IMPERIAL

Neural LiDAR 3D Reconstruction

Author
C.Y.A. Hon
CID: 01260366

Supervised by
Dr S. Wang

A Thesis submitted in fulfillment of requirements for the degree of
Master of Science in Applied Machine Learning

Department of Electrical and Electronic Engineering
Imperial College London
2024

Abstract

Neural Radiance Field (NeRF) have recently gained interest for achieving remarkable view synthesis and 3D reconstruction result from sparse RGB images. The idea of NeRF has, however, not yet been fully explored in the context of LiDAR data. This project sheds light on NeRF's capability for 3D reconstruction and proposes a pipeline for 3D reconstruction from LiDAR data with NeRF methodology.

This project proposed an upsampling method to eliminate noise around unscanned region in the LiDAR dataset, a volume sampling method for training the NeRF model on LiDAR data, and designed a loss function for reconstruction. The proposed method was tested on three dataset collected with Livox LiDAR sensor on outdoor objects and scene around Imperial College London White City campus. This project then shows that dynamic objects are omitted in reconstruction result, while achieving a promising 3D reconstruction of outdoor object and scene. Results in this project can lead to applications in mapping, urban planning or infrastructure management.

Declaration of Originality

I hereby declare that the work presented in this thesis is my own unless otherwise stated. To the best of my knowledge the work is original and ideas developed in collaboration with others have been appropriately referenced.

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Acknowledgments

I would like to dedicate my special thanks to Dr. Sen Wang for his guidance and support through out this project, this project has been an enjoyable and fruitful experience with his supervision. I would like to thank my mother and father for funding my MSc degree. I am very fortunate to have their unconditional support.

I am thankful for the support from Ka Ching at Sense Robotics Lab, who helped me with the technical details of this project. I would like to thank my course mates for their support along the way and for every discussion we have had about computer vision. I would like to thank my family and friends for their support of my studies.

Contents

Abstract **i**

Declaration of Originality **iii**

Copyright Declaration **v**

Acknowledgments **vii**

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**

1.1 Background 1

1.2 Motivation 1

1.3 Project Scope 2

1.4 Major Contribution 2

1.5 Outline of the Thesis 3

2 Literature Review **5**

2.1 LiDAR Technology 5

2.2 Point Clouds 6

2.3 LOAM and Dense Mapping 6

2.4 3D reconstruction and Novel View Synthesis 7

2.5 Neural Radiance Field 8

2.6 NeRF methodology on LiDAR 10

2.7 Loss functions for NeRF Models 11

2.8 Volume Sampling 12

2.9 Implicit and Explicit 3D Representations 13

3 Methodology **15**

3.1 Problem Definition 16

3.2 Projective Upsampling 17

3.3 Truncated Inverse Sigmoid Volume Sampling 20

3.4 Loss Functions 21

3.5 Model Architecture 24

3.6 Rendering the NeRF Model 25

4 Experiment & Results **27**

4.1 Data Collection 27

4.2 Data Processing 28

4.3 Deploying Projective Upsampling 29

4.4 Model training 30

4.5 Model Rendering 30

4.6 Qualitative Results 31

4.7 Quantitative Results 35

5 Discussion **39**

5.1 Quantitative Evaluation 39

5.2 Learning Curve 40

5.3 Loss Functions 41

5.4 Omission of dynamic object 41

5.5 Varying Batch Size 43

5.6 The Inertia of NeRF Models 45

5.7 Pseudo Sensors 47

5.8 Hidden Diemnsion 48

5.9 Sampling Variance 48

5.10 Registration Quality 49

Conclusions **51**

5.11 Future Work 51

A Implementation Detail for Projective Upsampling **53**

B Implementation Detail for Model Training **57**

Bibliography **59**

List of Figures

1.1 Pipeline from physical object to 3D mesh with NeRF framework. 2

2.1 The NeRF 3D reconstruction pipeline [16]. 8

2.2 The LiDAR NeRF pipeline [16]. 10

3.1 Overview of the proposed pipeline in this project. 15

3.2 Illustration of blind spots during data collection (left), and reconstruction result without using projective upsampling, resulting in noise on top of the object (right). 17

3.3 A chosen location of a pseudo sensor (left), when more then one point are projected onto a pseudo pixel, the closest point is sampled (middle), sampled points of projective upsampling of the pseudo sensor(right). 18

3.4 Fibonacci Lattice points on a sphere (left) [37], illustration of pseudo sensor pose with a given point cloud (right). 19

3.5 Applying S^* to near plane and far plane to get $t_{\min}$ and $t_{\max}$ (left), and applying S^* to stratified uniform random samples to get sampled positions in u-space. . . . 21

3.6 Illustration of sample distribution with different σ_s, with 20 samples, near plane $u_n = 0$, far plane $u_f = 3$, termination distance $u_d = 1.5$. 21

3.7 Relation and graph of target cumulative transmittance T^* and termination distribution h^* to a LiDAR ray. 23

3.8 Architecture of the NeRF model. 24

4.1 object of interest in box plant dataset (left), round plant dataset (middle) and building dataset (right). 28

4.2 LiDAR sensor setup for data collection. 28

4.3 Illustration of walking route and sensor pose in during data collection for round plant and box plant dataset (left) and building dataset (right). 29

4.4 Trimming of a single frame of point cloud for box plant dataset. 29

4.5 Point cloud of 500,000 randomly sampled points in box plant dataset (left), building dataset (middle) and round plant dataset (right). 32

4.6 Reconstruction result of the proposed method, compared to using only $\mathcal{L}_d$, only $\mathcal{L}_h$ and only $\mathcal{L}_T$. 32

4.7 Reconstruction result with different hidden dimensions, rendering threshold for box plant model with hidden dimension of 8 is rendered with threshold = 1 as it fails to visualize with threshold = 5. 33

4.8 Reconstruction result with different batch sizes. 33

4.9 Reconstruction result with different number of pseudo sensors. 34

4.10 Reconstruction result with different σ_t. 34

4.11 Setup for measuring edge length of the plant container in box plant dataset. 35

4.12 Selecting points with axis-aligned view angle for box plant mesh (left), selected points (red dots) for edge length calculation (right). 36

4.13 Selecting points with axis-aligned view angle for round plant mesh (left), selected points (red dots) for diameter calculation (right). 36

5.1 Plot of loss vs iteration of model trained on round plant dataset with default settings, first 500,000 iteration. 40

5.2 point cloud of box plant dataset with ghost trail (left), reconstruction result with dynamic object omitted (right). 42

5.3 Illustration of dynamic object occluding minority of rays at certain position. 42

5.4 Illustration of voting mechanism to eliminate dynamic objects in reconstruction. . 43

5.5 Illustration of failure of voting mechanism due to small σ_s. 43

5.6 Learning curve of models trained with round plant dataset with different batch size. 44

5.7 Illustration of small batch size relaxes reconstruction requirement (left) and large batch size require more points to be simultaneous fitted (right). 44

5.8 Architecture of the alternate model. 45

5.9 Reconstruction result of augmented model trained on raw round plant dataset and rendered with density threshold of 15. 45

5.10 Image of an array of bikes (left), point cloud of the array of bikes (middle) and reconstruction result of the bikes (right). 46

5.11 Illustration of outer points dominating points on surface in projective upsampling (left) reconstruction result using only projective upsampled data, causing lose of reconstruction details (right). 47

5.12 Image of the fine structure in building dataset (left) Reconstruction result of the fine structure with proposed σ_s (middle), and reconstruction result of the fine structure with constant $\sigma_s = 3$ (right). 48

5.13 Reconstruction result with proposed σ_s (left), reconstruction result with $\sigma_s = 0.01$ (right). 48

5.14 Close up image of the sign in box plant dataset (left), registered point cloud of the sign from 500,000 randomly sampled points (middle) and reconstruction result of the sign (right). 49

List of Tables

3.1 Variables and their descriptions 16

4.1 Meta data of datasets 28

4.2 Proposed hyperparameters 31

4.3 Reconstruction dimension for box plant container edge length, ground truth length = 1.02m. 36

4.4 Reconstruction dimension for box plant container upper edge diameter, ground truth diameter = 2.12m. 37

A.1 Description of the training dataset 53

1 Introduction

Contents

1.1 Background 1
1.2 Motivation 1
1.3 Project Scope 2
1.4 Major Contribution 2
1.5 Outline of the Thesis 3

1.1 Background

Light Detection and Ranging (LiDAR) is an important element of modern computer vision technology, providing 3D measurements of the physical world. On the other hand, Neural Radiance Field (NeRF) have been an active research area in recent years. NeRF has achieved remarkable results in novel view synthesis and 3D reconstruction from sparse RGB images and is capable of producing a compact implicit representation of scenes. In this project, the idea of NeRF was applied to dense LiDAR point clouds to achieve a learnt implicit representation of 3D reconstructions from LiDAR point clouds.

1.2 Motivation

On top of the generic advantages of NeRF models such as memory compactness, LiDAR NeRF enjoys the advantages of LiDAR sensors. Compared to RGB cameras, LiDAR sensors are robust

against outdoor conditions, such as different ambient lighting conditions and weathers, and provide reliable depth information. This has drawn interest in utilizing LiDAR data for 3D reconstruction to provide promising 3D geometry of objects or scenes. An accurate and compact 3D reconstruction with LiDAR data can lead to applications including autonomous driving, urban planning and facility management with the need for outdoor 3D reconstruction.

1.3 Project Scope

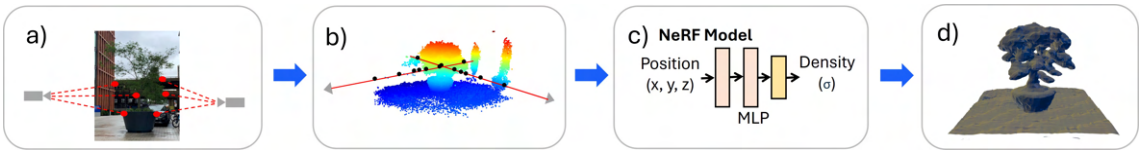

Figure 1.1: Pipeline from physical object to 3D mesh with NeRF framework.

Figure 1.1 illustrates the pipeline from LiDAR data to 3D reconstruction with NeRF framework. Which involves a) LiDAR data collection, b) point cloud processing and sampling, c) NeRF model training, and d) rendering of the model.

The body of work for this project will focus on b) and c). This project proposed projective upsampling, that creates pseudo sensors around a scene and projects the point cloud to corresponding pseudo image. This generates pseudo rays for NeRF model training, which eliminates noise in blind spots upon data collection. This project also proposed truncated inverse sigmoid volume sampling, which allows control of dense sampling around termination of LiDAR rays and improves sampling efficiency in NeRF model training. A loss function was designed for achieving a balanced reconstruction. Finally, work was done on selecting an appropriate NeRF model architecture and training configuration. Experiment was then conducted, covering the entire pipeline shown in figure 1.1.

1.4 Major Contribution

This project has achieved the following major contribution:

- Proposed projective up-sampling that remove noises in 3D reconstruction from LiDAR data in NeRF framework

- Proposed truncated inverse sigmoid volume sampling for NeRF model training with LiDAR data
- Designed loss function to achieve a balanced reconstruction result
- Show that reconstruction result from LiDAR NeRF model can omit dynamic objects
- Discuss effect of various hyper parameters on reconstruction quality

1.5 Outline of the Thesis

Chapter 2 reviews related works, providing readers with the necessary background to understand the work of this project.

Chapter 3 is the main body of this project, which details the proposed methodology, including projective up-sampling, truncated inverse sigmoid volume sampling, and the designed loss function for model training.

Chapter 4 details an experiment that deployed the method detailed in chapter 3 on 3 objects in White City Campus, Imperial College London. Visualisation of the reconstruction results and quantitative evaluation result are included.

Chapter 5 discuss the findings in chapter 4, with detailed discussion covering notable observations and their connection to other researches.

The conclusion summarises this project and addresses potential future work.

2 Literature Review

Contents

2.1 LiDAR Technology 5
2.2 Point Clouds 6
2.3 LOAM and Dense Mapping 6
2.4 3D reconstruction and Novel View Synthesis 7
2.5 Neural Radiance Field 8
2.6 NeRF methodology on LiDAR 10
2.7 Loss functions for NeRF Models 11
2.8 Volume Sampling 12
2.9 Implicit and Explicit 3D Representations 13

This chapter provides the background information to understand context of this project. This chapter also introduce related work that inspired and facilitated this project.

2.1 LiDAR Technology

Light Detection and Ranging (LiDAR) technology is an active and remote sensing technology. A LiDAR sensor would fires a short pulse of laser in a known direction. If the pulse hits a surface, it would be reflected. The intensity of the reflected pulse is monitored by the LiDAR sensor. By calculating the time difference between firing the pulse and receiving the returned pulse, the distance of that surface segment from the sensor can be estimated by: $d = \frac{c \cdot \delta t}{2}$, where d is the estimated distance, c is speed of light, and δt is the time difference. The surface segment is usually modelled as a 3D point. By repeating the above process, one can obtain a set of 3D points - a point cloud, in the frame of reference of the sensor.

LiDAR sensors are widely used in drone and autonomous vehicles, as they are robust outdoor conditions and provide depth information. A common LiDAR sensor design is to have the sensor rotating 360 degree, providing depth information across a horizontal field of view [1]. LiDAR data are notorious for its sparsity of measurement, which presents challenges for 3D reconstruction [2] [3] [4]. In this project, the LiDAR data from the Livox LiDAR sensor will be used, which provides a dense point cloud measurement, with field of view coverage of approximately 90% for 1 second scan [5]. This overcomes the problem of sparsity when utilising LiDAR data for 3D reconstruction.

2.2 Point Clouds

Point cloud serve as the fundamental data type for LiDAR sensors, and techniques regarding point clouds have been well studied. In particular, point cloud registration refers aligning two or more point clouds collected from the same scene into a single frame of reference, such that they are consistent with the underlying geometry. A typical point cloud registration method utilizes iterative closest point algorithm [6]. The registration quality of point cloud plays an important role for reconstruction quality in this project, which will be discussed in chapter 5.

Contrary to image data, point clouds contains 3D information. This project gains interest in the 3D information that LiDAR point cloud contains and utilize the properties of point clouds to achieve robust 3D reconstruction.

2.3 LOAM and Dense Mapping

LiDAR Odometry and Mapping (LOAM) [7] is a fundamental framework for processing LiDAR data to achieve precise mapping and localization. Combining with techniques from Simultaneous Localization and Mapping (SLAM) such as loop closure, an accurate pose estimation and mapping from LiDAR data can be achieved.

LOAM techniques contribute to the creation of dense mapping from LiDAR data, which are critical for applications requiring high-resolution environmental models. Typical pipeline for creating a dense map includes collecting LiDAR point clouds, utilizing the LOAM algorithm or other algorithm to estimate sensor poses, creating a sparse map from LiDAR point clouds, and finally applying surface reconstruction techniques such as poisson surface reconstruction [8], truncated signed distance function (TSDF) [9] or signed distance function (SDF) [10] to convert the sparse

map into a dense map. Recently, researches have been made to use the dense map representation for odometry and mapping, achieving real time dense mapping [4] [11] [12].

While LiDAR dense mapping techniques are able to achieve decent 3D reconstruction quality, the representation obtained are often explicit or consume non-trivial amount of memory. For example, SHINE-Mapping utilizes oct-tree embedding to provide input to a shallow neural network, and predict signed distance function (SDF) for a local region [10]. NeRF-LOAM utilizes voxel embedding and neural networks as SDF to achieve LOAM [4]. Both of which consume non-trivial amount of memory for the embedding.

Dynamic objects are also a challenge in dense mapping. The typical technique to tackle with dynamic objects is to apply a filter to points from dynamic objects [13] [14], which is prone to filtering error and require extra data processing.

Methodologies from LOAM dense mapping provided great insights to this project, which focuses on developing an implicit 3D representation for scenes using NeRF framework. This project will show and discuss the capability of the proposed method of omitting dynamic objects in reconstruction. Additionally, LOAM-derived mapping and localization have been utilized as a source for obtaining accurate pose information, which was used in training NeRF models for 3D reconstruction in this project.

2.4 3D reconstruction and Novel View Synthesis

3D reconstruction and view synthesis are two closely related and long standing problems in computer vision. 3D reconstruction involves reproducing the underlying 3D representation of an object or a scene from sparse information, such as images or a point cloud. View synthesis refers to creating consistent views from novel view point.

In the past, analytical methodologies such as structured from motion, structured light or photometric stereo have been developed to reconstruct depth maps or 3D scenes using multiple images or active lightning. For example, COLMAP are capable of producing 3D points and camera poses from given set of RGB images of a scene by feature matching [15]. These methods provides the foundation for modern 3D reconstruction techniques.

Recently, remarkable progress have been made by learnt novel view synthesis from differentiable rendering of RGB images. The method of neural radiance field (NeRF) achieved remarkable results

by optimizing neural network that synthesis views with volume rendering [16]. Later, 3D Gaussian Splatting [17] also achieved remarkable view synthesis quality with significant improvement in rendering speed.

In the framework of RGB NeRF or 3D Gaussian Splatting, the primary goal is novel view synthesis, and 3D reconstruction is a by-product. For instance, NeRF used the predicted optical density to reconstruct the 3D mesh of objects [16]. Although 3D geometry objects can be reconstructed, RGB NeRF framework has no obligation for the reconstruction to be loyal to the ground truth geometry of objects or scenes.

With this in mind, this project approached the NeRF framework with LiDAR data. This project utilized LiDAR data from multiple viewpoints to reconstruct scenes using NeRF models. Which provides a more robust reconstruction to the geometry of objects compared to RGB NeRF.

Researches in RGB NeRF has provided a problem framework for this project, and technical details used in researches such as NeRF model architectures and pipeline serve as valuable reference for this project. Researches in RGB NeRF also provide insight for improvement of reconstruction results achieved in this project, which will be included in the future work of this project.

2.5 Neural Radiance Field

Neural Radiance Field [16], has been a popular field of research recently. It has achieved remarkable scene reconstruction by representing a scene using Neural Network. Figure 2.1 illustrates the pipeline for training the original NeRF model.

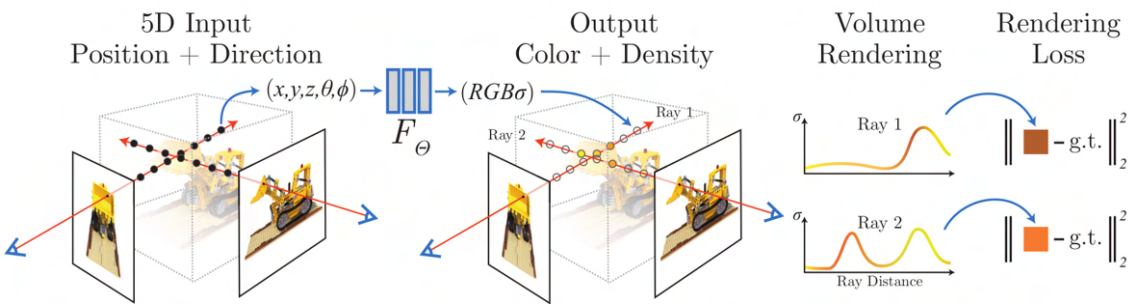

Figure 2.1: The NeRF 3D reconstruction pipeline [16].

Where F is a multilayer perceptron (MLP) that accepts position and view direction as inputs and returns optical density and colour as outputs. Using pixel colours from a given set of sparse

image from a scene as training data, the model was trained to minimize the pixel-wise photo-metric loss and can achieve novel view synthesis.

Formally, consider a camera located at $\mathbf{o}$, a camera ray that passes through the camera centre and a given pixel is defined by the equation $\mathbf{r}(t) = \mathbf{o} + t \cdot \mathbf{d}$, $t \in \mathbb{R}$. NeRF estimates the colour of that pixel with the following quadrature approximation of the volume rendering equation[18]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i \delta_i))\mathbf{c}_i \tag{2.1}$$

Where:

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

$\hat{C}$ is the colour estimated for that pixel, $\mathbf{c}_i$ and σ_i are the colour and density at the point $\mathbf{r}(t_i)$ observed from direction $\mathbf{d}$. $\delta_i = t_{i+1} - t_i$ is the distance between two sampled points. One could regard $exp(-\sigma(t_i)\delta_i)$ as the transmittance, which is the probability that a camera ray "passes through" the volume between t_{i+1} and t_i. It immediately follows that $\alpha_i = 1 - exp(-\sigma(t_i)\delta_i)$ is the opacity, which is the probability that the ray "terminates" at t_i. The cumulative transmittance T_i is the probability that the ray "reaches" t_i, obtained by multiplying the transmittance of volumes that the ray passed through.

In NeRF models, $\mathbf{c}_i$ and σ_i are predicted with a successive MLP F parameterized by θ:

$$\sigma(t), \mathbf{c}(t) = F(\mathbf{r}(t), \mathbf{d}|\theta) \tag{2.2}$$

As equation 2.1 is differentiable with respect to $\mathbf{c}(t)$ and $\sigma(t)$, θ can be optimized by backpropagation and gradient descent on minimizing the loss between $\hat{C}$ and actual colour of each pixel from the given set of images. From a qualitative perspective, the NeRF model will remember mapping of density and colour of the scene as a function of position and view direction through training, thus a NeRF model can be regarded as an representation of the scene. To render a NeRF model, one can query the model for σ and c and render pixel colour using equation 2.1.

The formulation of equation 2.1 has highly inspired this project where the focus in this project would be to obtain the density reconstruction from the model. Colour reconstruction will be ignored as LiDAR data does not provide an explicit estimation of the colour of the scene. Implementation details will be discussed in Chapter 3 of this project.

A variety of improvements have then been developed to the original NeRF. Including NeRF view

synthesis for outdoor scenes[19] [20], eliminating the need for camera poses by jointly optimizing the colour loss and camera poses[21], improving aliasing problem for NeRF model [22] [23] and reducing the number of images needed by providing depth prior to the model[2]. These improvements provided inspiration for this projects to extend the work of NeRF from RGB image reconstruction (RGB NeRF) to LiDAR data and provide direction for future work.

2.6 NeRF methodology on LiDAR

The idea of applying LiDAR data in NeRF framework has been explored in recent research. PCGen [24] utilized an MLP as a surrogate model to predict the ray drop probability for LiDAR rays as a function of distance, incident angle and intensity of a LiDAR ray. LiDeNeRF [3] combined LiDAR data and RGB image to predict depth map for novel views. [25] used LiDAR data as a depth prior and combined with Conditional Adversarial Networks to improve novel view image quality. NeRF-LiDAR [26], LiDAR-NeRF [27] and Neural LiDAR Fields [28] adopted the idea of volume rendering for novel view synthesis with LiDAR sensors.

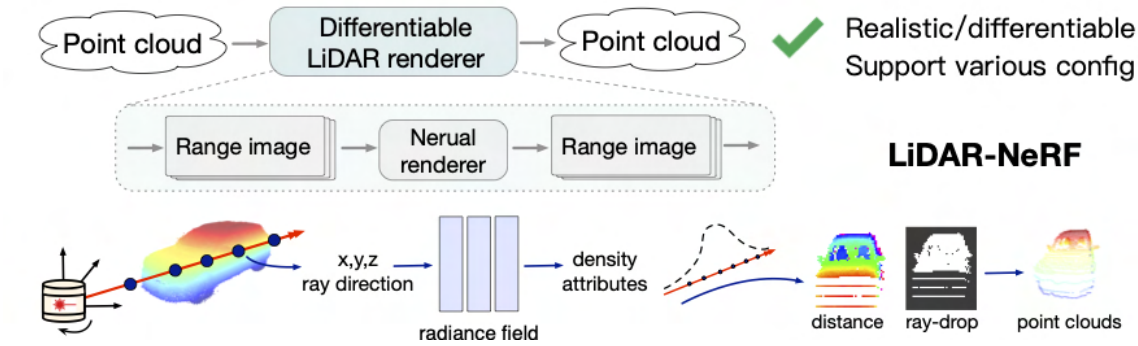

Figure 2.2: The LiDAR NeRF pipeline [16].

A detailed review will be dedicated to LiDAR-NeRF [27] as their work highly inspired this project. Given a set of LiDAR scan of an object (e.g. a car) from different view point, LiDAR-NeRF focuses on LiDAR view synthesis of the object. Figure 2.2 outlines the pipeline of LiDAR-NeRF.

First, a range image was constructed for each frame of the LiDAR data. This is done by projecting each 3D point in a frame onto a pseudo image:

$$\begin{pmatrix} h \\ w \end{pmatrix} = \begin{pmatrix} \left(1 - (\arcsin(z,d) + |f_{\text{down}}|)\, f_v^{-1}\right) H \\ \frac{1}{2}\left(1 - \arctan(y,x)\pi^{-1}\right) W \end{pmatrix} \tag{2.3}$$

Where h and w are the pixel coordinates of the projected 3D points, x, y, z are the Cartesian

coordinate of each 3D point, d is the distance of each 3D point from the LiDAR sensor, H and W are the height and width of the pseudo image. When more than one points was projected onto the same pseudo-pixel, point with smallest distance was kept, and pixels with no points projected onto was filled with zeros.

The NeRF framework were then applied with each pseudo image as input, where a NeRF model predicts the density, ray-drop probability and intensity for a given position and view direction. Given a LiDAR ray defined by $\mathbf{r}(u) = \mathbf{o} + u \cdot \mathbf{d}$, an expected distance $\hat{D}$ of the ray can be modelled with:

$$\hat{D}(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \cdot \delta_i)\right) u_i \tag{2.4}$$

Where:

$$T_i = exp(-\sum_{j=1}^{i-1} \sigma_i \delta_j) \tag{2.5}$$

Where $\hat{D}$ is the expected termination distance of a LiDAR ray, u_i are positions sampled on the ray, σ_i is the density at u_i and δ_i is the distance between consecutive samples. T_i can be regarded as the probability that the LiDAR ray did not hit anything before reaching u_i, and $\alpha_i = (1 - exp(-\sigma_i \cdot \delta_i))$ as the probability of the LiDAR ray terminates at u_i. In LiDAR-NeRF, the NeRF model was optimized to minimise loss between $\hat{D}$ and the actual depth measured by LiDAR data along with other losses.

The mentioned researches achieved remarkable results for LiDAR data view synthesis. This project, on the other hand, will shed light on the scene reconstruction capability of NeRF models. The focus of this project will be on training a NeRF model that predicts appropriate σ to reconstruct a given scene from LiDAR data.

2.7 Loss functions for NeRF Models

In RGB NeRF model training, the photometric loss function is typically used, i.e. $\left\|\hat{C}(\mathbf{r}) - C(\mathbf{r})\right\|_2^2$, the L2 loss between predicted colour $\hat{C}$ and actual colour C of each pixel in the input set of images. With LiDAR data, a different approach was taken. LiDAR-NeRF and NeRF-LiDAR used the distance loss to train the NeRF model [27] [26] :

$$\mathcal{L}_{\text{distance}}(\mathbf{r}) = \sum_{\mathbf{r} \in R} \left\|\widehat{D}(\mathbf{r}) - D(\mathbf{r})\right\|_1 \tag{2.6}$$

Where $\hat{D}$ is given from equation 2.4, and R is the training batch of LiDAR rays.

DS-NeRF proposed the ray distribution loss, which optimize the termination distribution of rays by minimizing the KL divergence between rendered termination distribution and a normal distribution [2], which was derived to be:

$$\mathcal{L}_{\text{Distribution}} = \mathbb{E}_{x_i \in \mathcal{X}_j} \int \log h(t) \exp\left(-\frac{(t - \mathbf{D}_{ij})^2}{2\sigma_i^2}\right) dt \tag{2.7}$$

$h(t)$ is the predicted termination distribution of a ray, $\mathbf{D}$ is the ground truth depth given by COLMAP a point cloud, σ is a chosen variance of the normal distribution which models the ground truth termination distribution.

SHINE-Mapping proposed to use BCE loss to optimize sigmoid mapping of a SDF [10]. similarly, [29] proposed to use BCE loss for optimizing the occupancy probability to a log normal distribution:

$$\mathcal{L}_d(\tilde{\mathbf{p}}_i(z)) = \begin{cases} -\log(1 - f_\theta(\tilde{\mathbf{p}}_i(z))), & \text{for } z < z_i \\ -P_{\text{occ}}(z) \log(f_\theta(\tilde{\mathbf{p}}_i(z))), & \text{for } z \geq z_i \end{cases} \tag{2.8}$$

Where z_i is the termination distance of a ray, P_{occ} is the target occupancy probability, and f_θ is the neural network model.

Exploiting the 3D nature of point clouds, it is observed that one can formulate loss functions on cumulative transmittance or weight contribution to the expected depth directly. This is particularly important for designing loss functions in training NeRF model with LiDAR data. The reviewed literature have provided important reference and inspiration for loss function design in this project.

2.8 Volume Sampling

Volume sampling strategy is an important topic in NeRF model training. To train a NeRF model with photometric-loss[16] or depth-loss [27], one would have to select a set of positions in the scene, that is, to deploy a volume sampling strategy.

Consider equation 2.2, a set of position along the ray, $t_i, i \in [0, N]$, is chosen to produce σ and $\mathbf{c}$ for rendering the expected colour $\hat{C}$. In the original NeRF paper, a stratified hierarchical sampling strategy was adopted. In which a "coarse" model was trained using the following stratified sampling

strategy[16]:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)\right] \tag{2.9}$$

Where t_i are scalars sampled from a uniform random distribution within N evenly spaced bins from the near plane t_n, to the far plane t_f. Using the prediction of the "coarse" model, a "fine" model was then trained with denser samples at positions that contributes more to the expected colour predicted in the "coarse" model. Author of the original NeRF research addressed a stratified sampling strategy could avoid the NeRF model only being queried at a certain set of location, and avoid limiting its representation resolution. Training the "fine" model with expected colour contribution weighted samples could reduce the NeRF model to be trained on free or occluded space, which improves its representation efficiency.

Various improvements on the volume sampling strategy were then proposed. Mip-NeRF [22] samples a series of conical frustums from a cone cast from the camera centre to each pixel, and showed that it reduced the aliasing problem in the original NeRF with improved image reconstruction quality. Zip-NeRF [30] extended Mip-NeRF's work, and adopted isotropic gaussian multiple sampling strategy in the conical frustum which improves model training speed compared to Mip-NeRF.

While researches such as NeRF, mip-NeRF and zip-NeRF proposed various volume sampling startegy for NeRF model training, their method was designed for RGB image settings and assumed that depth of the scene is unknown [16] [22] [30]. In this project, truncated inverse sigmoid volume sampling strategy was proposed and detailed in Chapter 3, which account for the fact that depth is known in LiDAR data.

2.9 Implicit and Explicit 3D Representations

There has been discussion about implicit and explicit representation. For example, the equation $x^2 + y^2 + z^2 = r^2$ is an implicit representation a sphere in $[x, y, z] \in \mathbb{R}^3$, while a 3D mesh that approximates the sphere would be an explicit 3D representation.

Examples of explicit representations include voxels, surfels, point clouds or meshes, researches on learnt explicit 3D reconstruction have been made [31] [32]. There has been increase in interest for learnt implicit 3D representation, such as Signed Distance Function [33] [34], Occupancy Function [35], or NeRF models.

Implicit representations generally produce better resolution, are more compact, and can potentially be easier for manipulation. For example, to join two SDF geometry, one can simply take the minimum between the two SDF. In this project, the advantages for implicit representations serves as a motivation for creating a 3D scene implicit representation of NeRF model from LiDAR data.

3 Methodology

Contents

3.1 Problem Definition 16
3.2 Projective Upsampling 17
3.3 Truncated Inverse Sigmoid Volume Sampling 20
3.4 Loss Functions 21
3.5 Model Architecture 24
3.6 Rendering the NeRF Model 25

This chapter will first introduce the problem definition, then explain details of the proposed methods that utilize a NeRF model with LiDAR data for 3D scene reconstruction. Figure 3.1 illustrates the overall pipeline proposed in this project. A detailed description will be provided to each part in figure 3.1. Table 3.1 include a summary of selected variables used in this chapter.

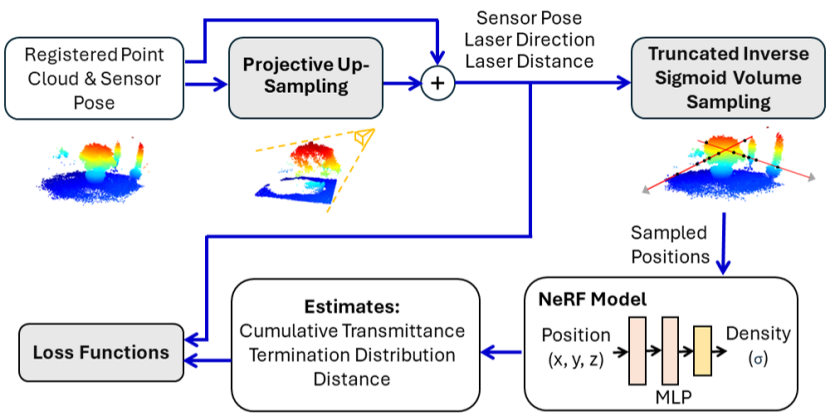

Figure 3.1: Overview of the proposed pipeline in this project.

Symbol	Description
${}^{\pi}P$	point cloud in sensor frame
π	sensor extrinsic
τ	pseudo sensor extrinsic
P_{raw}	registered point cloud from all point clouds in each sensor frame
P_{up}	projective upsampled point cloud
P_{combined}	registered point cloud combined with P_{raw} and P_{up}
u_n	near plane in volume sampling
u_f	far plane in volume sampling
u_d	termination distance of a LiDAR ray
σ_s	variance controller for volume sampling
$\mathcal{L}_T$	cumulative transmittance loss
$\mathcal{L}_h$	termination distribution loss
$\mathcal{L}_d$	distance loss

Table 3.1: Variables and their descriptions

3.1 Problem Definition

Formally, given N frame of point clouds collected from an object or scene of interest, expressed in corresponding LiDAR sensor frame: $\{{}^{\pi_i}P_i, i \in [0, N]\}$ and their sensor poses $\{\pi_i, i \in [0, N]\}$, the goal will be to reconstruct the scene with density representation with a NeRF model. Optimization of the NeRF model can be denoted as:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\hat{\sigma}|({}^{\pi}P, \pi)) \tag{3.1}$$

$$\hat{\sigma} = \mathcal{F}(\mathbf{p}|\theta)$$

Where:

$\mathbf{p}$ is a 3-dimensional vector representing position in the scene.

$\mathcal{F}$ is the NeRF model, which is an MLP. θ is the weights of the NeRF model.

θ^* is the optimal parameter for the NeRF model.

$\hat{\sigma}$ is the estimated density, which is a function of $\mathbf{p}$ parameterized by θ, the function is approximated with $\mathcal{F}$.

$\mathcal{L}$ is a loss function to be designed to achieve a desired reconstruction quality of the object or scene.

Furthermore, the following qualitative assumptions were made in this project. The reconstruction for a more generalized settings will be included in future work of this project:

1. The scene is static.
2. The LiDAR sensor pose is object tracking during data collection, meaning that it points towards the object of interest.

3.2 Projective Upsampling

Since sensor poses π are assumed given, the point cloud in each sensor frame can be registered into one single frame of reference. Which gives ${}^{w}P_{\text{raw}}$, where w is the world frame and P_{raw} is the registered 3D points from all sensor frame.

While one can proceed the pipeline in figure 3.1 directly with ${}^{w}P_{\text{raw}}$, it was found that this would give a noisy reconstruction result. This is because during data collection, the sensor would typically be held at a level from ground. As shown in figure 3.2, volumes such as the top of the object was not scanned by the sensor and will lead to a blind spot in data collected.

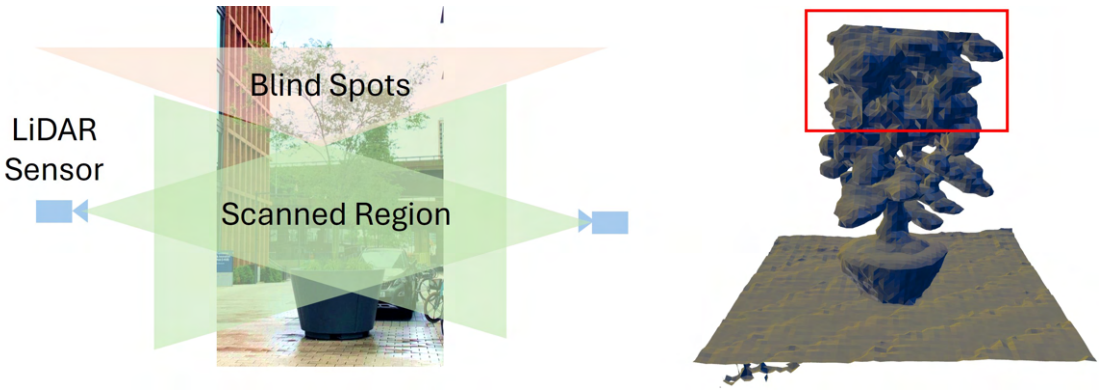

Figure 3.2: Illustration of blind spots during data collection (left), and reconstruction result without using projective upsampling, resulting in noise on top of the object (right).

To tackle with this problem, this project propose *perspective upsampling*. Inspiration was drawn from equation 2.3 from LiDAR NeRF [27] and through project discussion. The core idea is to create a set of pseudo LiDAR sensor around the scene, and project ${}^{w}P_{\text{raw}}$ to pseudo image of each pseudo sensor while keeping the points with smallest depth in each pseudo pixel. Doing this, blind spots in the dataset can be eliminated.

Formally, given a pseudo sensor with extrinsic τ and intrinsic K, ${}^{w}P_{\text{raw}}$ was first transformed to the pseudo sensor reference frame to give ${}^{\tau}P_{\text{raw}}$. Then ${}^{\tau}P_{\text{raw}}$ can be projected onto a pseudo image by:

$$\hat{P} = \frac{1}{z} K \cdot^{\tau} P_{\text{raw}} \tag{3.2}$$

Where z is the z-axis position of 3D points in the sensor frame which represents the depth of the point, $\hat{P}$ is the projected points on the pseudo image in homogeneous coordinate. In each pseudo

pixel, points in $\hat{P}$ within that pixel with smallest z in ${}^{\tau}P_{\text{raw}}$ are kept:

$$\tilde{p}_{h,w} = \min_{z}(p \in \hat{P} | h < \hat{u} < h + \delta_h, w < \hat{v} < w + \delta_w) \tag{3.3}$$

Where h and w are height and width position of that pixel, δ_h and δ_w are the height and width of that pixel, $\hat{u}$ and $\hat{v}$ are the position of the projected point on the pseudo image. Finally, point in ${}^{\tau}P_{\text{raw}}$ corresponding to each $\tilde{p}_{h,w}$ will be sampled. The sampled points will then be regarded as LiDAR ray fired from the pseudo sensor and terminated at the sampled point's position. Figure 3.3 illustrates projective upsampling with a single pseudo sensor.

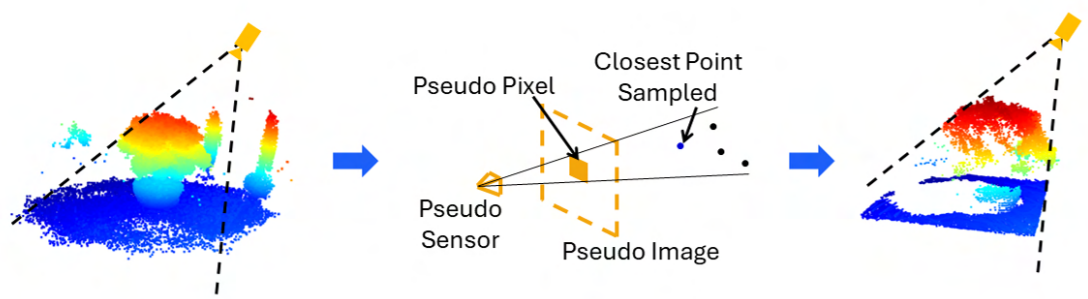

Figure 3.3: A chosen location of a pseudo sensor (left), when more then one point are projected onto a pseudo pixel, the closest point is sampled (middle), sampled points of projective upsampling of the pseudo sensor(right).

To generalize the procedure of projective upsampling across different scenes, a set of N pseudo sensors can be used. Without loss of generality, one can assume that P_{raw} is normalized and centre at world frame origin. Then, the pseudo sensor positions were chosen to be evenly separated on a sphere with radius r, centered at the origin and contain P_{raw}, and each pseudo sensor was designed to point towards the origin.

An even distribution of camera position on a sphere can be achieved with the Fibonacci Lattice formula [36]. The sensor position o_i of the ith pseudo sensor is given by:

$$\phi = arccos(1 - \frac{2 \cdot i}{N})$$
$$\theta = \pi \cdot (1 + \sqrt{5}) \cdot i$$

$$o_i = [x_i, y_i, z_i] = r \cdot [sin(\phi) \cdot cos(\theta), sin(\phi) \cdot sin(\theta), cos(\phi)] \tag{3.4}$$

Since each sensor is pointing towards the origin, it follows immediately that the sensor direction

o_i is given to be:

$$d_i = -\frac{t_i}{r} \tag{3.5}$$

Figure 3.4 illustrates how the pseudo sensor locates with the proposed method. Algorithm 1 shows the brief pseudo code of projective upsampling.

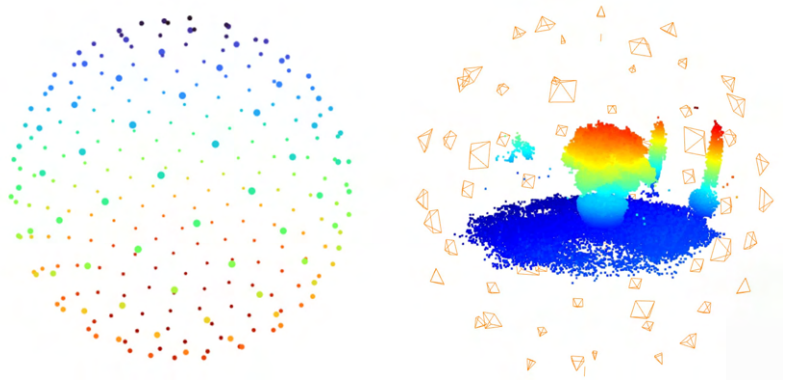

Figure 3.4: Fibonacci Lattice points on a sphere (left) [37], illustration of pseudo sensor pose with a given point cloud (right).

Algorithm 1 Brief Projective Upsampling

Input:
N: Number of pseudo sensors
P_{raw}: Registered point cloud collected from sensor
π: sensor pose
Output: Project Upsampled point cloud and sensor poses

```
i ← 0                                          ▷ Initialize count
P_up ← []                                      ▷ Initialize upsampled points as empty array
π_combine ← π                                  ▷ Initialize combined sensor pose as input sensor pose
for i < N do
    o, d ← fibonacciLattice(i, N)
    P_up ← {P_up, projectiveSampling(P_raw, o, d)}
    π_combine ← {π_combine, [o, d]}
    i ← i + 1
end for
P_combine ← {P_up, P_raw}
return P_combine, π_combine
```

It should be noted that the pseudo sensor pose shall also be combined with the original sensor pose. Together, P_{combined} and π_{combine} serve as the input point cloud and sensor poses in equation 3.1, where the registration of point clouds can be considered to be absorbed into the loss function. It was found that when using a large number of pseudo sensors or only using P_{up} and τ to train the NeRF model, the reconstruction result will be blurred. Thus the number of pseudo sensor N is a hyperparameter, which details will be discussed in Section 5.7.

3.3 Truncated Inverse Sigmoid Volume Sampling

NeRF addressed that querying the model at free or occluded space is inefficient, as they do not contribute much to the rendered colour[16]. Similarly, SHINE-Mapping addressed that free or occluded space does not contribute much to the expected distance of a LiDAR ray, and that dense sampling near surfaces would improve training efficiency [10]. With this in mind, this project propose a simple volume sampling strategy, *truncated inverse sigmoid volume sampling*, for sampling positions along a LiDAR ray. With a known termination distance of the each ray, samples can be controlled to be dense around occupied space.

Consider a LiDAR ray defined by the equation $\mathbf{r}(t) = \mathbf{o} + u \cdot \mathbf{d}$, terminate at distance u_d, and a given near plane u_n and far plane u_f. First, a sigmoid function parameterized by σ_s, centered at u_d can be constructed in the 1 dimensional subspace of $u \in \mathbb{R}$, namely the u-space. Formally:

$$S^*(u) = S(\frac{u - u_d}{\sigma_s}) \tag{3.6}$$

Where $S(x) = 1/(1 + exp(-x))$ is the sigmoid function, $S^*(\cdot)$ is the constructed function, and σ_s is a variable that controls the variance of sampled positions. Shown in the left of figure 3.5 visualizes the constructed function and its relation to the LiDAR ray.

An upper bound $t_{\max}$ and a lower bound $t_{\min}$ for stratified sampling are then obtained by $t_{\max} = S^*(u_f)$ and $t_{\min} = S^*(u_n)$. A stratified uniform random samples with M number of bins can then be generated between $t_{\min}$ and $t_{\max}$, where t_i is the ith sample:

$$t_i \sim \mathcal{U}\left[t_{min} + \frac{i-1}{M}\left(t_{max} - t_{min}\right), t_n + \frac{i}{M}\left(t_{max} - t_{min}\right)\right] \tag{3.7}$$

Where $\mathcal{U}$ is the uniform random distribution. Then, applying inverse of $S^*(\cdot)$ to t_i gives the ith sample in the u-space:

$$u_i = S^{-*}(t_i) \tag{3.8}$$

Where $S^{-*}(x) = ln(\frac{x}{1-x}) \cdot \sigma + u_d$ is the inverse of $S^*(\cdot)$. Finally, the sampled point in the 3D space are given by $\mathbf{p}_i = \mathbf{o} + u_i \cdot \mathbf{d}$, which completes the sampling procedure. The right of figure 3.5 illustrates how $S^{-*}(t)$ is converted to sampled positions in the 3D space. Figure 3.6 illustrates distribution of samples with different σ_s. As a sigmoid function is the cumulative distribution function of logistic distribution, the proposed method effectively samples points with the logistic distribution around the termination of the ray, truncated with a near and far plane. σ_s is a

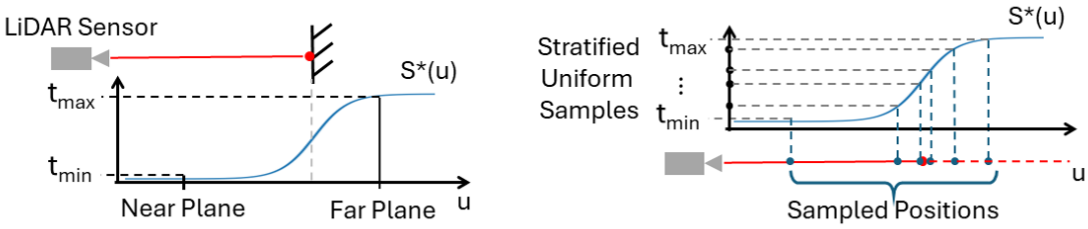

Figure 3.5: Applying S^* to near plane and far plane to get $t_{\min}$ and $t_{\max}$ (left), and applying S^* to stratified uniform random samples to get sampled positions in u-space.

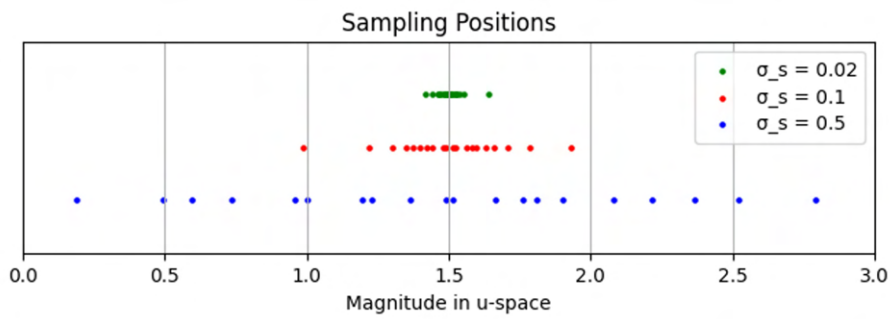

Figure 3.6: Illustration of sample distribution with different σ_s, with 20 samples, near plane $u_n = 0$, far plane $u_f = 3$, termination distance $u_d = 1.5$.

hyperparameter that controls the variance of the sampled points along the ray, where a larger σ_s results in a larger sampling variance, vice versa. σ_s could be reduced in the training schedule along epoch, which improve the sampling efficiency in model training. Discussion about varying σ_s is included in Section 5.9.

3.4 Loss Functions

This section introduces three loss functions: the distance loss $\mathcal{L}_d$, the termination distribution loss $\mathcal{L}_h$ and the cumulative transmittance loss $\mathcal{L}_T$, for optimizing the NeRF model. $\mathcal{L}_d$ was used in LiDAR NeRF and NeRF-LiDAR [27] [26]. $\mathcal{L}_h$ was proposed in DS-NeRF [2], and $\mathcal{L}_T$ was proposed in [29], in which modification was made for the setting of this project. The proposed loss function is a weighted sum of $\mathcal{L}_T$, $\mathcal{L}_h$ and $\mathcal{L}_d$.

Distance Loss $\mathcal{L}_d$

Recall that in LiDAR NeRF, an expected distance of a ray $\hat{d}$ can be rendered with equation 2.4.

Knowing that the ground truth termination distance d^* of a LiDAR ray is be given by:

$$d^* = \sqrt{x^2 + y^2 + z^2}$$

Where x, y, z are the position of the 3D point in the sensor frame. A loss function between the expected distance and ground truth distance can then be used to optimize the NeRF model, such as L1 loss as mentioned in equation 2.6. This project follows the distance loss in equation 2.6 of using L1 loss with the same expected distance rendering method given in equation 2.4.

Cumulative Transmittance Loss $\mathcal{L}_T$

SHINE-Mapping models the scene with signed distance function and mapped the SDF with a sigmoid function, the author addressed that the mapped value could be regarded as an indicator for the occupancy probability [10]. Similarly, [29] adopt the idea of occupancy field and applied binary cross entropy (BCE) loss to the occupancy function for optimising NeRF model as reviewed in equation 2.8. Inspired by SHINE-Mapping and [29], this project extend their idea and propose to optimise the cumulative transmittance with a mirrored sigmoid function.

Consider a LiDAR ray that terminated at distance u_d from the sensor, one could model the cumulative transmittance T^* as:

$$T^*(u) = 1 - S(\frac{u - u_d}{\sigma_T}) \tag{3.9}$$

Where $1 - S(\cdot)$ is the mirrored sigmoid function, $u \in \mathbb{R}$ is a scalar that lives in the u-space as discussed in Section 3.3 and σ_t controls how sharp we model the cumulative transmittance drop upon termination of the ray. Plot of T^* and the relationship between T^* with the LiDAR ray is shown in figure 3.7. Effectively, T^* models the survival function representing the probability that the ray travels further than u_i. Clearly $0 < T^* \leq 1$. On the other hand, using equation 2.5 as the predicted cumulative transmittance $\hat{T}_i$, since density σ_i and distance between consecutive samples δ_i are non-negative, it is also true that $0 < T_i \leq 1$. Thus, the binary cross entropy can be applied to T_i^* and $\hat{T}_i$ as a loss function:

$$\mathcal{L}_T = T_i^* \cdot ln(\hat{T}_i) + (1 - T_i^*) \cdot ln(1 - \hat{T}_i) \tag{3.10}$$

From discussion in SHINE-Mapping, the BCE loss is more sensitive for samples near the surface compared to losses such as L1 or L2 loss [10]. The effect of $\mathcal{L}_T$ will be shown in Chapter 4.

Termination Distribution Loss $\mathcal{L}_h$

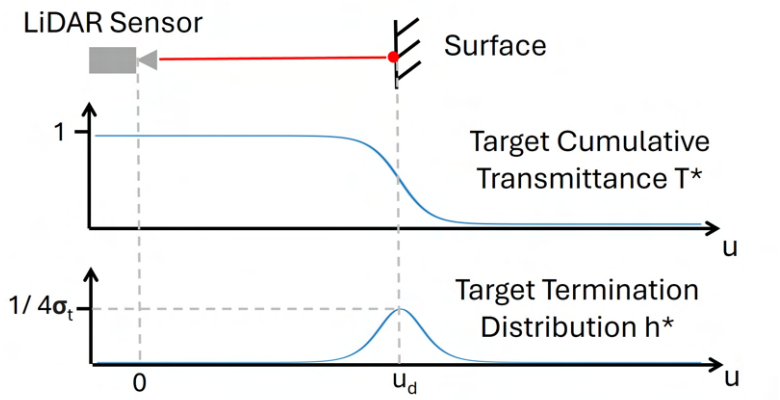

Figure 3.7: Relation and graph of target cumulative transmittance T^* and termination distribution h^* to a LiDAR ray.

DS-NeRF proposed to use a termination distribution loss as reviewed in Section 2.6 equation 2.7. While DS-NeRF used a COLMAP estimated depth to formulate a ground truth termination distribution [2], this is also applicable to LiDAR data. Following equation 2.4, the predicted termination probability mass function is given to be:

$$\hat{h}(u_i) = T_i \cdot \alpha_i \tag{3.11}$$

Where $h(u_i)$ is the probability that the ray terminates at u_i, and α_i is the opacity at u_i. On the other hand, since the ground truth cumulative transmittance is modelled with T^* as a survival function, $1 - T^*$ will be the cumulative distribution function. It follows that the ground truth probability density distribution $h(u)$ for the ray terminating at u_i shall be the gradient of $1 - T^*$. Noting that gradient of the sigmoid function is $S'(x) = S(x) \cdot (1 - S(x))$ gives:

$$h(u) = \frac{d(1 - T^*)}{du} = (1 - T^*) \cdot (T^*) \cdot \left(\frac{1}{\sigma_t}\right) \tag{3.12}$$

Figure 3.7 shows the relationship between the LiDAR ray and $h(u)$, and a plot of $h(u)$. $h(u)$ can then be used to approximate the ground truth probability mass function:

$$h^*(u_i) = h(u_i) * \delta_i \tag{3.13}$$

Finally, the KL divergence loss can be used for comparing $h*$ and $\hat{h}$ to give the termination distribution loss:

$$\mathcal{L}_h = \sum_{i=1}^{N} h^*(u_i) \cdot ln\left(\frac{\hat{h}(u_i)}{h^*(u_i)}\right) \tag{3.14}$$

The proposed loss function $\mathcal{L}_{total}$ will be:

$$\mathcal{L}_{total} = \lambda_T \mathcal{L}_T + \lambda_h \mathcal{L}_h + \lambda_d \mathcal{L}_d \tag{3.15}$$

Where λ_T, λ_h and λ_d are weights for each losses. Note that the construction of $\mathcal{L}_T$, $\mathcal{L}_h$ and $\mathcal{L}_d$ are solely the function of $\hat{\sigma}$, ${}^{\pi}P$ and π. The u-space and construction of S^* for each ray is given by π and ${}^{\pi}P$. One can consider computation of $\hat{d}$, $\hat{h}(u)$ and $\hat{T}(u)$ to be absorbed into the loss function $\mathcal{L}$ in equation 3.1.

3.5 Model Architecture

A similar model architecture with NeRF was used proposed for $\mathcal{F}$ in equation 3.1 [16], which accepts a 3 dimensional input and returns a non-negative value for predicted density.

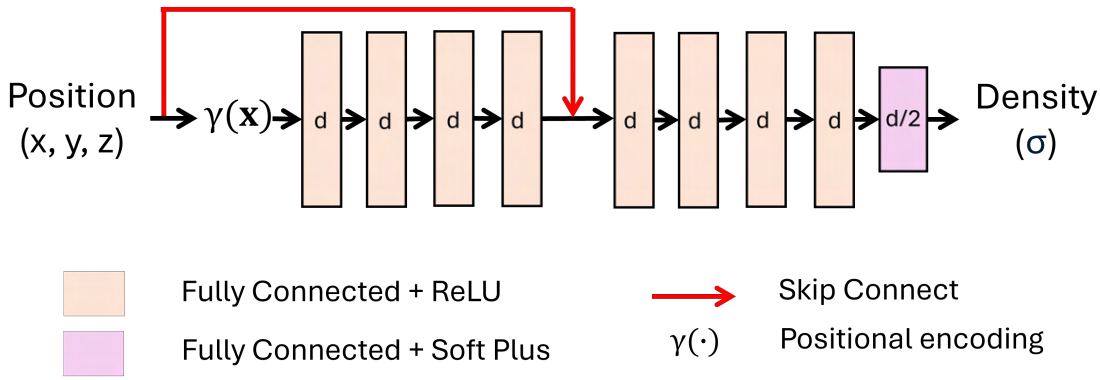

Figure 3.8: Architecture of the NeRF model.

Figure 3.8 illustrates the model architecture, where d is the hidden dimensions. The soft plus function was used as the output layer activation function to ensure the predicted density is non-negative while avoiding regions with zero-gradient. A skip connect was used to avoid gradient explosion or vanishing, and $\gamma(\cdot)$ is the positional encoding function that allows the model to handle high-frequency changes of density. The transformation for positional encoding is given to be [38]:

$$\gamma(p) = \left(\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right)\right)$$

Where L is the positional encoding dimension, which along with hidden dimension d are hyper parameters.

3.6 Rendering the NeRF Model

The same approach in NeRF was used to render the LiDAR NeRF model as a 3D mesh [16]. A density threshold was first chosen, where density of a position above that threshold would be considered occupied, and below it will be considered free space. By querying the NeRF model on a grid, the marching cube algorithm [39] can be deployed to render the predicted density as a mesh.

4

Experiment & Results

Contents

4.1 Data Collection 27
4.2 Data Processing 28
4.3 Deploying Projective Upsampling 29
4.4 Model training 30
4.5 Model Rendering 30
4.6 Qualitative Results 31
4.7 Quantitative Results 35

Experiment was conducted to validate the proposed pipeline in Chapter 3. This chapter details the experiment procedure, and shows the result of the experiment. The Code for implementing the experiment is available at: https://github.com/HonAnson/LiDAR_NeRF.git

4.1 Data Collection

A total of three datasets was collected, named box plant, round plant and building. Shown in figure 4.1 are images of the object of interest in each dataset.

A single Livox horizon LiDAR sensor [40] controlled with ROS software was used for data collection, with the setup shown in figure 4.2. The sensor was handheld, and LiDAR data was collected with the individual walking in route described in the top down view in figure 4.3.

The individual walked two rounds around the plant to collect the box plant and round plant dataset, and walked two round trips to collect the building dataset. The sensor was held such that it roughly pointed toward the object of interest and maintained approximately at horizontal

Figure 4.1: object of interest in box plant dataset (left), round plant dataset (middle) and building dataset (right).

elevation. An internal IMU sensor captured the motion of the Livox during data collection, which were then be used for odometry to provide the sensor pose. The data were stored in a rosbag file during data collection. Table 4.1 shows metadata of the datasets.

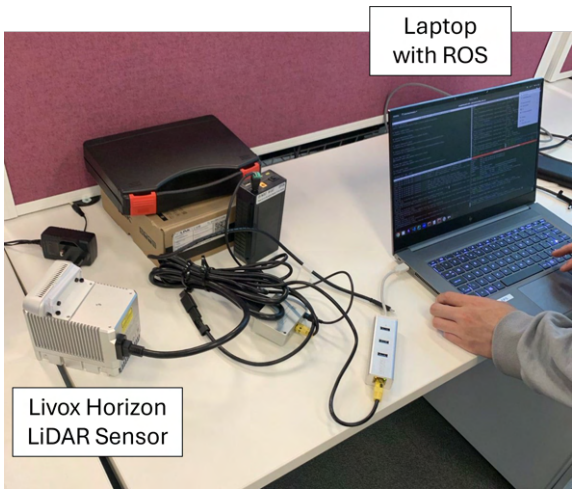

Figure 4.2: LiDAR sensor setup for data collection.

Dataset	Box Plant	Round Plant	Building
Number of Point Cloud Frames	1203	1107	971
Total number of points after trimming	4,205,092	6,932,455	10,893,642
Frame per second	10	10	10

Table 4.1: Meta data of datasets

4.2 Data Processing

The sensor pose was first estimated using PV-LIO odometry [41], which utilize both LiDAR data and internal IMU motion sensor information for estimation. Point cloud in each frame were then trimmed, with points at a distance between 0.1m and 10m from the sensor kept for box plant and

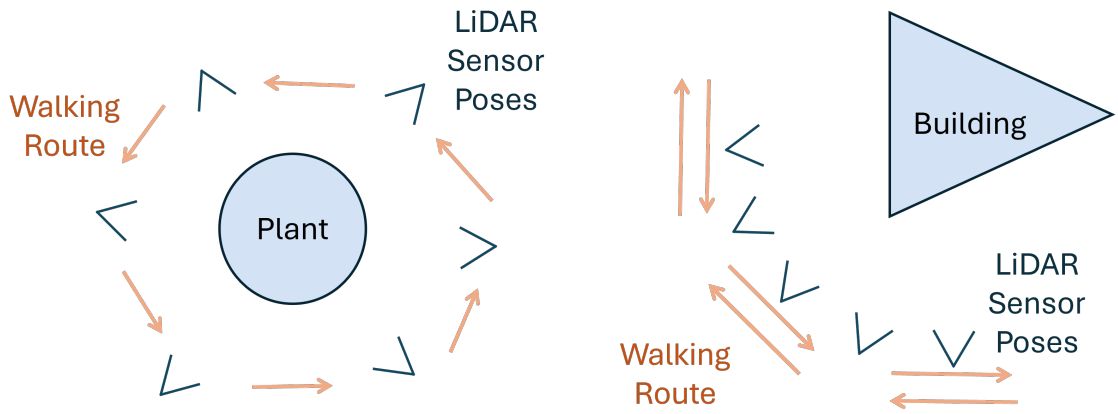

Figure 4.3: Illustration of walking route and sensor pose in during data collection for round plant and box plant dataset (left) and building dataset (right).

round plant dataset, and points at a distance between 0.1m and 20m from the sensor kept for building dataset. An example of trimming a frame of point cloud is shown in figure 4.4

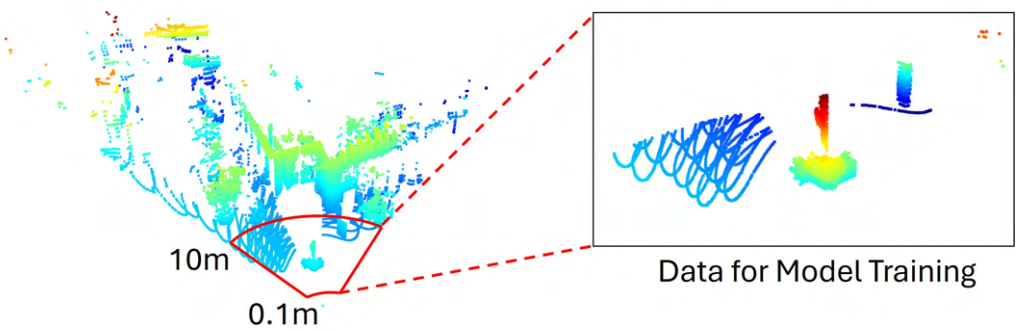

Figure 4.4: Trimming of a single frame of point cloud for box plant dataset.

Using the estimated sensor pose, the trimmed point clouds in different frames are registered the sensor pose in the first frame in each dataset. Next, the registered point cloud and sensor positions were normalized and scaled. The trimmed point cloud and sensor positions are normalized to center at [0,0,0] and scaled such that the maximum difference between the x, y or z axis positions of any two points or sensor positions is equal to 2. The normalized scaled point cloud and sensor position are P_{raw} and π_{raw}, which are the raw dataset for training.

4.3 Deploying Projective Upsampling

Projective upsampling was then applied to the raw dataset. Choosing r = 2, equation 3.4 and 3.5 gave the pseudo sensor position and direction. The intrinsic parameters of all pseudo sensors are

chosen with focal length of 0.0287, pseudo image width and height of 0.01, and number of pixel in width and height are 128. The sampled points were then regarded as a pseudo LiDAR ray fired from corresponding pseudo sensor, terminating at the 3D point. Detailed operations deploying projective upsampling are included in Appendix A.

All upsampled points, along with their corresponding pseudo sensor position, were combined to create a training dataset, which will be the ${}^{\pi}P$ and π in equation 3.1.

4.4 Model training

The PyTorch library was used for model training. A 9:1 split for training and validation dataset was used. The built-in PyTorch loss functions BCELoss(), KLDivLoss() and L1Loss() was used for implementing loss calculation for equation 3.10, equation 3.14 and the distance loss.

During training, truncated inverse sigmoid volume sampling was deployed with near plane chosen to be 0.05, the far plane chosen to be 3, number of samples per ray is 100, and σ_s decay to [1, 0.5, 0.2, 0.1] at [2, 4, 8, 16] epochs. All models were trained for 16 epoch. For computing target cumulative transmittance and termination distribution, σ_t was chosen to be 0.02, which is a relatively small value to give a sharp drop in cumulative transmittance upon ray termination and to approximate the Dirac delta function for target termination distribution.

A learning rate of 5e-6 was used, with a learning rate decay ratio of 0.5 at [2, 4, 8, 16] epochs. The positional encoding dimension L was chosen to be 15, which is the same as in NeRF in the Wild [19]. Training took approximately 6 hours for 16 epoch on a single Nvidia RTX 6000 GPU.

Implementation details for computing cumulative transmittance, termination distribution and truncated inverse sigmoid volume sampling are detailed in appendix B.

4.5 Model Rendering

The space within a chosen boundary l, centered at [0,0,0] was rendered with number of bins b. Formally, for $[x, y, z] \in \mathbb{R}^3$ where:

$$\begin{aligned} -l \leq x \leq l, \\ -l \leq y \leq l, \\ -l \leq z \leq l \end{aligned} \tag{4.1}$$

Where l is the boundary length, the Marching Cube Algorithm was deployed with cube size equal to $\frac{l}{b}$. For all models in the result, $l = 0.5$, $l = 0.7$ and $l = 0.6$ was used for rendering model trained with round plant dataset, box plant dataset, and building dataset respectively. Density threshold of 5 was used as default unless specified otherwise, $b = 64$ was used for all models.

4.6 Qualitative Results

The hyperparameters listed in table 4.2 were proposed with the listed values used as default. Through grid searching, qualitative evaluation such as evaluating number of floaters and completeness of reconstruction of the ground, the proposed loss weights of $\lambda_T = 1$, $\lambda_h = 1$ and $\lambda_d = 8$ was chosen to focus on elimination of dynamic objects, while balancing the reconstruction quality.

Variable Name	Symbol	Proposed Value
Batch Size	-	64
Hidden Dimension	d	512
Sampling Variance	σ_s	Update to 1, 0.5, 0.3, 0.1 at 1,2,4,8 epoch
Target Variance	σ_t	0.02
Cumulative Transmittance Loss weight	λ_T	1
Termination Distribution Loss weight	λ_h	1
Distance Loss weight	λ_d	8
Positional Encoding Dimension	L	15
Number of Pseudo Sensor	N	64
Number of Samples per Ray	-	100

Table 4.2: Proposed hyperparameters

Figure 4.5 shows the registered point cloud for model training. Figures 4.6 to 4.10 are visualisations of models trained on each dataset with selected hyperparameters varied. For all results in figures 4.6 to 4.10, the top row shows the reconstruction result of box plant dataset, the middle row shows the reconstruction result of round plant dataset, and the bottom row shows the reconstruction result of building dataset.

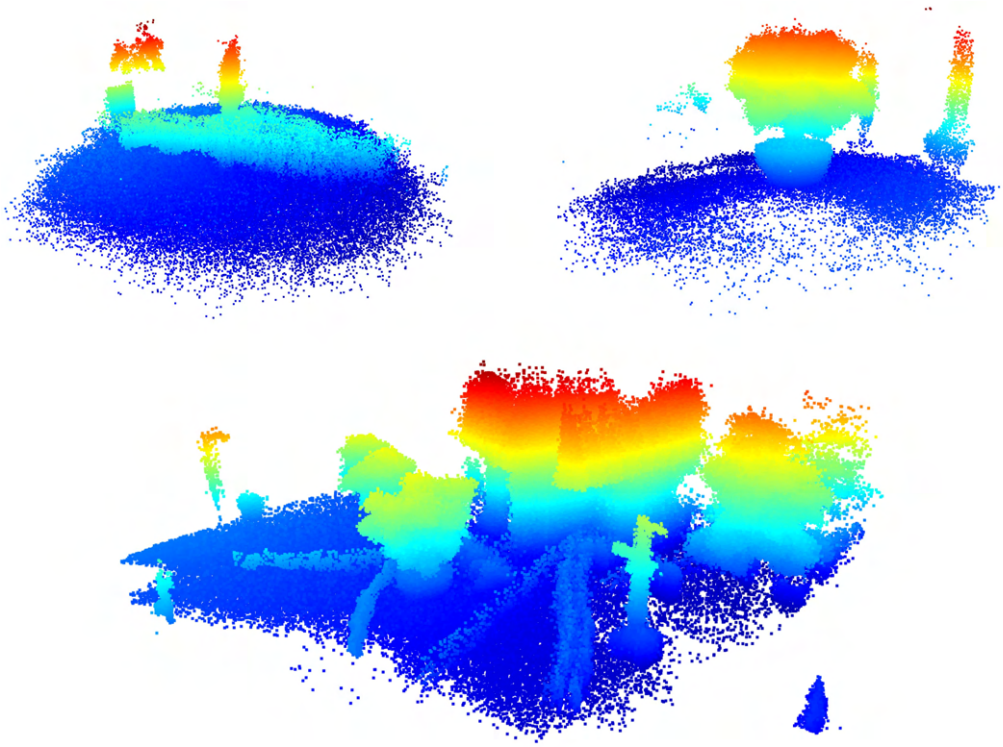

Figure 4.5: Point cloud of 500,000 randomly sampled points in box plant dataset (left), building dataset (middle) and round plant dataset (right).

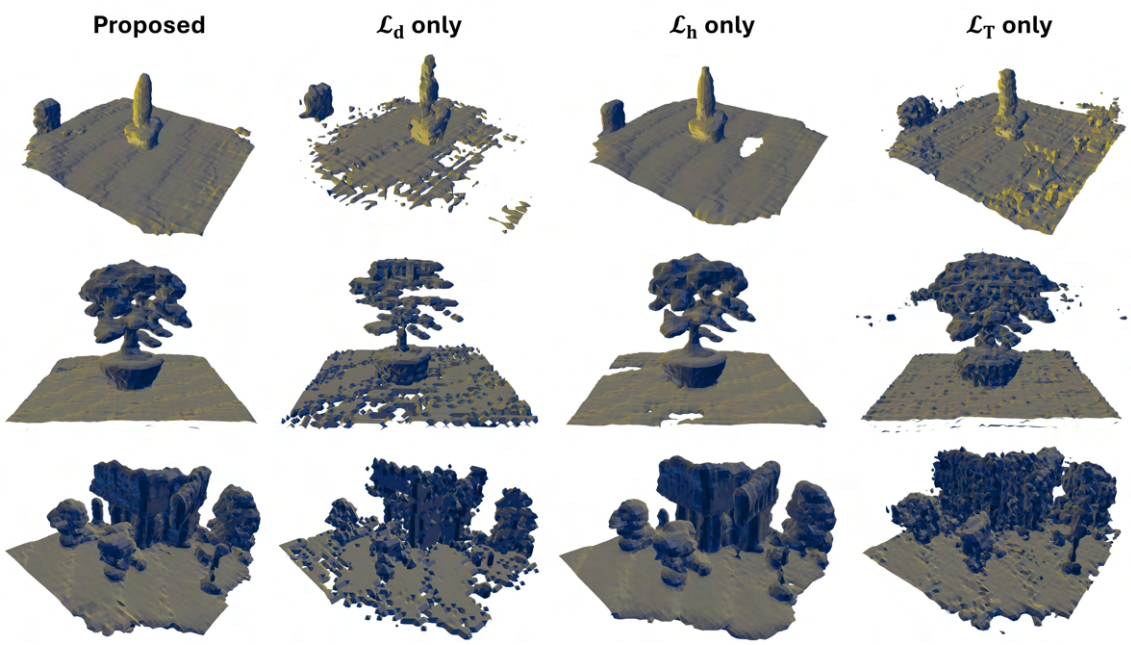

Figure 4.6: Reconstruction result of the proposed method, compared to using only $\mathcal{L}_d$, only $\mathcal{L}_h$ and only $\mathcal{L}_T$.

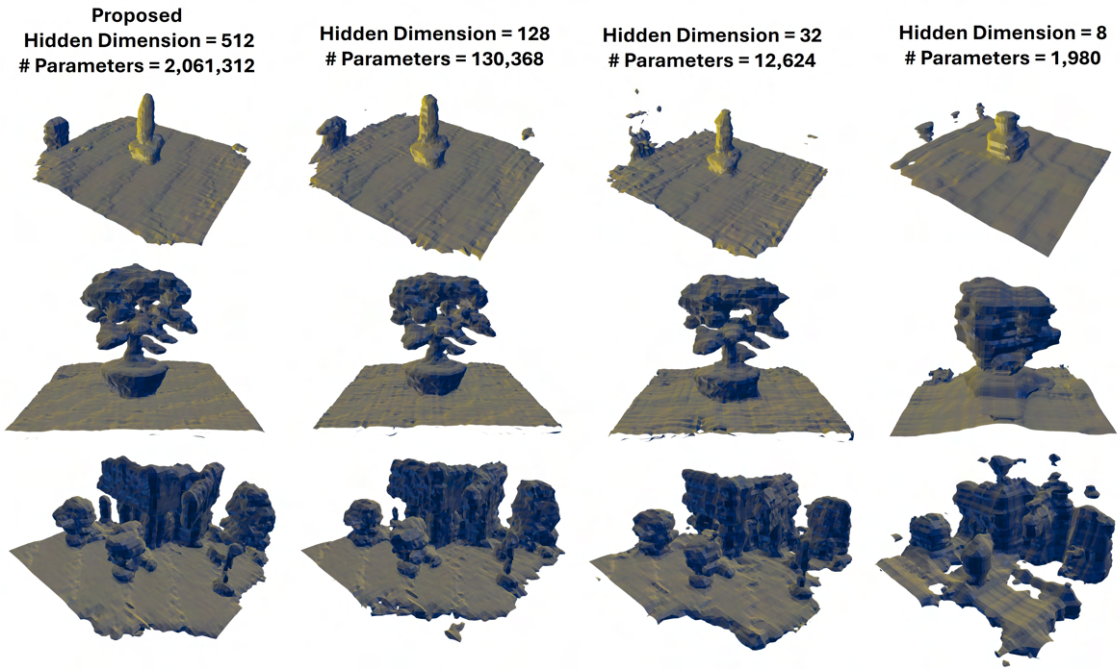

Figure 4.7: Reconstruction result with different hidden dimensions, rendering threshold for box plant model with hidden dimension of 8 is rendered with threshold = 1 as it fails to visualize with threshold = 5.

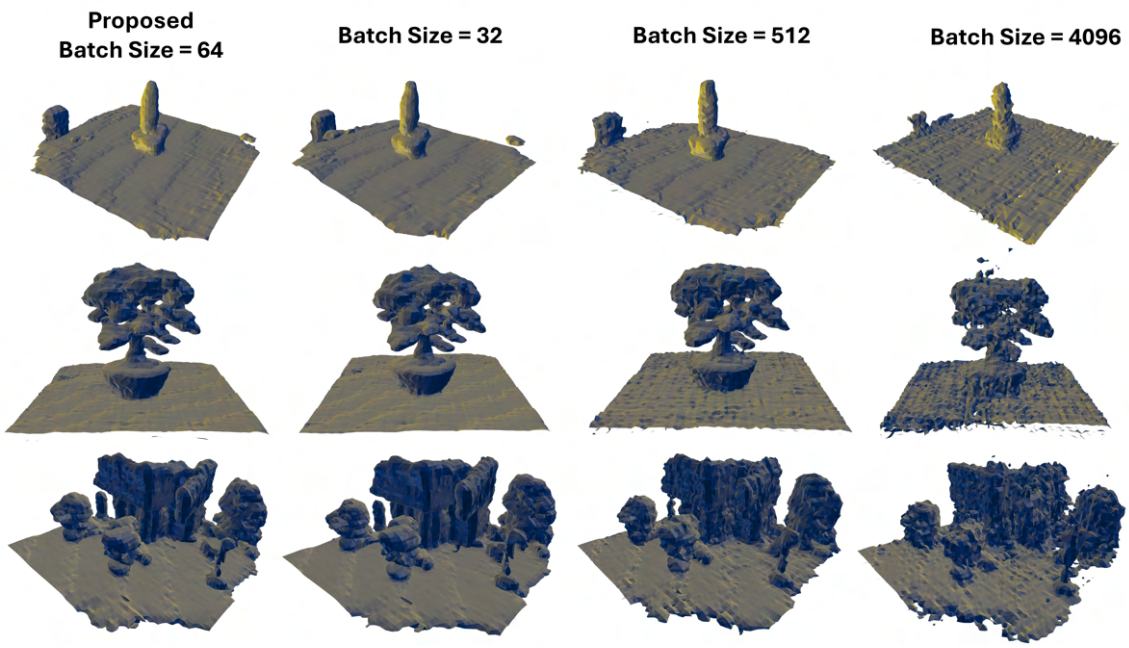

Figure 4.8: Reconstruction result with different batch sizes.

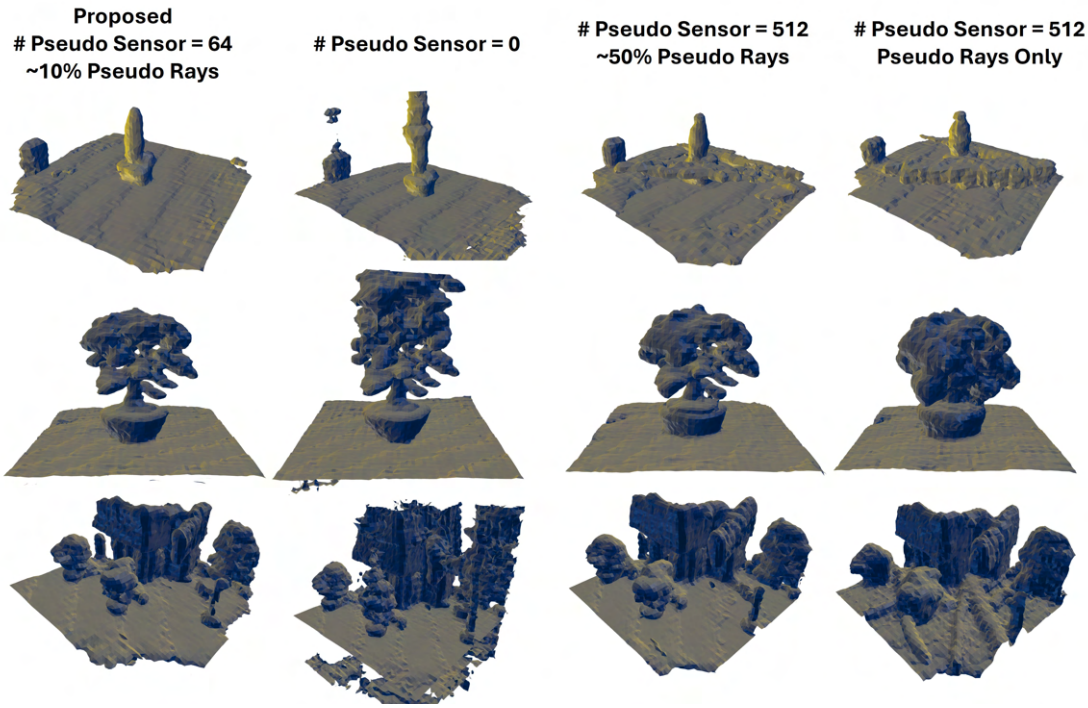

Figure 4.9: Reconstruction result with different number of pseudo sensors.

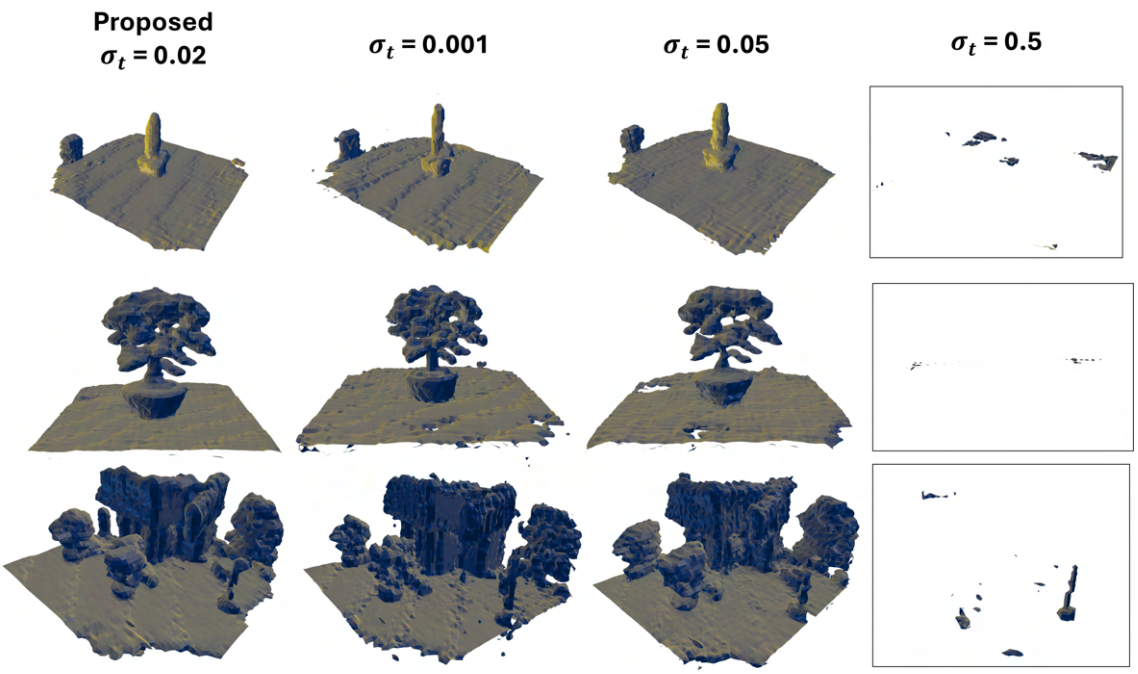

Figure 4.10: Reconstruction result with different σ_t.

4.7 Quantitative Results

This section provides details of quantitative evaluation by comparing the dimensions of the actual objects with the reconstructed mesh. The edge length of the plant container in box plant dataset was measured to be 1.02m, and the diameter of the upper edge of the plant container in round plant dataset was measured to be 2.12m. Figure 4.11 shows the setup for measuring the physical dimensions of the plant container in box plant dataset.

Figure 4.11: Setup for measuring edge length of the plant container in box plant dataset.

The round plant dataset and box plant dataset models were then rendered with boundary $l = 0.5$ and number of bin $b = 128$. Vertices in the rendered mesh was extracted and visualised with the Livox Viewer software [42]. By setting the view to align with x axis, a small batch of vertices was selected around region of interest, as shown in figure 4.12 and 4.13. Finally, the maximum x-axis difference within each batch of selected points was computed to estimate the corresponding dimensions.

Table 4.3 and 4.4 detailed the computed dimensions and their percentage error (corrected to 3 sig. fig.) to the actual dimension. Results for $\sigma_t = 0.5$ is not included due to insufficient number of vertices and result for using pseudo rays only in box plant models is not included due to severe occlusion of dynamic objects.

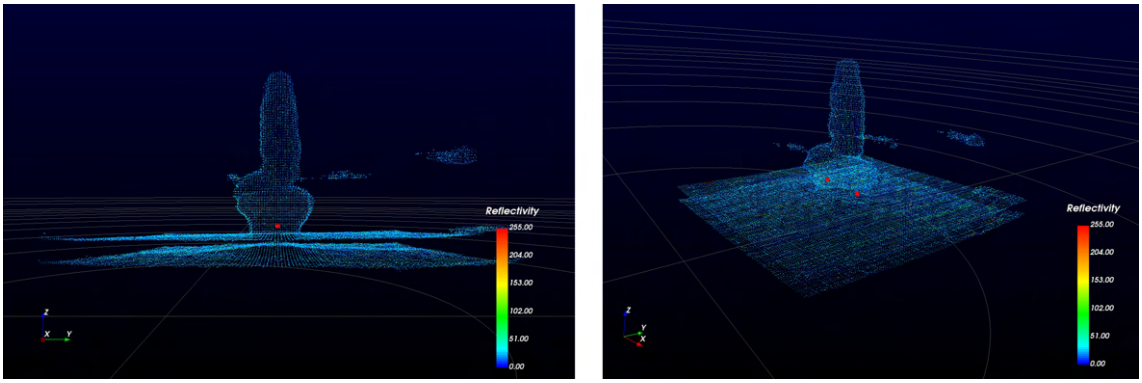

Figure 4.12: Selecting points with axis-aligned view angle for box plant mesh (left), selected points (red dots) for edge length calculation (right).

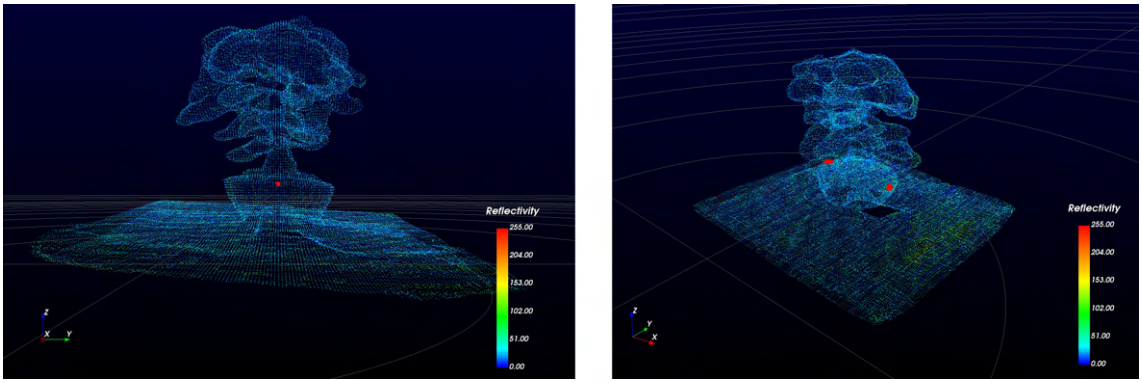

Figure 4.13: Selecting points with axis-aligned view angle for round plant mesh (left), selected points (red dots) for diameter calculation (right).

Box Plant Dataset Models	**Edge Length in Reconstructed [m]**	**% Error**
Proposed	1.41	38.2
$\mathcal{L}_d$ only	1.05	2.94
$\mathcal{L}_h$ only	1.59	55.9
$\mathcal{L}_T$ only	1.31	28.4
hidden dim = 128	1.44	41.1
hidden dim = 32	1.56	52.9
hidden dim = 8	2.03	99.0
batch size = 32	1.46	43.1
batch size = 512	1.49	46.1
batch size = 4096	1.53	50
# Pseudo Sensor = 0	1.40	37.3
# Pseudo Sensor = 512	1.58	54.9
# Pseudo Sensor = 512, Pseudo rays only	-	-
$\sigma_t = 0.001$	1.29	26.5
$\sigma_t = 0.05$	1.40	37.3
$\sigma_t = 0.5$	-	-

Table 4.3: Reconstruction dimension for box plant container edge length, ground truth length = 1.02m.

Round Plant Dataset Models	**Diameter in Reconstructed [m]**	**% Error**
Proposed	2.81	32.5
$\mathcal{L}_d$ only	2.31	9.43
$\mathcal{L}_h$ only	2.73	28.8
$\mathcal{L}_T$ only	2.94	38.7
hidden dim = 128	2.75	29.7
hidden dim = 32	2.63	24.1
hidden dim = 8	3.49	64.6
batch size = 32	2.75	29.7
batch size = 512	2.72	28.3
batch size = 4096	2.56	20.8
# Pseudo Sensor = 0	2.75	29.7
# Pseudo Sensor = 512	2.67	25.9
# Pseudo Sensor = 512, Pseudo rays only	2.96	39.6
$\sigma_t = 0.001$	2.38	12.2
$\sigma_t = 0.05$	2.86	34.9
$\sigma_t = 0.5$	-	-

Table 4.4: Reconstruction dimension for box plant container upper edge diameter, ground truth diameter = 2.12m.

5 Discussion

Contents

5.1 Quantitative Evaluation 39
5.2 Learning Curve 40
5.3 Loss Functions 41
5.4 Omission of dynamic object 41
5.5 Varying Batch Size 43
5.6 The Inertia of NeRF Models 45
5.7 Pseudo Sensors 47
5.8 Hidden Diemnsion 48
5.9 Sampling Variance 48
5.10 Registration Quality 49

This chapter will discuss the observations in the results from chapter 4, and dedicate further discussion to notable observations. Overall, the proposed method achieved a reasonable 3D reconstruction on all scenes. A qualitative comparison was primarily conducted due to the lack of an openly available dataset with ground truth 3D model for Livox LiDAR data. A more extensive quantitative comparison will be included as the future work for this project.

5.1 Quantitative Evaluation

As shown in table 4.3 and 4.4, the error in dimensions shows consistency between box plant and round plant datasets. All reconstructed dimensions overestimates the ground truth dimensions, this suggests that the overall pipeline tends to blurs the reconstruction result. Blurring could potentially be due to poor reconstruction quality of point clouds.

Using $\mathcal{L}_d$ only and choosing smaller σ_t is effective in improving the sharpness of reconstruction. Where using $\mathcal{L}_d$ only have achieved the lowest error. This is consistent to the qualitative observation of improved sharpness in reconstruction as shown in figure 4.6 and 4.10.

The quantitative evaluation approach conducted is prone to error in selecting points for measuring reconstructed dimension. The approach also does not consider factors such as noises in blind spots, floaters and holes in ground reconstruction. Thus, a qualitative approach was undertaken for selecting the proposed hyperparameters. A future work of this project will be to explore a more robust quantitative evaluation approach.

5.2 Learning Curve

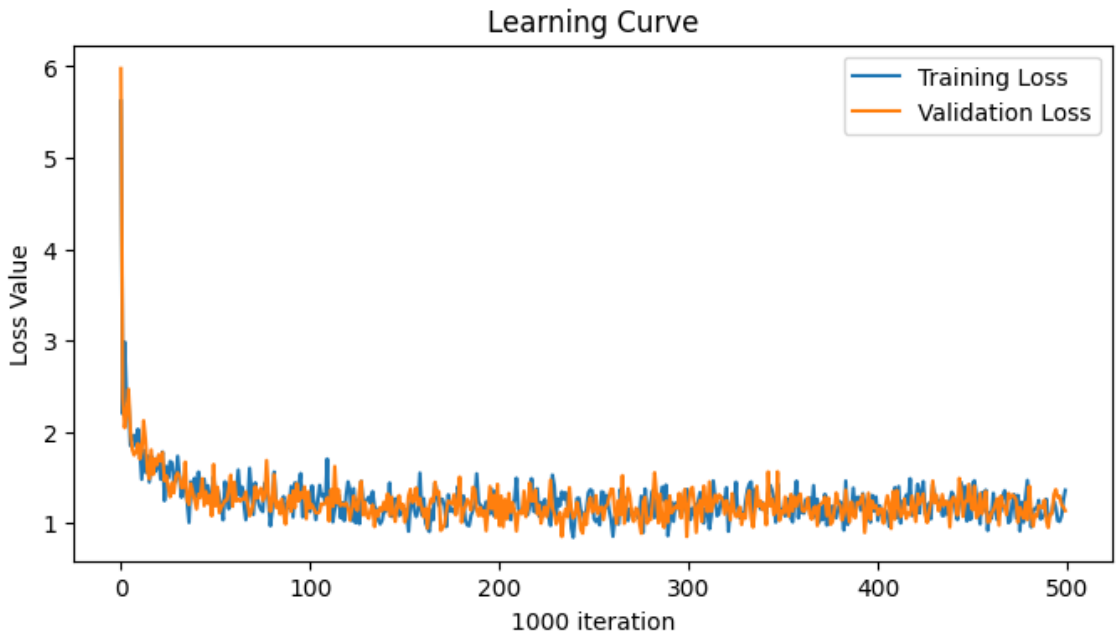

Figure 5.1: Plot of loss vs iteration of model trained on round plant dataset with default settings, first 500,000 iteration.

Figure 5.1 shows the learning curve of model train on the proposed configuration with round plant dataset. The losses converged within approximately 100,000 iteration. This is because the data is highly repetitive from similar sensor pose and the highly overlapping field of view between consecutive frames. The dataset is also exhaustive where the sampled positions will exhausts the bounded scene space. Thus, a small epoch can be equivalent to multiple passes of data points in the space and a fast convergence across iteration was achieved.

5.3 Loss Functions

To gain an idea on how loss functions affect reconstruction results, the anatomy of each loss function shall first be considered. $\mathcal{L}_d$ creates no obligation for the NeRF model to reproduce the ground truth density, but demands an accurate expected termination distance of rays. As the majority of the rays lands on the ground truth surface, $\mathcal{L}_d$ is thus capable for eliminating floaters and dynamic object but could cause sparse reconstruction due to its lack of requirement on a sharp density.

With reference from discussion in SHINE-Mapping, $\mathcal{L}_T$ is sensitive around surface and is strict on separating free space and occluded space [10], thus encourages a polarized density prediction and reconstruction details.

Finally, $\mathcal{L}_h$ focuses on the overall termination distribution of each ray, which relaxes the reconstruction of fine details and produce a smooth reconstruction and is effective in eliminating the reconstruction of dynamic objects.

Reconstruction result using different losses in figure 4.6 empirically shows that using only $\mathcal{L}_d$ results in a sharp but sparse reconstruction, using only $\mathcal{L}_h$ results in a smooth reconstruction but lead to holes on ground reconstruction, and using only $\mathcal{L}_T$ gives a generally complete reconstruction, but with noises and floaters. The combination of the loss functions is thus a trade-off problem between reconstruction of details, sparsity and elimination of dynamic objects. As discussed in Section 5.1, a qualitative approach was undertaken to select the proposed loss combination.

Noting that all losses encourages density to be concentrated around the termination of each ray, they should have ideally achieve similar results. While attempt have been made to explain different reconstruction results due to different losses, there is room for further experiment on how different loss functions dictate the reconstruction quality.

5.4 Omission of dynamic object

Refer to the reconstruction result in proposed setting in figure 4.6, dynamic object are omitted in the proposed reconstruction, even in heavily occluded scenario such as the box plant dataset. This project propose to explain the omission of dynamic objects in reconstruction with the underlying voting mechanism of machine learning models, where the NeRF model decideds to satisfy majority of the sampled positions and omit the minority in training. Figure 5.3 illustrates how dynamic

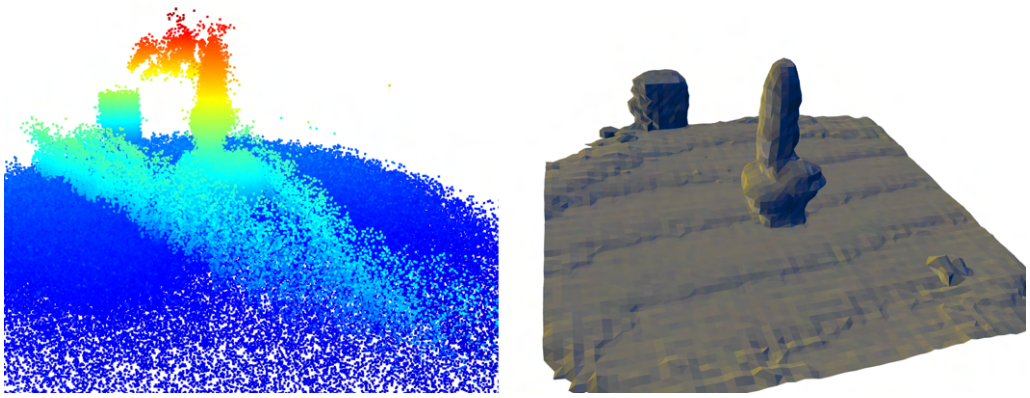

Figure 5.2: point cloud of box plant dataset with ghost trail (left), reconstruction result with dynamic object omitted (right).

objects only occupy a position in small number of frames by nature. Thus, only a small number of rays that passes through the position will be occluded, while for majority of the rays that passes through that volume, the position will remain free space. As shown in figure 5.4, majority of the samples will "vote" for free space, while minority of samples will "vote" for dense space in regions temporarily occupied by dynamic objects.

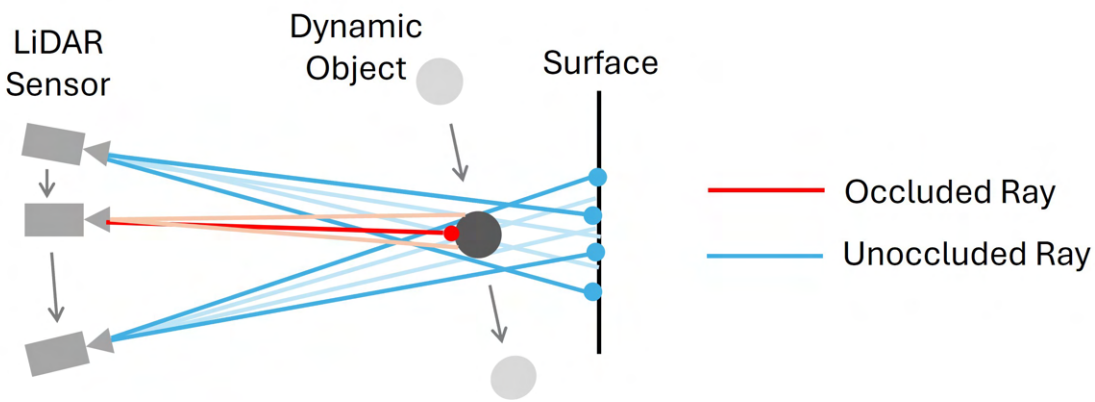

Figure 5.3: Illustration of dynamic object occluding minority of rays at certain position.

If σ_s is chosen to be too small, dynamic objects will be reconstructed, as show in figure 5.5. This is because the sampled positions are too concentrated in termination region around each rays. Thus sampled positions in rays land on ground truth surface has no interference with the samples from occluded rays. With no interference, the voting mechanism will fail and the dynamic object will fail to be omitted.

On the other hand, dynamic objects will also be reconstructed if there are too many pseudo sensors. This is because 3D points from dynamic objects will dominate the other points in projective upsampling procedure, as discussed in Section 3.2, and dilute the ground truth rays. These results

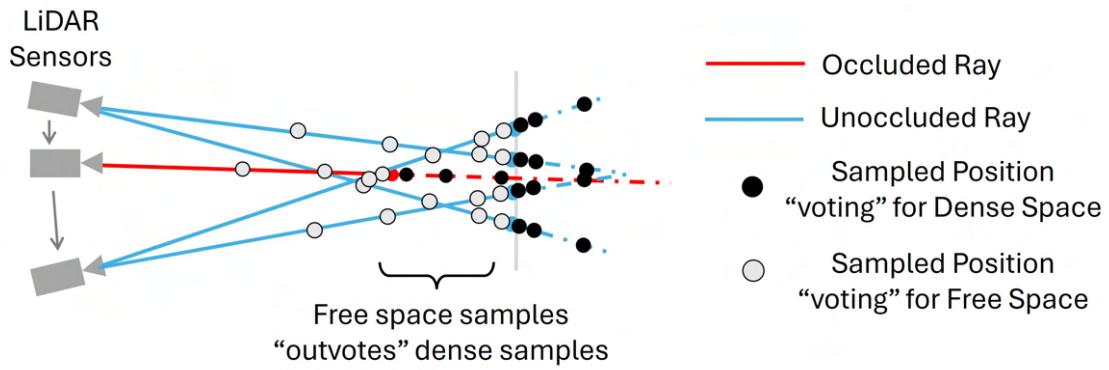

Figure 5.4: Illustration of voting mechanism to eliminate dynamic objects in reconstruction.

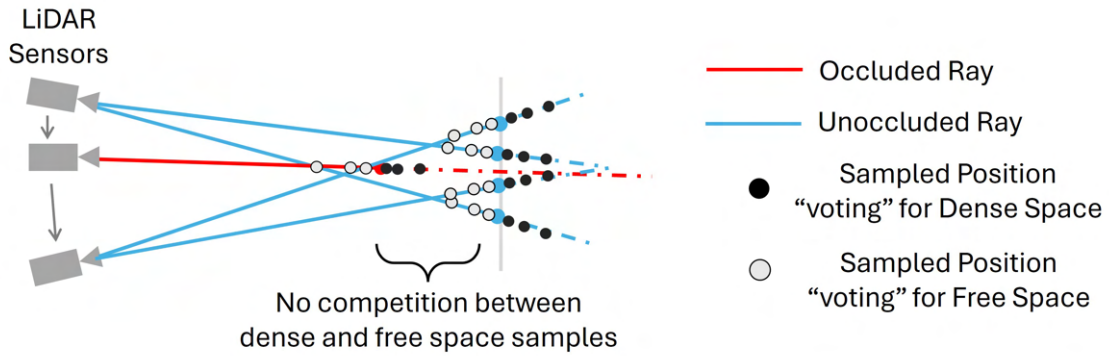

Figure 5.5: Illustration of failure of voting mechanism due to small σ_s.

support the proposed explanation of dynamic object omission by voting mechanism of neural networks.

5.5 Varying Batch Size

Reconstruction result with different batch size in figure 4.8 shows that the mesh produced is smoother for a small batch size, for a large batch size, the reconstructed mesh is rougher. Figure 5.6 shows the first 50,000 iteration of the models trained with different batch size, with round plant dataset for 16 epochs. Indicating that the losses have converges for model trained with batch size of 512 and 4096, while model trained with batch size 32 has significantly more iterations to be trained. This eliminates the reasoning of the rough reconstruction is due to insufficient training iterations. This project propose to explain the rough reconstruction from larger batch size by the registration error in point clouds. As shown in figure 5.7, when the batch size is large, more points are simultaneously required to be fitted and inconsistency in registration between frames causes the model to simultaneously satisfy poorly registered points. When the batch size is small,

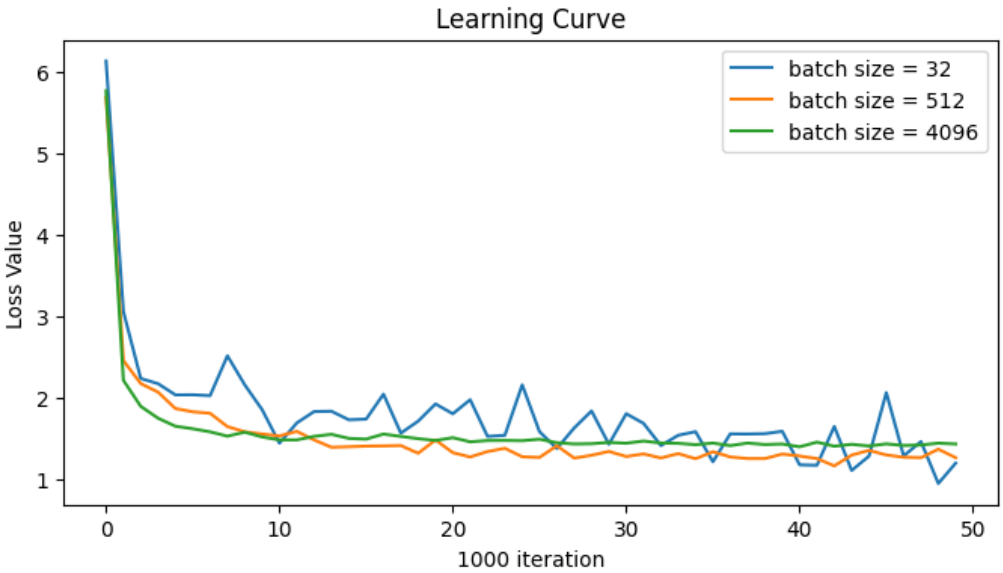

Figure 5.6: Learning curve of models trained with round plant dataset with different batch size.

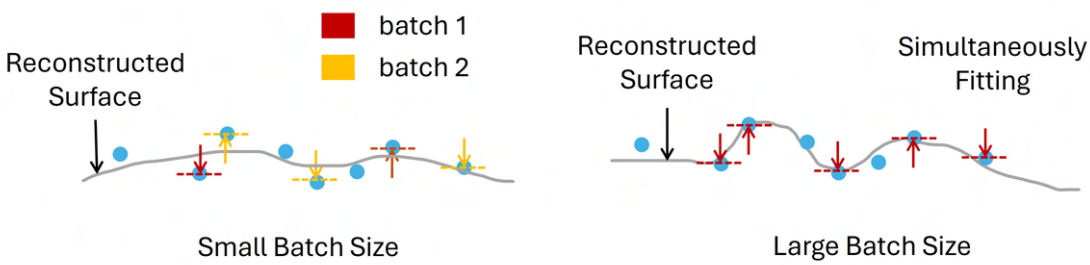

Figure 5.7: Illustration of small batch size relaxes reconstruction requirement (left) and large batch size require more points to be simultaneous fitted (right).

such condition is relaxed. The reconstruction will then be averaged across all points and produce a smoother surface.

The smoother reconstruction result from a small batch size suggests a connection with the findings in [43] and [44]. [43] empirically showed that a smaller batch size of 2 to 32 can lead to a more generalized test accuracy of convolutional neural network on image classification tasks, and explained it with batch size's affect on gradients for neural network training. The observation in this project suggests a different angle of explaining the findings in [43] geometrically. Where one can consider the reconstructed surface as an decision boundary between dense and free space, and a more generalized decision boundary is equivalent to a smooth reconstruction result.

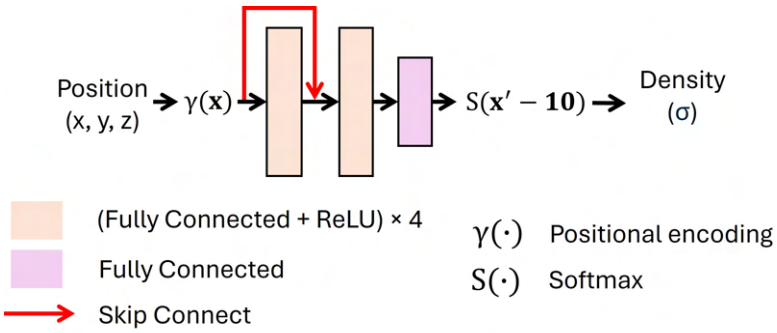

Figure 5.8: Architecture of the alternate model.

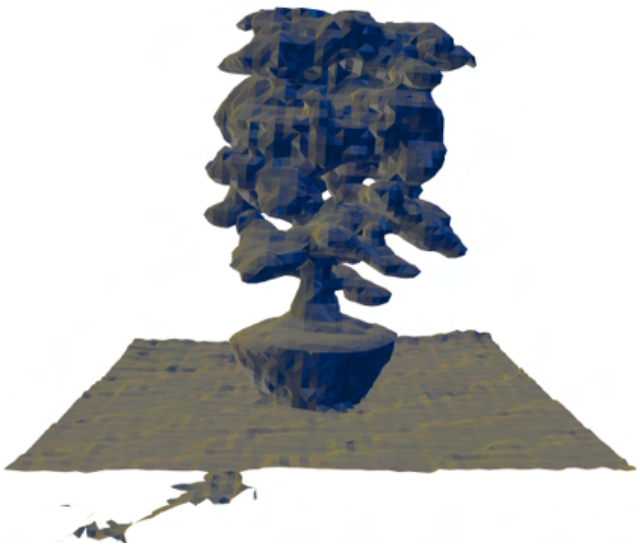

Figure 5.9: Reconstruction result of augmented model trained on raw round plant dataset and rendered with density threshold of 15.

5.6 The Inertia of NeRF Models

Consider the noise in blind spots shown in figure 3.2, it raises the question: Is it possible to initialize the NeRF model to only predict free space ($F(\mathbf{p}) \approx 0, \mathbf{p} \in \mathbb{R}^3$), and train the model without using any pseudo sensor, such that only the scanned region will be reconstructed, while the unscanned region will remain free space?

To investigate, an augmented NeRF model was designed to initialize at predicting free space. This is done by deducting the scalar value by 10 before applying it to the soft plus output function, with rest of the model architecture same as proposed. The model architecture is illustrated in figure 5.8. Upon initialisation and without any training, the augmented model predicted an average density of 8.8×10^{-5}, grid sampled across the cube centered at [0,0,0] with edge length 2 and

sampling bins of 64 in each dimension. The augmented model was then trained with the raw round plant dataset using default settings, the reconstruction result is shown in figure 5.9, rendered with density threshold 15 and the other rendering setting are same as Section 3.6.

It can be seen that the noises in blind spots still exist. This provide evidence that the noises is due to inertia of neural networks to prioritise using low-frequency features despite the availability of high-frequency features from positional encoding. And the reconstruction of the plant lead to jointly increase of density in blind spots.

The problem of NeRF model's inertia can also be observed in reconstruction of fine structures. For small free spaces such as the strips or nearby objects, they are connected to one dense region in reconstruction. Figure 5.10 shows that free spaces between an array of bikes are joint with dense region, which is a consequence of inertia of NeRF models.

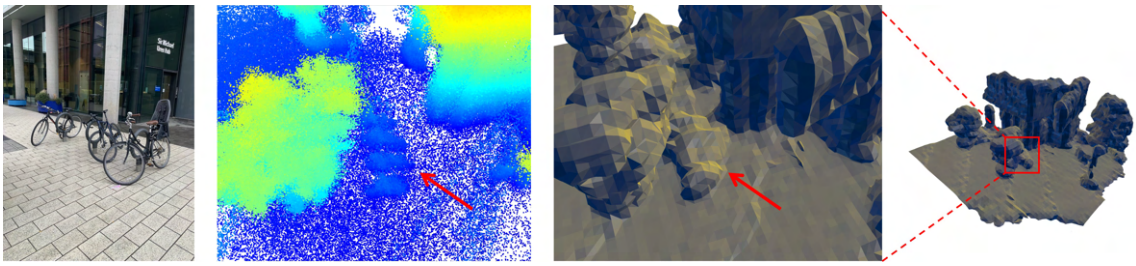

Figure 5.10: Image of an array of bikes (left), point cloud of the array of bikes (middle) and reconstruction result of the bikes (right).

Researches have addressed similar issue as *spectral bias* of neural networks, where low frequency features are first learnt by neural networks[45]. While spectral bias can be mitigated by mapping low dimensional features to Fourier features [38], this project has shown that the sole availability of high frequency features (from positional encoding) is insufficient in eliminating the inertia of neural network to affect unknown space.

While this further justifies the need for pseudo sensors to mitigate noises in blind spots, the method of pseudo sensor create extra computational costs and might not be suitable for real time application. The designing of neural network features such that unknown space remain unaffected or a neural network that is event triggered would be a direction for future work.

5.7 Pseudo Sensors

Observing figure 4.9 of reconstruction result with different number of pseudo sensors, applying pseudo sensors are clearly effective in eliminating noise around blind spots in reconstruction, but also causes blurring in the reconstruction. Note that with 50% pseudo rays have led to the reconstruction of dynamic objects in the heavily occluded box plant dataset.

To explain this, consider a point cloud containing registration error or ghost points, such as the box plant dataset in figure 4.5. The outer points, due to registration error or points from dynamic objects, will dominate the projective upsampling procedure, as illustrate in figure 5.11. With a large number of pseudo sensors, this will blur the reconstruction result or lead to the reconstruction of dynamic objects. Thus, a combination of the raw data and projective upsampled data is needed to balance noise mitigation and preservation of details in the original data, and the number of pseudo sensor plays an important role for optimal reconstruction quality.

The proposed sensor position is not universal. It would be limited in, for example, indoor settings, where the scene is bounded by walls and projective upsampling will be dominated by points on the walls. A more sophisticated design for pseudo sensor positioning that satisfying the need of different scenes should be explored in future work.

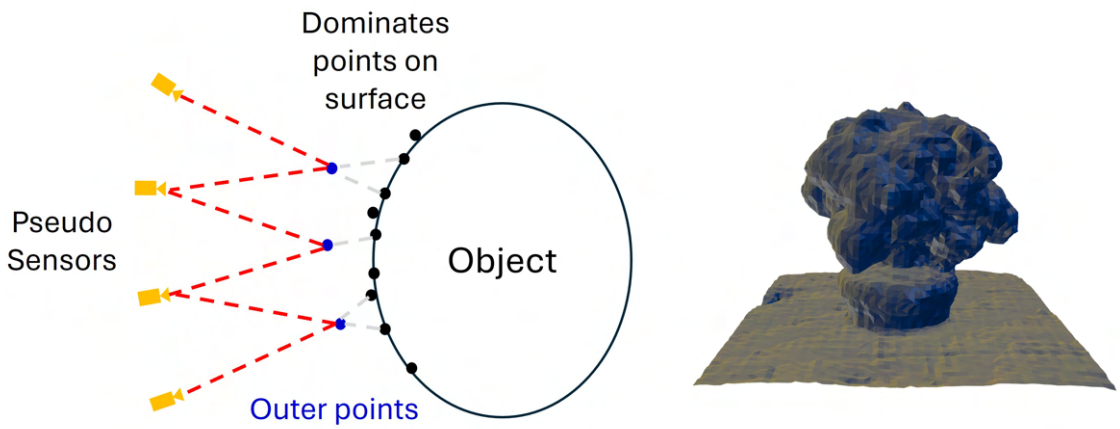

Figure 5.11: Illustration of outer points dominating points on surface in projective upsampling (left) reconstruction result using only projective upsampled data, causing lose of reconstruction details (right).

5.8 Hidden Diemnsion

From figure 4.7, the reconstruction result is robust against parameter reduction of the underlying model. A reasonable result can be seen with hidden dimension of 32, which uses 12,624 parameters, suggesting that the resultant model is a highly compact representation of the object.

5.9 Sampling Variance

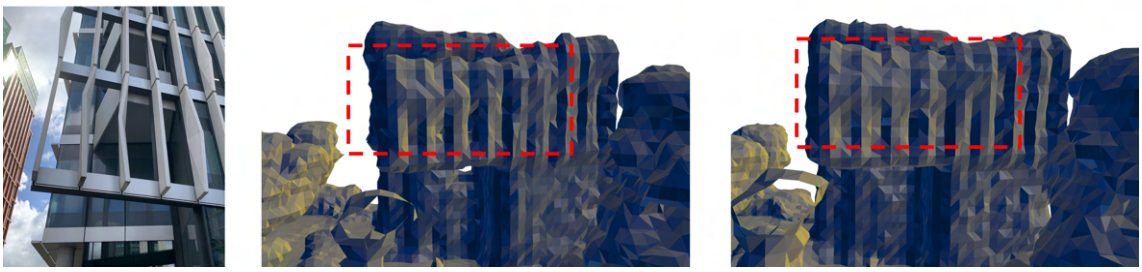

Figure 5.12: Image of the fine structure in building dataset (left) Reconstruction result of the fine structure with proposed σ_s (middle), and reconstruction result of the fine structure with constant $\sigma_s = 3$ (right).

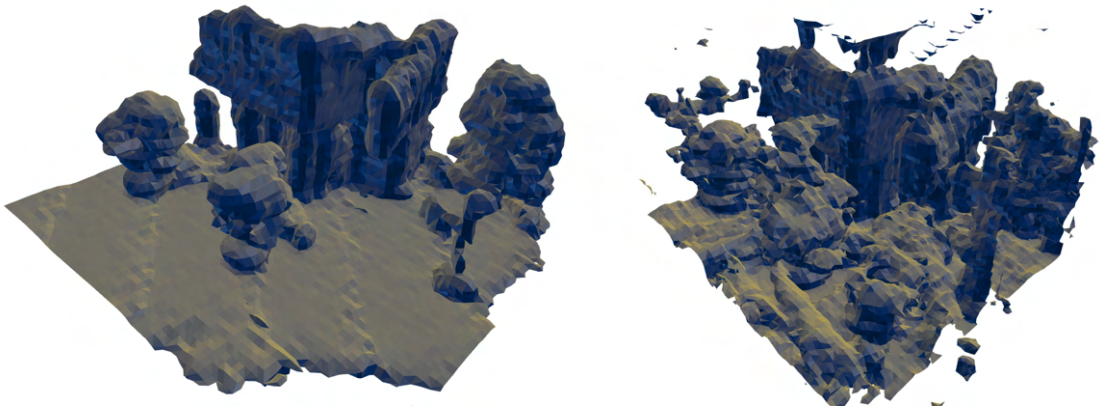

Figure 5.13: Reconstruction result with proposed σ_s (left), reconstruction result with $\sigma_s = 0.01$ (right).

In the proposed setting, σ_s was reduced gradually. Figure 5.12 compares the reconstruction of using the proposed σ_s arrangement against using $\sigma_s = 3$ without any decay. It can be observed decaying σ_s is effective in achieving a sharper reconstruction. Figure 5.13 compares the reconstruction with proposed setting against $\sigma_s = 0.01$ without any decay. If σ_s is chosen to be too small, the reconstruction result will be noisy with appearance of floaters and ghost trail from dynamic objects, which can be explained by discussion in Section 5.3. Overall, the proposed method trun-

cated inverse sigmoid sampling serves as a convenient approach approach for controlling focus of sampled positions around surfaces and has been shown to be effective in improving reconstruction details.

5.10 Registration Quality

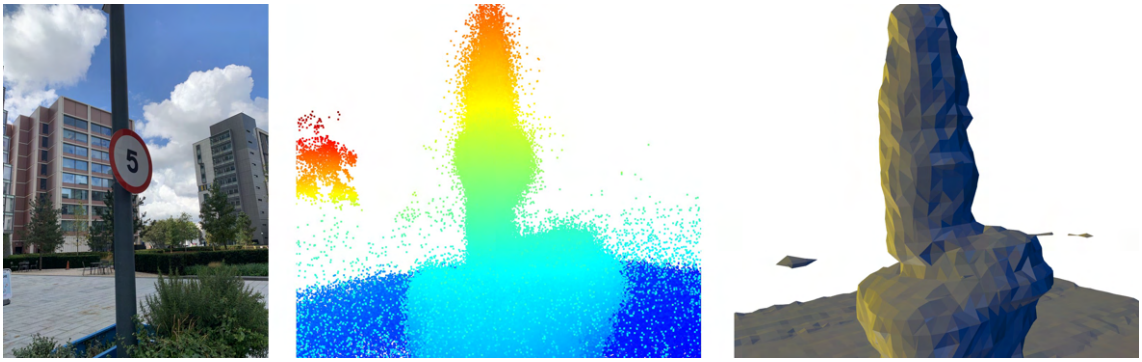

Figure 5.14: Close up image of the sign in box plant dataset (left), registered point cloud of the sign from 500,000 randomly sampled points (middle) and reconstruction result of the sign (right).

The point cloud registration quality is the major source of error in the pipeline of this project. Shown in figure 5.14 is the reconstruction result of the sign in box plant dataset. Due to poor registration quality, the point cloud blurs the sign, and consequently, the reconstruction is also smoothed out. This suggest room for future work in refining sensor poses. In particular, NeRF–improved the original NeRF pipeline by jointly optimizing parameters in the NeRF model and camera poses[46]. Similar work can potentially be applied to the pipeline proposed in this project.

Conclusions

An overall promising reconstruction result was achieved from the proposed pipeline in this project. The proposed method of projective upsampling was shown to be effective in removing noise in the reconstruction by producing pseudo rays to cover blind spots in the raw data. Truncated inverse sigmoid volume sampling provided a convenient method to control volume sampling distribution and was shown to improve reconstruction detail. The designed loss function was shown to achieve promising reconstruction result, which was experimented on three datasets.

It is particularly interesting to observe the connection between reconstruction results and fundamental theories of neural networks. Including the effect of different loss functions, varying batch size and the cause of noises in blind spots. While previous research has shown how batch size affect the generalization of convolutional neural network and the nature spectral bias of neural networks, this project could provide a different angle of understanding the previous findings.

5.11 Future Work

With recent advancements of research in RGB NeRFs, there is potential to transfer work on RGB NeRF to extend this project. For example the discussed NeRF– proposal of jointly optimizing camera poses and reconstruction result [46], or the F2-NeRF proposal of adopting space wrapping to improve representation efficiency of the NeRF model [47].

As addressed in Section 5.2, a qualitative approach was used for choosing loss function weights due to the lack of availability of ground truth models. Thus, another direction for future work would be to conduct quantitative evaluation, such as constructing a CAD model of objects and compare the chamfer distance of reconstructed mesh with the ground truth mesh. Comparison can also be made with different settings such as indoor scenes, with different LiDAR sensors and against different 3D reconstruction algorithms. Improvement to the pipeline can also be made by designing a more universal pseudo sensor arrangement.

Finally, Section 5.4 and 5.5 addressed how the batch size affects reconstruction results and the inertia of NeRF models. Through discussion, they shows connection to the more fundamental

behaviour of neural networks, which are potential topics for deeper exploration.

A

Implementation Detail for Projective Upsampling

A

In practice, the raw training data are expressed as a size [m, 7] numpy array with m being the total number of 3D points. Details of the array are described in table A.1

Columns	Description
Column 0, 1, 2	x, y, z position of the sensor
Column 3, 4, 5	x, y, z, direction of LiDAR ray
Column 6	Distance from the sensor position where the ray terminates

Table A.1: Description of the training dataset

The pseudo code for projective upsampling is described in algorithm 2.

Several functions used in algorithm 2 shall be explained in detail. Using columns described in table A.1, sensor position can be denote as $\mathbf{o} = [\mathbf{o}_x, \mathbf{o}_y, \mathbf{o}_z]$, which is a size [m, 3] matrix, LiDAR ray direction as $\mathbf{d} = [\mathbf{d}_x, \mathbf{d}_y, \mathbf{d}_z]$ a size [m, 3] matrix, and termination distance as $\mathbf{l}$, a length m vector. The point cloud is given to be:

$$P_{raw} = [\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z] = \mathbf{o} + \mathbf{l}^T \cdot \mathbf{d} \tag{A.1}$$

Which satisfy the GetPointCloud() function in line 3 of algorithm 2. Using equation 3.4 and 3.5 the pseudo sensor position and direction are given, satisfying the Fibb() function in line 6 of algorithm 2.

Using the chosen focal length $f = 0.0287$ and image width and height of $w = h = 0.01$, the

A

Algorithm 2 Projective Upsampling

Input:
$data_{raw}$: Raw data
m: Number of points in P_{raw}
N: Number of pseudo sensor
K: Sensor intrinsic matrix
Output: Projective upsampled point cloud

```
i ← 0                                                          ▷ Iteration Count
j ← 0
P_raw ← GetPointCloud(data_raw)
data_sampled ← []                                              ▷ Initialize 0 ×7 empty array
for i < N do
    org, dir ← FibLattice(i, N)
    M ← GetExtrinsic(org, dir)
    ^π P ← Transform(P_raw, M)                                 ▷ Transform raw point cloud to sensor frame
    P_pseudo ← Projection(^π P, K)                             ▷ Project point cloud to pseudo image
    P_pseudo[:, 2] ←^π P[:, 2]                                 ▷ Preserve the depth
    P_sampled ← []                                             ▷ Initialize 0 ×3 empty array
    P_pseudo ← Sort(P_pseudo, key = 2, order = increasing)     ▷ Sort according to depth
    P_pseudo ← StableSort(P_pseudo, key = 1)
    P_pseudo ← StableSort(P_pseudo, key = 0)
    for j < m do
        if j = 0 then    ▷ Sampling first point of every different pixel, knowing that depth is sorted increasingly
            P_sampled ← vStack(P_sampled, ^π P[j, :])
        else if P_pseudo[j, 0] ≠ P_pseudo[j − 1, 0] or P_pseudo[j, 1] ≠ P_pseudo[j − 1, 1] then
            P_sampled ← vStack(P_sampled, ^π P[j, :])
        end if
        j ← j + 1
    end for
    l ← GetLength(P_sampled)
    org_sampled ← repeat(org, l)                               ▷ Tile origin to number of sampled points
    dir_sampled ← getDirection(P_sampled, org_sampled)
    dis_sampled ← getDistance(P_sampled, org_sampled)
    temp ← hStack(org_sampled,dir_sampled, dis_sampled )
    data_sampled ← vStack(data_sampled, temp)
    i ← i + 1
end for
result ← vStack(data_sampled, data_raw)
return result
```

Algorithm 3 GetExtrinsic

Input:
org: Position of the sensor
dir: Direction of the sensor

Output: The camera extrinsic matrix

1: $world_up \leftarrow [0, 1, 0]$ ▷ Define the world up vector
2: $right \leftarrow dir \times world_up$ ▷ Right vector is cross product of sensror direction and world up vector
3: $down \leftarrow dir \times right$ ▷ Down vector is cross product of sensror direction and up vector
4: $R \leftarrow [right^T, donw^T, dir^T]$ ▷ Construct the rotation matrix
5: $t \leftarrow -R \cdot org$ ▷ Compute translation vector
6: $E \leftarrow I_{4\times4}$ ▷ Initialize extrinsic matrix as identity matrix
7: $E_{3\times3} \leftarrow R$
8: $E_{3,0:3} \leftarrow t$
9: **return** E

sensor intrinsic matrix K is constructed:

$$\mathrm{K} = \begin{bmatrix} \mathrm{f} & 0 & \frac{\mathrm{h}}{2} \\ 0 & \mathrm{f} & \frac{\mathrm{w}}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

Which is the required input for algorithm 2.

A

A

B

Implementation Detail for Model Training

B

Truncated Inverse Sigmoid Volume Sampling

In practice, t_{min} and t_{max} are bounded to avoid sampling at infinity due to rounding error after applying stratified samples to the inverse sigmoid function:

$$t_{min} = max(t_{min}, 0.0001)$$

$$t_{max} = min(t_{max}, 0.9999)$$

Computation of Predicted Termination Distribution and Predicted Cumulative Transmittance

In this project, the distance between consecutive samples δ_i is given to be:

$$\delta_i = u_{i+1} - u_i$$

Where i is index for samples on a ray and u_i are the sampled position. This is ambiguous with finite number of samples. In practice, for sampling N points on a ray, δ for the last sample will be:

$$\delta_N = u_N - u_{N-1} \tag{B.1}$$

Which is equivalent to defining $\delta_N = \delta_{N-1}$, this is a special case to handle the ambiguity.

To ensure the termination distribution of a ray to be valid, i.e. $\sum h_i = 1$, the weights in termination distribution of each ray is divided by their sum:

$$h_i \leftarrow \frac{h_i}{\sum_{i=1}^{N} h_i + \epsilon} \tag{B.2}$$

Where ϵ is chosen to be $= 10^{-}8$ to avoid singularity.

Bibliography

[1] N. Mehendale and S. Neoge, "Review on lidar technology," *SSRN Electronic Journal*, Jan. 2020. DOI: 10.2139/ssrn.3604309.

[2] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, *Depth-supervised nerf: Fewer views and faster training for free*, 2022. arXiv: 2107.02791 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2107.02791.

[3] P. Wei, L. Yan, H. Xie, *et al.*, "Lidenerf: Neural radiance field reconstruction with depth prior provided by lidar point cloud," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 208, pp. 296–307, 2024, ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2024.01.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271624000261.

[4] J. Deng, X. Chen, S. Xia, *et al.*, *Nerf-loam: Neural implicit representation for large-scale incremental lidar odometry and mapping*, 2023. arXiv: 2303.10709 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2303.10709.

[5] Livox, *Livox horizon user manual*, Accessed: 2024-08-09, 2024. [Online]. Available: https://www.livoxtech.com/3296f540ecf5458a8829e01cf429798e/assets/horizon/Livox%20Horizon%20user%20manual%20v1.0.pdf.

[6] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," Jun. 2009. DOI: 10.15607/RSS.2009.V.021.

[7] J. Zhang and S. Singh, "Loam : Lidar odometry and mapping in real-time," *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, Jan. 2014.

[8] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson surface reconstruction for lidar odometry and mapping," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5624–5630. DOI: 10.1109/ICRA48506.2021.9562069.

[9] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "Vdbfusion: Flexible and efficient tsdf integration of range sensor data," *Sensors*, vol. 22, no. 3, 2022, ISSN: 1424-8220. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/1296.

[10] X. Zhong, Y. Pan, J. Behley, and C. Stachniss, *Shine-mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations*, 2023. arXiv: 2210.02299 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2210.02299.

[11] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto, "Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 227–234, Jan. 2020, ISSN: 2377-3774. DOI: 10.1109/lra.2019.2953859. [Online]. Available: http://dx.doi.org/10.1109/LRA.2019.2953859.

[12] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "Pin-slam: Lidar slam using a point-based implicit neural representation for achieving global map consistency," *IEEE Transactions on Robotics*, pp. 1–20, 2024, ISSN: 1941-0468. DOI: 10.1109/tro.2024.3422055. [Online]. Available: http://dx.doi.org/10.1109/TRO.2024.3422055.

[13] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Yalla, "Robust method for removing dynamic objects from point clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 765–10 771. DOI: 10.1109/ICRA40945.2020.9197168.

[14] Y. Cong, C. Chen, J. Li, W. Wu, S. Li, and B. Yang, "Mapping without dynamic: Robust lidar-slam for ugv mobile mapping in dynamic environments," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B1-2020, pp. 515–520, Aug. 2020. DOI: 10.5194/isprs-archives-XLIII-B1-2020-515-2020.

[15] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113. DOI: 10.1109/CVPR.2016.445.

[16] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, 2020. arXiv: 2003.08934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.08934.

[17] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, vol. 42, no. 4, Jul. 2023. [Online]. Available: http://www-sop.inria.fr/reves/Basilic/2023/KKLD23.

[18] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995. DOI: 10.1109/2945.468400.

[19] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, *Nerf in the wild: Neural radiance fields for unconstrained photo collections*, 2021. arXiv: 2008.02268 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2008.02268.

[20] K. Rematas, A. Liu, P. P. Srinivasan, *et al.*, *Urban radiance fields*, 2021. arXiv: 2111.14643 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2111.14643.

[21] W. Bian, Z. Wang, K. Li, J.-W. Bian, and V. A. Prisacariu, *Nope-nerf: Optimising neural radiance field with no pose prior*, 2023. arXiv: 2212.07388 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2212.07388.

[22] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, *Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields*, 2021. arXiv: 2103.13415 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.13415.

[23] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, *Mip-nerf 360: Unbounded anti-aliased neural radiance fields*, 2022. arXiv: 2111.12077 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2111.12077.

[24] C. Li, Y. Ren, and B. Liu, *Pcgen: Point cloud generator for lidar simulation*, 2022. arXiv: 2210.08738 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2210.08738.

[25] M. Chang, A. Sharma, M. Kaess, and S. Lucey, "Neural radiance field with lidar maps," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 17 914–17 923.

[26] J. Zhang, F. Zhang, S. Kuang, and L. Zhang, *Nerf-lidar: Generating realistic lidar point clouds with neural radiance fields*, 2024. arXiv: 2304.14811 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.14811.

[27] T. Tao, L. Gao, G. Wang, *et al.*, *Lidar-nerf: Novel lidar view synthesis via neural radiance fields*, 2023. arXiv: 2304.10406 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.10406.

[28] S. Huang, Z. Gojcic, Z. Wang, *et al.*, *Neural lidar fields for novel view synthesis*, 2023. arXiv: 2305.01643 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2305.01643.

[29] D. Yan, X. Lyu, J. Shi, and Y. Lin, *Efficient implicit neural reconstruction using lidar*, 2023. arXiv: 2302.14363 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2302.14363.

[30] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, *Zip-nerf: Anti-aliased grid-based neural radiance fields*, 2023. arXiv: 2304.06706 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.06706.

[31] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, *3d-r2n2: A unified approach for single and multi-view 3d object reconstruction*, 2016. arXiv: 1604.00449 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1604.00449.

[32] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, *Atlasnet: A papier-mâché approach to learning 3d surface generation*, 2018. arXiv: 1802.05384 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1802.05384.

[33] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, *Deepsdf: Learning continuous signed distance functions for shape representation*, 2019. arXiv: 1901.05103 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1901.05103.

[34] R. Chabra, J. E. Lenssen, E. Ilg, *et al.*, *Deep local shapes: Learning local sdf priors for detailed 3d reconstruction*, 2020. arXiv: 2003.10983 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.10983.

[35] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, *Occupancy networks: Learning 3d reconstruction in function space*, 2019. arXiv: 1812.03828 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1812.03828.

[36] Á. González, "Measurement of areas on a sphere using fibonacci and latitude–longitude lattices," *Mathematical Geosciences*, vol. 42, no. 1, pp. 49–64, 2009. DOI: 10.1007/s11004-009-9257-x.

[37] A. Sch, *Fibonacci lattices*, https://observablehq.com/@meetamit/fibonacci-lattices, Accessed: 2024-09-12.

[38] M. Tancik, P. P. Srinivasan, B. Mildenhall, *et al.*, *Fourier features let networks learn high frequency functions in low dimensional domains*, 2020. arXiv: 2006.10739 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2006.10739.

[39] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987, ISSN: 0097-8930. DOI: 10.1145/37402.37422. [Online]. Available: https://doi.org/10.1145/37402.37422.

[40] Livox, *Livox mid-40 quick start guide multi v1.4*, 2019. [Online]. Available: https://www.livoxtech.com/3296f540ecf5458a8829e01cf429798e/downloads/20190530/Livox%20Mid-40%20Quick%20Start%20Guide%20multi%20v1.4.pdf.

[41] H. V. Tsoi, *Pv-lio: Probabilistic voxelmap-based lidar-inertial odometry*, https://github.com/HViktorTsoi/PV-LIO, Accessed: 2024-09-02, 2024.

[42] Livox, *Downloads center*, https://www.livoxtech.com/downloads, Accessed: 2024-09-10, 2024.

[43] D. Masters and C. Luschi, *Revisiting small batch training for deep neural networks*, 2018. arXiv: 1804.07612 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1804.07612.

[44] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, 2017. arXiv: 1609.04836 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1609.04836.

[45] N. Rahaman, A. Baratin, D. Arpit, *et al.*, *On the spectral bias of neural networks*, 2019. arXiv: 1806.08734 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1806.08734.

[46] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, *Nerf–: Neural radiance fields without known camera parameters*, 2022. arXiv: 2102.07064 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2102.07064.

[47] P. Wang, Y. Liu, Z. Chen, *et al.*, *F^2-nerf: Fast neural radiance field training with free camera trajectories*, 2023. arXiv: 2303.15951 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2303.15951.