbol{k}_{d5} \end{bmatrix}, \{\boldsymbol{L}^t\} = \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \\ \boldsymbol{p}_4 \\ \boldsymbol{p}_5 \end{bmatrix}, \{\boldsymbol{N}^t\} = \begin{bmatrix} \boldsymbol{n}_1 \\ \boldsymbol{n}_2 \\ \boldsymbol{n}_3 \\ \boldsymbol{n}_4 \\ \boldsymbol{n}_5 \end{bmatrix} \tag{27}$$

$$\{E^{t+1}\} = \{E^t\} \odot \{\boldsymbol{L}^t\} \cdot \{\boldsymbol{N}^t\}\{\boldsymbol{k_d}\} = \begin{bmatrix} E^t_{p_1}\boldsymbol{p}_1 \\ E^t_{p_2}\boldsymbol{p}_2 \\ E^t_{p_3}\boldsymbol{p}_3 \\ E^t_{p_4}\boldsymbol{p}_4 \\ E^t_{p_5}\boldsymbol{p}_5 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{n}_1 & \boldsymbol{n}_1 & \boldsymbol{n}_1 & \boldsymbol{n}_1 & \boldsymbol{n}_1 \\ \boldsymbol{n}_2 & \boldsymbol{n}_2 & \boldsymbol{n}_2 & \boldsymbol{n}_2 & \boldsymbol{n}_2 \\ \boldsymbol{n}_3 & \boldsymbol{n}_3 & \boldsymbol{n}_3 & \boldsymbol{n}_3 & \boldsymbol{n}_3 \\ \boldsymbol{n}_4 & \boldsymbol{n}_4 & \boldsymbol{n}_4 & \boldsymbol{n}_4 & \boldsymbol{n}_4 \\ \boldsymbol{n}_5 & \boldsymbol{n}_5 & \boldsymbol{n}_5 & \boldsymbol{n}_5 & \boldsymbol{n}_5 \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{k}_{d1} \\ \boldsymbol{k}_{d2} \\ \boldsymbol{k}_{d3} \\ \boldsymbol{k}_{d4} \\ \boldsymbol{k}_{d5} \end{bmatrix} \tag{28}$$

$$= \begin{bmatrix} \sum_j^{N^3} E^0_{p_j}\boldsymbol{p}_j \cdot \boldsymbol{n}_1\boldsymbol{k}_{p_1} \\ \sum_j^{N^3} E^0_{p_j}\boldsymbol{p}_j \cdot \boldsymbol{n}_2\boldsymbol{k}_{p_2} \\ \sum_j^{N^3} E^0_{p_j}\boldsymbol{p}_j \cdot \boldsymbol{n}_3\boldsymbol{k}_{p_3} \\ \sum_j^{N^3} E^0_{p_j}\boldsymbol{p}_j \cdot \boldsymbol{n}_4\boldsymbol{k}_{p_4} \\ \sum_j^{N^3} E^0_{p_j}\boldsymbol{p}_j \cdot \boldsymbol{n}_5\boldsymbol{k}_{p_5} \end{bmatrix} = \begin{bmatrix} E^t_{p_1} \\ E^t_{p_2} \\ E^t_{p_3} \\ E^t_{p_4} \\ E^t_{p_5} \end{bmatrix} \cdot \begin{bmatrix} 0 & T(V_1,V_2) & T(V_1,V_3) & T(V_1,V_4) & T(V_1,V_5) \\ T(V_2,V_1) & 0 & T(V_2,V_3) & T(V_2,V_4) & T(V_2,V_5) \\ T(V_3,V_1) & T(V_3,V_2) & 0 & T(V_3,V_4) & T(V_3,V_5) \\ \vdots & \vdots & \vdots & 0 & T(V_4,V_5) \\ T(V_5,V_1) & \cdots & \cdots & \cdots & 0 \end{bmatrix} = \begin{bmatrix} E^{t+1}_{p_1} \\ E^{t+1}_{p_2} \\ E^{t+1}_{p_3} \\ E^{t+1}_{p_4} \\ E^{t+1}_{p_5} \end{bmatrix} \tag{29}$$

Finally, derived from equation (8), the rendering equation also could be interpreted as the sum of radiance of total bounce of light [8], the irradiance that a voxel final receive should be:

$$E_{V_i} = E^0_{p_i} + E^1_{p_i} + \cdots . E^t_{p_i} \tag{30}$$

4. Experiment Result

This time, instead of using a large network to simulate the linear function, we used a smaller one to encode a voxel grid and then predicted a set of sphere harmonic coefficients to encode the light that the voxel had received.

Only the valid voxels that contain the mesh will be fed to the neural network in order to reduce the amount of data. Due to the arbitrary nature of voxel length, the neural network will include an adjustable pool layer to either limit or expand the data length. The output node of the hash grid should have a total of 27 coefficients, and the other layers of the neural network should all be linear. This is the objective function:

$$Loss = \|\boldsymbol{Y}_{model} - \boldsymbol{Y}_{data}\|_2^2 + \lambda \cdot \|\boldsymbol{W}\|_2^2 \tag{31}$$

The final element, L2 norm, is included in the objective function because it has the potential to significantly reduce overfitting.

All attributes were standardized to a range of [-1,1] before being fed into the network and the activation function was replaced to tanh to predicted negative number. Noteworthy, the Rasnet and MLP networks could fit the distribution well, but CNN could not unless extra two linear layers were used to reconnect it [10]. Figure 3 below illustrates the network's fundamental structure, Figure 4 illustrates the significant precision difference between the two types of networks.

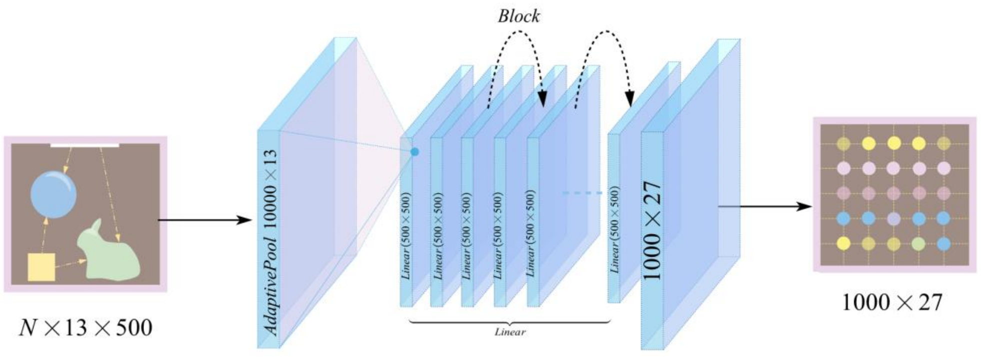


Figure 3 the network's fundamental structure

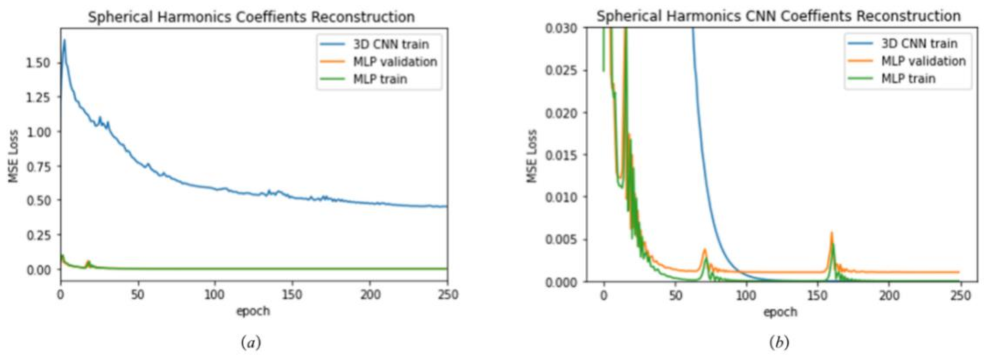
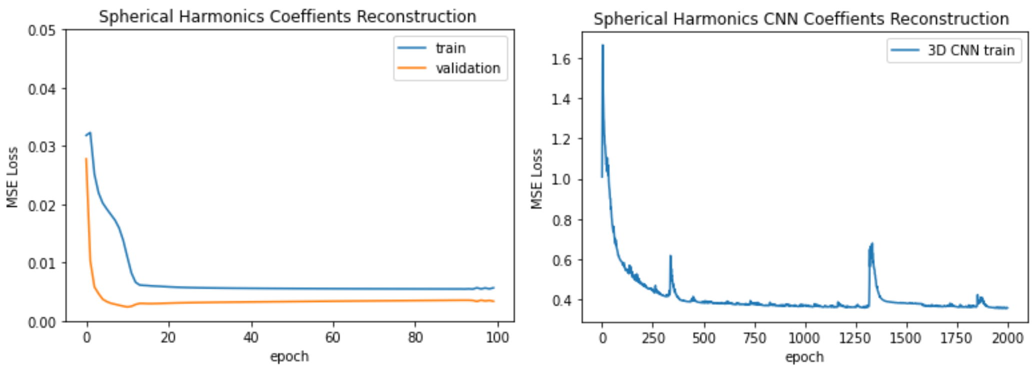


Figure 4. (a) a 3D convolution network was constructed and was very far from the MLP network; (b): after adding another two linear full connected layer, the loss was able to lower down but still unstable.

This demonstrates the fact that a neural network will require global information in order to recreate a radiance field.



(a) (b)

Figure 5 (a)Spherical Harmonics Coeffients Reconstruction; (b)Spherical Harmonics CNN Coeffients Reconstruction

The total Mean Squared loss with MLP and Rasnet layers, especially with regularization, might converge to less than 0.01, however the loss with CNN is about 0.4(Figure 5).

4.1 Datasets

Each of the five scenes in the data set we have created has 10,000 voxels. Ten properties, including locations, colors, emissions, and a special hash code, are present in each voxel. Because normals are implicit, the dataset has this time eliminated them. Each scene differs in terms of object placement and mesh color, allowing for a rough depiction of the scene. Additionally, each scene has one or two emitters that cast area light on the item and a directional light shadow on the cornel box border.

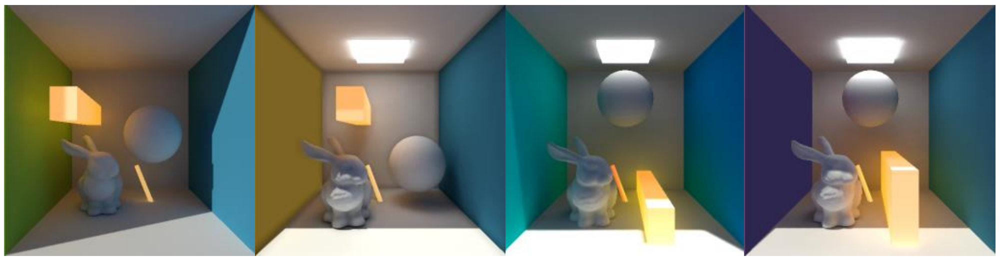


Figure 6 reference of dataset rendering with ray tracing

In order to construct a new scenario for the test dataset, a new object was added to the scene, and a white wall was converted to a pink wall.

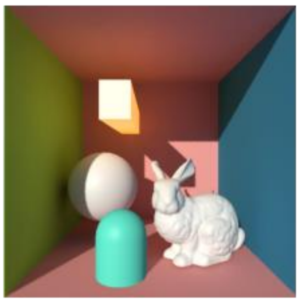


Figure 7 Test data set reference render with ray tracing

Table 1. mean square error of reconstruction using different model

Model	Train Loss	Test Loss	Without regularization
Rasnet-10	0.001	0.006-0.009	0.1
MLP	0.009	0.003-0.007	0.07
RNN	0.03	0.005-0.008	0.04
CNN	0.75	0.6-0.78	0.9-1.2
CNN-MLP	0.006	0.005-0.009	0.05

5. Conclusion

The work first proposed a discrete representation of light transition function. Then it demonstrates a method to represent a radiance field using spherical harmonics, and map voxel 3d scenes to a grid of irradiance. With a neural network-generated 3d function that maps voxel attributes to 27 coefficients of Spherical harmonics, the mesh could get the intermediated light bounced multiple times from other objects and also light in real time.

It has been demonstrated that even a small network can learn the light transition function, although there is further work to be done.

The accuracy of the network could be improved by separating visibility from color prediction using another neural network because the work in the study abstracted the light transition function. The BRDF function of the material should be taken into account for realistic rendering, as well as a full-size network, which has a stable-size node to depict the scene without any information loss.

Second, voxelization is some way to structure the complex scene, it is still necessary to reduce the data that be fed into the network.

The direction of future research could lie on the more complex network such as GNNs (Graph neural networks) which have stable nodes that could simulate light transition function in terms of times and space, and build a renderer that was differentiable to the network to render the scene. In addition, more sophisticated techniques such as raytracing could be adapted to be converted to a function simulated by the network.

Reference

[1] Cohen, M. F., Wallace, J. R., & Hanrahan, P. (1993). Radiosity and realistic image synthesis. Morgan Kaufmann.

[2] Mildenhall, B., Srinivasan, P. P., Tank, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1), 99-106.

[3] Yu, A., Fridovich-Keil, S., Tancik, M., Chen, Q., Recht, B., & Kanazawa, A. (2021). Plenoxels: Radiance fields without neural networks. arXiv preprint arXiv:2112.05131.

[4] Ramamoorthi, R., & Hanrahan, P. (2001, August). An efficient representation for irradiance environment maps. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (pp. 497-500).

[5] Sloan, P. P., Kautz, J., & Snyder, J. (2002, July). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques (pp. 527-536).

[6] Chen, W., Ling, H., Gao, J., Smith, E., Lehtinen, J., Jacobson, A., & Fidler, S. (2019). Learning to predict 3d objects with an interpolation-based differentiable renderer. Advances in Neural Information Processing Systems, 32.

[7] Crassin, C., Neyret, F., Sainz, M., Green, S., & Eisemann, E. (2011, September). Interactive indirect illumination using voxel cone tracing. In Computer Graphics Forum (Vol. 30, No. 7, pp. 1921-1930). Oxford, UK: Blackwell Publishing Ltd.

[8] Pharr, M., Jakob, W., & Humphreys, G. (2016). Physically based rendering: From theory to implementation. Morgan Kaufmann.

[9] Kaplanyan, A., & Dachsbacher, C. (2010, February). Cascaded light propagation volumes for real-time indirect illumination. In Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (pp. 99-107).

[10] Learning Attentions: Residual Attentional Siamese Network for High Performance Online Visual Tracking