# Generating Neural Radiance Fields from Point Clouds for Novel Image Synthesis

Karthick Subramanian
*1223408524*
*Arizona State University*
Tempe, Arizona
ksubra25@asu.edu

Prithvi Hemanth
*1225935284*
*Arizona State University*
Tempe, Arizona
phemanth@asu.edu

Praveen Paidi
*1225713099*
*Arizona State University*
Tempe, Arizona
ppaidi@asu.edu

***Abstract*—The task of reconstructing 3D scenes from image data and producing new photo-realistic perspectives is a crucial aspect of computer vision. While classical NeRF [1] uses images and pose data for volumetric rendering, our model leverages point clouds. However, due to the sparsity of point clouds, creating complete images becomes challenging. To overcome this challenge, we employ a recurrent neural network pipeline that transforms point clouds into dense point clouds, allowing for feasible construction even in subpar areas. Additionally, we propose point cloud registration using a neural network and COLMAP to extract the optimal point cloud in sparse areas. To address the training time limitations of Classical NeRF, we introduce a RANSAC model that avoids void space training by identifying the object's near space probability from multiview stereo probability maps. To complete this pipeline, we implement it in a volumetric rendering algorithm to obtain novel views.**

## I. Introduction

Applications involving autonomous driving, robotics, and virtual/augmented reality require reconstructing 3D scenes using sensor fusion of images, LiDAR data, etc. There have been advancements in this regard with the development of methods such as bundle adjustment[19], structure from motion, multi-view stereo, and methods such as Neural Radiance Fields(NeRF) that leverage deep learning. The NeRF algorithm has played a significant role in advancing the field of Neural Rendering. This is enabled by its flexible nature in terms of its framework and areas of application and, its unprecedented quality of synthesized images.

NeRF uses 2D images for regression in Multi Layer Perceptron (MLP) with known poses. After long training on image data, it produces artifacts of blurring due to single ray marching per pixel. Increasing the number of rays per pixel in volumetric rendering would significantly increase the training time. Various alternatives to 2D image regression have been suggested, including voxel-based, mesh-based, and point cloud-based methods. Voxel-based methods are rudimentary for 3D reconstruction. They infer the occupancy or color of each point in a grid space but are computationally intensive and have difficulty with surface representation. Additionally, meshes are popular representations because they are easy to use and can store information efficiently, but they are difficult to capture or optimize detailed topologies and geometry. Since point clouds are capable of accurately representing arbitrary surfaces on the outdoor scale and high level of detail representation capability, we leverage them in our research to counter-mentioned and relevant to the application as well.

Point cloud generation can be achieved using COLMAP. COLMAP provides tools and algorithms for various tasks, including 3D reconstruction, camera calibration, image matching, and dense point cloud generation. There has been work on generating point clouds using Deep Neural Networks as well [14]. With the help of structure from motion, multi-view stereo was developed and trained on huge datasets, allowing dense point clouds to be generated by DNNs by pre-trained models.

In this work, the two-point clouds obtained from COLMAP and the multi-view stereo are registered using deep global representation[16]. We propose to add two complete point clouds of cross-source instead of fragment point clouds and make the optimal point cloud with more detail. The resulting point cloud is then passed through a set of recurrent levels for feature extraction indicating a sequential approach, and these features are used to refine the Chamfer distance metric, which measures the similarity between the source and target point clouds. By leveraging registration and RNNs, the pipeline aims to improve the alignment of the point clouds and preserve the original surface shapes, resulting in dense accurate point clouds.

We apply regression on this point cloud in MLP instead of 2D images to reduce the loss in NeRF training and to generate novel views of the 3D scene through volumetric rendering. This holistic approach enables us to leverage both image data and point cloud information for enhanced 3D scene reconstruction and novel view generation, presenting a promising avenue for advancing computer vision.

Despite strides in improving the visual quality of NeRF, addressing the challenges related to training time remains an important research focus. We advocate the utilization of Random Sample Consensus (RANSAC) in conjunction with depth maps and confidence values generated by Multi-view stereo net. For some sets of iterations in volumetric rendering, we neglect training on low-confidence points indicating low volume density. This allows us to efficiently remove the need for training in void or empty spaces.

The proposed approach builds upon the ongoing efforts to

enhance NeRF's training efficiency and adaptability to novel data, an area that has garnered substantial attention in recent years.

## II. Related Work

This section covers the past work on key concepts that constitute our proposed work under the headings stated below.

### A. Neural Radiance fields

Classical NeRF [1] generates novel views of complex scenes by optimizing volumetric scenes. This method introduces blur in the rendered images due to limited ray marching per pixel and has a significant training time. MIP-NeRF [7] uses the frustum of rays as multivariate Gaussian to approximate the shapes in low resolution as well. For the reflective surfaces, Ref-NeRF [8] was introduced by training MLP on scenes with various materials and textures.

Taking NeRF from static to dynamic scenes, D-NeRF[17] was proposed by rendering views at arbitrary times and producing novel views at different times. Furthermore, instead of using known camera poses, the authors of NeRF–[18] propose the camera parameters can be jointly optimized as learnable parameters with NeRF training using photometric reconstruction.

### B. Point cloud

The main bottleneck for NeRF training time is ray marching and point sampling. To overcome this limitation, data representations such as point clouds, 3D meshes, depth maps, etc ., were developed. Point clouds alone in the place of 2D images don't provide satisfactory results. The work in [2] demonstrates this. Points2NeRF [2] is a method that directly generates a neural radiance field from point cloud data. The paper shows that there were issues while using standalone point clouds. The authors suggest using a hypernetwork to generate radiance fields simultaneously from point clouds and 2D images and train loss function. They developed auto encoders to generate 3D representations from point clouds by training hypernetworks.

The work in [5] produces 3D point clouds from a multiview stereo (MVS) net. Classical MVS net involves extracting image features, building cost volumes, and differentiable homography. MVS net can operate on four types such as voxel-based, surface-based, patch-based, and depth map-based. The final method is appropriate for our approach. It estimates depth maps for a set of images and appends all the depth maps to construct a point cloud. The algorithm performs 3D CNN cost regularization to produce the probability maps that result in depth maps.

Recurrent Forward Network (RFNet) [4][27] uses sparse point clouds and then generates dense point clouds. The algorithm comprises the following units - Recurrent Feature Extraction, and Raw Shape Protection. Recurrent feature extraction extracts features from incomplete point clouds. Forward-dense completion produces a dense point cloud. Finally, Raw Shape Protection fine-tunes the point cloud by adding details from the incomplete point cloud.

Our method involves generating a point cloud from depth maps of MVSNet and COLMAP, performing complete cross-cloud registration, and using RFNet to produce subtle output.

### C. Neural Point clouds

Although dense point clouds produce reasonable results, using pixel graphic techniques from computer graphics produces improved results. These techniques have been in use since the early 2000's. Image synthesis using neural textures [11] is one such technique. It proposes an approach that learns object-specific neural textures that can be interpreted by a neural renderer. It proposes neural maps similar to feature maps. Instead of storing low-dimensional hand-crafted features, they store learned higher-dimensional feature maps. These were used in encoders and deferred neural rendering to generate novel images.

Point clouds with neural descriptors were proposed in [6]. Here 2D image features were extracted, a point cloud was generated from image data, and the output image was rendered using a rasterizer. The data is downsampled in a pyramid network from coarse to fine mechanism to add reliable surface details and image features to the neural cloud.

Point Nerf analysis [3] is another technique for applying NeRF on neural clouds. Here feature vectors corresponding to the local geometry of the unprojected 2D image are assigned to each point in the point cloud. These points also have an associated depth confidence value.

MVS - Nerf [10][26] proposes a similar technique where it takes three images that are in proximity of each other and applies a CNN to extract features. It executes cost volume sweeping to build a voxel-based radiance field. Although it provides for faster convergence, it cannot be used to render images from any viewpoint. The 3D object representation is limited. The work in [3] takes inspiration from this idea and builds a 3D representation from images of different viewpoints around the object.

Our method proposes fusing point cloud data with CNN feature vectors obtained from 2D image pyramids at different levels.

### D. Accelerated NeRF Algorithms

To reduce the training time, Fast-nerf [12] was proposed. It divides the MLP of the traditional NeRF into two different MLPs. The input to the first MLP is the 3D coordinates, while the second takes the viewing directions as the input. This reduces the exponential training time. The idea is motivated by spherical harmonics.

In NerfingMVS[13], COLMAP is used to extract sparse depth priors in the form of a point cloud which is used to get a depth map. This depth map is used to supervise volume sampling by only allowing sampling points at the appropriate depth. During volume rendering, the ray is divided into N equal bins with the ray bounds clamped using the depth priors. Once the image is rendered, the error between the

rendered image and the original image is obtained to develop a confidence-based filter to improve the rendered depths.

There is some work done in eliminating training on void space by the introduction of Depth supervised ray termination loss proposed by DS-NeRF [9][25]. In addition to images and poses, the algorithm inputs a sparse point cloud and the reprojection error computed from SFM or COLMAP[24]. In general, there is a distribution for depth in NeRF using ray marching which tries to match to ground truth. Using depth uncertainty, DS-NeRF creates another depth distribution and develops a loss function to decrease divergence between ray marching distributions which helps in decreasing the training time.

Our method suggests taking confidence level from MVSNet and then using it to neglect the outliers in volume rendering.

## III. Method of Approach

### A. MVS NET for Point cloud

It takes N images as a input and generates different point clouds using MVSNet. For training we use ground truth depth values as well.

*1) Image features Extraction:* In order to extract deep features $[F_i]_{i=1}^N$ from N input images $[I_i]_{i=1}^N$, a 2D CNN with 8 layers is applied to get feature maps of 32 channel pixel descriptor.

*2) Cost Volume, Differentiable Homography:* One of $[I_i]_{i=1}^N$ is taken as reference frame and then other image feature maps are warped into reference frame using image extrinsic, rotations, translations $[K_i, R_i, t_i]_{i=1}^N$ making N feature volumes.All feature maps are warped into different front parallel planes of the reference camera to form $[V_i]_{i=1}^N$ Volume maps using the projection matrix $P_i$ in 3D using depth values initiated for reference image.The resulting feature volumes are aggregated to make variance cost volume C.

$$C = \frac{1}{N}\sum_{i=1}^{N}(V_i - \overline{V})^2$$

*3) Cost volume regularization:* To refine noise, a Multi Scale 3D CNN applied on feature map of 32 channel pixel descriptor at different depth levels to get a single layer output at each depth level, and finally proceeds to softmax application in depth direction which gives single probability volume P that can be used as confidence estimation at each depth.

*4) Depth Map:* From the probability map P, depth map can be achieved by applying softargmax.

$$D = \sum_{d=d_{min}}^{d_{max}} dXP(d)$$

Where P(d) is the probability estimation for all pixels at depth d.The output shape of depth values is same size as the 2D feature maps.With the use of depth map, respective images point cloud is formed mostly inspired from Multiview stereo.

*5) Loss function:* This is used to train and apply on novel dataset by apply using trained weights on loss following loss function.

$$\text{Loss} = \sum_{p \in p_{\text{valid}}} \left\| d(p) - \hat{d}_i(p) \right\| + \left\| d(p) - \hat{d}_r(p) \right\|$$

Where $p_{valid}$ denotes the generated depth input for reference frame, d(p) the ground truth depth value, $\hat{d}_i(p)$ the initial depth estimation and $d_r(p)$ the refined depth estimation.

### B. COLMAP for Point Cloud

COLMAP is a tool which provides a command line and graphical user interface for generating point clouds and also rendering surface meshes. It works based on structure-from-motion(SfM) and Multi-View Stereo pipeline for generation of point cloud.

- **Feature Detection and Extraction**
  - **Input**: Images
  - **Output**: Keypoints, descriptors of extracted features.
  - **Summary**: The first step of the Sfm pipeline includes extracting features from the input images. SIFT is used for this purpose and the extracted features are given as input to the next stage for the pipeline.
- **Feature matching and geometric verification**
  - **Input**: Keypoints, descriptors of extracted features
  - **Output**: Pair of correspondences stored in a data file.
  - **Summary**: The second step includes the process of finding matches and correspondences between feature points in different images. COLMAP provides various feature matching modes, amongst which we have used **Exhaustive Matching** since the number of images that we have are relatively less.
- **Sparse reconstruction**
  - **Input**: Keypoint correspondences
  - **Output**: Sparse 3D point cloud
  - **Summary**: Using the database file COLMAP seeds the reconstruction from an initial pair of images. the scene is incrementally extended by registering new images and triangulating new points.
- **Dense reconstruction**
  - **Input**: keypoint correspondences and sparse cloud.
  - **Output**: Dense 3D point cloud
  - **Summary**: COLMAP has an integrated dense reconstruction pipeline to produce depth and normal maps for all registered images, to fuse the depth and normal maps into a dense point cloud with normal information using the a sparse representation of the scene and the camera poses of the input images.

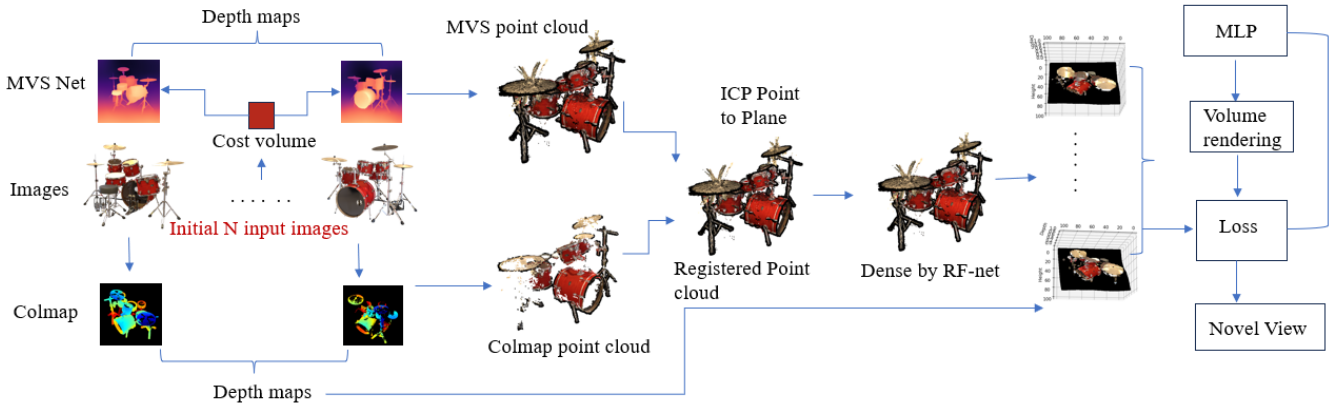


Fig. 1. Project pipeline and progress: Successfully generated point cloud, depth maps

### C. DNN for Point cloud

[28]

Point clouds can also be generated using Deep Neural Networks.We generated this to check the quality and to use the best among three clouds for registration.

First, network generates a depth map using the disparity of the images. Using this depth map, the point cloud is generated. MiDas[29](Multiple Depth Estimation Accuracy with Single network) is a deep learning based residual model built atop Resnet101 for depth estimation. The deep neural network[30] uses an encoder-decoder style architecture.

**Input**: RGB images

**Output**: Depth map of the scene

**Summary**: The loss function is calculated in two parts : First is the scale and shift invariant loss for each sample:

$$\mathcal{L}_{ssi}(\hat{d}, \hat{d}^*) = \frac{1}{2M} \sum_{i=1}^{M} \rho(\hat{d}_i - d_i^*)$$

where $\hat{d}, \hat{d}^*$ are the scaled and shifted versions of predicted and ground truth disparity and $\rho$ defines a specific type of loss function.Second is the scale-invariant gradient matching term to the disparity. The gradient matching term is:

$$\mathcal{L}_{reg}(\hat{d}, \hat{d}^*) = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{M} (|\nabla_x R_i^k| + |\nabla_y R_i^k|)$$

where $R_i = \hat{d}_i - \hat{d}_i^*$ and $R^k$ denotes the difference of disparity maps at scale k. Generally, K=4 scale levels. This halves the image resolution at each level.The final loss function is defined below, where $N_l$ is the training set size and $\alpha$ = 0.5. :

$$\mathcal{L}_1(\hat{d}, \hat{d}^*) = \frac{1}{N_l} \sum_{n=1}^{N} \mathcal{L}_{ssi}(\hat{d}, (\hat{d}^*)^n) + \alpha \mathcal{L}_{reg}(\hat{d}, (\hat{d}^*)^n)$$

### D. Point Cloud registration

Iterative Closest Point algorithm is used for cross source point cloud registration.Point cloud P is treated as fixed one and Q as moving one, where P and Q are points clouds observed from MVSNet and Colmap.We implemented Point to Plane correspondences for fast convergence.

Assuming distances from $p_i$ to $q_j$ be infinite, kdtree is applied to get the k nearest point distances from P to Q for all points.Then selecting n best corresponding points making sure equally spaced among the all points.Normals $n_p = n_x, n_y, n_z$ are estimated to the selected points using the eigen values of the covariance matrix C. Planarity is measured as well.

$$C = \frac{1}{n-1} \cdot (X - \bar{X})^T \cdot (X - \bar{X})$$

$$C = V \Lambda V^T$$

V is a matrix whose columns are the eigenvectors of C, $\Lambda$ is a diagonal matrix containing the eigenvalues.Arbitrary initialization of Transformation T is done for initial estimation of point-to-plane distances between corresponding points P and T.Q.Mean absolute devaition is used for the removing outliers in the point to plane distance estimation.Then estimating Rigid body parameters through the least square estimation.Loss function is optimized through

$$E(T) = \sum_{(p,q) \in K} ((p - Tq) \cdot \mathbf{n}_p)^2$$

Here $n_p$ is the normal at point p $p_i$ and $q_j$ is the normal direction at the point qi. The final transformed moving points is stacked with fixed points to achieve Registered point cloud.

### E. RF net for Point cloud density

The ouput of registered cloud is made dense using RF net.Recurrent Forward Network (RFNet) uses a multi-layered approach to convert a sparse point cloud to a dense point cloud.

**Input**: Sparse point cloud data

**Output**: Dense point cloud data

**Recurrent Feature Extraction**

- Extract and enhance feature representation for dense completion
- Shared Parameters: Parameter-sharing, state features for adaptivity

**Forward Dense Completion**

- Completes the point cloud using seeds and offsets
- **Initializes Cell**:
  - Creates a basic structure, generate seeds
  - Enhance accuracy
- **Decode Cell**:
  - Learn a parameter-shared transformation
  - Predict K offsets for each point

**Raw Shape Protection**

- Prevent results from deviating too far from original shapes
- **Merge Layer**:
  - Introduce original model information, drive points towards neighbors
  - Merge network outputs with incomplete model details
- **Refine Cell**:
  - Fine-tune point positions
  - Concatenate input data, create a global feature for tuning

### *F. NERF implementation on the point cloud*

Given N images along with each pose matrix $P_{4\times4}$, denoted as $N_i$ with dimensions $(H \times W \times 3)$, and the rotation matrix $R_{3\times3}$ obtained from each pose matrix $P_i$, ray marching is performed in the direction of the camera. Images are synthesized by sampling 5D coordinates $(x, y, z, \theta, \phi)$ along camera rays, resulting in a $(H \times W \times D \times 3$ tensor, where $D)$ represents the depth values along the rays for each pixel.

The generated ray marching output $(H \times W \times D \times 3)$ is approximated using an MLP to produce a radiance $\mathbf{c}$ $(R, G, B)$, and a volume density $\sigma$. The MLP, denoted as $F_\Theta$, processes the input 3D coordinate $x$ through 8 fully connected layers with ReLU activations and 256 channels per layer. It outputs $\sigma$ and a 256-dimensional feature vector. This feature vector is concatenated with the camera ray's viewing direction and passed through an additional fully-connected layer with a ReLU activation and 128 channels, producing the view-dependent RGB color.

The radiance field regresses on each point cloud data $p_1, p_2, \ldots, p_{N_q}$ obtained per $q$ views. These point clouds are generated in MVSNet using the unprojection of depth maps.

Using the radiance fields generated by the MLP, volumetric rendering is performed to obtain novel views. The volumetric rendering equation is given by the following -

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i\delta_i))\mathbf{c}_i$$

where,

$$T_i = exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j)$$

Here, $\hat{C}(\mathbf{r})$ is the computed color along ray $\mathbf{r}$, $T_i$ is the probability that the ray did not hit any of the previous bins, and $\mathbf{c}_i$ , $\sigma_i$ and $\delta_i$ is the radiance, volume density and the sampling resolution in the $i^{th}$ bin.

## IV. Implementation Details

Classical NERF takes in RGB values from the image and the 2D volumetric rendered values to get the loss.

$$L = \sum_{r\in\mathcal{R}} ||\hat{I}_v(r) - I(r)||_2^2$$

where $\hat{I}_v(r)$ is predicted from volumteric rendering and $I(r)$ is the reference.Instead of the taking the loss with 2D images, we considered point cloud from each view, which is precisely combined depth map and image, which makes RGBD, each of these RGBD are used for the making of the final point cloud. Instead of considering a whole point cloud, we compared loss with each cloud.

RGB loss being same as the Classical NERF, we implemented mean square loss of depth at different levels and penalized with regularizer[20][22].

$$L = \lambda_1.(\frac{1}{N}\sum_{r\in\mathcal{R}} ||\hat{I}_v(r) - I(r)||_2^2) + \lambda_2.(\frac{1}{N}\sum_{r\in\mathcal{R}} ||\hat{D}_v(r) - D(r)||_2^2)$$

where, $\hat{D}_v(r)$ is depth after volumetric rendering, $D(r)$ being individual point cloud depths at different levels which are referred to as depth clouds in this paper. $\lambda_1$ and $\lambda_2$ are hyper parameters. In most cases, $\lambda_1$ is 1, and $\lambda_2$ acts as a regularizer. Here, $N$ is the cardinality of set $\mathcal{R}$.

The LEGO and Drums datasets' individual point clouds, as indicated in the dataset section below, are created for loss and looks like of low quality data.

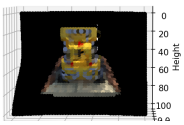


Fig. 2. depth cloud

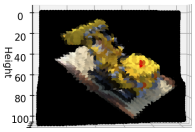


Fig. 3. depth cloud

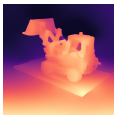

Fig. 4. depth map

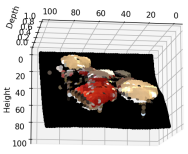


Fig. 5. depth cloud

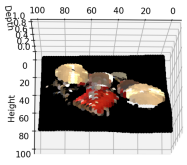


Fig. 6. depth cloud

Fig. 7. depth map

## V. Experiments

### A. Datasets and Network

We are primarily using the Tiny NeRF dataset and the synthetic NeRF dataset to conduct experiments. The synthetic NeRF dataset consists of 8 different objects and each object has a training and testing set. The dataset also has information regarding the camera pose, focal length, and the image transformation matrix for all images of a given object. Each image is of size 800 x 800 pixels.

The two objects that were chosen for our experiments are the LEGO backhoe loader and the drum set. The reason these two objects were chosen is because the LEGO dataset is dense and the drum set dataset is sparse. The LEGO dataset is extensively used as a baseline for checking the quality of rendered images. The drum set dataset has the same number of images as the LEGO dataset.

The tiny NERF dataset [reference] is a version of the original LEGO dataset which keeps quality intact although images are downsampled. The resolution of the images is 100 x 100. The images are stored in the image keys. Along with the individual keys, the poses and the focal length are stored as a .npz file. This file is created for rapid testing.

In our experiments, we emulate the tiny NERF Data dataset and create a similar dataset for the drum set. In essence, our dataset consists of multiple .npz files each having 100 x 100 resolution. The poses and the focal lengths are also saved in the same file.

We also experimented with the NERF neural network architecture. Two architectures containing similar hyperparameters are used one being a 4-layered network and the other being a 9-layered network. The main comparison is to understand the effect of skipped connections.

The skip connections in the NERF MLP are present at the 4th layer and the 9th layer. We divided it into different sets of layers and experimented with layer size, point cloud density and RGB and RGBD loss functions back propagations in the following sections.

The LEGO dataset consists of 100 training images and 200 testing images. Some examples of the images are shown below:

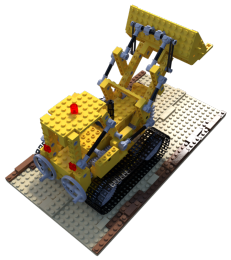

Fig. 8. LEGO

Fig. 9. DRUMS

The following figure depicts the architecture of the NeRF MLP.

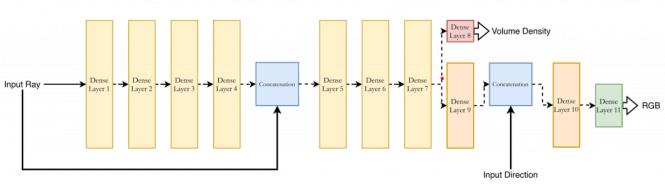


Fig. 10. MLP

### B. Training and results of LEGO dataset

*1) 4 Layer results:* The following (fig 11) is the plot of the 4-layered network when we back-propagated classical NERF loss. In parallel, we collected our loss between the point cloud and the rendered point cloud from volumetric rendering. We observed that although RGB loss is converging, depth from the point cloud is not fully converging and is greater than RGB loss in the case of the LEGO dataset. Loss is compared for 5000 epochs and 128 depth layers for all the 4-layered networks. We have also trained for 10000 and 20000 epochs for certain outputs.

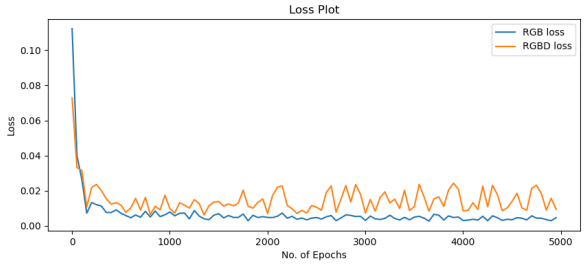


Fig. 11. LEGO dataset with RGB loss back propagation

In the case of the drum set dataset, (fig 12) the depth loss is significantly lesser due to the sparse nature of points available. Although loss is converging for both RGB and RGBD, on the whole, it makes penalized RGBD loss lower than classical NERF loss.

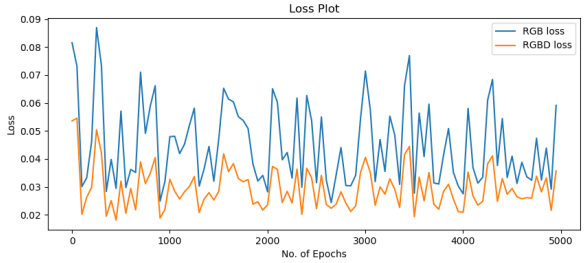


Fig. 12. Drums dataset with RGB loss back propagation

This plot (fig 13) is when we back-propagate our point cloud(RGBD) loss while collecting classical NERF loss (RGB). We observed that RGB along depth in the point cloud is converging which produces better results. Due to back-propagating depth along with RGB, both converge with an expectation of better results.

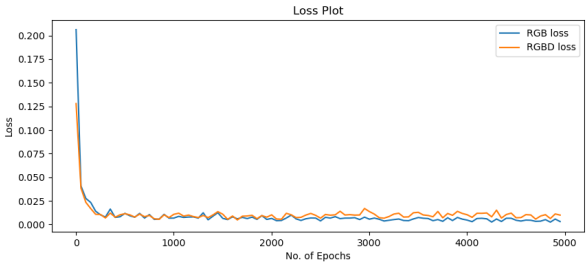


Fig. 13. LEGO dataset with RGBD loss back propagation

In the case of the drum set dataset, (fig 14) the RGBD is still lesser than the RGB loss, maintaining less loss in depth. It stabilized well in comparison with RGB loss back-propagation. Convergence is also much better in this case compared to Figure 12.

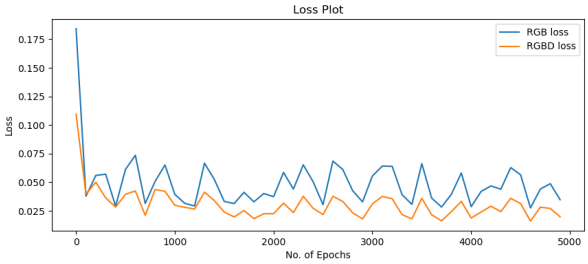


Fig. 14. Drums dataset with RGBD loss back propagation

Overall, the 4-layer network showed promising results for better performance of loss convergence and generating novel images. This is shown in the conclusion.

*2) 9 Layer results:* The following is the plot (fig 15) of a 9-layered network when we back-propagate classical NERF (RGB) loss. In parallel, we have collected our loss (RGBD) between the point cloud and the rendered point cloud from volumetric rendering. The number of epochs was 200 for 9 all layer network for the loss comparison. Due to the residual layer, we observe that the RGB loss converges and the depth loss from the point cloud (RGBD) also converges.

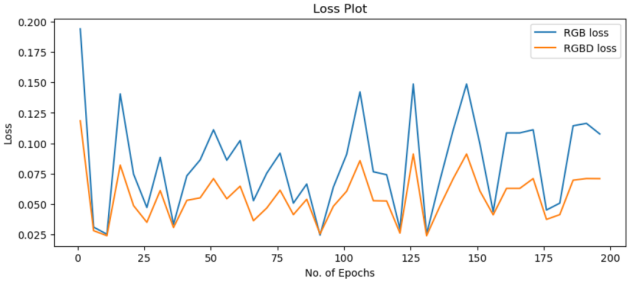


Fig. 15. LEGO 9 layer RGB loss

With the drum set dataset (fig 16), again RGBD loss is lesser than the RGB loss. This is due to 2 reasons. One being residue and the other being low depth loss due to sparse points. A supporting plot (fig 17) to low depth loss is shown below.

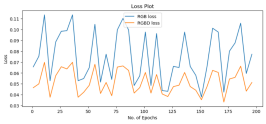

Fig. 16. Drums 9 layer RGB loss

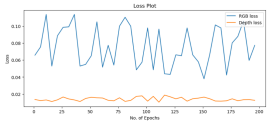

Fig. 17. RGB and Depth loss in 9 layer

This plot (fig 18) shows when we back-propagated point cloud loss (RGBD) and the classical NERF loss (RGB) for the LEGO dataset. We observe that RGB along depth in the point cloud is converging as the case with the RGB loss. We can observe that the RGB and the RGBD converge in the same pattern whereas more fluctuation was there in RGB and depth loss in Figure 15.

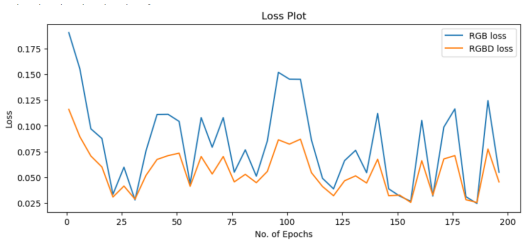


Fig. 18. LEGO 9 layer RGBD loss

Here it's the same as above back propagating RGBD loss while collecting RGB loss in parallel, but with the drum data set. Again loss function doesn't perform well in convergence for both losses, although it converges for RGBD. With the sparse cases (less amount of objects in the image) more hyperparameter tuning is required to get a better convergence of losses.

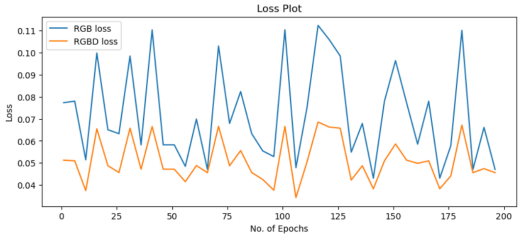


Fig. 19. RGB and Depth loss in 9 layer network by backpropogating our loss

*3) Point Cloud registration:* Instead of overall point cloud registration, we perform individual depth cloud registration. As the ground truth point cloud is 100 x 100 x $(R, G, B, D)$. The registered dense depth point cloud is 100 x 100 x ( $R_1, G_1, B_1, D_r, D_{V_r}$) where $D_r$ is the registered depth and the $D_{V_r}$ is registered depth for each point $X, Y$.

Figure 20 shows the complete dense depth cloud analysis. In each ray, some points are empty, one point is from the depth

point cloud. One or more points may be from the dense RF net which in turn gets registered with a small error just beside the original point as shown.

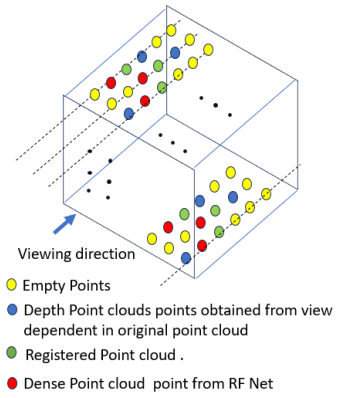


Fig. 20. Representation of Depth point cloud

The following (fig 21) is the loss plot between the registered dense depth point cloud and the general depth point cloud. Registered loss is almost equal to or greater than the general loss as it has more depth points due to density increment and registration. This increased loss at a few points and where there were no additional points were generated, it stayed the same as the general depth point cloud loss.

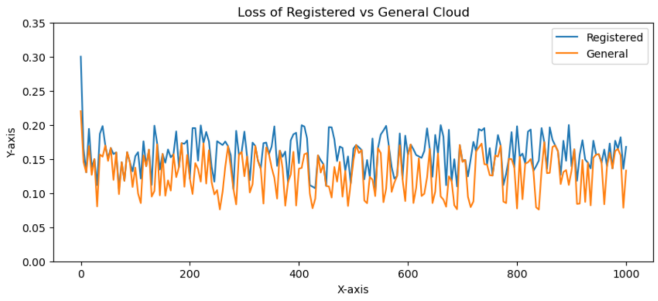


Fig. 21. Loss of Registered vs General point cloud

### C. Novel Image results

The following results are with LEGO dataset, comparison between original image, point cloud generated, 4 layered and 9 layered outputs. Although the point cloud is decent, we implemented 10000 iterations on 4-layered and 9-layered networks for depth layers of 128.

Fig. 22. Input LEGO image

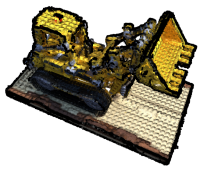

Fig. 23. Input LEGO point cloud

The generated outputs of the 4 layered with classic NERF loss, 4 layered with our loss, and 9 layered Our loss are shown below. There is improvement in the 4-layered network due to better depth loss convergence, but the 9-layered network is almost the same and we may get better results if we had better resolution.

Fig. 24. Classic loss 3 layers

Fig. 25. Our Loss 3 layers

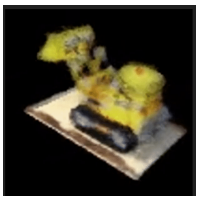

Fig. 26. Input LEGO point cloud

There is no tiny dataset available for drums, we implemented 150000 iterations to get a subpar output in 3-layered and 9-layered networks. With sparse downsampled images, it was very challenging to run a lot of iterations with the heavy computation time of over 10 hours on GPU. The input image, point cloud, and respective results of 3-layered and 9-layered are shown below. The novel image results of drums are not up to mark, though we compared the point cloud technique with the classical image technique using loss graphs.

Fig. 27. Input Drums image

Fig. 28. Drums Input Point cloud

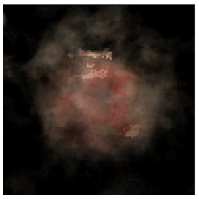

Fig. 29. Drums Input Point cloud

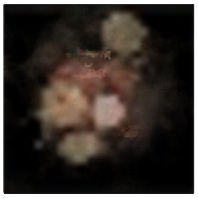

Fig. 30. Input Drums image

Fig. 31. Input Drums image

## VI. Conclusion and Future Work

Overall, in our paper, we present a hybrid approach of using loss function with point clouds. We had success with the loss function for dense or better-resolution images of tiny nerf. Our network is performing well for the 4 layers network and giving at-par results to classic NERF. For the 9-layered network, although it gave similar results, it can still be tested on good-resolution images to get competitive results. Some future scope work is to tune the hyperparameter of the regularizer in the loss function with a sparse dataset to get better results.

## REFERENCES


[1] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., Ng, R. (2020, August). Nerf: Representing scenes as neural radiance fields for view synthesis. In European conference on computer vision.

[2] Zimny, Dominik, T. Trzciński, and Przemyslaw Spurek. "Points2nerf: Generating neural radiance fields from 3d point cloud." arXiv preprint arXiv:2206.01290 (2022).

[3] Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U. (2022). Point-nerf: Point-based neural radiance fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5438-5448).

[4] Huang, T., Zou, H., Cui, J., Yang, X., Wang, M., Zhao, X., ... Liu, Y. (2021). Rfnet: Recurrent forward network for dense point cloud completion. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 12508-12517).

[5] Yi, H., Wei, Z., Ding, M., Zhang, R., Chen, Y., Wang, G., Tai, Y. W. (2019). Pyramid Multi-view Stereo Net with Self-adaptive View Aggregation. arXiv preprint arXiv:1912.03001.

[6] Aliev, K. A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V. (2020). Neural point-based graphics. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16 (pp. 696-712). Springer International Publishing.

[7] Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P. P. (2021). Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. arXiv preprint arXiv:2103.13415.

[8] Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J. T., Srinivasan, P. P. (2022). Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5491-5500).

[9] Deng, K., Liu, A., Zhu, J. Y., Ramanan, D. (2022, June). Depth-supervised NeRF: Fewer Views and Faster Training for Free. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 12872-12881). IEEE Computer Society.

[10] Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H. (2021). Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 14124-14133).

[11] Thies, J., Zollh¨ofer, M., Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. In: Proc. SIGGRAPH. (2019)

[12] Garbin, S. J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J. (2021). Fastnerf: High-fidelity neural rendering at 200fps. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 14346-14355).

[13] Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., Zhou, J. (2021). Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5610-5619).

[14] C. Chen, Z. Han, Y. -S. Liu and M. Zwicker, "Unsupervised Learning of Fine Structure Generation for 3D Point Clouds by 2D Projection Matching," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 12446-12457, doi: 10.1109/ICCV48922.2021.01224.

[15] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In European Conference on Computer Vision (ECCV), 2016

[16] Choy, C., Dong, W., Koltun, V. (2020). Deep global registration. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 2514-2523).

[17] Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F. (2021). D-nerf: Neural radiance fields for dynamic scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10318-10327).

[18] Wang, Z., Wu, S., Xie, W., Chen, M., Prisacariu, V. A. (2021). NeRF–: Neural radiance fields without known camera parameters. arXiv preprint arXiv:2102.07064.

[19] Chengzhou Tang and Ping Tan. BA-net: Dense bundle adjustment network. In Proc. ICLR, 2019.

[20] Kukačka, J., Golkov, V., Cremers, D. (2017). Regularization for deep learning: A taxonomy. arXiv preprint arXiv:1710.10686.

[21] tiny nerf data

[22] Yang, T., Zhu, S., Chen, C. (2020). Gradaug: A new regularization method for deep neural networks. Advances in Neural Information Processing Systems, 33, 14207-14218.

[23] NERF synthetic Data

[24] Rosinol, A., Leonard, J. J., Carlone, L. (2022). Nerf-slam: Real-time dense monocular slam with neural radiance fields. arXiv preprint arXiv:2210.13641.

[25] Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P. P. (2021). Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5855-5864).

[26] Rosu, R. A., Behnke, S. (2022, July). Neuralmvs: Bridging multi-view stereo and novel view synthesis. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.

[27] Chiang, C. H., Kuo, C. H., Lin, C. C., Chiang, H. T. (2020). 3D point cloud classification for autonomous driving via dense-residual fusion network. IEEE Access, 8, 163775-163783.

[28] Liu, B., Chen, X., Han, Y., Li, J., Xu, H., Li, X. (2019).

Accelerating DNN-based 3D point cloud processing for mobile computing. Science China Information Sciences, 62, 1-11.

[29] Wang, Z., Zhang, L., Zhang, L., Li, R., Zheng, Y., Zhu, Z. (2018). A deep neural network with spatial pooling (DNNSP) for 3-D point cloud classification. IEEE Transactions on Geoscience and Remote Sensing, 56(8), 4594-4604.

[30] Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P., Song, S. (2019). Deepvcp: An end-to-end deep neural network for point cloud registration. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 12-21).