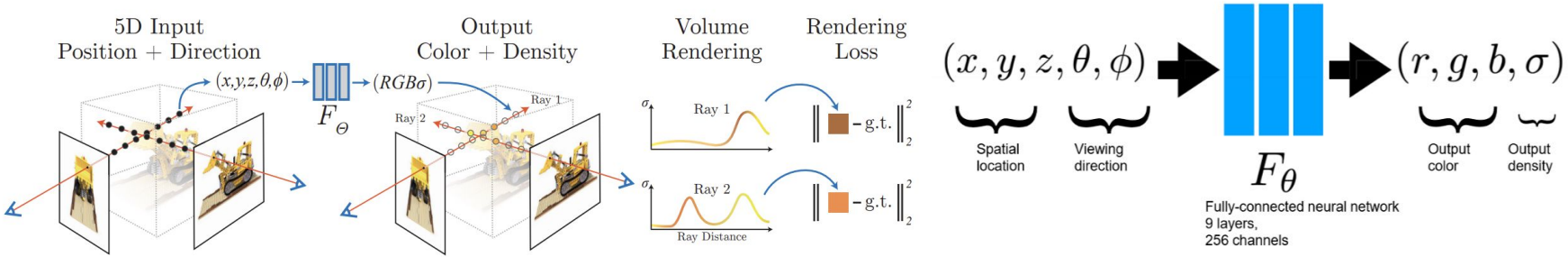
DLCV HW4 Report

NTU CSIE, R12922051
資工碩一 陳韋傑

Q1: NeRF

NeRF learns a model (MLP) that takes spatial locations and view directions and output the predicted colors and densities of a scene. NeRF model will then use a technique called 'volume rendering' to render 2d image by projecting predicted color and density into it, hence makes it possible to render 2d image in any viewpoints.

Since volume rendering is a differentiable process, by computing losses between the rendered images and the corresponding real images, NeRF model can update its weights by stochastic gradient descent and learns a good representation for the given scene.



Q1: NeRF

Personally I think the ‘differentiable volume rendering’ is the most important part in NeRF model. Since NeRF model learns a 3d representation and 3d datasets are hard to acquire, it would be very beneficial if we can directly utilize 2d images to learn 3d representation.

By rendering 2d images using scene representation in a differentiable way, we can compute the losses between predicted 2d images and real 2d images, enabling us to further calculate gradients, update model weights, and learn 3d representation of given scene.

Q1: NeRF

The other two provided papers (DVGO and InstantNGP) are based on NeRF and proposed some modifications to further improve the performance of NeRF. DVGO use voxel grid to replace original MLP used in NeRF, thus allows fast training and inference. InstantNGP proposed a multiresolution hash encoding to replace the original trigonometric frequency encoding in NeRF.

Provided Papers: NeRF: Representing Scene as Neural Radiance Fields for View Synthesis
Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction (DVGO)
Instant Neural Graphics Primitives with a Multiresolution Hash Encoding (InstantNGP)

Q1: NeRF

Pros

- NeRF only needs little 2d images for 3d modeling
- NeRF can obtain decent result and is easy to train

Cons

- Training NeRF is time-consuming
- Trained NeRF can only fit on the given scene, thus lose generality

Q2: Implementaion

I used the provided link (link2) and TA's code (dataset.py) to implement NeRF and train from scratch on the given dataset.

About implementation details please refer to my code and original github repository, or see next page for a brief explanation.

About parameter setting please see question 3 for more details.

Q2: Implementaion

For training dataset, we have rgb images and the corresponding xyz positions of the camera, and we will use dataset.py to transform them into NeRF's model input and output type (input: (x, y, z, theta, phi), output: (r, g, b, density)).

In train.py, the class NeRFSystem is comprised of two NeRF models (Coarse & Fine) and two Embedding Layers (For Positional Encoding). During training, the function render_rays() will first get the NeRF model output, and render them into 2d images, so that we can use them to calculate loss and update model weights.

In eval.py, we use batched_inference() (batched version of render_rays) to render 2d images from brand-new point of view.

Q3: Result

Setting	PSNR ↑	SSIM ↑	LPIPS (VGG) ↓
D=2, W=64	39.0944	0.9866	0.1169
D=4, W=128	42.0635	0.9925	0.1053
D=8, W=256	**43.4415**	**0.9941**	**0.0974**

N_importance = 64
Epochs = 10
Noise_std = 0
Batch Size = 1024
Optimizer = Adam

Img_wh = 256 256
Learning Rate = 5e-4
Scheduler = StepLR
Decay Step = 8
Decay Gamma = 0.5

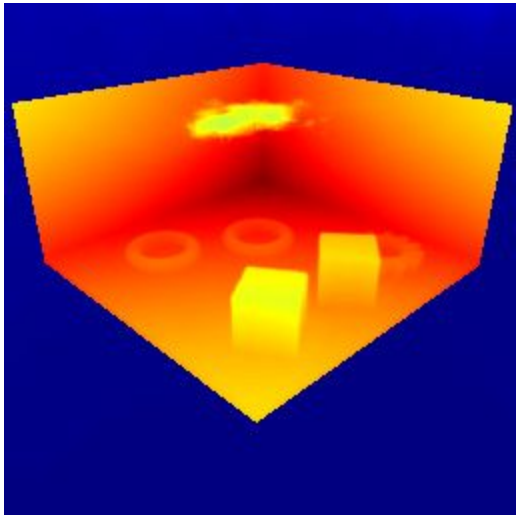
Q3: Metrics Explanation

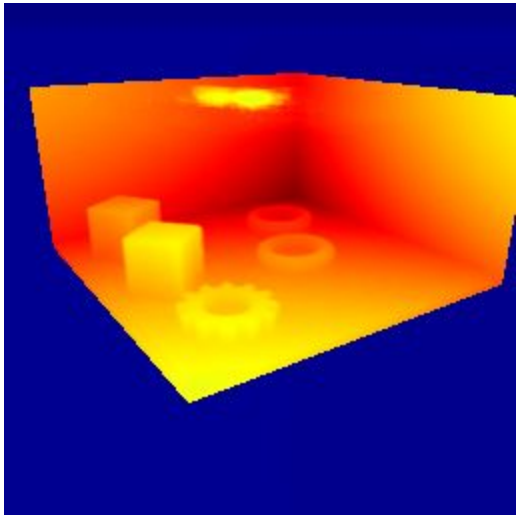
PSNR (Peak Signal to Noise Ratio): PSNR measures the ratio between the maximum possible value and the corrupting noise that affects the fidelity. Higher PSNR is better.

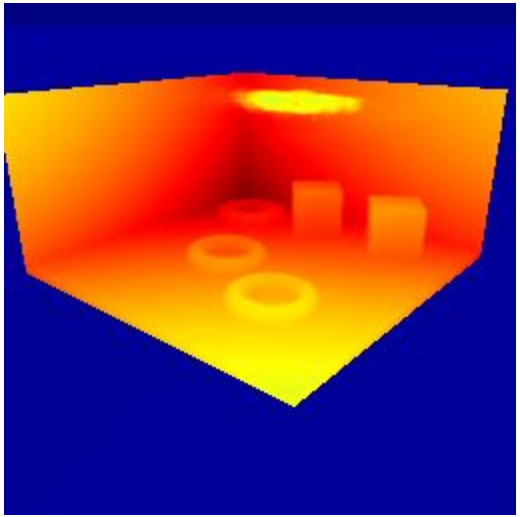
SSIM (Structural Similarity Index): SSIM measures the structural similarity between two images, it calculates luminance, contrast, and structure to provide a human-like assessment of image quality. Higher SSIM is better.

LPIPS (Learned Perceptual Image Patch Similarity): LPIPS uses DNN (backbones such as VGG, AlexNet) to compute a perceptual similarity metric, it first extracts image features at each layer of DNN, compute their distance and multiply by model layer weights, then average over each layer to get result. Compared to PSNR and SSIM, LPIPS aligns more closely with human visual perception. Lower LPIPS is better.

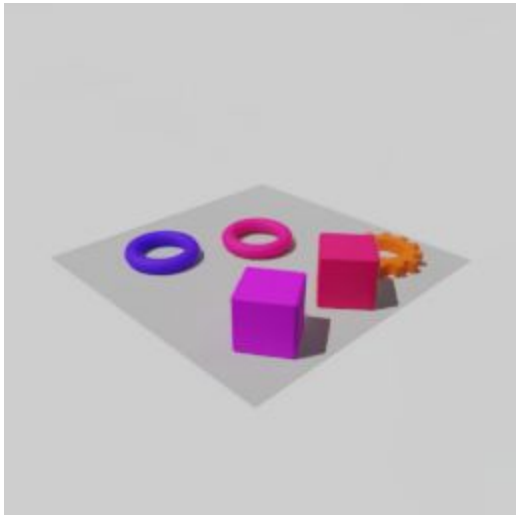
Q4: Depth Rendering

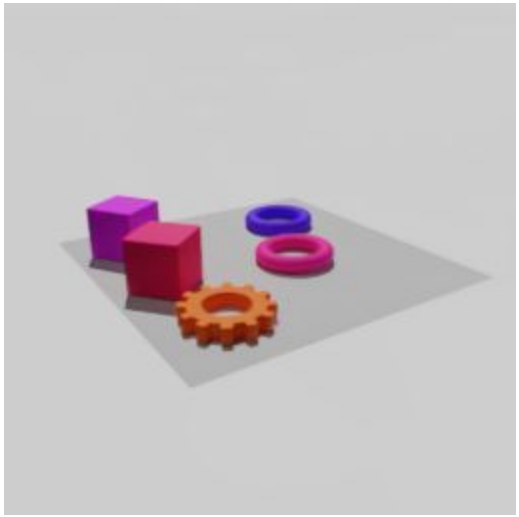


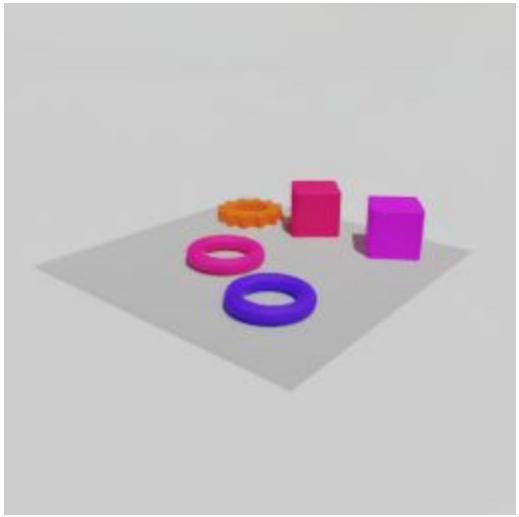




Appendix: RGB Rendering







Reference

Code: https://github.com/kwea123/nerf_pl

Report:

- NVS: https://zhuanlan.zhihu.com/p/486710656
- DVGO: https://zhuanlan.zhihu.com/p/584734270
- InstantNGP: https://zhuanlan.zhihu.com/p/575269971
- Metrics:
  - https://zhuanlan.zhihu.com/p/309892873
  - https://ithelp.ithome.com.tw/articles/10332547