# Cascade Cost Volume for Generalizable Neural Radiance Field

Tz-Wei Mo
Stanford University
tzwmo@stanford.edu

Yi-Chia Wu
Stanford University
yichiawu@stanford.edu

## Abstract

*In 2020, [9] first proposed the idea of neural radiance fields (NeRF), by using a MLP as a 5D vector-valued function for scene representation. The network takes in 3D coordinates in space* $\mathbf{x} = (x, y, z)$ *and 2D viewing directions (*$\theta$*,* $\phi$*) and predicts an emitted color* $c = (r, g, b)$ *and volume density* $\sigma$ *at that spatial location. A differentiable volume rendering method is then used to composite these output values along virtual camera rays to produce a final output pixel value for a synthesized view. Although the paper was able to produce high resolution results and represent fine details, it lacks the ability to generalize across scenes, and a new network has to be trained per scene, which can be time consuming. MVSNeRF [1] solves this issue by combining cost-volume based deep MVS technique with NeRF to enable efficient reconstruction of radiance fields for neural rendering, however the use of 3D CNN in the model architecture slows down the training process. In this project we resolved this by using cascade MVSNet [4] to refine the cost volume from coarse to fine using fewer depth layers for the 3D CNN to increase network training speed and decrease GPU usage.*

## 1. Introduction

The goal of view synthesis is to generate novel views of a subject or scene by supplying a set of input images taken from given point of views. Earlier works have attempted to solve this problem using interpolation methods [3], while others [5] try to generate arbitrary views of a scene by first reconstructing the 3D model from the input images.

The idea of Neural Radiance Fields (NeRF) was first proposed by [9] in 2020 which is a novel view synthesis and geometric reconstruction method using a trained Multi-Layer Perceptron (MLP) network to represent a 3D scene. The MLP takes in arbitrary points and viewing angles in space as input and predicts RGB colors and volume density at the given spatial location. Differentiable volume rendering [13] is then used to produce the final novel view image. Since the whole process is differentiable, gradient descent can be used to optimize the MLP model by minimizing the error between each observed image and the corresponding views rendered from the MLP and volume rendering procedure. Although NeRF was able to render high quality results, a new model has to be trained per scene, which can take up to 12 hours and at least 20 images are needed to train each scene. To overcome these shortcomings, [1] combines NeRF with MVSNet [15], which was first proposed inference depth map from multi-view images. The MVS-NeRF framework first constructs a cost volume by using a a deep 2D CNN to extract image features onto a plane sweep. Then a 3D CNN UNet with skip connections is used to reconstruct a neural encoding volume with per-voxel neural features. Finally, the neural features are fed into the NeRF framework MLP to regress volume density and RGB radiance at an arbitrary location and volume rendering is again used to produce the final output view. The whole process can be trained end-to-end for a general model, which can already achieve decent results and can be further improved by rapid fine-tuning on each scene.

In this project we aim to further improve MVSNeRF by optimizing the training speed of the general model by using a more efficient neural volume encoding from [15] but can still obtain the same quality after fine-tuning.

## 2. Related Work

### 2.1 Deep Learning for Novel View Synthesis

Recently, deep learning methods, such as DeepSDF [10] are also being used to synthesize high quality results and another work [6] uses only a single input image and generates visible regions from new view-points by shrink-wrapping a planar mesh sheet onto that image.

### 2.2 Recent Works on NeRF

Since the advent of the 2020 paper on Neural Radiance Field, many papers have tried to improve on the original NeRF paper in several ways: [9] used dynamic scene inputs instead of static scenes and trains a second network to handle deformities;[8, 11, 12] try to generalize NeRF to avoid per-scene training; and [17] enables relighting of the 3D

scene and synthesized views. Other approaches improves scalability by learning a generic view interpolation function that generalizes to novel scenes [14] or by using 3D CNN [1].

### 2.3 Learned Multi-View Stereo

In 2018, [15] proposed to predict depth maps of a scene by building 3D cost volumes. A 3D cost volume is built based on warped 2D image features obtained from a 2D CNN network and taking the variance across all views. Once the cost volume is built, 3D CNNs are applied for cost regularization and depth regression for the final depth map. Newer methods such as [2, 16, 4] all try to reduce the memory requirement and generate high resolution outputs. In this work however instead of using the cost volume for depth regression, we instead use it as a representation of the scene such that the NeRF MLP can learn to generalize from the representation.

## 3. Technical Approach

Following the approach of MVSNeRF, we will first train a general model able to extract useful representations from a scene using only 3 images and train it jointly with the NeRF MLP which will learn to output the correct RGB value from the 3D spatial position as well as our learned representation. Once we have this general model, we can then perform fast per scene optimization that can produce high quality results after just 20 minutes instead of 12 hours per scene training from the original NeRF. The pipeline for training the general model can be seen in Figure. 1, and can be separated into 3 parts: Feature Volume Generation as shown in Figure. 2, Neural Volume Encoding, and Volume Rendering.

### 3.1 Feature Volume Generation

To build the feature volume for our scene representation, we follow the work of [4], using 3 images from the same scene as input into a 2D CNN feature extraction network. The 2D CNN uses a feature pyramid network (fpn) structure where the output of the network are proportionally sized feature maps at different stages, which will all be used to generate our final feature volume. In our final implementation we used a total of 3 stages such that the $i$-th input image $I_i \in \mathbb{R}^{H\times W\times 3}$ will have output feature maps of $F_{i,1} \in \mathbb{R}^{H/4\times W/4\times C}$, $F_{i,2} \in \mathbb{R}^{H/2\times W/2\times C}$, and $F_{i,3} \in \mathbb{R}^{H\times W\times C}$, where $i = \{1, 2, 3\}$ are indices of the three input views, $H$ and $W$ are the image height and width and $C$ is the number of output channels of the CNN network.

To obtain 3D relations of the feature maps, we use camera intrinsic and extrinsic parameters of the $i$-th view: $\Phi_i = [K_i, R_i, t_i]$ to warp points from feature maps of 2 of the source views onto the fronto-parallel planes of the reference view ($i = 1$) at different depth values $z$. The differentiable homographic warping can be written as:

$$H_i(z) = K_i \cdot \left( R_i \cdot R_1^T + \frac{-R_1^T t_1 + R_i^T t_i}{z} n_1^T \right) \cdot K_1^{-1} \quad (1)$$

where $H_i(z)$ is the warping matrix at depth value $z$ and $n_1$ denotes the principle axis of the reference camera. Using the warping matrix and given a pixel coordinate in the reference view image, $[u, v, 1]^T$, we can find the corresponding pixel coordinate $[u', v', 1]^T$ in image $i$ using

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = H_i(z) \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (2)$$

Here, we parameterize $[u, v, z]$ using the normalized device coordinate (NDC) at the reference view. The cost volume $P_j$ for stages $j = 1, 2, 3$, can then be built by warping the feature maps at stage $j$ onto $D_j$ sweeping planes, and taking the variance across different views to aggregate the 2 warped feature maps and the reference feature map at stage $j$ into one cost volume. We can write the above operations as:

$$P_j(u, v, z) = Var(F_{i,j,z}(H_i(z) \cdot \begin{bmatrix} u & v & 1 \end{bmatrix}^T)) \quad (3)$$

where $P_j(u, v, z)$ is value of the cost volume $P_j$ centered at $(u, v, z)$. The cost volume after each stage will then go through a 3D CNN with a UNet architecture for cost regularization, which will produce a feature volume that can be used for depth inference or in the case of the last stage be used as the scene representation to train the NeRF MLP. This process is expressed as:

$$V_j = S_j(P_j) \quad (4)$$

where $S_j$ is a 3D CNN UNet at stage $j$ and $V_j$ is the feature volume produced at stage $j$.

In our implementation the hypothesis depth range of the first stage denoted by $R_1$ covers the entire depth (or disparity) range of the input scene. In the following stages, we can base on the feature volumes from the previous stages, estimate a depth map to narrow the hypothesis range. Consequently, we have $R_{j+1} = R_j \cdot w_j$, where $R_j$ is the hypothesis range at the stage $j$ and $w_j < 1$ is the reducing factor of hypothesis range. By shrinking the hypothesis depth range after each stage, we can get more accurate depth estimations by applying differential warping at relevant depth values to refine the feature volume for the next stage.

In addition to shrinking the hypothesis depth range, we also shrink the interval between the sweeping planes at each stage. Our depth interval in the first stage denoted as $L_1$ is larger than the single cost volume formulation in the

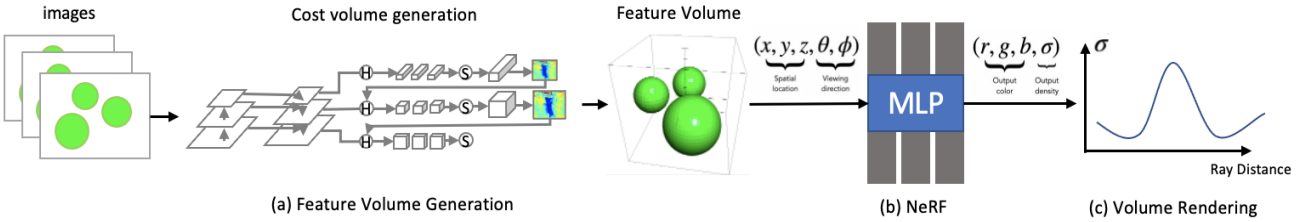


Figure 1. Pipeline for training a general model which can be divided into 3 stages: (a) Feature Volume Generation (b) Neural Volume Encoding (c) Volume Rendering.

original MVSNeRF for a coarser estimate. In the following stages, however, finer hypothesis plane intervals are applied to recover more detailed outputs, such that we have: $L_{j+1} = L_J \cdot p_j$, where $L_j$ is the hypothesis plane interval at the $j$th stage and $p_j < 1$ is the reducing factor of hypothesis plane interval.

Given the hypothesis range $R_j$ and hypothesis plane interval $L_j$ at stage $j$, the corresponding number of hypothesis planes $D_j$ for homographic warping is determined by the equation: $D_j = R_j/L_j$. In MVSNeRF, a total of 128 hypothesis planes was used to generate their single cost volume, leading to a larger 3D CNN network for cost volume regularization. However, in our project the number of planes for each stage are 48, 32, and 8 respectively such that even though we have three 3D CNN networks, the size of each network is comparatively smaller than MVSNeRF and by doing the coarse to fine update after each stage, we can still achieve similar results by using less computation time and memory usage. For more details on the update of depth intervals and depth ranges using depth estimates from the previous stage please refer to [4], particularly the warping operation in Eq. 3 of their paper.

Since the original feature volume MVSNet and Cascade MVSNet did not contain color information, MVSNeRF appended per-view colors as additional channels to the cost volume which we also adopted this approach into our method of using Cascade MVSNet.

### 3.2 Neural Volume Encoding

After obtaining the feature volume $V_3$ generated by the last stage of the cascade MVSNet described in the previous subsection, we will use it as a scene representation to feed into the NeRF MLP. Instead of feeding the entire feature volume into NeRF, we will extract useful information from the feature volume. Particularly, using the spatial location, $\mathbf{p} = (x, y, z)$, in the reference view NDC coordinate, we will index the feature volume value $f = V(\mathbf{p})$ at $\mathbf{p}$ as additional features. Aside from the color information appended in the feature volume, MVSNeRF also indexed rgb values $c$ using the spatial coordinates $\mathbf{p}$ projected onto different source views as features to the MLP. The MLP can then be written as:

$$\sigma, r = A(\mathbf{p}, d, f, c) \tag{5}$$

where $A$ represents the MLP. $\sigma$ is the density output and $r$ is the rgb value at the input spatial location $\mathbf{p}$. $d$ is the viewing direction, $f$ is the volume feature value indexed at $\mathbf{p}$ and $c$ is the rgb color value indexed at $\mathbf{p}$ at different source views. Positional encoding is used for both $\mathbf{p}$ and $d$ as described in [9].

### 3.3 Volume Rendering

Using the output of the MLP, we can evaluate the differentiable ray marching equation to generate a targe image using the same procedure in [9]. In particular, the pixel's radiance value in the target image can be computed by marching a ray through the pixel and accumulating radiance at sampled shading points on the ray, given by:

$$\begin{aligned} c_t &= \sum_k \tau_k (1 - \exp(-\sigma_k)) r_k, \\ \tau_k &= \exp(-\sum_{j=1}^{k-1} \sigma_j), \end{aligned} \tag{6}$$

where $c_t$ is the final pixel color output, and $\tau$ represents the volume transmittance.

To train the general model the entire pipeline above is trained in an end-to-end fashion and we supervise the entire model with the ground truth pixel colors, using an L2 rendering loss:

$$L = \|c_t - \tilde{c}_t\|_2^2 \tag{7}$$

where $\tilde{c}_t$ is the ground truth pixel color sampled from the target image.

### 3.4 Per Scene Fine-tuning

After training a general model. We also apply per scene fine-tuning to obtain high quality novel view synthesis results of new scenes. For fine-tuning, we do not need to update the entire network, instead, we only have to update the feature volume generated and the NeRF MLP which allows

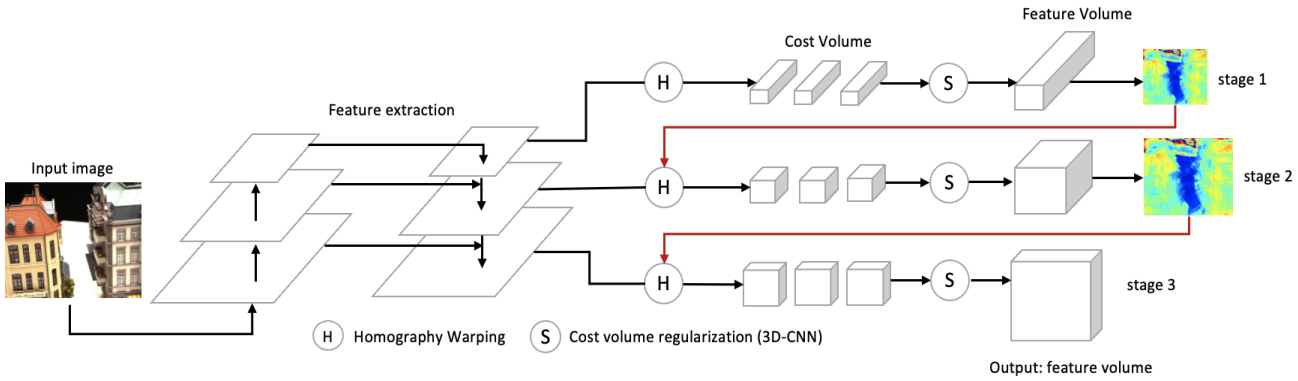


Figure 2. Volume Feature generation of a scene using 3 views as input.

us to produce superior results in less than 20 minutes as can be seen in the next section. During fine-tuning more views can also be added if available to improve the viewing range of the synthesized views. For our obtained results, we fine-tuned the model on 16 source views on unseen scenes during training on the general model.

### 3.5 Other implementation Details

#### 3.5.1 Deviation from Original NeRF

The MLP used in our project has fewer layers than the original NeRF paper consisting of only 6 layers. We also did not implement hierarchical volume sampling as the results obtained from our project can already achieve good results.

#### 3.5.2 Dataset

We trained our network on the DTU [7] dataset which contains a total of 124 scenes which we divided up into 88 scenes for training and 36 scenes for testing. Each scene contains 343 RGB images taken from 49 different positions with 7 different brightness and resolutions of $640 \times 512$. During training we randomly select a batch of scenes where in each scene we used three different views as an input to train the general model. We then tested our trained model on the testing dataset to evaluate the effectiveness and generalizability of our trained network.

## 4. Results

In this section we first compare our trained general model against MVSNeRF. We trained both models on the DTU dataset for 6 epochs, where it took our method 36 hours to train whereas MVSNeRF took 72 hours using Google Cloud platform with NVIDIA Tesla K8. We then show fine-tuned results of our own model which shows great improvements both qualitatively and quantitatively. For all of the results obtained, we used PSNR and SSIM as our metric to quantify the performance of our model.

For qualitative results, we also evaluate our geometry reconstruction quality by comparing depth reconstruction results, generated from the volume density by a weighted sum of the depth values of the sampled points on marched rays using the same method in [9].

### 4.1 Comparing general model with MVSNeRF

Here we show the quantitative and qualitative results of the performance of our model against MVSNeRF. We can see from Fig. 3 and Table. 6 that our results of the general model without fine-tuning is worse than the results of the original MVSNeRF across all test scenes after only training 6 epochs. We see more artifacts and blurriness in our results. From the depth estimation we also see that the contour of the objects aren't as accurately captured. This initial degradation of generalizability across scenes might be caused by the decreased depth range hypothesis from our smaller 3D CNN networks. However, after just one epoch of fine-tuning, which takes about 1 minute, we see that Cascade MVSNeRF can achieve high PSNR close to the performace of the original MVSNeRF.

### 4.2 Fine-tuning on the Cascade MVSNeRF

From the quantitative results seen in Table. 6 we see after fine-tuning for 20 epochs, which takes about 20 minutes to train on each scene, we can achieve high PSNR comparable to the results in the original MVSNeRF papers. We can also see the improvements by looking at Figure 4, where we see a reduction in blurriness and a better estimated depth map. The performance saturates at around 40 epochs where we only see a slight improvement compared to the 20 epochs fine-tuning. You can find rendered videos of our results at: https://drive.google.com/drive/folders/1b8PIgxzuQsbr1AOkrA7VFZdfR4XJNJ2X?usp=sharing

| Method | Setting | DTU Scene | PSNR | SSIM |
|---|---|---|---|---|
| Cascade MVSNeRF | no fine tuning | Bucket | 17.7226 | 0.7657 |
| | | Ball | 19.3203 | 0.8136 |
| | | House | 14.2223 | 0.6303 |
| | | Pig | 17.0606 | 0.8376 |
| | fine tuning (1 epoch) | Bucket | 23.9647 | 0.8767 |
| | | Ball | 25.6056 | 0.8618 |
| | | House | 19.5293 | 0.7963 |
| | | Pig | 27.1065 | 0.9319 |
| MVSNeRF | no fine tuning | Bucket | 22.6295 | 0.8863 |
| | | Ball | 21.0985 | 0.8651 |
| | | House | 17.5893 | 0.8026 |
| | | Pig | 22.0656 | 0.9241 |
| | fine tuning (1 epoch) | Bucket | 26.6090 | 0.9133 |
| | | Ball | 27.5977 | 0.8803 |
| | | House | 21.1742 | 0.8543 |
| | | Pig | 29.0708 | 0.9474 |

Table 1. Quantitative results of Cascade MVSNeRF and MVSNeRF. We show averaged results of PSNRs and SSIMs on different 4 scenes of the DTU dataset. The table shows no fine-tuning results between two networks and compares with the results with 1 epoch fine-tuning.

| Cascade MVSNeRF | DTU Scene | PSNR | SSIM |
|---|---|---|---|
| fine-tuning 20 epochs | Bucket | 29.1938 | 0.9571 |
| | Ball | 31.7760 | 0.9643 |
| | House | 24.5467 | 0.9520 |
| | Pig | 31.8754 | 0.9801 |
| fine-tuning 40 epochs | Bucket | 29.5759 | 0.9607 |
| | Ball | 32.5395 | 0.9705 |
| | House | 25.1341 | 0.9581 |
| | Pig | 32.5701 | 0.9823 |
| no fine tuning | Bucket | 19.8908 | 0.7549 |
| | Ball | 20.0483 | 0.8530 |
| | House | 15.3283 | 0.6664 |
| | Pig | 19.4937 | 0.8256 |

Table 2. Quantitative results of Cascade MVSNeRF. We show averaged results of PSNRs and SSIMs on different 4 scenes of the DTU dataset. The table shows no fine-tuning results, 20 epoch fine-tuning results and 40 epoch fine-tuning results.

## 5. Conclusion and Future Work

In this project we present a way to speed up the general model training time for MVSNeRF by replacing the MVSNet for feature volume generation with Cascade MVSNet, which uses smaller models, but refines the generated representation after every stage. We were able to train the model in half the time and although initially the out of the box model seems to have a poorer performance, after fine-tuning on a single scene for even one epoch shows great improvement. After 20 epochs, which takes around 20 minutes, we achieve PSNR comparable to the original MVSNeRF paper and can render high quality videos from novel view points.

In this work we did not have time to analyze the trade off between the number of hypothesis planes and the general model performance. With more planes we suspect the model to have better performance without fine-tuning, but with a slower training time and higher memory requirements. We also didn't have time to try the model on self-captured data to test how well the model can generalize to real-world scenes.

## 6. Supplementary Materials

Here is the link to view the code used in the development of our project: https://github.com/wuyichia/Cascade_MVSNeRF

## References


[1] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo, 2021. 1, 2

[2] R. Chen, S. Han, J. Xu, and H. Su. Point-based multi-view stereo network. *CoRR*, abs/1908.04422, 2019. 2

[3] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996. 1

[4] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. *CoRR*, abs/1912.06378, 2019. 1, 2, 3

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 1

[6] R. Hu, N. Ravi, A. C. Berg, and D. Pathak. Worldsheet: Wrapping the world in a 3d sheet for view synthesis from a single image. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 1

[7] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014. 4

[8] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 3, 4

[10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1

[11] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 1

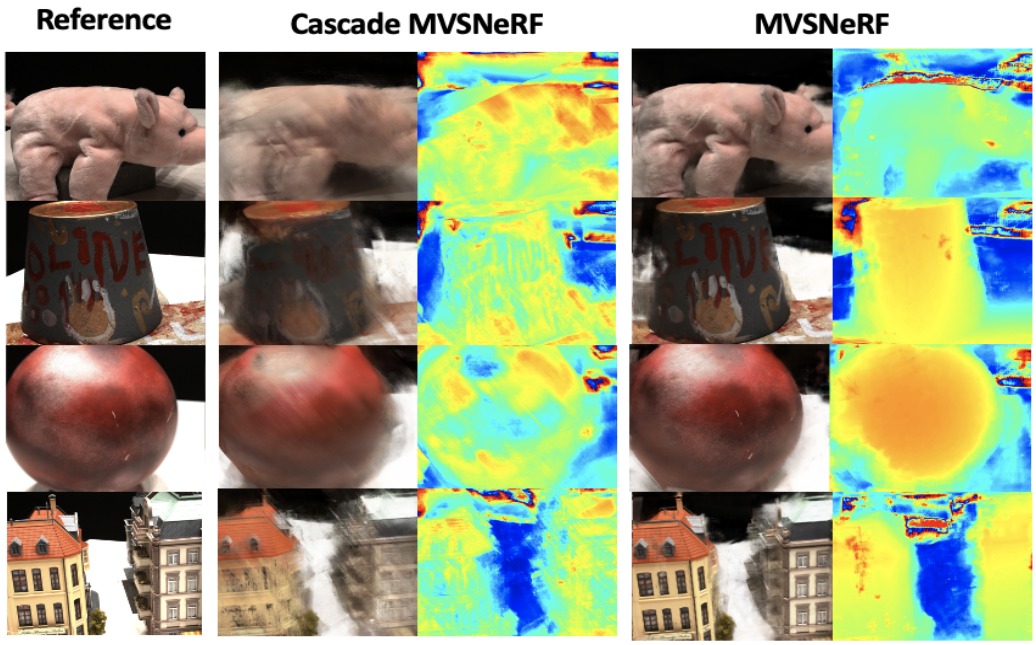


Figure 3. Qualitative result of our method and MVSNeRF. Without fine-tuning our model performance produces degraded results compared to MVSNeRF.

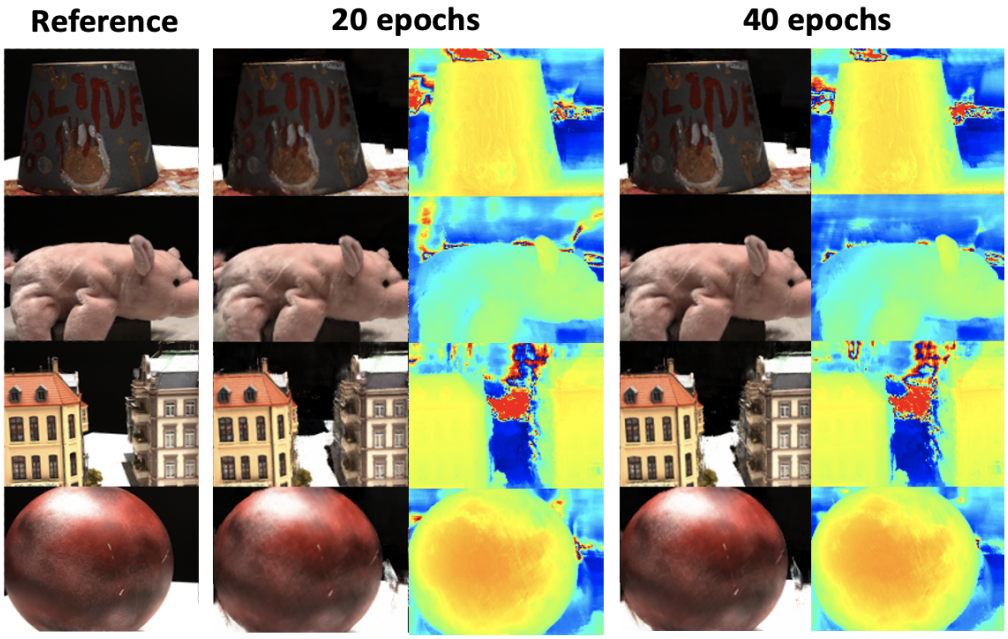


Figure 4. Results of our method after fine-tuning for 20 epochs and 40 epochs on a single scene. Results show great improvement and can obtain high PSNR.

[12] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1

[13] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wi-

ley Online Library, 2020. 1

[14] Q. Wang, Z. Wang, K. Genova, P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[15] Y. Yao, Z. Luo, S. Li, and T. F. andLong Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *CoRR*, abs/1804.02505, 2018. 1, 2

[16] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. *CoRR*, abs/1902.10556, 2019. 2

[17] K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting, 2021. 1