
TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

MAJOR PROJECT REPORT

ON

# 3D RECONSTRUCTION BASED VIRTUAL TOUR

**SUBMITTED BY:**

BISHAD KOJU (PUL075BCT025)

GAURAV JYAKHWA (PUL075BCT037)

KRITI NYOUPANE (PUL075BCT043)

LUNA MANANDHAR (PUL075BCT047)

**SUBMITTED TO:**

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

April 2023

# Page of Approval

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"3D Reconstruction Based Virtual Tour"** submitted by **Bishad Koju**, **Gaurav Jyakhwa**, **Kriti Nyoupane**, **Luna Manandhar** in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
**Jyoti Tandukar**
Associate Professor
Department of Electronics and
Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
**Person B**
Assistant Professor
Department of Electronics and
Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner
**Person C**
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

# Copyright

# Acknowledgement

We express our sincere gratitude to the Department of Electronics and Computer Engineering, Central Campus, Pulchowk, for allowing us to choose the major project per the fourth-year syllabus at IOE, Tribhuvan University.

We want to express our deepest gratitude to our supervisor Assoc. Prof. Jyoti Tandukar, Ph.D., for his continuous guidance, direction, and supervision. He has been exceptionally motivating and helpful to us while giving guidance, suggestion, and pointing out the optimal path, which helped us progress our project rapidly.

We want to express our sincere gratitude to Prof. Danda Pani Paudel, Ph.D., for the guidance and support throughout our project.

We thank all faculty teachers who provided us with the knowledge to build a foundation, which helped us study the feasibility of our major project and ignited our passion for doing this great project. We would also like to thank our parents and friends who've been exceptionally motivating and accompanying us throughout our work on the project.

Gratitude,

Bishad Koju

Gaurav Jyakhwa

Kriti Nyoupane

Luna Manandhar

# Abstract

Number of research and several methods have been proposed for 3D reconstruction from 2D images. The first is a triangulation approach based on determining the same points in images taken from different angles to approximate a point cloud in 3D space and then reconstructing the mesh. This is purely a computation-based approach. Another approach is to redefine 3D reconstruction problems as recognition problems and use the existing knowledge about 3D space and projection to reconstruct, much like how humans do. This knowledge is approximated using deep learning models. However, in these approaches, the mesh reconstruction part is extremely expensive. This cost can be reduced by trying to reconstruct the view rather than trying to reconstruct the mesh. Neural Radiance Field (NeRF) has been used to generate novel views. NeRF represents a scene using a fully-connected deep network, whose input is a spatial location and viewing direction and output is the volume density and view-dependent emitted radiance at that spatial location. We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. In this project, we have used the latter approach.

Keywords: *3D Reconstruction, NeRF, Point Cloud, Deep Learning*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **2D** | 2 Dimension |
| **3D** | 3 Dimension |
| **4D** | 4 Dimension |
| **5D** | 5 Dimension |
| **CPP** | C Plus Plus |
| **EVQ** | Enhanced Vector Quantizations |
| **FCN** | Fully Convolutional Network |
| **GPU** | Graphics Processing Unit |
| **LPIPS** | Learned Perceptual Image Patch Similarity |
| **MSE** | Mean Squared Errod |
| **MLP** | Multi-Layer Perceptron |
| **NeRF** | Neural Radiance Fields |
| **NN** | Neural Network |
| **OpenCV** | Open Computer Vision |
| **ORB** | Oriented Fast and Rotated Brief |
| **PSNR** | Peak Signal to Noise Ratio |
| **RBFNN** | Radial Basis Functional Neural Networks |
| **RGB-D** | Red Green Blue - Depth |
| **RNN** | Recurrent Neural Network |
| **SSIM** | Structural Similarity Index |
| **SFM** | Structure From Motion |
| **SIFT** | Structure Invariant Feature Transform |
| **SLAM** | Simultaneous Localization And Mapping |
| **SRN** | Scene Representation Network |

# 1. Introduction

## 1.1 Background

Making a 3D representation of an object from 2D photos that are as accurate as feasible is one of the most fundamental difficulties in image-based modeling and computer vision. It is the opposite of projecting a 3D object into a 2D environment. By using prior knowledge of the structure and shape of previously viewed objects, humans can resolve this issue with ease. With just one eye, they can roughly estimate the size and geometry of any object in three dimensions. However, computers lack this inherent knowledge of 3D space, making it challenging to extract the depth information needed for 3D reconstruction from 2D photos.
Numerous techniques for reconstructing 3D space from 2D photos have been presented, each with its own execution requirements, benefits, and drawbacks. To create algorithmic answers to the poorly presented inverse problem, the initial generation of techniques focused on comprehending and theoretically formalizing the 3D to 2D projection process. These methods approached the problem geometrically. By redefining the 3D reconstruction problem as a recognition challenge and utilizing preexisting expertise, the second-generation technique attempts to tackle this problem in a manner similar to how humans do it. This is accomplished by utilizing deep learning models. This method relies on a sizable training dataset because mesh-based 3D reconstruction is computationally expensive. This can be reduced by a view synthesis rather than trying to reconstruct the mesh.

## 1.2 Objectives

The objectives of the project are:

- To be able to simultaneously localize and map the environment.
- To synthesize a realistic 3D-aware view from the images of the given scene based on the mapping.

## 1.3 Problem Statement

With numerous uses in fields like gaming, virtual reality, and film production, the creation of 3D models has been a long-standing issue in the field of computer graphics. The manual creation of detailed models using conventional methods is labor- and time-intensive and frequently requires skilled artists and designers. As a result, automated 3D modeling methods have been created, with the goal of streamlining the creation of 3D models by utilizing recent developments in computer vision and machine learning. But using these automated technologies is still computationally expensive.

## 1.4 Scope of Project

A critical technology with widespread uses in many industries is 3D reconstruction. The capacity to create 3D models of the environment is essential for developing many sectors, from virtual tourism to augmented reality, from medicine to digital marketing, robotics mapping to gaming, and more. For instance, the creation of 3D models has expanded the possibilities for displaying items in a virtual exhibition in the fields of virtual environments and augmented reality. By taking pictures with a camera, an environment's 3D model can be built, and new items can then be added to this 3D-aware environment. By blending the actual world and the virtual one, this method gives viewers a more immersive and interactive experience.

The capacity to create 3D worlds has completely changed how we encounter digital objects within the realm of virtual reality. High-quality 3D models can be used to create immersive experiences that allow users to explore historical sites as if they were truly there. The ability to interact with virtual objects and environments as if they were real has created new opportunities in entertainment, education, and training.

# 2. Literature Review

## 2.1 Related Theory

The pipeline for 3D reconstruction consists of two main steps: mesh/texture extraction and point cloud creation. In 1999, David G. Low used Scale Invariant Feature Transform (SIFT), which enables corresponding features to be matched even with significant variations in scale and viewpoint, as well as under circumstances of partial occlusion and changing illumination, to match corresponding features in images and measure distances between them on the camera image plane d, d'. [1].

To discover interest points for the object recognition problem in 1997, Schmid & Mohr also used the Harris corner detector. From an orientation-invariant vector of derivative-of-Gaussian image measurements, they created a local picture descriptor at each interest point. These picture descriptors were used for accurate object detection by looking for a number of matched descriptors that complied with object-based alignment and placement limitations. This research was noteworthy for its capability to handle congested photos and for how quickly it could identify things in a sizable database [2].

Using the improved vector quantization (EVQ) technology, Stefano Ferrari and colleagues took an approach to reduce and filter the 3D point cloud in 2007 [3]. This phase must be completed before starting the mesh reconstruction process. In 2010, Fengxia Li et al. employed the Radial Basis Function Neural Network (RBFNN) technique to uniformly sample the point cloud in terms of the fitting line via nonlinear least square, reduce the point cloud, and then fill the holes that were the defect on the surface produced by the scanner [4]. But the outcome was still a point cloud. In 2011 Kun Zhou and colleagues introduced the octree structure to carry out the current Poisson surface reconstruction approach on the GPU [5].

The solution to more realistic transitions was to determine the depth of each pixel

using computer vision algorithms. In order to do this, it was necessary to examine each set of overlapping photos and compare items to determine their distance from the camera. The researchers were able to streamline the data and create 3-D surfaces from a comparatively small number of planes by computing depth for every pixel [6].

Realistic scenes with intricate geometry have not yet been able to be replicated using these techniques. A technique based on neural rendering was employed for 3D context-aware view creation rather than attempting to extract the 3D geometry. In [7], a technique called Neural Radiance Fields for View Synthesis was put out that, by using a small number of input views to optimize an underlying continuous volumetric scene function, produced state-of-the-art results for creating innovative views of complicated scenes. The views were generated via 5D coordinate queries along camera rays, and the resultant colors and densities were projected into an image using traditional volume rendering techniques.

The theoretical foundation for NeRF gets back to 1908 with the introduction of the plenoptic function.The plenoptic function describes the degrees of freedom of a light ray as a wave with the parameters: Irradiance (aka brightness), position, wavelength (aka color), time, angle, phase, polarization, and bounce. The equation was later simplified by removing polarization, bounce, time, and phase. But it was done in a discrete space so, NeRF presents a novel approach that outperforms existing methods in generating new views of intricate scenes. The technique involves optimizing a continuous volumetric function of the scene, with the help of only a small number of input views.[8]

BakedSDF is based on learning signed distance functions (SDFs) and is optimized for high-quality shape reconstruction and rendering. BakedSDF represents shapes as an SDF volume and a set of texture maps and can be used for a range of applications including shape editing, rendering, and animation. The authors build upon previous work in the field of shape representation and propose a novel method for learning SDFs that captures both local and global shape information. They train their model using a combination of supervised and unsupervised learning and introduce a novel optimization method that improves the quality of the SDFs and texture maps.[9]

## 2.2 Related Work

### 2.2.1 Microsoft's Photosynth

Microsoft's Photosynth was one of the most popular 3D model reconstruction programs on the Internet. The Photosynth cloud service allows users to upload many images of the same object or real location, which is then processed into a "synth"—a composite of overlapping photos that creates a 3-D model of the location with extra depth and transitional images for natural 3-D viewing[10]. The paper titled Piecewise Planar Stereo for Image-based Rendering proposed a way to create more realistic transitions when moving from photograph to photograph i.e panorama mode. Essentially, the flat screen after the flat screen was joined together. But in real life, you'd see objects from different angles and depths as you move along [11].

### 2.2.2 Variants of NeRF

MLP NeRFs can capture complex geometry, appearance, and lighting effects in a single model, they are computationally expensive and slow to train and render. To address these challenges, recent methods have proposed using alternative representations such as voxel grids [12], grids of small MLPs [13], low-rank [14] or multiscale hash encoding with a small MLP[15]. Karnewar et al. proposed using voxel grids to represent the scene in their paper "GridNeRF: Fast Neural Radiance Fields from Voxel Grids" cite[12]. Sun et al. also used voxel grids in their work "NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis". Reiser et al. proposed using a grid of small MLPs to represent the scene in their paper "VLocNet++: Deep Voxel Localization Networks for Large-Scale Indoor and Outdoor Navigation"[13]. Chen et al. proposed using a low-rank grid representation in their paper "Low-Rank Neural Radiance Fields"[14] Müller et al. proposed using a multiscale hash encoding equipped with a small MLP in their paper "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation"[15].

### 2.2.3 Nerfstudio

Neural radiance fields (NRFs), a kind of deep learning model for 3D scene modeling and rendering, can be created using the modular Nerfstudio platform. NRFs are a hopeful tool for applications involving virtual and augmented reality because they can represent geometry, appearance, and lighting effects in a single model. The authors suggest a modular framework made up of scene representation, view synthesis, and rendering modules, building on earlier research in the area of neural radiance fields. The view synthesis and rendering modules are trained using the continuous volumetric representation that the scene representation module creates from 3D point clouds or models. The rendering module creates photorealistic views while the view synthesis module creates novel views of the scene from arbitrary viewpoints.[16]

# 3. Theoretical Background

## 3.1 ORB SLAM

The computational challenge of creating or updating a map of an unknown environment while simultaneously localizing an agent's location within it is known as simultaneous localization and mapping (SLAM). It is mostly used in autonomous vehicles and robots. The data is from various sensors like LIDAR, GPS, camera, Kinect, etc can be used for SLAM. Visual SLAM (VSLAM) is a subset of SLAM that uses data from visual sensors only. These techniques depend upon image processing for currently identifying various points in multiple images. One of the efficient VSLAM algorithms is ORB SLAM.

ORB SLAM (Oriented Fast and Rotated Brief SLAM) is a versatile algorithm capable of performing SLAM in real-time, and also able to perform loop-closing for a large area. Oriented fast is a fast and robust key point detector and while Rotated Brief is a visual descriptor that provides a unique identity to the detected key points. ORB SLAM first extracts ORB (Oriented FAST and rotated BRIEF) features then use a fast approximate nearest neighbor algorithm. After that PROSAC (Progressive Sample Consensus) algorithm is used for point matching in different images. For localization, the bundle adjustment algorithm is used to estimate the camera localization for sparse geometrical reconstruction. This process of simultaneously mapping the environment and localizing the camera pose is iteratively run on a loop, each progressive run creating a more accurate estimation, till the desired accuracy is achieved.

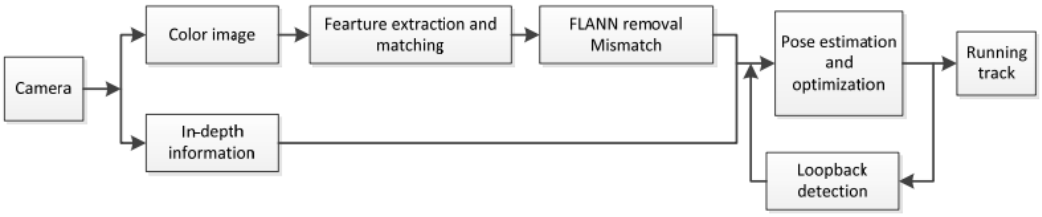


Figure 3.1: ORB SLAM Architecture

Image Source: Qiang Li, Jia Kang, Yangxi Wang, and Xiaofang Cao. An improved feature matching orb-slam algorithm. In Journal of Physics: Conference Series, volume 1693, page 012068. IOP Publishing, 2020.18

## 3.2 COLMAP

COLMAP is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline for reconstructing 3D models from ordered and unordered image collections. The method of reconstructing a 3D structure from its projections into a series of images is known as "Structure-from-Motion" (SfM). The input consists of a collection of sequential photos of the same object taken from various positions. The output is a 3-D reconstruction of the object, and the reconstructed intrinsic and extrinsic camera parameters of all images.

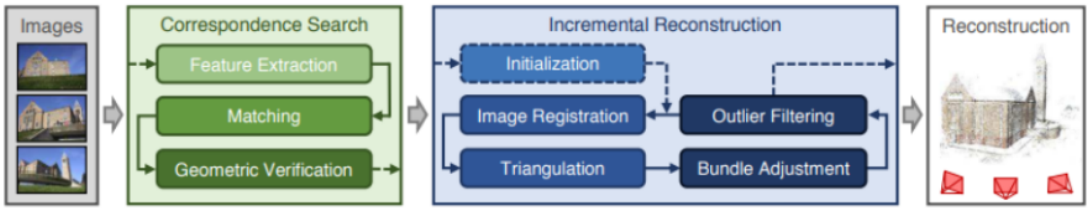


Figure 3.2: Structure From Motion (COLMAP)

Image Source: Schonberger, Johannes L., and Jan-Michael Frahm. "Structure-from-motion revisited." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

The first steps are often feature extraction and matching, then geometric verification. The resulting scene graph serves as the foundation for the rebuilding stage. Correspondence search refers to feature extraction, matching, and geometric verification. In overlapping images, correspondence search identifies scene overlap and

detects projections of the same points.

### 3.2.1 Feature Extraction

SfM finds a set of local features for each image at the locations indicated by an appearance description. So that SfM can detect the features in the image uniquely across different photos, the characteristics in the image must be invariant across radiometric and geometric changes.

### 3.2.2 Matching

Matching is the process of using features as an appearance description of the images to find images that share the same scene feature. The naive approach compares the appearance of each feature in each image pair to locate the most identical feature in order to find feature correspondences. This method checks each image pair for scene overlap.

### 3.2.3 Geometric Verification

In the third stage, the potentially overlapping image pairings are confirmed. Due to the fact that matching is only based on appearance, it is not guaranteed that related features will actually map to the same scene point. SfM attempts to estimate a transformation that uses projective geometry to map feature points between images to confirm the matches. Depending on how they are arranged spatially, several mappings describe the geometric relationship between a pair of doubles.

## 3.3 Affine Transformations

In the real world, several entities are arranged together to create a scene. Further, these entities are themselves collections of smaller parts that are assembled together and these objects are defined relative to each other. Complex pictures can be treated as a combination of straight lines, circles, ellipses, etc. and if these basic figures can be generated then combinations of them can also be generated. Transformation in computer graphics is one of the basic operations that is performed to change the position and orientation of an object. The set of operations providing for all such transformations is known as the affine transforms. The affines include

translations and all linear transformations, like scale, rotation, and shear.

Consider a point X= (x,y) in 2D space. The affine transformations of x are all transforms that can be written as following

$$X' = \begin{bmatrix} ax + by + c \\ dx + ey + f \end{bmatrix} \tag{3.1}$$

where a,b,c,d,e and f all are scalars.

Consider a,e=1 and b,d=0 then result is a pure translation

$$X' = \begin{bmatrix} x + c \\ y + f \end{bmatrix} \tag{3.2}$$

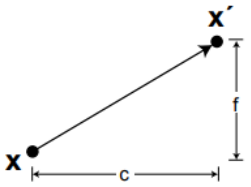


Figure 3.3: 2D Translation

Consider b,d,c,f=0 then result is a pure scaling

$$X' = \begin{bmatrix} ax \\ ey \end{bmatrix} \tag{3.3}$$

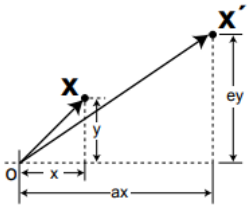


Figure 3.4: 2D Scaling

And, if a,e = $\cos\theta$, b = $-\sin\theta$, d = $\sin\theta$, and c, f = 0, then we result is a pure

rotation about the origin.

$$X' = \begin{bmatrix} x\cos\theta \text{ - } y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix} \tag{3.4}$$

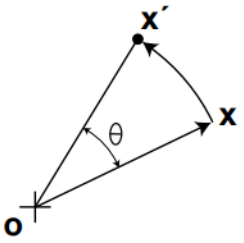


Figure 3.5: 2D Rotation

If a,e=1, and c,f=0 then result is a shear transformation

$$X' = \begin{bmatrix} x + by \\ y + dx \end{bmatrix} \tag{3.5}$$

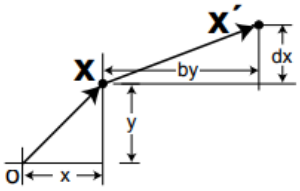


Figure 3.6: 2D Shearing

It would be very helpful to be able to represent all affine transforms by matrices since the matrix form is so useful for creating complex transforms from simpler ones. Matrix addition is required for translation, but matrix multiplication is required for scaling and rotation. Translation is not a linear transform, which is the problem in this situation. The solution to this dilemma is to convert the 2D problem into a 3D problem in homogeneous coordinates.

For this each Cartesian co-ordinate position (x,y) is represented with homogeneous triple coordinate ($x_h$,$y_h$,h) where x=$\frac{x_h}{h}$ and y=$\frac{y_h}{h}$. Thus, general homogeneous co-ordinate representation can also be written as ($x_h$,$y_h$,h)=(h.x,h.y,h)

where h may be any non zero value. But for convenience h=1 is used. So, 2D position is represented with homogeneous coordinates (x,y,1)

Now, the affine matrix can be represented in homogeneous matrix as following

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.6}$$

This is the same result as in 2D, with the exception of the extra w coordinate, which remains 1. The homogeneous representation place all the 2D points on the plane w = 1 in 3D space, and further all the operations are done on this plane but the operations are still 2D operations.

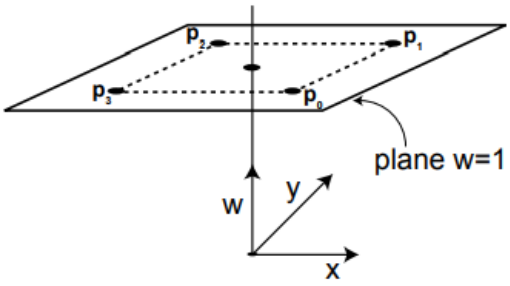


Figure 3.7: Homogenous Representation for 2D Coordinates

Image Source: `https://people.cs.clemson.edu/~dhouse/courses/401/notes/affines-matrices.pdf`

We can extend all of these ideas to 3D in the following way

- Convert all the 3D points to homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.7}$$

  The extra (4th) coordinate is again called the w coordinate

- Use matrices to represent the 3D affine transforms in homogeneous form

Any combination of translation, rotations, scalings and shearing in 3D space can be combined in a single 4 by 4 affine transformation matrix as shown above. The

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.8: Affine Matrix

matrix shown above represents the affine matrix. The last column of the matrix represents a translation (brown rectangle), and the upper-left 3 x 3 sub-matrix (red rectangle) represents a rotation transform. When used as a coordinate system, the upper-left 3 x 3 sub-matrix represents an orientation in space while the last column vector represents a position in space.

## 3.4 Quaternion

Quaternions provide another way to describe rotations. A quaternion is a four-dimensional quantity somewhat similar to a complex number. A complex number has an imaginary part i in addition to its real part, whereas a quaternion has a vector component with three imaginary parts i, j, and k. A quaternion is represented as $q{=}q_0{+}q_1i{+}q_2j{+}q_3k$

### 3.4.1 Convert Quaternion to Rotation Matrix

Given the rotation quaternion $q = (q_0, q_1, q_2, q_3)$, the corresponding rotation matrix is:

$$R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \tag{3.8}$$

Then the affine matrix can be obtained using the above rotation matrix and the

translation vector T=$(t_x, t_y, t_z)$ as following

$$R = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 & tx \\ 2q_1q_2+2q_0q_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3-2q_0q_1 & ty \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & 1-2q_1^2-2q_2^2 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

### 3.4.2 Convert Rotation Matrix to Quaternion

Given the rotation matrix R:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.10}$$

We can find the equivalent quaternion using two steps.

1. Find the magnitude of each quaternion component. This leaves the sign of each component undefined.
   $|q_0| = \sqrt{\frac{1+r_{11}+r_{22}+r_{33}}{4}}$
   $|q_1| = \sqrt{\frac{1+r_{11}-r_{22}-r_{33}}{4}}$
   $|q_2| = \sqrt{\frac{1-r_{11}+r_{22}-r_{33}}{4}}$
   $|q_3| = \sqrt{\frac{1-r_{11}-r_{22}-r_{33}}{4}}$

2. To solve for the sign, find the largest of q0, q1, q2, q3 and assume its sign is positive. Then calculate the remaining components as shown in the table below. Using the maximum size avoids division by small numbers that would reduce the precision of the number.

| If $q_0$ is largest: | If $q_1$ is largest: | If $q_2$ is largest: | If $q_3$ is largest: |
|---|---|---|---|
| $q_1 = \frac{r_{32}-r_{23}}{4q_0}$ | $q_0 = \frac{r_{32}-r_{23}}{4q_1}$ | $q_0 = \frac{r_{13}-r_{31}}{4q_2}$ | $q_0 = \frac{r_{21}-r_{12}}{4q_3}$ |
| $q_2 = \frac{r_{13}-r_{31}}{4q_0}$ | $q_2 = \frac{r_{12}+r_{21}}{4q_1}$ | $q_1 = \frac{r_{12}+r_{21}}{4q_2}$ | $q_1 = \frac{r_{13}+r_{31}}{4q_3}$ |
| $q_3 = \frac{r_{21}-r_{12}}{4q_0}$ | $q_3 = \frac{r_{13}+r_{31}}{4q_1}$ | $q_3 = \frac{r_{23}+r_{32}}{4q_2}$ | $q_2 = \frac{r_{23}+r_{32}}{4q_3}$ |

Figure 3.9: Rotation Matrix to Quaternion

Image Source: https://danceswithcode.net/engineeringnotes/quaternions/quaternions.html

## 3.5 Multi Layer Perceptron

An Artificial Neural Network (ANN) or simply a Neural Network(NN) is interconnected layers of small units called nodes that perform mathematical operations to detect patterns in data. The way NN algorithms are built reflects the way that human neurons function. A Neural Network is composed of multiple neurons arranged into layers. The building blocks of a Neural Network are as follows:

1. Neuron: It is a fundamental NN building block. It receives weighted values, applies mathematical calculations, and generates results. Other names for it are unit, node, and perceptron.

2. Input layer: The first layer, also known as the input layer, consists of n nodes for an n-dimensional input.

3. Output Layer: The output layer, on the other hand, consists of t neural units for a t-dimensional output.

4. Hidden Layer: Any layers between the input and output layers are called hidden layers, and the number of hidden layers determines the depth of the Neural Network.

5. Weights: These values describe how strong (important) a connection is between any two neurons.

6. Bias: It is a constant value that is added to the product of the input values and their corresponding weights. It is used to fasten or slow down the activation of a specific node.

7. Activation function: It is used to determine the output of the neural networks like yes or no. The resulting values are mapped between 0 and 1 or -1 and 1, etc (depending upon the function).

### 3.5.1 Sigmoid or Logistic Activation Function

The Sigmoid Function curve has an S-shaped shape. The sigmoid function outputs between (0 to 1). As a result, it is particularly used for models whose output is a probability prediction. The sigmoid is the best option because anything has a probability that only occurs between 0 and 1.

### 3.5.2 ReLU (Rectified Linear Unit) Activation Function

ReLU is the activation function that is employed the most globally. Since practically all convolutional neural networks and deep learning systems employ it. The ReLU is half rectified (from the bottom). When z is less than zero, f(z) equals zero, and when z is more than or equal to zero, f(z) equals z.

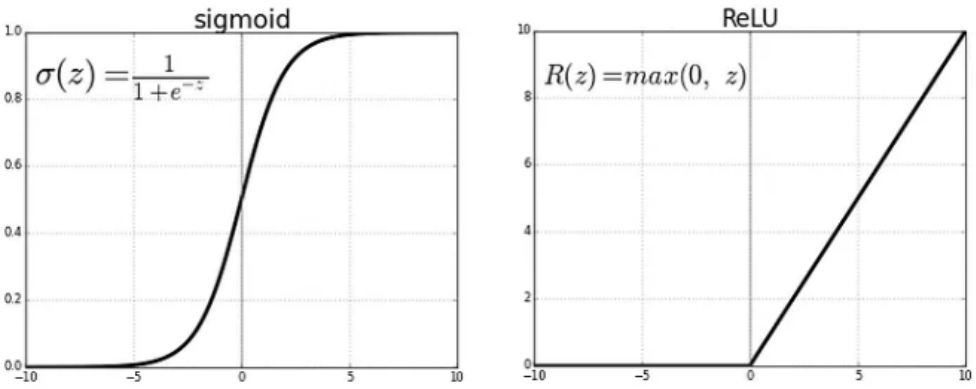


Figure 3.10: ReLU vs Sigmoid

Image Source: `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`

Note: The trainable parameters in NN are weights and biases; i.e. the network learns patterns by adjusting these parameters to get the best predictions.
An artificial neuron receives input values (which may include multiple values with weights attached). The weighted inputs are added up inside the node, and an activation function is then used to produce the outputs. The node's output is transmitted to the other nodes or, in the case of the network's final layer, becomes

the network's overall output. The basic architecture of neural networks contains a single input and output layer with multiple hidden layers. The following diagram shows the architecture of a neural network with 2 hidden layers.

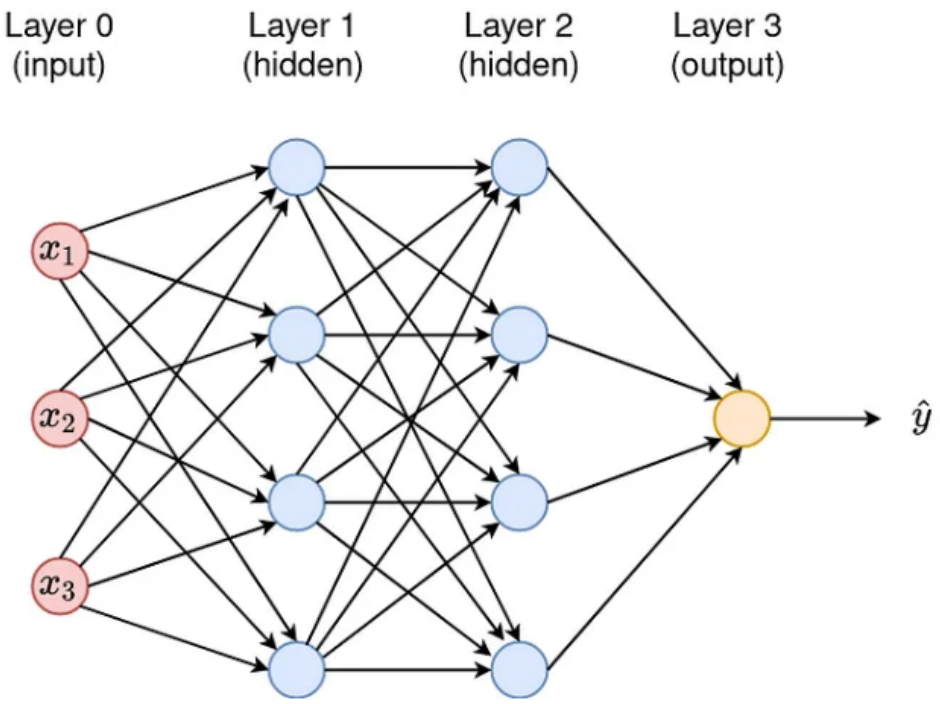


Figure 3.11: Simple Architecture of Neural Network

Image Source: `https://towardsdatascience.com/the-basics-of-neural-networks-neural-network-series-part-1-4419e343b2b`

## 3.6 NeRF

NeRF (Neural Radiance Fields) is a revolutionary technique in computer graphics and computer vision that allows for the high-quality reconstruction of 3D scenes from 2D photos or videos. Unlike traditional methods, NeRF models a static scene as a 5D vector-valued function. The input is a 3D location (x, y, z,) and 2D viewing direction($\Theta$, $\phi$) that produces a volume density ($\sigma$) and emitted color (r,g,b). Volume Density represents the presence/absence of an object at that point and functions as a differential opacity, controlling how much light is accumulated by a ray passing through the point. Direction is represented as a 3D cartesian vector d. NeRF outputs the radiance emitted in each direction at each point in space.
To depict this function, NeRF uses a Multi-Layer Perceptron (MLP) model to

regress from a single 5D coordinate to a single value volume density and view-dependent RGB color. This enables NeRF to represent scenes with high detail and complexity, including fine geometric details and intricate lighting effects. Furthermore, NeRF can handle non-Lambertian surfaces, such as shiny or translucent objects, and can generate photorealistic images with accurate lighting and shading.

By using NeRF, any scene can be rendered from a specific point of view by determining the radiance and opacity along each ray passing through the scene. This enables the generation of novel views of the scene from any viewpoint or lighting condition. The process of using NeRF (Neural Radiance Fields) to generate a novel view of a scene involves several steps.

(a) NeRF march camera rays through the scene to generate a sampled set of 3D points. This involves determining the direction of each ray based on the desired viewpoint and generating a set of equidistant points along the ray.

(b) These sampled sets of 3D points and their corresponding 2D viewing directions are fed as input to the neural network. The neural network then produces an output set of colors and densities for each point, which represents the radiance field at that point. This is achieved by using a Multi-Layer Perceptron (MLP) model to regress from a single 5D coordinate to a single value volume density and view-dependent RGB color.

(c) The classical volume rendering technique is used to accumulate those colors and densities into a 2D image. This involves integrating the radiance field along each ray passing through the scene to determine the color and opacity of each pixel in the final image. This results in a photorealistic image that accurately represents the scene from the desired viewpoint and lighting condition.

(d) The loss function in NeRF is designed to achieve two goals: optimizing the coarse network and optimizing the fine network. This is accomplished by using the L2 distance with the RGB values as the loss, which can be estimated using the ground truth 3D model. Since every step is differentiable, the network can be optimized by the predicted RGB value of rays.

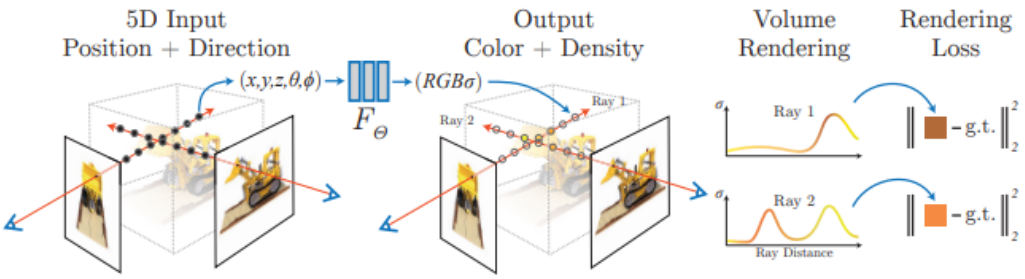


Figure 3.12: Neural radiance field scene representation and differentiable rendering procedure

Image Source: Mildenhall, Ben, et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. 2020.

NeRF alone is insufficient to achieve state-of-the-art quality. In order to portray complex high-resolution scenes, concepts like positional encoding and hierarchical sampling are used in NeRF. The positional encoding of the input coordinates helps the MLP express high-frequency functions. Hierarchical sampling enables an efficient sample of the high-frequency representation.

### 3.6.1 Positional Encoding

NeRF results in poor performance in representing high-frequency variation in color and geometry when directly operated on the 5D coordinates. Positional encoding facilitates the network to optimize the parameters by mapping input to higher-dimensional space easily. NeRF showed that using a high-frequency function for mapping original input enables better fitting of data that contains high-frequency variation.[7] The encoding function is given as

$$\gamma(p) = [\sin(2^0\pi p), \cos(2^0\pi p), ..., \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)] \tag{3.11}$$

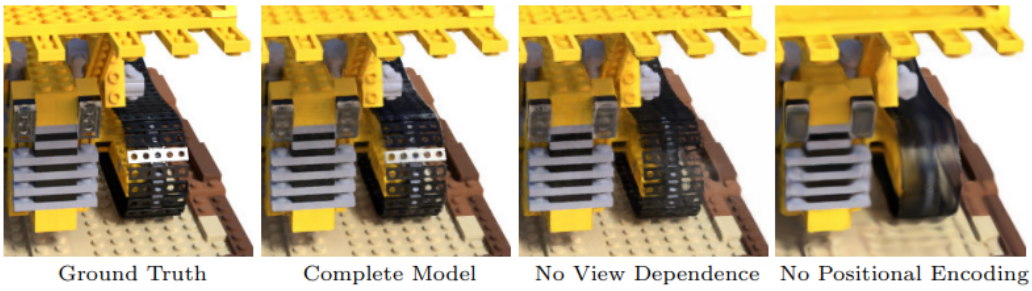


Figure 3.13: Positional Encoding to capture fine details of the image

Image Source: Mildenhall, Ben, et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. 2020.

### 3.6.2 Ray Sampler

Along the rays of the cameras, sampling is done along the field and aggregated to predict the pixel value (ie. color). In the ideal world, many dense samples are computed along a ray. Unfortunately, each additional sample adds a computation cost to the system as it needs to be processed by the field which is often a neural network. As a result, it is common for NeRF methods to use on the order of 100 samples. Therefore, sample placement in the scene must be optimized. Different techniques of sampling are used that are best suited based on their application.

### 3.6.3 Stratified Sampling

In NeRF, stratified sampling is used to improve the quality of the training data. Stratified sampling aims to ensure that the training data represents the entire scene and covers all possible viewpoints. This is done by dividing the scene into smaller regions or "strata" and sampling from each stratum based on its importance. Most samplers have the option to stratify the samples. When stratified, each sample is randomly perturbed.

The magnitude of the perturbation is such that the sample order remains consistent and the overall distribution statistics are not changed. Using stratified samples during training generally improves the reconstructions as it helps prevent overfitting. During inference, stratified sampling should be disabled as it can cause noisy artifacts when the camera moves.


Figure 3.14: Stratified Sampling Vs Normal Sampling

Image Source: `https://docs.nerf.studio/`

### 3.6.4 Hierarchical Sampling

In NeRF (Neural Radiance Fields), hierarchical sampling is used to speed up the rendering process. Since NeRF generates high-quality 3D renderings by sampling the underlying radiance field, it can be computationally expensive to render a large scene with a high level of detail. It is important to sample the scene where it has content otherwise the reconstruction quality will be reduced. Hierarchical sampling starts with a coarse-level sampling of the scene and progressively refines the sampling at finer levels. This allows the algorithm to allocate computational resources more efficiently, focusing on areas of the scene that are most likely to contribute to the final rendering.

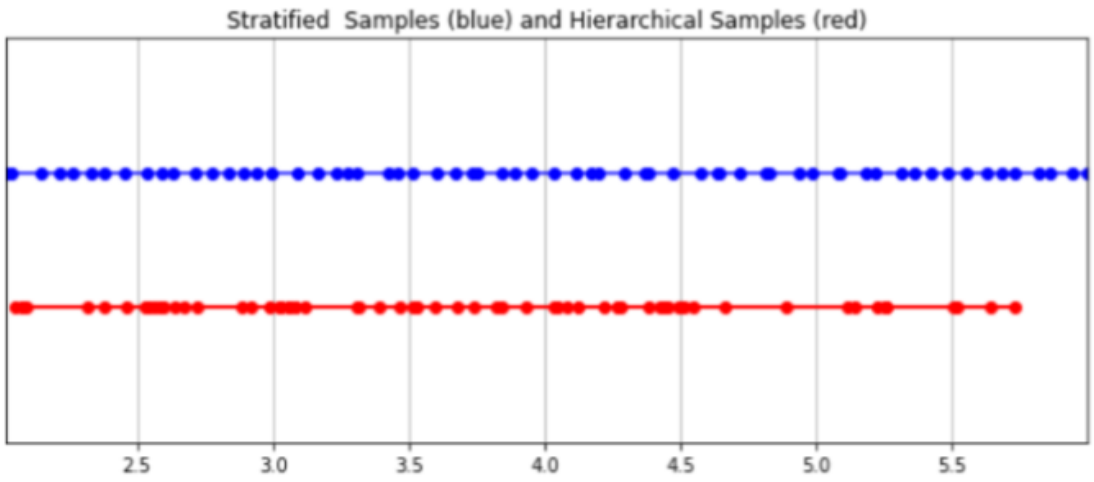


Figure 3.15: Hierarchical Sampling

## 3.7 NeRF Architecture

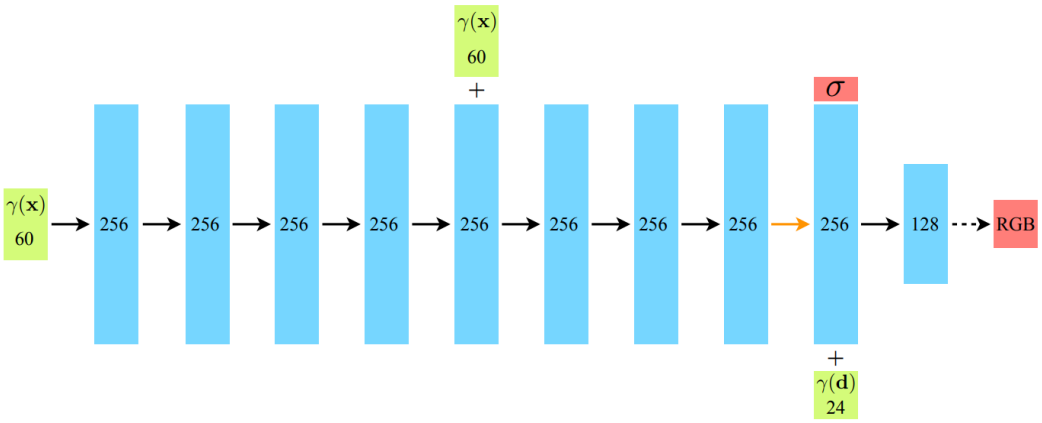


Figure 3.16: Architecture of NeRF

Image Source: Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radi- ance fields for view synthesis. In European conference on computer vision, pages 405–421. Springer, 2020.

The given diagram depicts the architecture of NeRF, which is a simple fully-connected model. The green blocks represent the input vectors, the blue blocks represent the intermediate hidden layers, and the red blocks represent the output vectors. The number inside each block indicates the dimension of the vector. The model consists of standard fully-connected layers with different types of activations, such as ReLU, sigmoid, and concatenation.

The input location's positional encoding ((x)) passes through eight fully-connected ReLU layers, each having 256 channels. The output of the last layer is the volume density , which is rectified using ReLU to ensure non-negativity and a 256-dimensional feature vector. This feature vector concatenates with the positional encoding of the input viewing direction ((d)), and the resulting vector goes through an additional fully-connected ReLU layer with 128 channels. Finally, a final layer with sigmoid activation computes the emitted RGB radiance at position x as viewed by a ray with direction d.

## 3.8 Nerfacto

For real data captures of static scenes, Nerfstudio uses the NerFacto model as the default model. The model have been found to work well for real data.

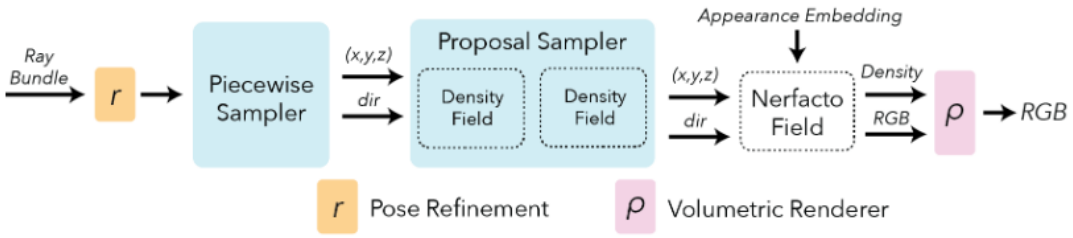


Figure 3.17: Nerfacto Pipeline

Image Source: `https://docs.nerf.studio/`

### 3.8.1 Pose Refinement

It is common to encounter inaccuracies in predicted camera positions, especially when using poses obtained from devices like phones. These misaligned poses can lead to hazy artifacts in the scene and a decrease in clarity and detail. The NeRF framework provides a way to compute loss gradients that can be back-propagated to refine and optimize the input pose estimations.

### 3.8.2 Piecewise Sampler

To generate the initial samples of a scene, a Piecewise sampler is employed that divides the samples into two groups. The first half of the samples are distributed evenly up to a distance of 1 from the camera, while the other half is distributed such that the step size between each sample increases progressively. The step size is selected such that the frustums of each sample are scaled versions of one another. By increasing the step size gradually, objects that are far away can effectively be sampled while maintaining a dense set of samples for objects that are nearby.

### 3.8.3 Proposal Sampler

To enhance the quality of the final render, a proposal sampler is employed that focuses on the areas of the scene that have the most significant impact. This approach effectively consolidates the sample locations to the regions that contribute

most to the reconstruction. However, the proposed network sampler requires a density function for the scene, which can be implemented in various ways. It is found that using a small fused MLP with hash encoding is a fast and sufficiently accurate method for this purpose. Moreover, multiple density functions can be chained together to further consolidate the sampling, but it is observed that using more than two density functions does not yield significant improvements.

### 3.8.4 Density Field

To guide the sampling process, a basic density field is sufficient, and it can be created using a hash encoding and a small fused MLP from tiny-cuda-nn to efficiently query the scene. The encoding dictionary size and the number of feature levels can be reduced to increase efficiency, without significantly affecting the quality of the reconstruction. This is because the density function does not need to capture high-frequency details in the early stages.

### 3.8.5 Hash Encoding

Hash Encoding is a technique used in NeRF to speed up the rendering of 3D scenes by reducing the computational cost of evaluating the network at many points along the camera ray. The hash table is constructed by partitioning the camera ray into segments and computing a hash code for each segment based on the network output at a single point within the segment. The hash table can then be used to efficiently retrieve the network output for any point along the camera ray by looking up the corresponding hash code and interpolating between the network outputs for the neighboring segments. This results in a significant speedup in rendering time without sacrificing image quality, making NeRF practical for real-time applications.

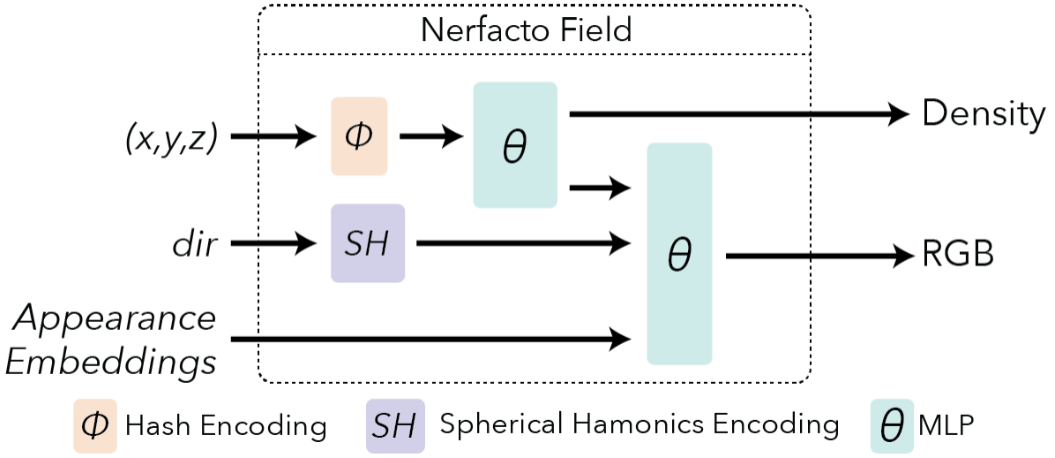


Figure 3.18: Nerfacto Field

Image Source: `https://docs.nerf.studio/`

## 3.9 Camera Models

Given a set of images, every pixel of the images is projected into 3D space. This is accomplished by computing the camera ray for each pixel given information about the type of camera and the location of the camera. Each image should have an associated pose that consists of two properties, intrinsic and extrinsic parameters.

- Intrinsics: All of the parameters internal to the camera such as lense or sensor properties.
- Extrinsic: All of the parameters external to the camera such as the location and rotation relative to the world frame.

### 3.9.1 Perspective Camera Model

In computer graphics, the perspective camera model is a mathematical representation that simulates how an actual camera functions. To create a projected image of a 3D scene, the model converts 3D coordinates into 2D coordinates. According to the perspective camera model, light enters a camera through a lens and travels in straight lines. A film or digital sensor, or another photosensitive plane inside the camera, is where the light is focused by the lens. The size of the photosensitive plane and the focal length of the lens determines the camera's field of view.

The perspective camera model is widely used in computer graphics to produce realistic 3D scenes and animations. Objects in front of the camera seem bigger than anything behind it. This effect is known as perspective foreshortening. 3D points in the scene are converted to 2D points on the photosensitive plane of the camera using the perspective projection matrix. The projection matrix determines the appropriate perspective distortion for each object in the scene by considering the camera's position, orientation, field of view, and other factors.

| **Intrinsic** | **Description** |
|---|---|
| cx | Number of pixels in the x dimension |
| cy | Number of pixels in the y dimension |
| fx | Focal length in the x dimension |
| fy | Focal length in the y dimension |

Table 3.1: Perspective Camera Model

## 3.10 Evaluation Metrics

There are several metrics used to evaluate the quality of reconstructed images. Some of the commonly used metrics are:

### 3.10.1 PSNR (Peak Signal to Noise Ratio)

A metric called PSNR is used to evaluate how well a video signal or image has been compressed or reconstructed. It calculates the difference between a signal's highest possible power and the power of the noise that degrades the accuracy of its representation. The PSNR, which is measured in decibels (dB), is determined by comparing the original and reconstructed or compressed signals. The clarity of the compressed or reconstructed signal improves as PSNR increases. The PSNR equation is:

$$PSNR = 10log_{10}(\frac{MAX^2}{MSE}) \tag{3.12}$$

where MAX is the maximum possible pixel value of the image, and MSE is the mean squared error between the original image and the reconstructed or compressed image. The equation calculates the ratio of the maximum possible power of a signal to the power of the noise that affects its fidelity. The result is ex-

pressed in decibels (dB). The higher the PSNR value, the better the quality of the reconstructed or compressed signal.

### 3.10.2 SSIM(Structural Similarity Index)

Structural Similarity Index, or SSIM, is a metric used to evaluate the quality of an image or video signal that has been compressed or reconstructed. Contrasting the luminance, contrast, and structural information of the two pictures determines how similar they are structurally.

SSIM is a better indicator of perceptual quality than conventional measures like PSNR since it is created to match the human visual system's sensitivity to changes in these parameters. The three variables that make up the calculation are luminance, contrast, and structure. A number between -1 and 1, where 1 denotes complete resemblance between the two pictures, is used to express the SSIM index. The quality of the compressed or reconstructed signal increases with the SSIM value.

The SSIM equation is:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.13}$$

where x and y are the two images being compared, $\mu_x$ and $\mu_y$ are the mean values of x and y respectively, $\sigma_x$ and $\sigma_y$ are the standard deviations of x and y respectively, $\sigma_{xy}$ is the covariance between x and y, and c1 and c2 are small constants to avoid division by zero.

### 3.10.3 LPIPS(Learned Perceptual Image Patch Similarity)

Learned Perceptual Image Patch Similarity, or LPIPS, is a metric used for evaluating the perceptual quality of an image or video signal that has been compressed or reconstructed. It calculates the perceptual distance between two images using deep learning as a foundation.

To extract characteristics from both the original and the reconstructed or compressed image, LPIPS uses a neural network that has been trained on a large dataset of images. Then, the network calculates how similar the characteristics of the two images are. An outcome is a single number, with a lower value denoting

a greater perceptual quality, that indicates the perceptual distance between the two pictures.

## 3.11 Image Segmentation

Many systems for visual comprehension depend on image segmentation. It involves partitioning images (or video frames) into multiple segments or objects. Numerous applications, such as medical image analysis (including tumor boundary extraction and measurement of tissue volumes), autonomous vehicles (e.g. navigable surface and pedestrian detection), video surveillance, and augmented reality, heavily rely on segmentation. The number of image segmentation algorithms, ranging from the earliest techniques, such as thresholding, histogram-based bundling, region growing, k-means clustering, and watersheds, to more sophisticated ones, such as active contours, graph cuts, conditional and Markov random fields, and sparsity-based methods has been developed till date. The classification of pixels with semantic labels (semantic segmentation) or the partitioning of individual objects (instance segmentation) is two ways to frame the issue of segmenting an image. While image classification predicts a single label for the entire image, semantic segmentation performs pixel-level labeling with object categories (e.g., human, car, tree, sky) for all image pixels. As a result, it is typically a more difficult task.


Figure 3.19: Image Segmentation

The U-Net, suggested by Ronneberger et al.[17], for segmenting pictures from biological microscopy. Their network and training technique depends on data

augmentation to efficiently learn from the low number of annotated photos. The symmetric expanding path that allows for exact localization and a contracting path that captures context make up the U-Net architecture. With a 3 X 3 convolutional architecture, the downsampling or contracting portion extracts features. Up-convolution (or deconvolution) is used during the up-sampling or expanding phase to decrease the number of feature maps while increasing their dimensions. To prevent losing pattern information, feature maps from the network's down-sampling portion are replicated in the up-sampling part. Finally, a 1 X 1 convolution processes the feature maps to generate a segmentation map that categorizes each pixel of the input image.

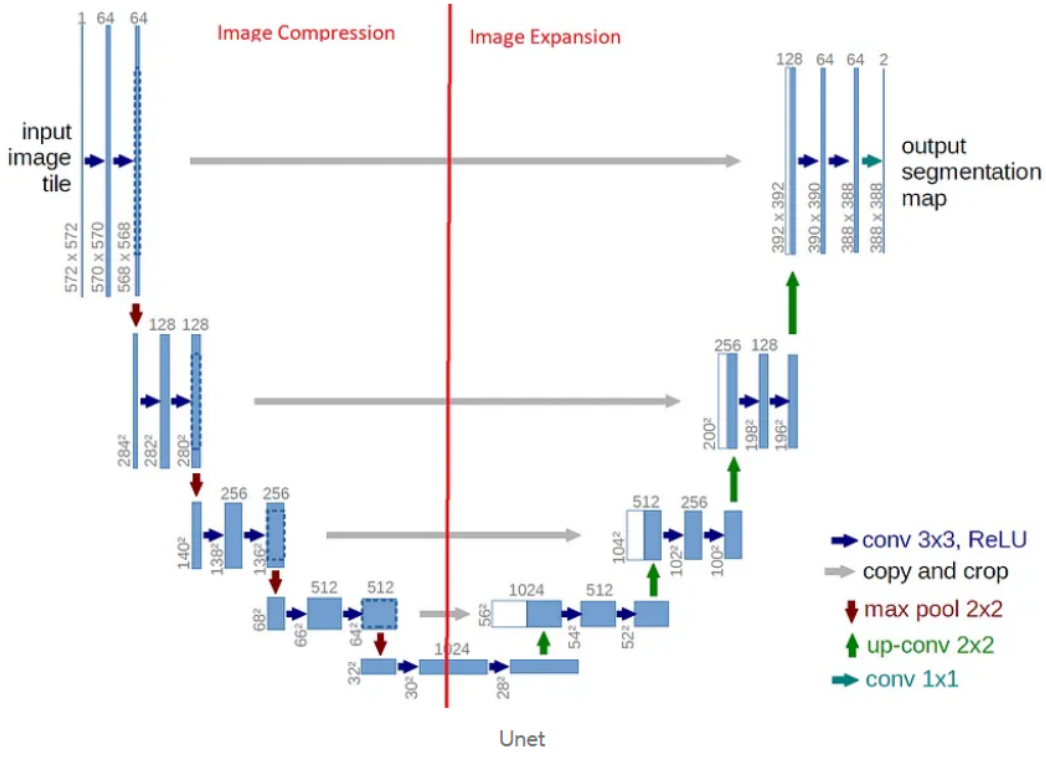


Figure 3.20: U-Net Architecture

Image Source: Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015.

## 3.12 Technologies Used

### 3.12.1 React

React is a popular JavaScript library used for building user interfaces. It was developed by Facebook and is currently maintained by Facebook and a community of individual developers and companies. React allows developers to build reusable UI components that can be easily composed to create complex interfaces. It uses a declarative approach, which means that developers describe what the interface should look like, and React takes care of updating the DOM as necessary. One of the key benefits of React is its performance. It uses a virtual DOM, a lightweight representation of the actual DOM that minimizes the number of updates that need to be made to the page. This makes React applications fast and responsive. React is also highly extensible and can be used with other libraries and frameworks. It is commonly used with tools like Redux for managing application state, and React Router for handling navigation. Overall, React is a powerful tool for building modern, dynamic user interfaces.

### 3.12.2 ThreeJS

ThreeJS is an open-source JavaScript library used for creating 3D graphics and animations in web applications. It is an open-source project that is actively maintained by a community of developers. It is a powerful tool that makes it easy to build complex 3D scenes and interactive browsing experiences. The key features of ThreeJS include:

1. Scene graph: Three.js uses a scene graph data structure to manage objects in a 3D scene. It makes easy addition, manipulation, and removal of objects in the scene.

2. Geometry and materials: Three.js provides a range of built-in geometric shapes and materials, including spheres, cubes, and textures. It also supports importing 3D models created in other software like blender.

3. Lighting and shadows: Three.js supports a range of lighting options, including directional, point, and spotlights. It also supports casting and receiving

shadows that help provide a more realistic view.

4. WebGL Rendering: Three.js uses WebGL for hardware-accelerated 3D rendering in the browser that allows high-performance graphics.

### 3.12.3 Redux

Redux is a popular JavaScript library used for managing application state in front-end web applications in the React ecosystem. It follows a unidirectional data flow pattern, where the application state is stored in a single object called the store. The store is modified by dispatching actions, which are plain JavaScript objects that describe the changes to be made to the state. One of the key benefits of Redux is its ability to manage complex application states in a predictable and maintainable way. By centralizing the application state in the store, it becomes easier to reason about how changes to the state affect the rest of the application. This also makes it easier to debug and test the application. It also provides a range of middleware and extension points that can be used to extend its functionality. Its popularity and strong community support make it a valuable skill for developers working with React and other front-end frameworks.

### 3.12.4 COLMAP

COLMAP is an open-source software package for computer vision and photogrammetry. It used two techniques, structure from motion (SfM) and multi-view stereo (MVS) to reconstruct 3D models from 2D images. COLMAP provides a range of tools and methods, such as feature extraction, matching, and bundle adjustment for processing image data. It also includes a user-friendly graphical interface for viewing and editing reconstructed models. The key features of COLMAP include:

1. Structure from motion (SfM): COLMAP automatically reconstructs 3D models from a set of 2D images using the SfM technique. This involves estimating the camera parameters and 3D geometry of the scene from the image data.

2. Multi-view stereo (MVS): COLMAP performs multi-view stereo reconstruction, which involves estimating the depth information of the scene from multiple views of the same object.

3. Dense reconstruction: COLMAP produces dense 3D reconstructions by fusing multiple sparse reconstructions and refining the results using MVS techniques.

4. Texturing: COLMAP also applies high-quality textures to the reconstructed models using image-based rendering techniques.

### 3.12.5 Nerfstudio

Nerfstudio is an open-source software package for working with neural radiance fields (NeRFs), a technique for representing 3D scenes using neural networks. The University of California, Berkeley research team created Nerfstudio with the goal of making it easier to train and employ NeRFs for 3D scene reconstruction and rendering. The range of tools and features required for working with NeRFs include:

1. Data preparation: Nerfstudio provides tools for preparing and processing images and 3D data for use with NeRFs. This includes tools for aligning and normalizing images. It also includes tools for preparing 3D meshes and point clouds.

2. Training and evaluation: Nerfstudio includes a user-friendly interface for training and evaluating NeRF models using TensorFlow. It includes pre-trained models for common datasets, making it easier to get started with NeRFs.

3. Rendering: Nerfstudio includes tools for rendering high-quality images and videos of 3D scenes using NeRF models. This includes support for path tracing, importance sampling, and other advanced rendering techniques.

4. Visualization: Nerfstudio includes a wide range of visualization tools for exploring and interacting with 3D scenes reconstructed using NeRFs. This includes tools for viewing 3D point clouds, meshes, and textured surfaces.

### 3.12.6 Pytorch

PyTorch is an open-source machine-learning framework that is known for its simplicity, flexibility, and ease of use. It is based on the Torch library and provides

an efficient way to perform numerical computing and build deep learning models. This library is used for all machine-learning components of the system.

### 3.12.7 GoogleColab

Google Colab is a cloud-based data science workspace that provides a powerful resource to perform machine learning operations. Hence, we have trained our models on this platform.

# 4. Methodology

This project aims to reconstruct a 3D-aware view from a sequence of 2D images. In order to achieve this, the Simultaneous Localization and Mapping (SLAM) technique is used to generate point clouds and estimate the camera pose. Using these data, a Neural Radiance Field (NeRF) network can be optimized. This optimized NeRF is capable of creating novel views.

## 4.1 System Block Diagram

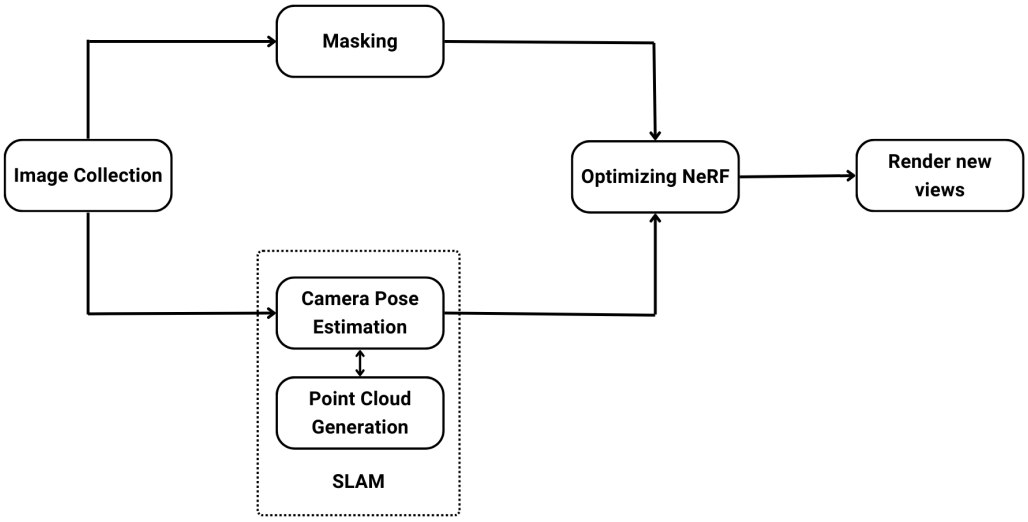


Figure 4.1: System Block Diagram

## 4.2 Image Collection

Multiple images of the scene to be rendered are captured. These images must be captured from slightly different viewing angles and should have a high overlap(nearly about 70%)[11]. For complete reconstruction, the images should capture each and every surface of the scene. The images can be captured by a standard smartphone camera. To get a better-reconstructed model from the sequence of images 360 cameras are preferred.

## 4.3 Point Cloud Generation and Camera Pose Estimation

By using a set of images of the same scene captured from slightly different points of view, SLAM systems like ORB SLAM/COLMAP are able to create sparse 3D maps of key points, which is also known as the point cloud., and simultaneously estimate the pose of the camera capturing those images. Under the hood, the SLAM system first detects important points in the images and represents them as a unique vector using techniques like scale-invariant feature transform. Then, it tries to find the same points in different images by calculating the difference between the vectors associated with the points. Once it finds the matching points in different views, it estimates the position of that point in 3D space and the pose of the camera by triangulation. These data are then used by the NeRF to generate a novel view of a complex scene.

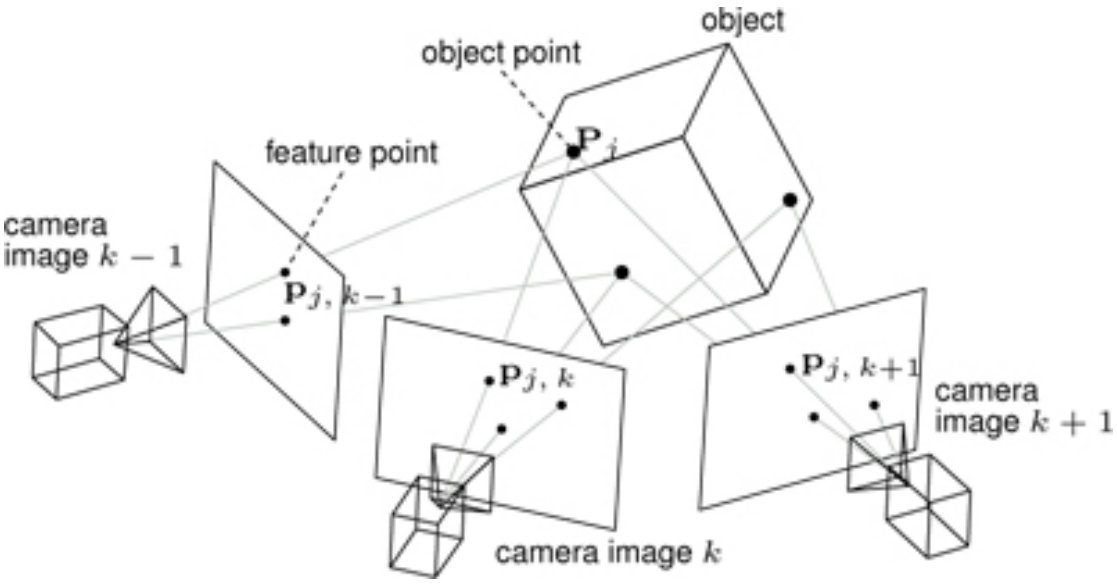


Figure 4.2: Camera Pose Estimation

Image Source: Thormählen, Thorsten, et al. "Registration of Sub-Sequence and Multi-Camera Reconstructions for Camera Motion Estimation." Volume 7, 2010

## 4.4 Masking

Outdoor scenes contains several dynamic occluders(like a walking person). As NeRF assumes its environment to be static the dynamic occluders can create a

problem during the training of the NeRF like floating artifacts that generate inconsistent views in the result. So these occluders are masked and corresponding pixels in the images are ignored during the training process of the NeRF. The majority of the dynamic occluders were people in the outdoor scene, so the UNet model trained on the OCHuman datasets was used for image segmentation(masking).


Figure 4.3: Masking

## 4.5 Optimizing NeRF

Neural Radiance Field, or NeRF, is a technique for creating new perspectives on complex scenes. NeRF takes a set of input photos from a scene and interpolates between them to render the entire scene. The result is a volume whose color and density are determined by the direction of view and the radiance of emitted light at that place. We get an output volume for each ray, and all of these volumes make up the complicated scene.

## 4.6 Render New Views

NeRF tries to represent a continuous scene as 5D vector-valued function. The input is a 3D location (x, y, z,) and 2D viewing direction($\Theta$, $\phi$) that produces a volume density ($\sigma$) and emitted color (r,g,b). Volume Density represents the presence/absence of an object at that point. Direction is represented as a 3D cartesian vector d. This function is approximated using a Multi-Level Perceptron (MLP) network, which is the heart of the algorithm. Using each pixel as a data

point, the MLP is overfitted which provides complete 3D information about the scene.

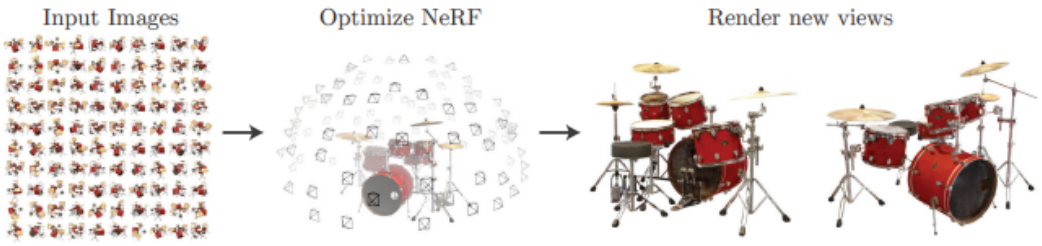


Figure 4.4: NeRF pipeline

Image Source: Mildenhall, Ben, et al. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. 2020.

Scene Representation Networks (SRN) represent a continuous scene as an opaque surface, which is implicitly defined by an MLP that maps each (x, y, z) coordinate to feature vectors. A recurrent neural network is trained to march along a ray across the scene representation by predicting the next step size along the ray using the feature vector at any 3D point. The final stage decodes the feature vector into a single color for that spot on the surface.

### 4.6.1 Virtual Tour

The novel view generated by the NeRF is used to create an immersive virtual tour through the viewer. The framework for the viewer is constructed with ThreeJS and integrated into a ReactJS application. To establish a connection between the client viewer application and the server, a websocket is employed. The server operates on the local machine and facilitates the generation of an environment through NeRF. Interacting with other objects in the environment is made possible through ThreeJS within the client application. To offer a clear understanding of the viewer framework, a detailed depiction is shown in the accompanying figure.

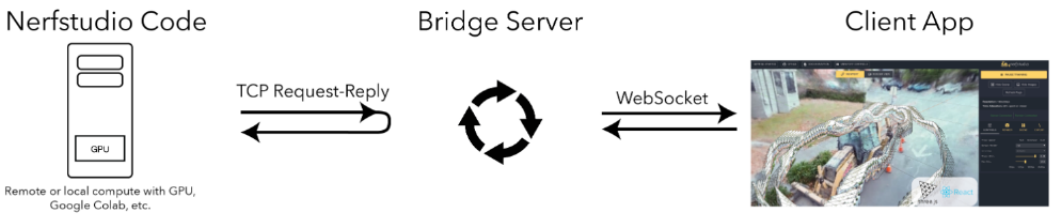


Figure 4.5: Application Architecture

The viewer consists of two modes: edit and play. The edit mode allows users to create custom virtual tours by augmenting the customized 3D objects. These tours can then be shared and viewed by other people through the playboard.
The basic application workflow can be described as follows:

1. The Bridge Server connects nerfstudio code with the Client App
2. The server has a TCP Request/Reply connection with the Viewer object
3. The Viewer class sends commands to the Bridge Server and receives replies
4. The Bridge Server dispatches commands to the Client App via a websocket
5. WebSockets are used for drawing primitives, setting object transformations, and properties

# 5. Results & Discussion

## 5.1 Dataset

### 5.1.1 Open Source Dataset

During the initial phase of the project, NeRF model was trained using the open-source dataset obtained from NeRF data repository. This was done in order to evaluate the performance of NeRF independent of the localization errors that might have been generated from COLMAP when using custom data. The dataset from NeRF repository used in the project was

| **Dataset** | **Image Dimension** | **No. of images** | **Scene** |
|---|---|---|---|
| Synthetic Lego | 100*1000px | 106 | 360 |
| Fern | Fern Dataset | 20 | Forward Facing |

Table 5.1: Open Source Dataset Description

### 5.1.2 Custom Dataset

In order to train the NeRF model on a custom dataset several indoor and outdoor scenes were captured using a normal mobile camera. Along with this, synthetic images were also generated using a unity3D. Camera pose estimation for synthetic images was also exported using unity3D.

#### Synthetic Dataset

The synthetic dataset consists of the 360-degree scene of the table created using unity3D. It consists of 100 images, each image is an RGB image with a resolution of 800*800 px.

#### Indoor Scene Dataset

These datasets are a collection of images that represent the inside of a building or an enclosed space focusing on some particular items like flower vase, ukulele,

pencil holders, stupa, and so on.

| Dataset | Image Dimension | No. of images | Scene |
|---|---|---|---|
| Flower Vase | 640*480px | 62 | Forward Facing |
| Ukulele | 500*800px | 34 | Forward Facing |
| Pencil Holder | 1920*1080px | 113 | 360 |
| Stupa (Swayamabhu) | 576*768px | 60 | 360 |
| Stupa (Krishna Mandir) | 500*700px | 250 | 360 |
| LICT Hall | 3024*4032px | 84 | 360 |
| Office | 3456*4608px | 154 | 360 |

Table 5.2: Indoor Scene Dataset Description

**Outdoor Scene Dataset**

The outdoor scene dataset consists of a collection of images captured in various outdoor environments within the campus like Locus Ground, Architecture Department, ICT Building, and from outside the campus like offices and temples.

| Dataset | Image Dimension | No. of images |
|---|---|---|
| Locus Ground | 3000*4000px | 62 |
| ICT Roof | 1530*2040px | 92 |
| Architecture Ground | 1920*1080px | 113 |
| White House Day | 1476*3280px | 155 |
| White House evening | 3456*4608px | 155 |
| Architecture Building | 3456*4608px | 123 |
| Sundarichowk | 576*1024px | 128 |
| Bhaktapur Durbar Square | 440*979px | 643 |
| Patan Ground | 720*1280px | 218 |

Table 5.3: Outdoor Scene Dataset Description

## 5.2 Training

In this project, a series of experiments were conducted to create a 3D model of various scenes using the NeRF and Nerfacto models with the dataset mentioned

above. The models underwent multiple iterations of training and were evaluated using various metrics.

### 5.2.1 Open Source Dataset

#### Lego Dataset

The initial training was carried out using the Lego dataset, and the results were promising. The reconstructed image was found to be almost identical to the ground truth image. Furthermore, the vanilla NeRF model was trained using both stratified and non-stratified sampling techniques. This was done to find out the effect of sampling on the overall result. The Lego dataset was trained for 10000 iterations.

Figure 5.1: Synthetic Lego Dataset

| Dataset | PSNR | Training Time |
|---|---|---|
| Synthetic Lego(stratified sampling) | 24.3456 | 1 hr 02 mins 09 secs |
| Synthetic Lego(no stratified sampling) | 24.345 | 3 hrs 05 mins 20 secs |

Table 5.4: Training Synthetic Lego

From this experiment, it was found that the stratified sampling technique helps to reduce the computational time in training the model to achieve the same result.

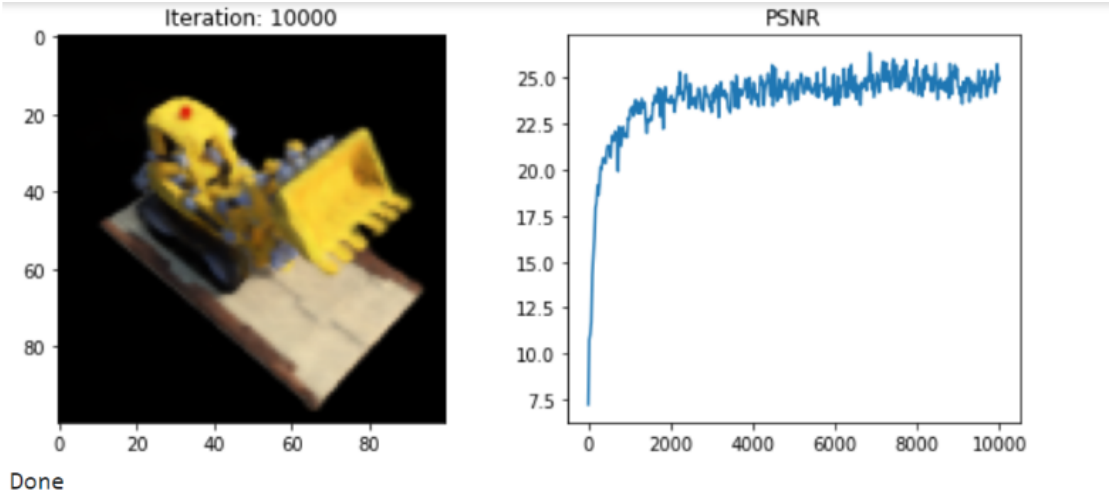


Figure 5.2: Training on Lego Dataset(No Stratified Sampling)

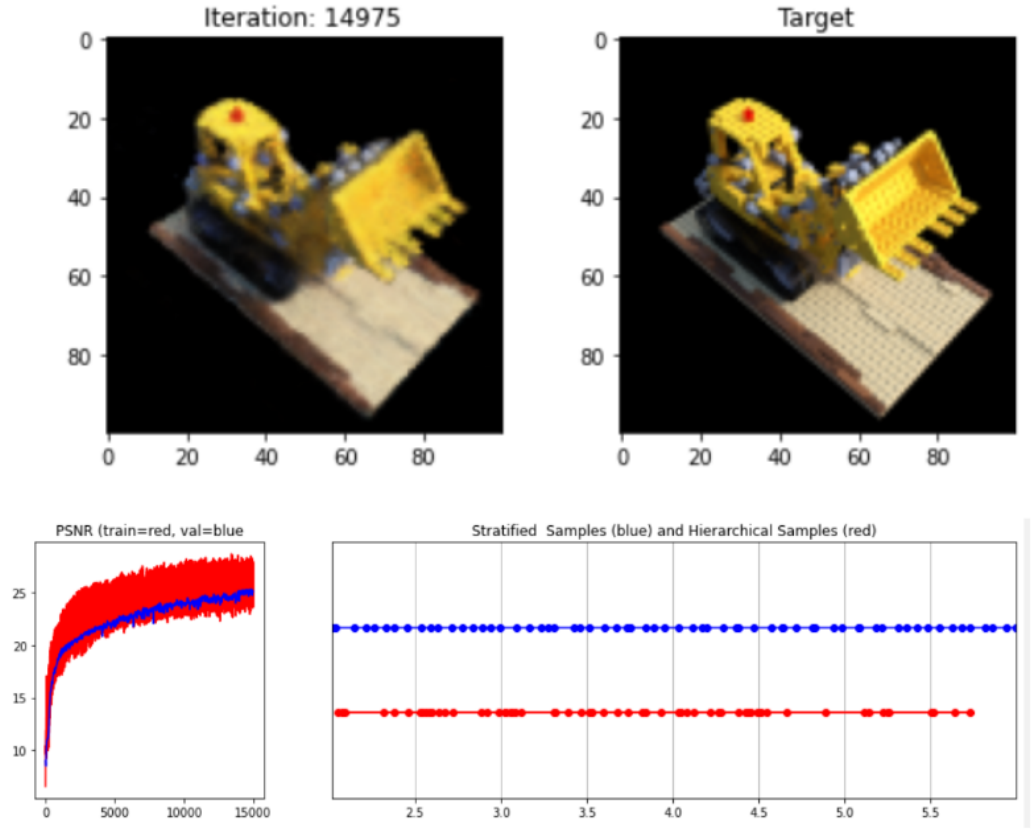


Figure 5.3: Training on Lego Dataset(Stratified Sampling)

### Fern Dataset

After successfully training on a synthetic dataset, the experiment was carried out on real-life images,i.e.Fern dataset. This dataset contains 20 high-resolution front-

facing images, used by COLMAP for camera pose estimation. Also, the 3D mesh was extracted for the given dataset.

Figure 5.4: Fern Dataset

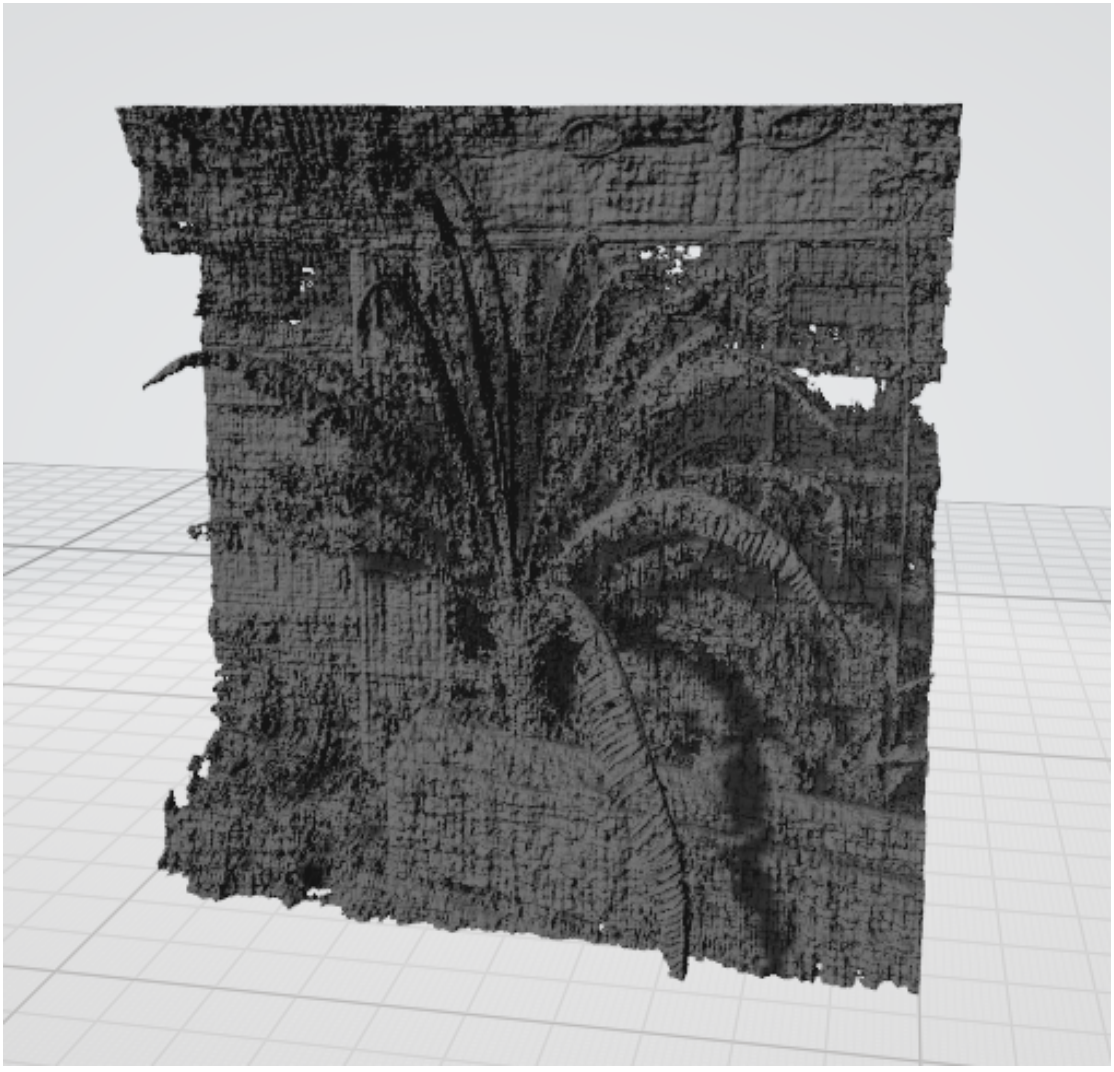

Figure 5.5: Mesh for Fern Dataset

It was observed that NeRF was able to reconstruct even from real-life images. Furthermore, the extracted 3D mesh captured fine details. This is most likely due to the dataset being prepared in a controlled condition.

### 5.2.2 Indoor Scene Dataset

#### Flower Vase Dataset

Later, the NeRF model has trained on a custom indoor scene dataset(Flower Vase). While training, two methods (ORB SLAM and COLMAP) were utilized to localize the camera points. The goal was to compare the performance of NeRF on these two SLAM techniques.

Figure 5.6: FlowerVase Dataset

It was found that the results obtained using COLMAP were significantly better than ORB SLAM. The results from the ORB SLAM were blurry which indicates a high camera pose estimation error. This is more likely due to the fact, ORB SLAM tries to localize in real time while COLMAP takes a little longer. COLMAP was used in all further experiments.

Figure 5.7: Results from ORB SLAM (Left) vs COLMAP(Right)

| **Dataset** | **PSNR** | **Training Time** |
|---|---|---|
| Flower Vase(ORB SLAM) | 21.126 | 1 hr 35 mins 30 secs |
| Flower Vase(COLMAP) | 30.2 | 1 hr 23 mins 05 secs |

Table 5.5: Indoor Scene Training

### Ukulele, Pencil Holder Dataset

The training process was replicated for other custom datasets, including pencil holders, ukuleles, stupas, and more. These experiments were specifically designed to verify the assumptions derived from previous experiments.

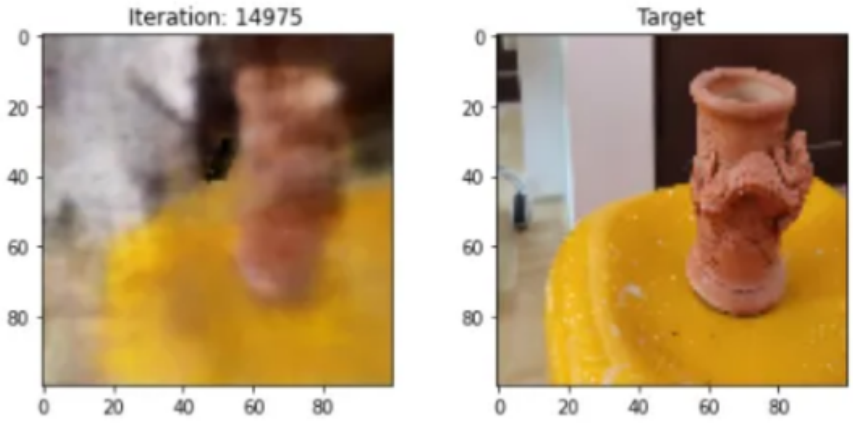


Figure 5.8: Pencil Holder Dataset

Figure 5.9: Ukulele ground truth(left) vs output(right)

| Dataset | PSNR | Training Time |
|---|---|---|
| Pencil Holder | 18.225 | 1 hr 36 mins 51 secs |
| Ukulele | 25.916 | 1 hr 17 mins 08 secs |

Table 5.6: Indoor Scene Training

### 5.2.3 Outdoor Scene Dataset

#### Locus Ground

Till now vanilla NeRF was used as a base model, but this model had two major limitations. First, the computation time was very high($\sim$ 18-24hrs), and secondly, due to the less file size($\sim$ 5MB), it couldn't capture large areas with sufficient details. As a result, a more efficient version of NeRF, known as Nerfacto, was used as a base model instead. Nerfacto requires far less computation time($\sim$ 5-10 mins) and much large file size($\sim$ 300MB). As a result, more details were captured even on large scenes. This model was first trained on Locus Ground(Dronacharya Ground) dataset.

Figure 5.10: Locus Ground ground truth(left) vs output(right)

The Nerfacto model was able to capture the fine details of the unbounded outdoor scene but still, there were floating artifacts that appeared most likely due to inadequacy in distinct viewpoints while collecting the datasets.

### Architecture Ground

To verify the above hypothesis, this dataset was captured from several different directions so that every distinct viewpoint was included in the dataset. Firstly, the images were captured from different positions pointing towards in assumed center of the scene. Then some outward-facing images were also captured.

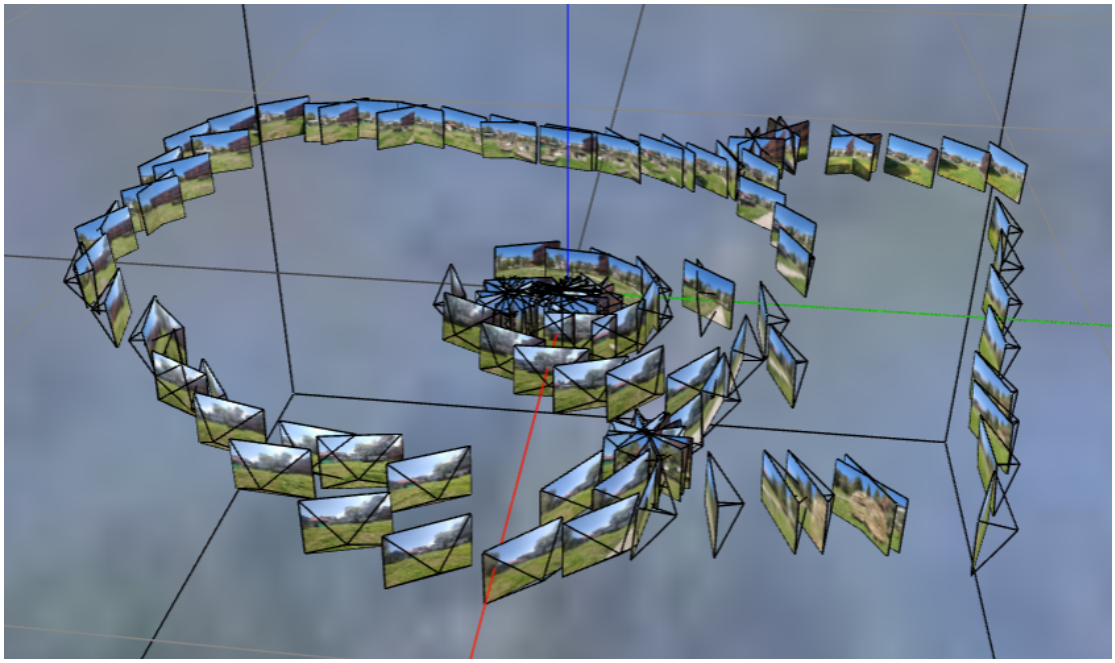

Figure 5.11: Images taken in inward direction

Due to careful considerations while creating the dataset, highly realistic results containing no artifacts were obtained. A similar approach was used in dataset collection while capturing images of other locations like Sundari Chowk, and Patan Museum Ground. All these results were also highly realistic and are attached in the appendix section.

Figure 5.12: Architecture backyard ground truth(left) vs output(right)

### ICTC Rooftop

The goal of this experiment was to observe the number of training iterations required to obtain a satisfactory reconstruction of the scene. For this ICTC Rooftop dataset consisting of 92 images, each f dimension 2448*3264 pixels were used. Peak Signal Noise Ratio(PSNR) metric was used as an evaluation criterion.

| **Dataset** | **Iterations** | **PSNR** |
|---|---|---|
| ICTC Roof Top | 4000 | 26.32 |
| ICTC Roof Top | 6000 | 26.71 |
| ICTC Roof Top | 16000 | 26.92 |
| ICTC Roof Top | 20000 | 27.13 |
| ICTC Roof Top | 29000 | 27.29 |

Table 5.7: Outdoor Scene Training

Figure 5.13: ICTC ground truth(left) vs Output(right)

### Durbar Square

Till now all experiments were done on static scenes i.e.containing no dynamic objects in the captured image. However, this condition may not always be true, especially while trying to capture culturally significant places. So, this experiment was carried out to observe the reconstruction results in a scene with dynamic

occluders.

For this experiment, the dataset of Bhaktapur Durbar Square was used. Since the area of the square is quite large, it was not possible to capture the entire square in a single go. Thus, 9 different zones were created, and each zone was independently captured. These captures were later combined to train the NeRF.

Figure 5.14: Zonal Division of Durbar Square(Left) and COLMAP(Right)

Figure 5.15: Durbar Square Ground Truth(Left) Vs Output(right)

Because of the dynamic occluders, the reconstructed views contained a lot of floating artifacts resulting in inconsistent views. However, the reconstruction of static objects like buildings and statues was still satisfactory.

### Masked Durbar Square Dataset

Since the dynamic occluders were responsible for reducing the reconstruction quality, the occluders were removed using masking. The masked pixels were completely discarded during NeRF training. As most of the dynamic occluders were people, a U-Net architecture-based segmentation model was used to generate masks.
The resulting reconstruction was much cleaner with hardly any floating artifacts. In addition to this, even the static occluders (like persons sitting in *falcha*) were removed from the reconstruction. As different zones were captured on different days, in the combined dataset the static occluders virtually behaved as the dynamic occluders.


Figure 5.16: Output after masking


Figure 5.17: Static Occluders

## 5.3 Experiment

To further investigate the effects of varying the number of images used to train a NeRF model, a small experiment was conducted. The distance of the camera from the object was fixed while the camera position varied around a circle with a specified radius. The experiment was performed for different numbers of images, and a COLMAP was generated for each iteration. The results were analyzed using evaluation metrics such as PSNR, SSIM, and LPIPS. The model was trained for 20000 iterations on the same stupa dataset from various positions, and the resulting 3D model was compared to the ground truth to determine the accuracy of the reconstruction. All results and findings can be found in the appendix section of this report.

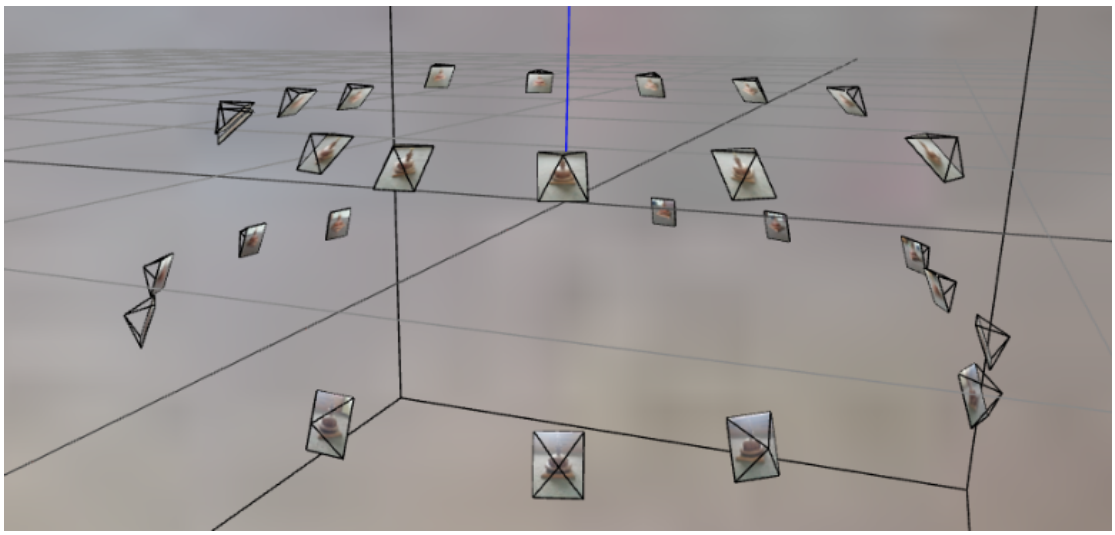

Figure 5.18: Colmap of 32img16pos@30cm

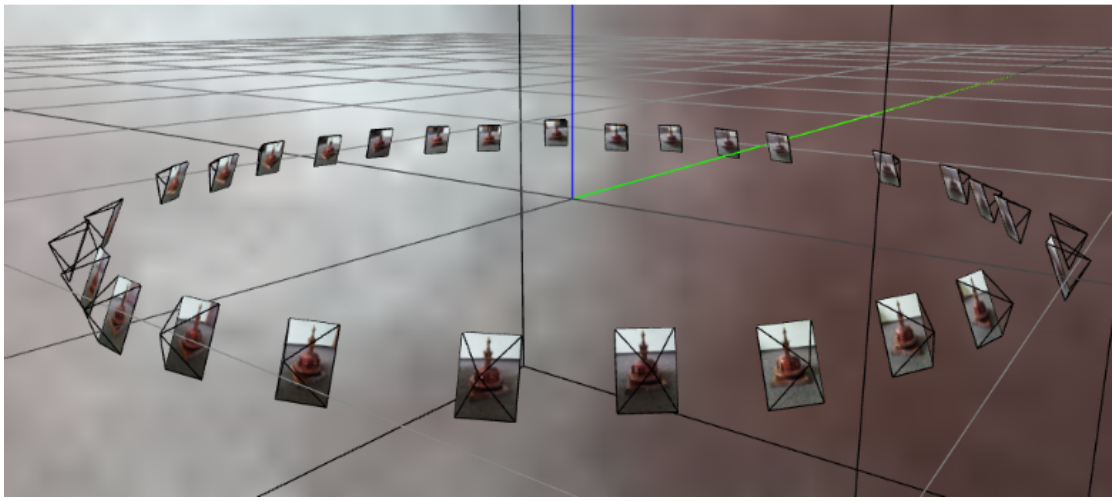

Figure 5.19: Colmap of 32img32pos@30cm

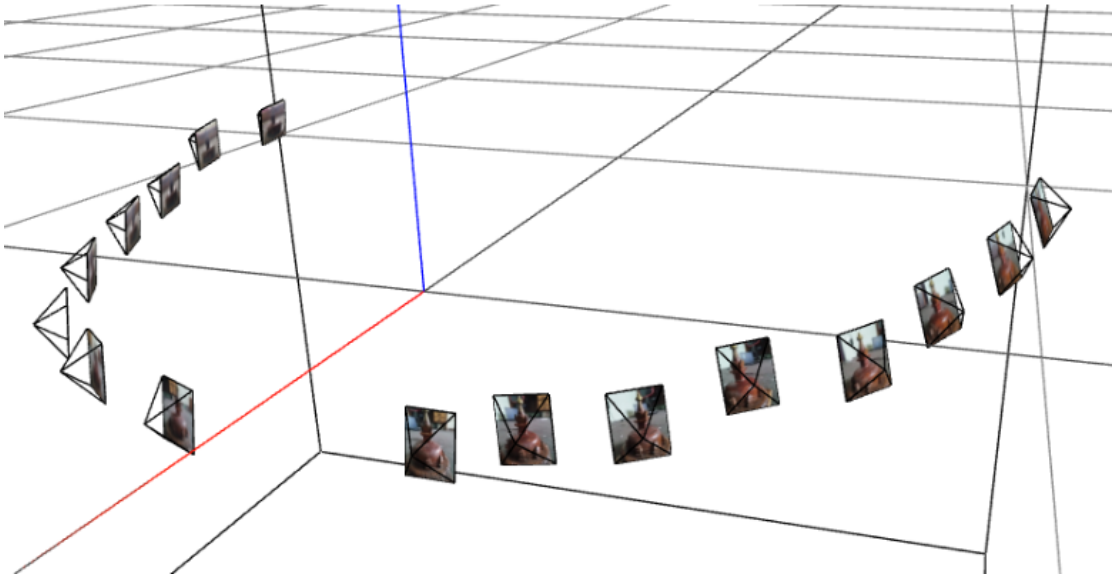

Figure 5.20: Colmap of 32img32pos@20cm

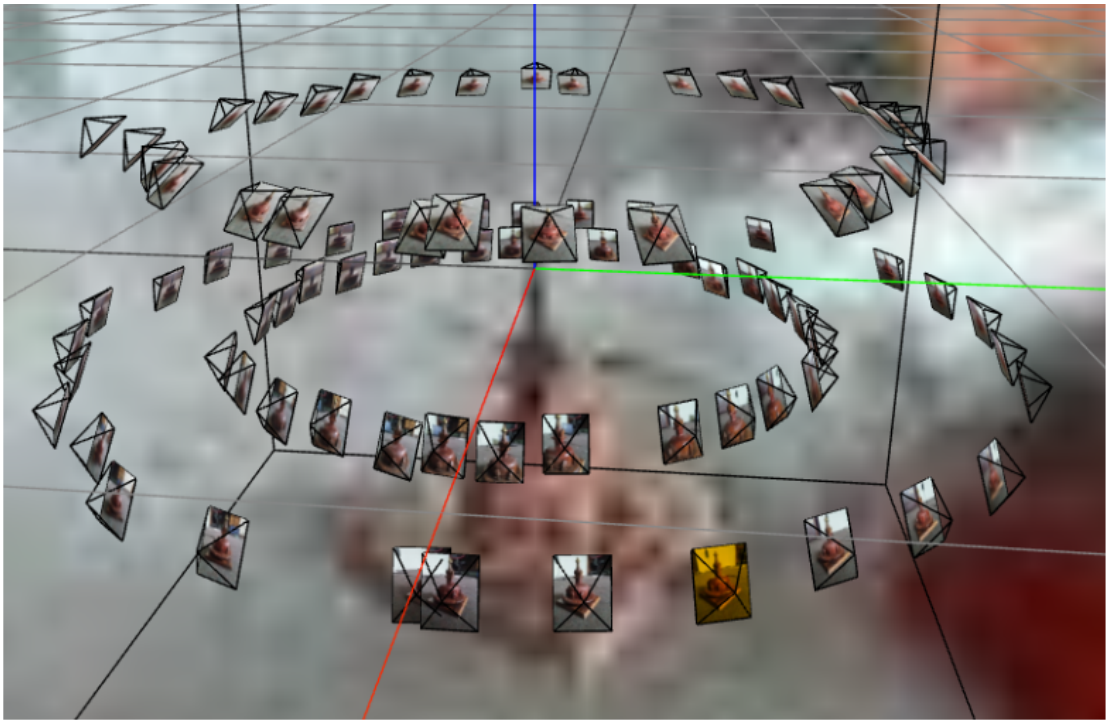

Figure 5.21: Colmap of combined experiment

Figure 5.22: Colmap of 64img32pos@20cm

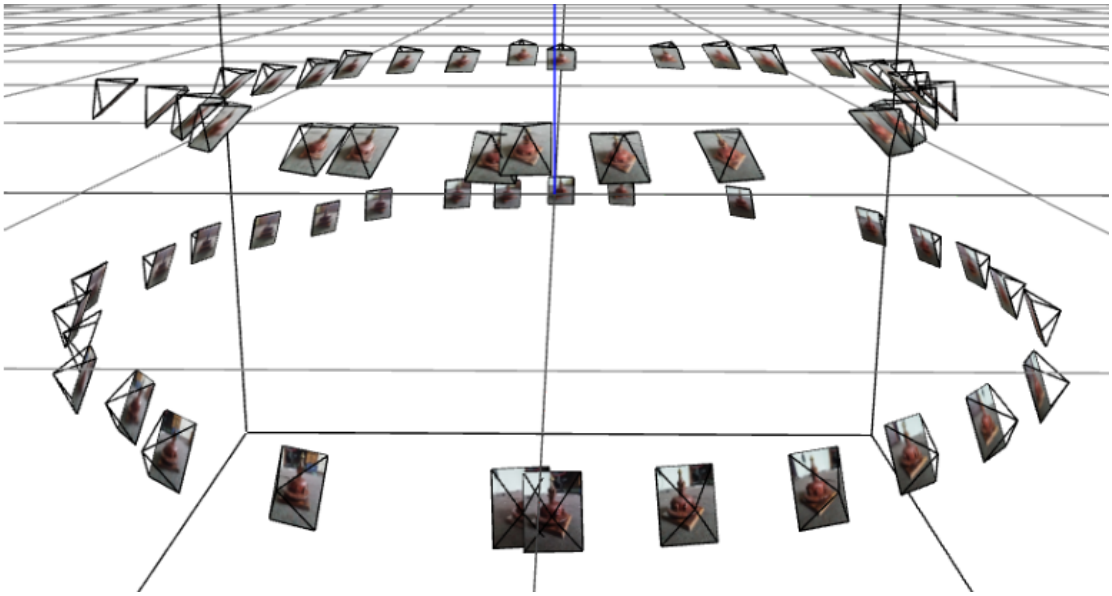

Figure 5.23: Colmap of 64img32pos@30cm

| Name | Distance | Training Time | PSNR | SSIM | LPIPS |
|---|---|---|---|---|---|
| 16img8pos | 30cm | - | - | - | - |
| 16img16pos | 30cm | 24mins | - | - | - |
| 32img16pos | 30cm | 26mins | 14.9916868 | 0.28583031 | 0.68830150 |
| 32img32pos | 30cm | 25mins | 13.9053392 | 0.12411347 | 0.47026792 |
| 64img32pos | 30cm | 25mins | 17.8696765 | 0.68059253 | 0.12411347 |
| 16img8pos | 20cm | - | - | - | - |
| 32img32pos | 20cm | 26mins | 11.0986557 | 0.14666181 | 0.60418546 |
| 64img32pos | 20cm | 27mins | 19.1051521 | 0.76696425 | 0.23837579 |
| Combined | Both | 26mins | 20.0390663 | 0.76530569 | 0.13428217 |

Table 5.8: Experiment on Stupa Dataset

The 16img8pos experiments for both 30cm and 20 cm distances were not able to generate the COLMAP because the images were not continuous enough to generate the COLMAP. The 16img16pos experiment was able to generate the COLMAP but it was not able to reconstruct the image. Further, the best result for the experiment was obtained when both the 64img32pos@30cm and 32img32pos@20cm were combined. The 3D reconstructed mesh for the best experiment is shown below.

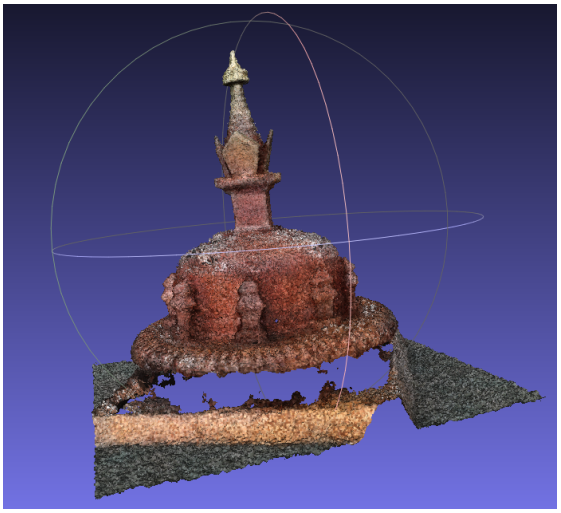

Figure 5.24: 3D reconstructed Mesh of Stupa

## 5.4 Web Viewer

The viewer is a web application that utilizes React and Three.js to create a virtual environment. Through the utilization of WebSockets, it communicates with a Nerf engine via a bridge server, transmitting camera coordinates and associated transformation matrices. The Nerf engine generates corresponding images and relays them back to the viewer, while the viewer itself is responsible for creating and augmenting the interactive 3D space. Additionally, the viewer serves as a visualization tool for keyframes.

Figure 5.25: Output from the viewer

Figure 5.26: Output from the viewer next camera location

# 6. Epilogue

## 6.1 Conclusion

This project attempts to develop a system that can produce accurate 3D views of situations from a series of 2D photos, which is a difficult job in the field of computer vision. The difficulty comes from the fact that the 2D photographs only convey a portion of the scene's 3D geometry and structure, such as how the scene appears from a single point of view. The project uses the COLMAP, which is frequently used in robotics and autonomous vehicles, to create a 3D point cloud and determine the camera pose from the 2D photos in order to get around this problem. The point cloud represents the 3D structure of the scene, while the camera pose estimates the position and orientation of the camera relative to the scene.

The project then uses the Neural Radiance Field (NeRF) network to model the radiance field, which describes how light interacts with the scene, based on the 3D point cloud and camera pose data. The NeRF network is a deep learning technique that learns to represent the scene's appearance and texture from different viewpoints by modeling the radiance field. The optimized NeRF is then used to synthesize novel views of the scene from arbitrary viewpoints. The system can generate high-quality 3D views of both indoor and outdoor scenes that accurately capture the geometry, structure, and appearance of the scene from different viewpoints. Finally, any interactive virtual system is created on top of the synthesized views to augment other objects.

## 6.2 Limitations and Further Enhancement

While the system presented in this project is a significant advancement in 3D-aware novel view synthesis, there are still some limitations to consider.

1. NeRF provided a realistic 3D-aware novel view, but it was not able to reconstruct in real time.

2. The mesh generated by the NeRF model consists of a lot of noise, containing several floating artifacts.

3. The generation of a novel view using NeRF is GPU intensive.

# References


[1] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. IEEE, 1999.

[2] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):530–535, 1997.

[3] Stefano Ferrari, Giancarlo Ferrigno, Vincenzo Piuri, and N Alberto Borghese. Reducing and filtering point clouds with enhanced vector quantization. *IEEE Transactions on Neural Networks*, 18(1):161–177, 2007.

[4] Fengxia Li, Rong Tang, Chen Liu, and Haikun Yu. A method for object reconstruction based on point-cloud data via 3d scanning. In *2010 International Conference on Audio, Language and Image Processing*, pages 302–306. IEEE, 2010.

[5] Kun Zhou, Minmin Gong, Xin Huang, and Baining Guo. Data-parallel octrees for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 17(5):669–681, 2010.

[6] Qiang Li, Jia Kang, Yangxi Wang, and Xiaofang Cao. An improved feature matching orb-slam algorithm. In *Journal of Physics: Conference Series*, volume 1693, page 012068. IOP Publishing, 2020.

[7] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[8] History of neural radiance fields. `https://neuralradiancefields.io/history-of-neural-radiance-fields/`.

[9] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv preprint arXiv:2302.14859*, 2023.

[10] A new spin for photosynth. `https://www.microsoft.com/en-us/research/blog/new-spin-photosynth/`.

[11] Thorsten Thormählen, Nils Hasler, Michael Wand, and Hans-Peter Seidel. Registration of sub-sequence and multi-camera reconstructions for camera motion estimation. *JVRB-Journal of Virtual Reality and Broadcasting*, 7(2), 2010.

[12] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[13] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345, 2021.

[14] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022.

[15] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[16] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*, 2023.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[18] Sudipta Sinha, Drew Steedly, and Rick Szeliski. Piecewise planar stereo for image-based rendering. In *2009 International Conference on Computer Vision*, pages 1881–1888, 2009.

[19] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, et al. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*, 2023.

[20] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[21] Affine transformations. `https://people.cs.clemson.edu/~dhouse/courses/401/notes/affines-matrices.pdf`.

[22] Rotation quaternions. `https://danceswithcode.net/engineeringnotes/quaternions/quaternions.html`.

[23] Basics of neural network. `https://towardsdatascience.com/the-basics-of-neural-networks-neural-network-series-part-1-4419e343b2b`.

[24] Nerf: Representing scenes as neural radiance fields for view synthesis. `https://towardsdatascience.com/nerf-representing-scenes-as-neural-radiance-fields-for-view-synthesis-ef1e8cebace4`.

[25] Activation functions in neural networks. `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`.

[26] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning:

A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542, 2021.

# 7. Appendix

Figure 7.1: Table Dataset

Figure 7.2: Stupa(Krishna Mandir) Dataset

Figure 7.3: Stupa(Swayambhu) Dataset

Figure 7.4: LICT Dataset

Figure 7.5: Office Dataset

Figure 7.6: White house day Dataset

Figure 7.7: White House Evening Dataset

Figure 7.8: Output of Stupa Dataset