Mathematisch-
Naturwissenschaftliche
Fakultät

Lernbasierte Computer Vision

Masterarbeit

Exploring Depth-guided 3D Reconstruction with Radiance Fields

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Shrisha Bharadwaj, shrisha.bharadwaj@student.uni-tuebingen.de, 2022

Bearbeitungszeitraum: 01/12/2021 - 07/06/2022

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter: Prof. Dr. Gerard Pons-Moll, Universität Tübingen

Abstract

Reconstructing coherent and complete point clouds of real-world scenes is challenging because existing scene representation methods rely on explicit 3D supervision. To combat this, emerging research focuses on coordinate-based neural representations that can be trained with only 2D images. Our key objective is to leverage pixel-wise information of 2D images with its complementary sparse point cloud information, to improve the quality of 3D reconstruction. Neural Radiance fields (NeRF) demonstrate impressive results in view synthesis by learning a density-based model from posed 2D images; but, there is a drop in performance for large scenes with complex geometry. NerfingMVS extend NeRF by efficiently sampling points centered around a depth prior (from a monocular depth network). The size of each sampling interval is decided by the uncertainty of the predicted depth prior. However, we observe and show that the modeled uncertainties are not meaningful of the errors present in the depth prior. Consequently, the range of the sampling interval is directly affected, which decreases the quality of reconstructions. We address this by formulating a bi-directional depth reprojection error and empirically show that the modeled uncertainties encode most of the depth inaccuracies. Additionally, we study the benefits of utilizing stereo depth priors and observe that they specifically capture fine geometric details. Finally, we regularize the geometry by supervising the expected depth and imposing a depth smoothness constraint. We observe that utilizing stereo depth priors captures thin structures, while depth supervision improves the quality of reconstruction and combining the advantages of both enables the model to learn a compact point cloud with fine geometric details.

Contents

1 Introduction

1.1 Motivation

Obtaining a digital representation of the world in 3D is crucial for applications such as the navigation of autonomous agents, digital heritage preservation, terrestrial surveying, and virtual tourism. Recently, learning-based approaches have demonstrated impressive results for 3D reconstruction through learning a rich prior during the training process [MON+19, CZ18, PFS+19, FSG16, LPZR18, GFK+18]; specifically, [FSG16] pioneered reconstructing 3D point clouds as an output representation from 2D image input. Point clouds are compact, expressive, and have the ability to store complex geometric details without requiring large amounts of memory. Thus, point-based output representations are preferred for reconstructing the scene geometry of large scenes, especially those of urban environments [MWA+13]. However, learning-based 3D reconstruction approaches require accurate 3D models for the training process and often depend on synthetic datasets [CFG+15, XHG+21] or utilise 3D sensors to acquire point clouds of real-world scenes.

However, each acquisition method produces point clouds with imperfections such as nonuniform sampling, misaligned scans, and noisy measurements [BTS+14]. A fundamental limitation that is common to all sensors is *missing data*, which primarily occurs due to occlusion and the physical limitations of the range sensors. It is challenging to acquire coherent and complete point clouds of outdoor environments because buildings are often located in streets that are surrounded by other buildings that act as obstructions. Additionally, various objects such as cars, vegetation, and street signs are present in overlapping regions and constantly occlude one another. Fig 1.1 (top) demonstrates the incomplete nature of point clouds acquired by accumulating points from three sensors taken from the KITTI-360 dataset [LXG21].

Due to the difficulties in acquiring complete 3D models of real-world scenes, image-based 3D reconstruction is being investigated. Imagery data is available in abundance, particularly as the cost of cameras has fallen significantly and their use in consumer devices is now widespread. Learning-based approaches for image-based 3D reconstruction take multi-view images as inputs and design a differentiable renderer that maps 3D coordinates to 2D images [NMOG20, YKM+20, SL20, SZW19, JJHZ19]. Image-based scene-level reconstruction of a real scene, however, is challenging because of textureless regions, occlusions, and reflective surfaces where arbitrarily

Figure 1.1: **Example from the KITTI-360 dataset [LXG21]**: Top: A point cloud of an urban environment acquired from three sensors. Bottom: An image captured at the same camera pose. This figure demonstrates the incompleteness of the accumulated point cloud. Observe the image at (a) red building (b) roof of the white building (c) an entire building next to the red one; the fused point cloud fails to capture these regions due to the physical limitations of the sensors (LiDAR and SICK).

different reconstructions satisfy the input images. Furthermore, without prior information, it is challenging to recover the exact size of an object from only images, as objects may appear similar regardless of their size, depending on the camera angle.

To address the ambiguities of image-based reconstruction, adding sparse point cloud information as an input source is a promising direction that provides geometry information to the reconstruction process [LCOZ+11]. Several RGB-D based methods for indoor environments propose encoding image features into a latent embedding and fusing them with depth features [SYZ+16, LLG+19, JLY20, LZL+20, LHZ+18]. Sparse point clouds obtained from sensors (e.g. LiDAR, RGB-D) contain real-world measurements with a high level of accuracy and this is beneficial to guide the reconstruction process. Alternatively, cameras do not require direct physical contact with the real world and this enables them to capture color information in regions that are unreachable to the sensors (e.g. LiDAR, SICK). Observe Fig. 1.1 containing an image and a point cloud captured at the same camera pose. The point cloud of the red building and roof/window region of the white building is incomplete due to the physical limitation of sensors (e.g. LiDAR, SICK), whereas the image contains dense color information for the missing regions in the point cloud. Motivated by this, we investigate the impact of leveraging pixel-wise information from high-resolution images with its complementary depth information for the task of 3D reconstruction.

To achieve this, we leverage learning-based approaches by considering coordinate based neural representations. Neural Radiance Fields (NeRF) [MST+20] achieve

impressive results on image synthesis by learning a volumetric representation. NeRF encodes the color and density information of each scene as the parameters of a single MLP network. By utilising classical volume rendering techniques, NeRF learns a scene representation from 2D images. However, for large scene with complex geometry, the extracted geometry of NeRF is not noise-free and is susceptible to shape-radiance ambiguity, where the model tends to learn arbitrary geometry information for textureless regions to satisfy the color information of that region [ZRSK20].

NerfingMVS [WLR+21] address the shape-radiance ambiguity by proposing a sampling strategy to guide the optimization process of NeRF. First, they acquire point clouds of each scene by running COLMAP [SZPF16, SF16] and project them to view-specific depth maps. Further, points queried for each region are designed to be concentrated around depth priors that are acquired from a monocular depth network. The range of the sampling interval is decided by estimating the uncertainty of the depth priors. DS-NeRF [DLZR22] acquire sparse points from structure-from-motion and propose to incorporate depth supervision along with color supervision synchronously. However, in practise depth and image frames are recorded asynchronously when we consider LiDAR sensor (based on time-of-flight). Motivated by the strategy of utilising projected depth for constraining the sampling interval, we base our formulation on NerfingMVS.

In summary, the goal of this thesis is to reconstruct dense point clouds from 2D images as well as sparse input point clouds of static urban environments provided by the KITTI-360 dataset [LXG21]. We are primarily interested in investigating the potential of combining image information with point clouds acquired by 3D sensors for the task of 3D reconstruction. We follow NeRF and represent each scene as a neural radiance field using a coordinate-based model and learn to reconstruct dense point clouds for 10 scenes from the KITTI-360 dataset [LXG21]. More specifically, we adopt the framework of NerfingMVS [WLR+21] which is designed to improve the expected depth prediction of NeRF. However, we observe that the sampling interval is incorrect when NerfingMVS underestimates the uncertainty of the depth priors and this decreases the quality of the extracted geometry. Hence, we propose the following modifications to tailor it for 3D reconstruction:

1. We design a toy setting by considering 10 chairs from the ShapeNet dataset [CFG+15], where we analyse the limitations of the sampling strategy of NerfingMVS. We show, that the uncertainty estimation of the depth prior is crucial for the entire learning process and NerfingMVS underestimates the uncertainty.

2. Based on the insights from the toy setting, we formulate the bi-directional depth reprojection error to perform a geometric consistency check of the depth priors. This modification enables the network to learn a consistent representation.

3. To enforce a stronger prior on the 3D modeling process, we exploit the

availability of stereo-pairs in the KITTI-360 dataset and study the benefits of utilising stereo priors over monocular depth priors. We observe that stereo priors specifically improve the quality of the reconstructed point cloud where fine geometric details that are not scanned by the sensors are captured.

4. Finally, to analyse the advantages of directly supervising the expected depth, we utilise the depth smoothness regularizer [NBM+22] in addition to depth supervision [DLZR22]. We demonstrate qualitatively and quantitatively that incorporating depth supervision results in sharp reconstructions with distinct discontinuities at object boundaries.

In conclusion, we investigate a framework for reconstructing dense 3D point clouds by providing synchronized supervision of image and depth for each pixel. Firstly, we study the effectiveness of the sampling strategy for 3D reconstruction and compare NeRF, NerfingMVS, +bi-directional depth reprojection error on chair setting. Furthermore, we investigate the benefits of stereo depth priors over monocular depth priors and additionally study the individual advantages of incorporating only depth supervision, only depth prior, and both on the quality of the reconstructed point clouds. Finally, the combination of utilising stereo depth priors with depth supervision on the framework, with our bi-directional depth reprojection error, qualitatively and quantitatively outperforms our baseline methods.

Outline

We describe the outline of this thesis. Related works are covered in Chapter 2. We provide a detailed review of Neural Radiance Fields and NerfingMVS in Chapter 3. Additionally, we cover the fundamentals of depth projection in Chapter 3. In Chapter 4 we discuss the overview, and analyse the limitations of sampling interval in detail. Followed by the details of our formulation and describe the optimization objective we use. The details of data preparation can be found in Chapter 5. Furthermore, we discuss the results and our studies on stereo prior vs monocular prior, depth supervision vs depth prior in Chapter 6. Conclusion and future work is described in Chapter 7.

2 Related Works

In this chapter, we first discuss learning-based approaches for 3D reconstruction and cover different output representations used in the past, and the recent scene representations with 3D supervision. Next, we cover 3D reconstruction from multi-view images, the advances in the field and describe the recent work (NeRF) which is central to this thesis. Further, we discuss the various extensions that are build on neural radiance fields and cover the most relevant works. Finally, we discuss the literature of depth estimation as we utilise depth priors in this thesis.

2.1 3D Representations

Learning-based approached for 3D reconstruction are broadly classified based on their output representation of the geometry. Combining deep learning with point-based representations were pioneered by [FSG16], where they proposed a network to reconstruct the geometry of simple objects from a single image. They designed a hierarchical neural network architecture and for the first time, introduced an optimization objective that enforces the network to predict points close to the ground-truth point cloud. Chamfer Distance and Earth-Mover Distance were introduced by [FSG16] and they are widely used as an optimization objective or as an evaluation metric by several 3D reconstruction methods. In this paper, we utilise Chamfer Distance as the evaluation metric. Further, [QSMG16] introduced an MLP architecture for processing point clouds such that the network adheres to the properties of point clouds as a representation. The network was designed with a combination of fully-connected layer and global pooling operations such that it is permutation invariant. Even though [QSMG16] was introduced for the task of classification and semantic segmentation, the Encoder network is widely adapted as a backbone by multiple 3D modeling/reconstruction methods [MON+19, PNM+20, OMN+19]. Despite the advantages of point-based representations such as its compact and lightweight nature, an ambiguity that still exists, is its unorderedness that ignores topological information.

3D reconstruction and modeling was revolutionized with the introduction of implicit surface representation by [MON+19, PFS+19, CZ18]. Implicit representations tackle the limitations of explicit representations (point-based, voxel-based and mesh-based representation) to represent geometry at arbitrary topology with a constant memory

footprint. Occupancy Networks [MON+19] use a single neural network to learn a function mapping between a 3D point and its occupancy probability [0,1] where the decision boundary represents the surface. Similarly, [PFS+19] model the surface as a signed distance function, where the neural network takes a 3D point as input and outputs the points distance to the closest surface, where the sign of the function indicates whether the point is inside (negative) or outside (positive) the surface. The introduction of these works where they model a continuous function as the parameters of neural network led to significant improvement in 3D reconstruction results. Follow-up works [PNM+20, CAP20] focus on scene-level reconstruction and utilise low dimensional features to improve reconstruction detail. But, these methods still require explicit 3D supervision for training. 3D models of synthetic datasets with CAD models are available [CFG+15], but it is non-trivial to obtain watertight models of real-world scenes. In Chapter 1, we described the difficulties of obtaining perfect 3D models of real data because of the imperfections of 3D sensors. Point clouds obtained by 3D sensors contain defects such as nonuniform sampling, noisy measurements and missing data. Thus, in the next section we discuss image-based 3D reconstruction as an alternative to reconstructing geometry purely from depth data.

2.2 3D Reconstruction from Multi-view Images

Reconstructing 3D geometry from 2D images has been a long-standing goal of the computer vision community [HZ04]. The goal of this task is to recover the depth information that is lost during the image formation process. In most of the cases, 2D images are captured from a calibrated camera and the poses of each image is available. When the 2D images contain a single scene with multiple views of the scene along with their camera poses, the task if referred as Multi-View Stereo (MVS). Classical algorithms that aim to reconstruct 3D geometry from 2D poses images are based mostly on epipolar geometry. Image-based 3D reconstruction systems traditionally have the following steps: feature extraction, feature matching step, predicting depth from images, depth map fusion, where feature matching step forms the basis of the entire pipeline. The goal of feature matching/extraction is to find matching features between image pairs. The algorithm must have the ability to match local, salient features belonging to the same 3D world points but captured from multi-view cameras. To find matching features, nearest-neighbour search was initially utilised, but the cost of computing matching features for the entire image is very high. Later on, an algorithm was proposed to match patches of features by random sampling and leveraging the coherence of images to quickly propagate similar matches to surrounding areas was proposed [BSFG09]. As a follow-up, [BRR11] extended PatchMatch to incorporate view propagation where planes were extended to the left and right views of stereo pairs. The seminal work by [SF16, SZPF16] (COLMAP) use structure-from-motion to estimate 3D geometry from images and release their open sourced code. The goal of their pipeline is to

jointly solve for camera poses along with 3D geometry from 2D unstructured images. They follow the traditional reconstruction pipeline where the first step is to match features, that is also termed as Correspondence Search where the algorithm finds matching features between pairs of images, followed by incremental reconstruction where they start with 2 images and then add images incrementally. But, to obtain the final point cloud, predicted depth maps are fused using volumetric fusion [CL96].

Learning-based multi-view stereo algorithms usually replace one component of the MVS pipeline with a method that exploits deep learning. Convolutional Neural Networks (CNNs) have demonstrated extraordinary performance in scene understanding and are able to learn contextual features that are beneficial for the feature matching step of MVS pipeline. Some approaches exploit CNN to learn local features of the same surface points with respect to different camera views and different orientations [LFB18, HGH+17, LSU16]. Deep learning for depth estimation has been explored (more of this will be covered in a separate section), but some methods perform depth estimation task purely from images for the MVS pipeline [YLL+18a, YLL+18b, YLL+19].

Compared to explicit representation, recent approaches based on neural scene representations, represent the surface with a neural network by by learning a continous representation based on occupancy values or signed distance fields. These methods learn from 2D supervision and do not suffer from discretization. Among the line of these works, [NMOG20] learn an occupancy grid representation by deriving analytical gradients with respect to the parameters of the network representing shape and texture information. [YKM+20] propose a surface normal and view-dependent based rendering network that captured view-dependent effects. They additionally demonstrate sphere-tracing with positional encoding and are able to reconstruct complex geometry with high reconstruction details. [SZW19] design an LSTM based ray-marcher and generalize across 3D scenes from a single input image. These methods learn from only 2D posed images and demonstrate impressive reconstruction results. However, these methods show results on object-level reconstruction and additionally require accurate object masks.

Neural Radiance Fields

Neural Radiance Fields (NeRF) [MST+20] demonstrate impressive results on view-synthesis by representing a scene with a neural network that maps a 3D point and viewing direction to density and color value. NeRF is based on classical volume rendering techniques and learns a density-field to represent the underlying geometry. To train NeRF, posed RGB images are used and a photometric based loss is used. For input scenes containing large number of camera views captured in a controlled environment, and for scenes that are small, NeRF predicts highly detailed novel

views. There are several follow-up works that address the limitations of NeRF, that we review in the next section separately. Since our formulation is based on NeRF, we provide complete details of its method under a section in Chapter 3.

2.3 NeRF-related

One of the limitations of NeRF is the requirement of a large number of input camera views per scene, and additionally NeRF is only able to optimize a single scene at a time. To address this, [YYTK20, CBLPM21, RMBF21] propose conditional generative models. [YYTK20, CBLPM21] utilise a CNN and initialize it with ImageNet weights to condition each scene with local CNN features. [RMBF21] learn a generative model trained explicitly on a synthetic dataset with ground-truth voxel grids and then generalize to real scenes of the same object category. Both of these methods are demonstrated for object-level view synthesis and show impressive performance to train with sparse inputs. [NBM+22] introduce regularization strategies to NeRF framework, where they regularize the geometry with a smoothness constraint and the appearance regularization is done with a flow-based model. We adapt the geometry regularizer of [NBM+22] that applies patch-based depth smoothness loss and does not require additional ground-truth depth information.

Several approaches explicitly model the surface by combining an implicit surface representation with the volume rendering framework of NeRF. UNISURF [OPG21] unify volumetric rendering with surface rendering by modeling an occupancy field instead of a density field, i.e the output of the neural network predicts an occupancy probability for each input 3D point and viewing direction. Similarly, [WLL+21] combine a signed distance field representation with volume rendering and incorporate it to NeRF framework. Both of these methods achieve remarkable 3D reconstruction results but are once again restricted to object-level reconstruction. Recently, [AMBG+22] was released in 2022 with the goal of surface reconstruction of large-scale indoor scenes obtained from RGB-D cameras. Similar to [WLL+21], they represent the surface as signed distance fields, but incorporate an additional pose refinement module that optimizes for the camera poses. They also demonstrate results on large scenes from ScanNet dataset [DCS+17] similar to [WLR+21].

The idea of combining depth input to neural radiance fields to improve the underlying geometry was simultaneously explored by [DLZR22, WLR+21]. We adapt both of these frameworks in our study and explain them in depth in Chapter 3.NerfingMVS [WLR+21] use point clouds obtained from the multi-view stereo step of COLMAP [SZPF16] and project points to obtain view-specific depth maps. These depth maps are utilised to fine-tune a monocular depth network and obtain a dense depth prior. Further, these depth priors are used to guide the optimization process of NeRF by setting adaptive sampling ranges for each ray. DS-NeRF [DLZR22] used the sparse point clouds obtained by running structure-from-motion (SfM) to directly supervise

the neural network. [DLZR22] proposed a depth based loss function where the expected depth prediction from NeRF was supervised with the sparse points from SfM. Motivated by DS-NeRF, Urban Radiance Fields (URF) [RLS+22] was released in 2022. The motivation of URF is to reconstruct the geometry of outdoor environments captured by lidar and images. They propose lidar based loss functions and explicitly model exposure parameters and model the sky with a separate neural network. They produce remarkable and realistic 3D meshes of outdoor environments. However, their code and dataset is not publicly available and we are unable to include URF in our studies. A method very close NerfingMVS/our experiments was released in 2022, titled Dense Depth Priors for Neural Radiance Fields with Sparse Input Views [RBM+22].

Dense Depth Priors for Neural Radiance Fields with Sparse Input Views [RBM+22]

In this work, the replace the monocular depth network of NerfingMVS with a depth completion network with convolutional spatial propagation networks [CWY18] (CSPN). CSPN is a depth estimation method that ranks under top 20 in the KITTI Benchmark [GLU12]. This depth completion network predicts an uncertainty map of the depth prior that indicates the depth inaccuracies of the predicted depth prior. The sampling interval is modeled as a Gaussian distribution with the mean centered at the depth prior and the standard deviation is modelled by the predicted uncertainty. This is very similar to our studies where individual components like the depth network and the range of the sampling interval is defined differently. They report results on ScanNet dataset [DCS+17].

2.4 Depth Estimation

Since we adapt the framework of NerfingMVS for our studies and compare mo nocular priors with stereo priors, we include a seperate section of depth estimation methods.

Monocular Depth Estimation

The task of monocular depth estimation is to predict the depth, given a single (mono-) image. Deep learning methods were first explored for depth estimation by [EPF14]. They introduced a multi-scale deep network for estimating monocular depth from RGB images. [EPF14] additionally introduced the scale-invariant loss to address the problem of global scale ambiguity of monocular predictions. In our experiments for training the monocular depth network, we use the scale-invariant loss of [EPF14] following [WLR+21]. [CFYD16] introduced a dataset containing images randomly taken from the internet, where each image contains relative depth annotations. They

designed an hour-glass CNN network for processing depth maps and the network is adapted by multiple works as a baseline [WLR+21, LDC+19]. In this thesis, we use the network architecture proposed by [CFYD16]. CSPN [CWY18] is a spatial propogation module that is able to propogate information to neighbouring pixels. CSPN module is proposed for stereo estimation as well as depth completion task. However, for our study we use [CFYD16] as the monocular depth network and compare monocular prior with a prior from a stereo depth network.

Stereo Depth Estimation

Given a pair of rectified images pairs (I_L,I_R), the goal of stereo estimation is to compute disparity d for each pixel. The horizontal displacement between a pair of corresponding pixels from both the images is defined as disparity. For a given pixel (x_1,x_2) from image I_L, if the corresponding point is found at a disparity of ($x_1 - d$,x_2) of image I_R, then the depth of this pixel can be estimated as $\frac{fB}{d}$, where f is the focal length of the caliberated camera, and B is the baseline (distance between the two cameras).

Stereo matching is a long-standing goal of computer vision and has a long history [SSZ01]. The advantages of CNNs were leveraged to replace the traditional cost matching computation and was pioneered by [vL16]. Recent methods, leverage 3D CNNs to construct a 3D feature cost volume. HSMNet [YMHR19] propose a framework where they hierarchically search for correspondences with a coarse-to-fine strategy with the aim of estimating disparity for high resolution images. They propose a network architecture that is encoder-decoder based, where the resolution of the output is gradually increased by the decoder. PSMNet [CC18] utilise spatial pyramid pooling to exploit global contextual information and dilated convolution to enlarge the receptive fields. They combine resulting global and local features to construct the cost volume and achieve state of the art results on KITTI [GLU12, MG15] in 2018. [CC18] does not stand first on the leaderboard of KITTI anymore, but we observed that it is most often used as a backbone network for multiple depth estimation methods in the KITTI leaderboard. Due to the availability of pretrained weights on KITTI, we adapt [CC18] as our stereo depth network (to obtain depth priors) and study the effects of stereo prior in comparison to monocular prior in Chapter 6.

3 Preliminaries

In this section we first provide a detailed review of Neural Radiance Fields in Section 3.1 and NerfingMVS in Section 3.2. Next, we cover the fundamentals of depth projection in Section 3.3 as it forms the basis for the bi-directional depth reprojection error that we introduce.

3.1 Neural Radiance Fields

A radiance field is a continuous 5D function representation f that maps a 3D location $\mathbf{x} \in \mathbb{R}^3$ and viewing direction $\mathbf{d} \in \mathbb{S}^2$ to a volume density $\sigma \in [0, \infty)$ and color value $\mathbf{c} \in [0,1]^3$. Neural Radiance Fields (NeRF) [MST+20] parameterizes this function with a multi-layer perceptron (MLP) with parameters θ to map each 5D input coordinate to its corresponding density and color value:

$$f_\theta : \mathbb{R}^3 \times \mathbb{S}^2 \mapsto [0,1]^3 \times [0, \infty) \tag{3.1}$$

To encourage the MLP to approximate high frequency functions, *positional encodings* $\gamma(\cdot)$ [TSM+20] are used. Positional encoding γ is a function mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$. This function is applied to $\mathbf{x}$ and $\mathbf{d}$ separately with L as a hyperparameter.

3.1.1 Volume Rendering

A pixel is rendered by casting ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ from the camera center $\mathbf{o}$ along the direction $\mathbf{d}$ through the pixel. The color value $\mathbf{c}_\theta$ and the density σ_θ[1] are predicted by the MLP network for points queried along the ray with near and far bounds t_n and t_f. The *expected* color value $\hat{\mathbf{c}}_\theta$ is calculated by classical alpha compositing as follows:

$$\hat{\mathbf{c}}_\theta(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma_\theta(\mathbf{r}(t))\mathbf{c}_\theta(\mathbf{r}(t), \mathbf{d})\, dt \tag{3.2}$$

where transmittance $T(\cdot)$, describes the amount of light attenuated when the ray reaches the camera and is calculated as:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma_\theta(\mathbf{r}(s))ds\right) \tag{3.3}$$

[1] For the sake of notational brevity, we henceforth denote $\mathbf{c}_\theta, \sigma_\theta = f_\theta(\mathbf{x}, \mathbf{d})$.

and the volume rendering integration weights are defined as:

$$w(t) = T(t) \cdot \sigma_\theta(\mathbf{r}(t)) \tag{3.4}$$

Following [Max95], the continuous integral (3.2) is evaluated using quadrature by querying points using stratified sampling approach. For N samples, the integral is approximated as follows:

$$\hat{\mathbf{c}}_\theta(\mathbf{r}) \approx \sum_{l=1}^{N} w_l \mathbf{c}_l \tag{3.5}$$

The parameters of the radiance field f_θ for a set of input camera rays $\mathcal{R}$ is optimized by minimizing the mean squared error between predicted color pixel and ground truth $\mathbf{c}_{gt}$ is the ground-truth color value .

$$\mathcal{L}_{MSE}(\theta, \mathcal{R}) = \sum_{r \in \mathcal{R}} \|\hat{\mathbf{c}}_\theta(\mathbf{r}) - \mathbf{c}_{gt}(\mathbf{r})\|^2 \tag{3.6}$$

3.1.2 Rendering Depth

NeRF renders color value for each camera ray, but in a similar fashion as (3.5), the expected depth can be computed as follows:

$$\hat{\mathbf{z}}_\theta(\mathbf{r}) \approx \sum_{l=1}^{N} w_l \mathbf{t}_l \tag{3.7}$$

where $\hat{\mathbf{z}}_\theta$ is the expected depth value of ray $\mathbf{r}$ and $\mathbf{t}_l$ is a point queried along the ray.

Network Architecture

We have employed the same architecture as NeRF [MST+20] with 8 fully-connected ReLU-layers having 256 channels each. The input for this is the 3D position $\mathbf{x} \in R^3$ of each point and its viewing direction $\mathbf{d}$. An additional ReLU-layer outputs the density σ to ensure it is non-negative, along with a feature vector. This feature vector is conditioned on the input viewing direction to model view-dependent effects and is processed by an additional MLP layer with 128 channels. The final layer has a sigmoid activation which outputs the color value. The architecture taken from the paper is shown in figure 3.1.

3.2 NerfingMVS

The goal of NerfingMVS is to estimate multi-view depth maps for indoor environments, where each scene consists of a room with multiple objects within. Their key observation is to utilize additional depth information obtained by running COLMAP

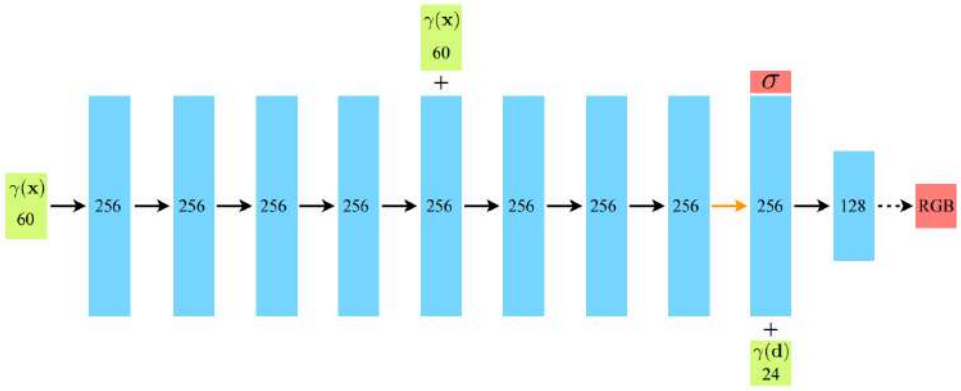

Figure 3.1: MLP architecture of NeRF. Image taken from NeRF's paper where all the layers are fully connected. Black arrows: indicate network with ReLU activation. Orange arrow: indicate network with no activation. "+" denotes vector concatenation. [MST+20]

[SZPF16, SF16] on each scene to guide the optimization process of neural radiance fields. This can be achieved by acquiring a per-view sparse depth map for each scene and train a monocular depth network to obtain dense depth priors. These scene-specific depth priors are further utilized to constrain the sampling interval of each ray. Thus, their strategy makes the optimization process more efficient and empirically outperforms NeRF to obtain finer depth maps. Since we extend NerfingMVS, we explain the preliminaries of their method below and an overview of their method can be found in Figure 3.2

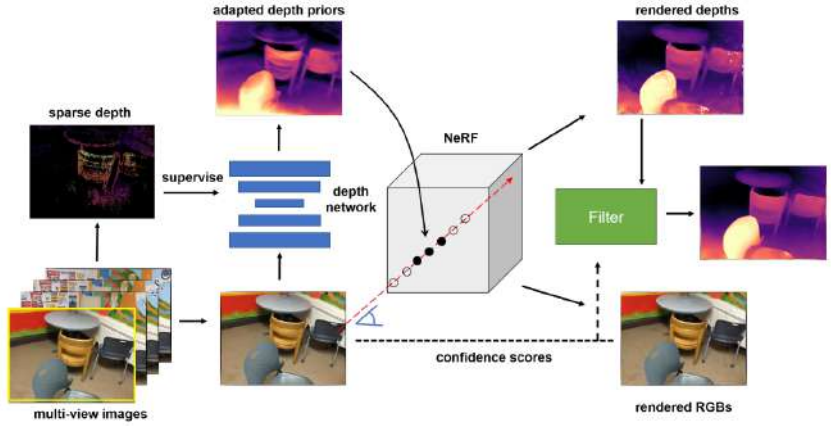

Figure 3.2: Overview of NerfingMVS [WLR+21] taken from the paper.

3.2.1 Monocular Depth Network

NerfingMVS adapt the network architecture of the monocular depth network proposed by [CFYD16] and is shown in Figure 3.3. The Monocular depth network utilizes relative depth supervision to predict the metric depth of monocular images. The depth maps obtained from COLMAP[SZPF16, SF16] are sparse and used as a supervision source for training the monocular depth network. Due to the global scale ambiguity of the prediction from the monocular network, the network is trained by the scale-invariant loss that is proposed by [EPF14] and applied in log-space:

$$\mathcal{L}_{\mathrm{SI}} = \frac{1}{L}\sum_{l\in L}|\log D^i(l) - \log D^i_{\mathrm{gt}}(l) + \alpha(D^i, D^i_{\mathrm{gt}})|, \tag{3.8}$$

where the loss is applied pixel-wise for all the pixels L in the sparse depth map D_{gt} that has a valid depth value. D is the depth prior predicted by the monocular depth network for all training images $i \in M$. The scale of the depth prior is aligned to the scale predicted by COLMAP with the scale factor $\alpha(D^i, D^i_{\mathrm{gt}})$ that is defined as follows:

$$\alpha(D^i, D^i_{\mathrm{gt}}) = \frac{1}{L}\sum_{l}(\log D^i(l) - \log D^i_{\mathrm{gt}}(l))$$

The fine-tuned monocular depth network acts as a strong prior than directly using the sparse maps from COLMAP [SZPF16, SF16] as it predicts an approximate depth value for each pixel in the image.

Network Architecture

The architecture is taken from [CFYD16] and is initialized with pretrained weights from the Mannequin challenge [LDC+19]. The architecture is visualized in Figure 3.3.

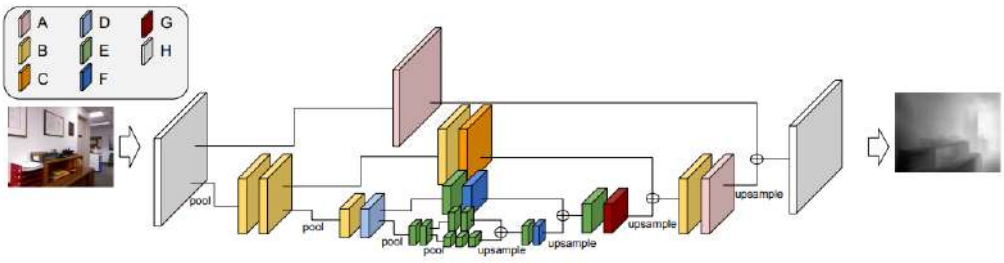

Figure 3.3: Monocular depth network architecture where the figure is taken from the paper [CFYD16]. Each block represents a convolutional layer following the Inception module where blocks sharing the same color are identical. $\bigoplus$ denotes element wise addition.

3.2.2 Guided Optimization

NeRF struggles on poorly textured areas like white walls of an indoor scene and this leads to shape-radiance ambiguity that is discussed in detail in [ZRSK20]. In a nutshell, shape-radiance ambiguity occurs when the network learns arbitrary geometry for each input view that is independent of the true surface. NerfingMVS additionally notice that queried points along the ray corresponding to a poorly textured area roughly predict the same RGB values. Motivated by this observation, a sampling strategy that explicitly limits the sampling range to be distributed around the depth priors is proposed. To do this, NerfingMVS first acquire per-view confidence maps (uncertainty maps) by projecting each depth prior to every other view and calculate the depth reprojection error to obtain relative pixel wise errors (We explain the calculation of depth reprojection error in detail in Section 3.3.3). These relative errors measure the uncertainty of the depth priors for each pixel and are further utilized to restrict the sampling bounds t_n and t_f to be concentrated around the depth priors. The sampling ranges are calculated (quoting from paper) as follows:

$$\begin{aligned} t_n &= D\,(1 - \text{clamp}\,(E\,,\,\alpha_l,\,\alpha_h)) \\ t_f &= D\,(1 + \text{clamp}\,(E\,,\,\alpha_l,\,\alpha_h)) \end{aligned} \tag{3.9}$$

where α_l and α_h are relative lower and upper bound used to control the size of the sampling interval and E is the error map obtained for each view. These adaptive sampling ranges are used to train the NeRF framework by restricting the sampling space to a smaller interval.

3.3 Depth Projection

In the following explanation, we consider a perspective camera projection model where the image plane captures the light rays emitted from a 3D point that exists in the world coordinate system. We also consider a single camera with known intrinsics K and extrinsics P. We have M depth maps $(D^i)_{i=1}^M \in \mathbb{R}^{H \times W}$ corresponding to each training view, and its respective extrinsics $(P_i)_{i=1}^M$.

Let us consider pose i as the reference camera pose, with its corresponding depth map

$$\mathbf{P}^i_{\text{ref}} = [\,\mathbf{R}^i_{\text{ref}} \,|\, \mathbf{t}^i_{\text{ref}}\,] \;;\; D^i_{\text{ref}}$$

and all the other poses are considered as the target camera poses $j = (1,..,M)$,

$$\mathbf{P}^j_{\text{target}} = [\,\mathbf{R}^j_{\text{target}} \,|\, \mathbf{t}^j_{\text{target}}\,] \;;\; D^j_{\text{target}}$$

and the intrincis where $s = 1$ given as:

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

3.3.1 Backprojected Depth

First, in order to transform the reference 2D depth map D^i_{ref} from the image plane to the world co-ordinate system we need to perform an "inverse" projection, i.e backproject the depth value $z \in D^i_{\text{ref}}$ associated to every pixel. For a particular pixel $\mathbf{x}_s = (x_1, x_2, 1)$ belonging to image plane (screen co-ordinates), with a depth value z, the corresponding 3D point in camera co-ordinate system $\mathbf{x}_c = (x_c^x, x_c^y, x_c^z)$ is given as:

$$\begin{aligned} x_c^x &= z \cdot \frac{x_1 - c_x}{f_x} \\ x_c^y &= z \cdot \frac{x_2 - c_y}{f_y} \\ x_c^z &= z \end{aligned} \tag{3.10}$$

For the entire image with a 2D grid of homogeneous screen co-ordinates $X_s \in \mathbb{R}^{3\times H\times W}$, pixels can be backprojected as

$$(\mathbf{K}^{-1} X_s) \cdot D^i_{\text{ref}}$$

The next step is to transform $\mathbf{x}_c$ from camera co-ordinate system to the world co-ordinate system and since $\mathbf{P}^i_{\text{ref}}$ is in camera-to-world reference, we simply multiply the 3D points with the pose:

$$\mathbf{X}^i_w = [\,\mathbf{R}^i_{\text{ref}} \,|\, \mathbf{t}^i_{\text{ref}}\,] \cdot (\mathbf{K}^{-1} X_s) \cdot D^i_{\text{ref}}$$

3.3.2 Projected Depth

In the next step, the points $\mathbf{X}^i_w$ in world co-ordinates are transformed to the camera co-ordinate system of the target poses $\mathbf{P}^j_{\text{target}}$:

$$[\,\mathbf{R}^j_{\text{target}} \,|\, \mathbf{t}^j_{\text{target}}\,]^{-1} \cdot \mathbf{X}^i_w$$

and finally, we perform perspective projection by multiplying the points with the focal length and dividing it by the depth values to obtain the screen co-ordinates of the target camera view:

$$S^{i\to j} = K \cdot [\,\mathbf{R}^j_{\text{target}} \,|\, \mathbf{t}^j_{\text{target}}\,]^{-1} \cdot \mathbf{X}^i_w \tag{3.11}$$

where $S^{i\to j} \in \mathbb{R}^{M\times 3\times H\times W}$ and the x- and y-axis represents the ($\mathbf{u}^{i\to j}$, $\mathbf{v}^{i\to j}$) co-ordinates and the z-axis is the projected depth $D^{i\to j}$. In a nutshell, the projected depth $D^{i\to j} \in \mathbb{R}^{M\times H\times W}$ is obtained by projecting the reference depth D^i_{ref} to j target camera views. Intuitively, when we project depth from $i \to j$, each pixel from i is mapped to j and we can obtain a depth value for all the pixels in i with respect to view j. Thus, all projected depth maps $D^{i\to j}$ are considered with respect to the reference view i, i.e $D^{i\to j}_{\text{ref}}$.

3.3.3 Depth Reprojection Error

A geometric consistency check can be performed by verifying whether the projected depth maps $D_{\text{ref}}^{i\to j} \in \mathbb{R}^{M\times H\times W}$ are consistent with respect to the depth maps of the target camera poses $D_{\text{target}}^{j} \in \mathbb{R}^{M\times H\times W}$. However, the projected depth maps are from the point of view of the reference pose. Therefore, all the depth maps belonging to the target camera poses need to be warped in order to match the reference view. To achieve this, D_{target}^{j} is sampled from the uv co-ordinates grid that we obtain from (3.11) as follows:

$$D_{\text{ref}}^{j} = D_{\text{target}}^{j} \sim [\mathbf{u}^{i\to j}, \mathbf{v}^{i\to j}] \tag{3.12}$$

The depth reprojection error is calculated as:

$$E_{\text{ref}}^{j} = \frac{D_{\text{ref}}^{j} - D_{\text{ref}}^{i\to j}}{D_{\text{ref}}^{i\to j} + \text{eps}} \tag{3.13}$$

where $E_{\text{ref}}^{j} \in \mathbb{R}^{M\times H\times W}$ is the per-view confidence map (that we refer to as the uncertainty map).

4 Method

4.1 Overview

Our goal is to reconstruct 3D point clouds with image and depth information. We adapt the framework of NerfingMVS due to its efficient sampling strategy and tailor it to the task of reconstruction by altering three components:

1. The uncertainty map E^k is calculated by the bi-directional depth reprojection error that we introduce.

2. We study the benefits of stereo priors in comparison to monocular priors by using a different network architecture in D_ϕ.

3. To regularize the geometry, we incorporate depth supervision $\mathcal{L}_{depth}$ and additionally a patch-based depth smoothness regularizer $\mathcal{L}_{DS}$.

We provide a detailed review of NeRF and NerfingMVS in Chapter 3. In this Chapter, we first conduct detailed analysis pertaining the limitations of NerfingMVS's sampling strategy in Section 4.2. Next, we introduce the bi-directional depth reprojection error and explain the sampling strategy in Section 4.3. Further, we introduce the stereo depth priors that we investigate in this study in Section 4.4. Finally, we discuss the loss functions and the optimization objective in Section 4.5. An overview of our framework is provided in Figure 4.1.

4.2 Analysis

In this section, we first conduct a detailed analysis of the following: (1) We discuss the advantages of constraining the sampling interval of NeRF in Section 4.2.1, i.e NerfingMVS's framework densely samples points close to the surface of the depth prior and avoids evaluating free-space and occluded samples. (2) We discuss the limitations of the sampling strategy of NerfingMVS in Section 4.2.2, i.e The uncertainty is underestimated and as a consequence real surface points lie outside the sampling interval. (3) Provide a detailed explanation of the cause of this limitation in Section 4.2.3, i.e The depth reprojection error does not capture the inaccuracies contained within each depth prior.

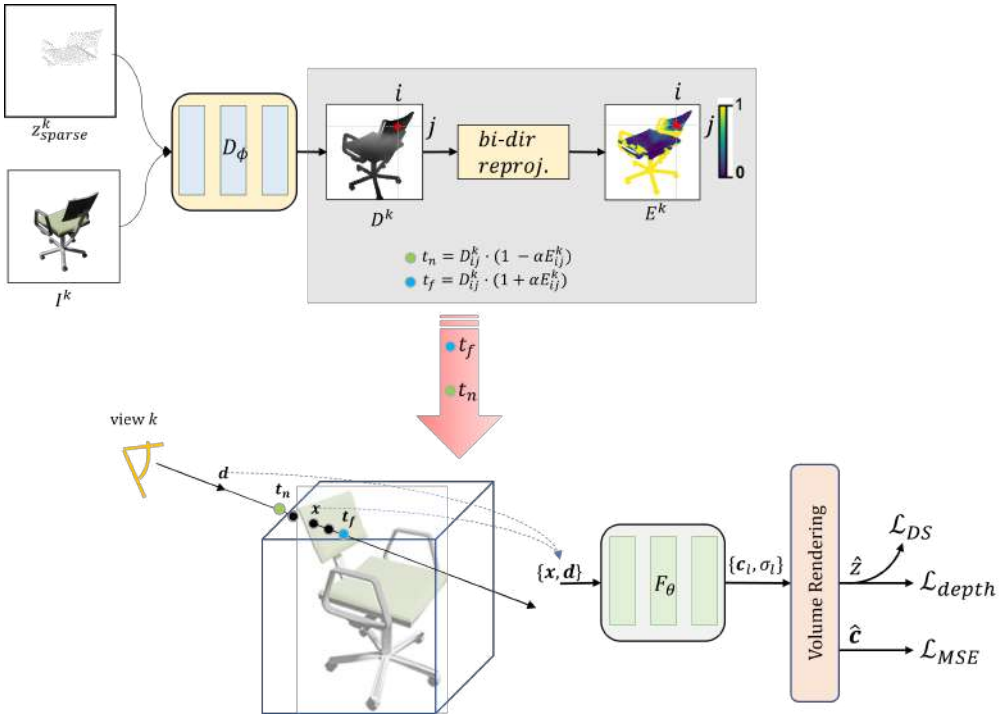

Figure 4.1: **Overview:** Depth network D_ϕ takes image I^k, sparse depth map z^k_{sparse} of camera view k, and predicts a depth map (depth prior) D^k. The bi-directional depth reprojection predicts the uncertainty map E^k for each D^k. For camera ray passing through pixel (i, j), queried points $\mathbf{x}$ are constrained to lie within an adaptive sampling interval (t_n, t_f) centered at D^k_{ij}. 3D coordinates $\mathbf{x}$ and viewing direction are evaluated by F_θ to obtain $\{\mathbf{c}_l, \sigma_l\}$ (l is the index of the number of queried samples). Finally, the expected color and depth values $(\hat{\mathbf{c}}, \hat{\mathbf{z}})$ are obtained after volume rendering and F_θ is optimized with color and depth supervision. Additionally, the geometry is regularized with depth smoothness loss $\mathcal{L}_{DS}$. The following components are different from NerfingMVS: (1) Bi-directional depth reprojection error to calculate E^k (2) D_ϕ: We use Monocular Depth Network as well as Stereo Depth Network (3) Regularize geometry with $\mathcal{L}_{depth}$ and $\mathcal{L}_{DS}$.

4.2.1 Constrained Sampling Interval

Scenes with large areas of interest containing few overlapping input views has been a challenge for NeRF which is trained in a controlled setting. One of the reasons is the sampling strategy where points are queried between defined bounds t_n and t_f such that the object of interest is contained within. NeRF uses the MLP network to evaluate each of these queried points that participate in the rendering equation. However, this strategy is inefficient because free space regions with zero density are sampled repeatedly. To increase the sampling efficiency, NeRF propose a hierarchical

sampling strategy where they simultaneously optimize two networks: "coarse" and "fine" for N_c and N_f set of points respectively. The former points are queried using stratified sampling and the latter points are sampled from a piece-wise constant probability density function obtained from the normalized integration weights. The "coarse" network is evaluated with N_c samples and the "fine" network at the union $N_c + N_f$. This strategy decreases the number of free-space or occluded points being evaluated by the "fine" neural network but it does not alleviate inefficient sampling entirely. However, another crucial feature to be considered is the sampling density of

Figure 4.2: **An example image from KITTI-360**. The box marked in black shows that every scene has thin structures occluding large objects.

each scene, i.e the distribution of points sampling the surface of the scene. Real-world scenes have high-frequency details causing sharp variation in density of points as well as dramatic depth discontinuities at object boundaries. Let us consider an example image from KITTI-360 shown in Figure 4.2. We can see that there are objects with thin structures that have dense points condensed to a small region. Juxtaposed to that, we find large buildings that contain dense points spanning uniformly. For instance, consider a ray passing through a pixel of a thin structure (Figure 4.2, pixel marked in red) occluding a larger object in the background. If we consider the regular formulation of NeRF, then N points are queried with stratified sampling between the intervals t_n to t_f such that the pole and the building are contained within. This is a challenging scenario because the network must learn to terminate the ray right after it passes the pole and two possible occurrences can hinder the learning process:

1. If the number of samples N is not sufficient, it is possible that the queried points do not intersect with the real surface and this can cause the network to learn arbitrary geometry to fit the training image leading to shape-radiance ambiguity [ZRSK20]. With enough input views and increase in the number of queried samples, the network will favor multi-view consistency. But, with scenes like KITTI-360 the number of overlapping views containing a particular object is very low (5-10 images) and thus less points are queried.

2. Furthermore, we can see that the building spatially occupies a larger part of the training images (this example and consecutive camera views) and this implies that a higher percentage of the camera rays and thus points, participate in the learning of the building in comparison to the pole. Due to this imbalance in the contribution of points between different objects, the network may learn to predict integration weights with higher confidence for all the points that are sampled near the building. When we consider this particular camera ray

that intersects both the objects, the weights of points near the building may overwhelm that of the weights learned near the pole. In this case, the depth value learned by the network at this pixel will belong to the building whereas the RGB value will belong to the color of the pole.

In conclusion, these factors reinforce the need to constrain the sampling space in a manner that reduces the number of points sampled in free space while simultaneously ensuring that the object of interest lies within the sampling interval.

4.2.2 Limitations

The predicted point cloud and depth maps of NerfingMVS [WLR+21] trained on a chair from the ShapeNet dataset are visualized in Figure 4.3. We can clearly see that the model predicts noisy point cloud for chair and learns multiple surface parts, i.e the legs of the chair and not a single solid surface (similar results on different scenes are provided in the results section 6.4). The depth maps from Figure 4.3 validate the extracted point cloud and we can see that the depth of the legs of the chair is predicted to be closer to the camera. In this section, we first analyze the reason behind this occurrence and further explain how it is caused and its repercussions.

We identify a limitation with the measure proposed by NerfingMVS [WLR+21]

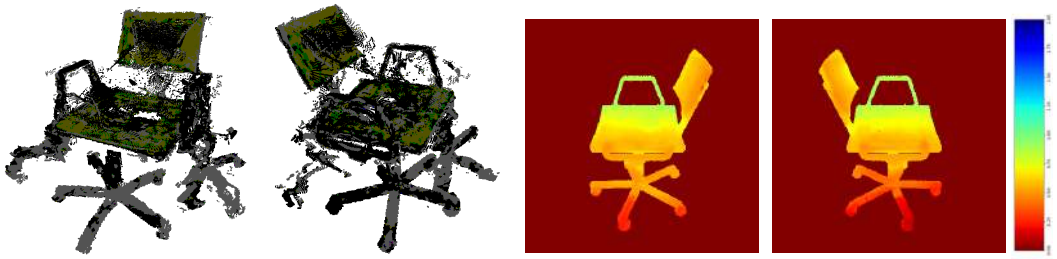

Figure 4.3: Demonstrates the inconsistent geometry learnt by NerfingMVS [WLR+21]. The depth maps on the right validate the extracted point clouds on the left.

(section 3.3.3) and observe that it assigns a lower uncertainty even to depth estimates that are not 3D-consistent. A plot of the uncertainty map of NerfingMVS [WLR+21], the depth prior, and the ground truth depth are visualized in Figure 4.4. From the figure and the resulting point cloud, we can see that the depth regions along the right handle, headrest, and the legs of the chair have inconsistent depth predictions compared to the ground-truth depth. The handlebar and the headrest are smoothed out against the seat of the chair, whereas the legs of the chair are predicted closer. Thus, the role of the uncertainty map is to capture these inaccuracies and assign a higher

uncertainty, but the measure proposed by NerfingMVS [WLR+21] underestimates the uncertainty.

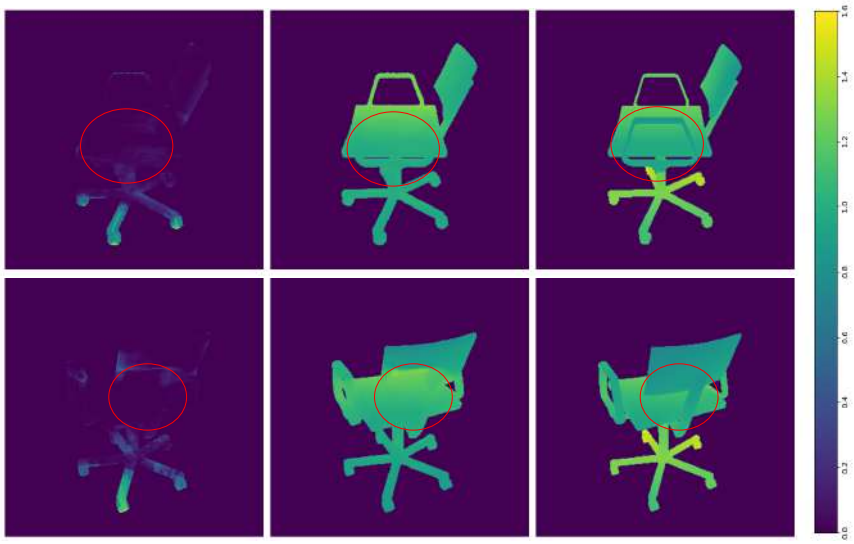

Figure 4.4: Visualizes the uncertainty map of NerfingMVS [WLR+21], depth priors from the monocular depth network and the ground-truth depth. Take from Chair 04.

Equation 3.9 shows that the size of the sampling interval for each ray is entirely dependent on the uncertainty maps E. In other words, E directly decides whether the sampling interval must be larger or smaller for each ray. Intuitively, if the value of the approximate surface depth at a particular pixel of view i is consistent with respect to all the views j, then we expect the uncertainty of this pixel to be low. Whereas, a high uncertainty value is *expected* if the depth value is inconsistent with respect to its projected depths. Hence, the role played by the uncertainty maps is pivotal for the entire learning process because we expect camera rays from multiple views belonging to the same 3D world point to coincide. When the uncertainty is lower for a particular pixel, then the sampling interval is tight and centered at the approximate depth estimate as illustrated in Figure 4.5 (c). Whereas, if the uncertainty is low for a depth estimate that is inconsistent then this implies that the approximate surface depth prediction is far from the real surface depth and the repercussions are twofold:

1. The sampling interval is tightly bound at the incorrect depth estimate and as a result the real surface often lies *outside the interval*. This is illustrated in Figure 4.5 (d).

2. When the sampling interval fails to contain the real 3D point for a particular pixel visible from view i, this implies that the camera ray for this pixel is

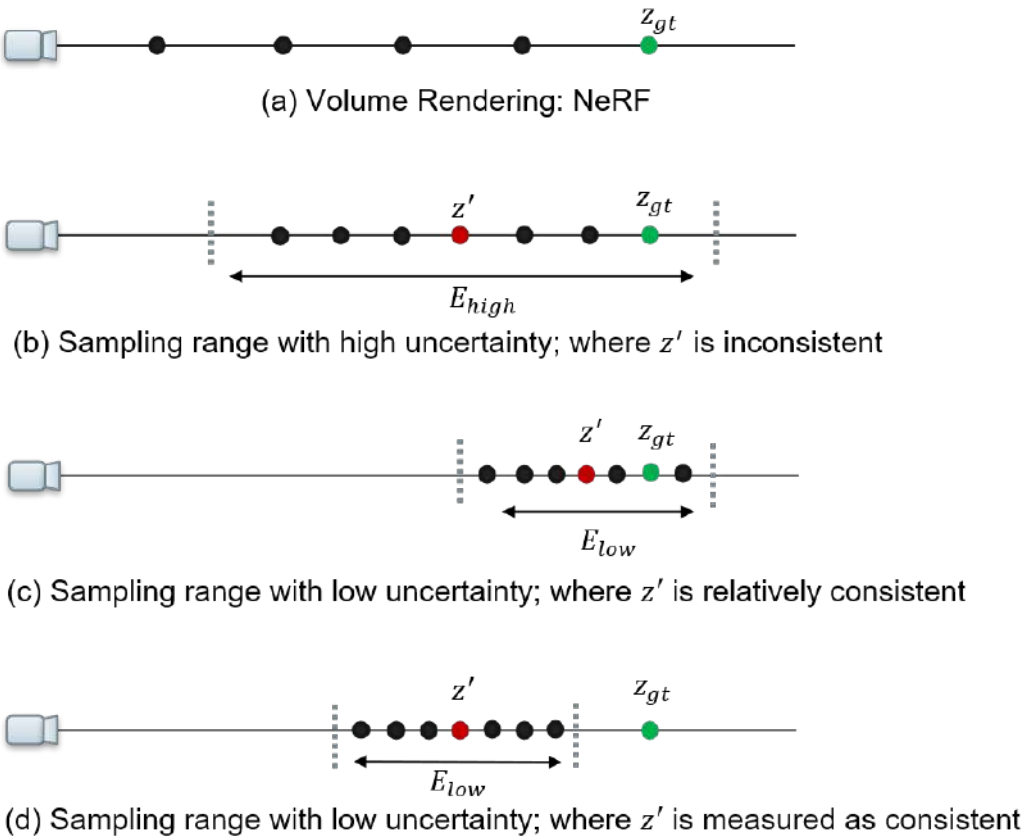

Figure 4.5: **Illustration of incorrect sampling:** This figure illustrates three cases of sampling strategy. (a) Shows NeRF, where it samples the entire space along a ray (b and c) When the uncertainty E is adequately estimated, the real surface depth z_{gt} is contained within the sampling interval (d) When the uncertainty is underestimated, the real surface depth z_{gt} lies outside the sampling interval.

terminated before colliding with the real surface. As a consequence of this, when a 3D world point is visible from multiple camera views, the rays from these different camera views do not intersect at their respective 3D world point but terminate before reaching the surface. This results in the model learning multiple surface parts that do not form a single, compact representation and we can see this from the extracted point cloud visualized in Figure 4.3.

In the following section 3.3 we first formalize the calculation of the uncertainty map by breaking down each step and investigating the failure modes.

4.2.3 Underestimated Uncertainty

We firmly believe that the inconsistent depth regions must be recognized by *atleast* one out of M uncertainty maps, because the camera poses are sampled uniformly to ensure a 360-degree view of the object. In order to thoroughly investigate the uncertainty maps, we qualitatively evaluate the projected depth maps ($D^{i\to j}_{\text{ref}}$) and the warped target depths (D^{j}_{ref}). The depth inaccuracies present in D^{i}_{ref} are also propagated to $D^{i\to j}_{\text{ref}}$, but, this is an expected behaviour because the maps are obtained by projecting $i \to j$. However, when we investigate D^{j}_{ref} in detail we notice that these depth maps are sampled from the uv coordinates that are obtained from $D^{i\to j}_{\text{ref}}$ (refer to equation (3.12). Thus, all the depth inaccuracies present in D^{i}_{ref} are also present in the grid of uv coordinates. To elaborate this further, we provide a visualization of the grid of uv coordinates in Figure 4.6. A particularly interesting occurrence is the absence of the right handlebar of the chair in the grid of uv coordinates of all the target views. This is due to the reference depth of this particular camera view that does not explicitly predict distinct depth values for the handlebar of the chair. From this visualization, we can see that the depth inaccuracies present in D^{i}_{ref} are also propagated to warped target depth D^{j}_{ref}.

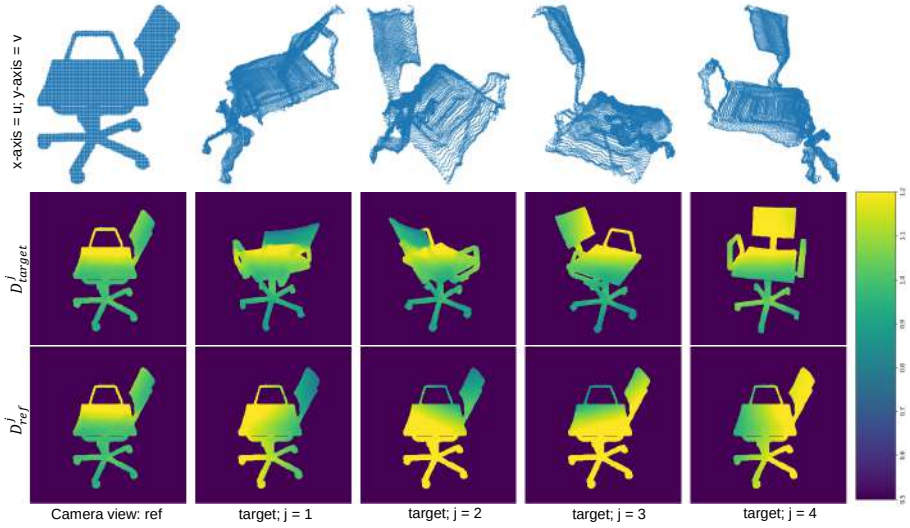

Figure 4.6: **Illustrates Reprojected Depth maps**. This figure containes the reference camera view along the first row, and the corresponding rows(2-4) contain target views. Notice that, the uv grid demonstrated in the top row does not contain the handle region of the chair, and due to this, the reprojected maps D^{j}_{ref} fails to assign an error to the handle region of the chair.

Since these inaccuracies are consistent across all the projected and target depth maps, they are not detected by the depth reprojection error. Hence, these uncertainty maps have a low uncertainty associated with these erroneous regions. But, in order to warp the target depths to the reference camera view, we require the grid of uv coordinates. In the next section, we describe the proposed solution that is able to capture these inaccuracies, assign a high uncertainty to these regions, and finally ensure that the surface depth lies inside the sampling interval.

4.3 Bi-directional Depth Reprojection Error

Reprojection error is a geometric cost function that is a gold-standard optimization objective that jointly optimizes the homography (camera pose) and pixel correspondences [HZ04, FHT17]. Pixel correspondences are defined as a pair of points from two images that are projected from the same 3D world point. In our case, a lower reprojection error between pixel correspondences implies that the depth values are in mutual agreement because we have access to camera poses from a calibrated camera. As we have established previously, depth inaccuracies are propagated from the reference depth to the projected target depth maps and they mutually agree with respect to the inaccuracies.

Our key observation is the fact that the depth reprojection error is applied with the intention of capturing the "transfer error" that occurs within each target depth map D^j_{target} when depth correspondences from the reference map are projected onto the target map. [HZ04] describe this as determining the transfer error in a single image with the assumption that points in the first map are captured accurately. However, the reference depth map also contains inaccuracies and it is crucial to determine the errors that occur within the reference map. [HZ04] describes the process of determining cost in both directions as the "symmetric transfer error", but their objective is to jointly minimize the homography and the correspondences. Motivated by their formulation, we propose an additional cost function that determines the "transfer error" that occurs in the backward direction to measure the inaccuracies within the reference depth map. In order to achieve that, we project the depth maps of the target poses to the reference pose and check if they mutually agree with the reference depth map. We explain the steps in detail below.

Depth Reprojection Error: Backward Direction

First, we backproject the depth maps belonging to the target camera poses to the world coordinate system by following the steps described in Section 3.3.1 as follows:

$$\mathbf{X}^j_w = [\,\mathbf{R}^j_{\text{target}} | \mathbf{t}^j_{\text{target}}\,] \cdot (\mathbf{K}^{-1} X_s) \cdot D^j_{\text{target}} \tag{4.1}$$

In the next step, we transform these 3D points to the camera coordinate system of the reference pose P^i_{ref} and then project the points to the screen coordinates of the reference camera view as follows:

$$S^{j\to i} = K \cdot [\,\mathbf{R}^i_{\text{ref}} \,|\, \mathbf{t}^i_{\text{ref}}\,]^{-1} \cdot \mathbf{X}^j_w \tag{4.2}$$

where the z-axis of $S^{j\to i} \in \mathbb{R}^{M\times 3\times H\times W}$ is the projected depth $D^{j\to i}_{\text{ref}} \in \mathbb{R}^{M\times H\times W}$ from the point of view of the reference pose. In order to compute the cost function we calculate the relative difference between the projected depth maps and the reference depth map. When we project pixels $j \to i$, each pixel in j is not mapped to i, hence $D^{j\to i}$ does not contain information for every pixel in the reference view. Thus, while computing the cost function we consider only these mapped pixels for D^i_{ref} as well. The reason we compute relative depth is because the depth values predicted by the depth network are ambiguous in scale and not tailor made for a multi-view data.

$$E^i_{\text{ref}} = \frac{D^{j\to i}_{\text{ref}} - D^i_{\text{ref}}}{D^i_{\text{ref}} + \text{eps}} \tag{4.3}$$

After obtaining M depth maps, we combine the depth reprojection error in the forward direction (3.13) with our proposed cost function in the backward direction from (4.3) to obtain $E^i = (E^j_{\text{ref}}, E^i_{\text{ref}}) \in \mathbb{R}^{2M\times\times H\times W}$. [HZ04] sum the cost functions of the forward and backward reprojection errors, but we consider a relative cost function and can not sum them. In order to aggregate pixel-wise error values and obtain a single uncertainty map, we follow a similar procedure as NerfingMVS [WLR+21], but instead consider K largest error values. Let $e_q \in \mathbb{R}^{M\times 1}$ be the error value of the q^{th} pixel for each error map, then the uncertainty $\forall q \in \mathbb{R}^{H\times W}$ is calculated as follows:

$$\underset{C\subset e_q, |C|=k}{\arg\max} \; \frac{1}{k}\sum_{c\in C} c \tag{4.4}$$

where we first sample a subset C containing k largest error values of the set e_q and consider the average of this subset C. We consider the largest error in order to capture the inaccuracies that are detected from different camera views. We provide qualitative and quantitative evaluation of the proposed error maps in the results section and show that our proposed error term is able to: (1) capture depth inconsistencies (2) ensure that the surface depth lies *inside* the sampling interval.

4.3.1 Sampling Interval

In order to constrain the sampling interval for each ray, we utilize the uncertainty map E to decide the size of the sampling interval around the approximate depth values (depth priors). The uncertainty map is a relative measure that lies in the closed interval $[0,1]$ and each pixel measures the relative uncertainty of the approximate depth values. Consider a camera view k, and its corresponding uncertainty map E^k

and depth prior D^k. For pixel (i, j), the measure $D^k_{i,j} \cdot E^k_{ij}$ estimates a rough deviation of the approximate depth value from the real depth. In order to obtain the near and far bound of this deviation for ray r through pixel (i, j), we can calculate it as follows:

$$\begin{aligned} t_n &= D^k_{i,j} - D^k_{i,j} \cdot E^k_{ij} \\ t_f &= D^k_{i,j} + D^k_{i,j} \cdot E^k_{ij} \end{aligned} \tag{4.5}$$

But the scale of a depth estimate is dependent on each scene and the real depth can have a larger deviation from the approximate depth (depth priors). With this formulation, the maximum deviation of the real depth from the approximate depth is at $E^k_{ij} = 1$, when $t_f = 2\ D^k_{i,j}$. But, when we consider a large-scale scene like KITTI-360, where each scene is approximately 60-80meters wide, it is possible that even doubling the approximate depth is insufficient to capture the entire scene. We provide an example in Figure 4.7. Thus, we introduce a hyperparameter α to

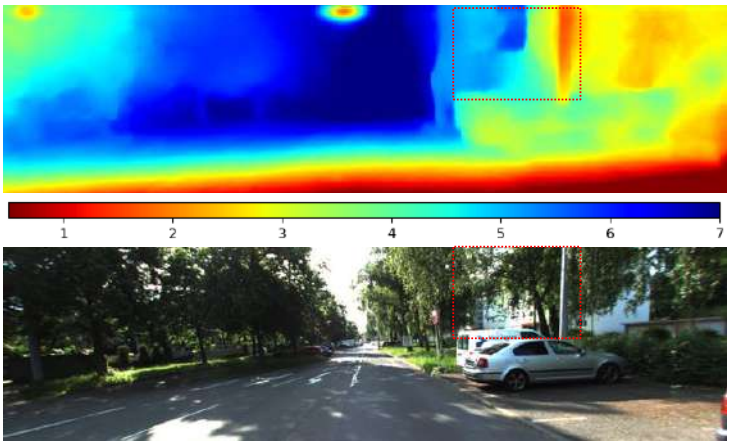

Figure 4.7: This figure provides an example to showcase that doubling the approximate depth is insufficient. The depth map in this visualization is scaled down by 10 times. Notice the depth of the region marked in the square, the approximate depth prediction is around 3-5 (which is 30-50m), but the RGB image shows the building in this region to be much further i.e approximately at 7-8 (70-80m). If the approximate depth at this region is 3, doubling it gives us 6 which is not sufficient to capture the real depth and learn a consistent 3D representation.

account for this scaling. This hyperparameter is responsible for the scale of the deviation and can be set according to each scene. We provide an ablation study of this hyperparameter in the results section. Thus, our final equation to calculate the near and far bound for each sampling interval is described below:

$$\begin{aligned} t_n &= D^k_{i,j} \cdot (1 - \alpha E^k_{ij}) \\ t_f &= D^k_{i,j} \cdot (1 + \alpha E^k_{ij}) \end{aligned} \tag{4.6}$$

Additionally, in order to ensure the bounds are positive we restrict the values to be greater than or equal to 0 as: min(0, t_n). To query points along each ray, we follow NeRF's [MST+20] approach of stratified sampling by partitioning $[t_n, t_f]$ to N evenly-spaced bins and draw one sample uniformly from each bin:

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\, t_n + \frac{i}{N}(t_f - t_n)\right] \tag{4.7}$$

4.4 Depth Prior

Monocular depth estimation is a challenging problem particularly because of scale ambiguity that arises from a single (monocular) image, i.e it is impossible to accurately predict metric depth from an image because visual tricks like parallax, perspective, occlusion, etc [SKB18]. Moreover, predicting absolute metric depth requires at least one reliable and known distance from the real-world, e.g. baseline between stereo cameras. Stereo depth estimation methods have superior performance in comparison to monocular estimation methods and this can be an unfair comparison because stereo methods have access to more information. But, this also implies that stereo image pairs have access to information that monocular images can never have and this information is crucial for recovering finer depth details. Thus, we exploit the availability of rectified stereo pairs in KITTI-360 dataset and utilise a stereo depth network to obtain approximate depth information for each training pair.

Stereo matching estimates depth via triangulation by estimating the disparity between matching pixel that can be obtained by recovering dense correspondences between image pairs. Obtaining depth from disparity will be beneficial for our task because of its ability to identify sharp depth discontinuities at object boundaries. Significant progress has been made in the recent years in applying deep learning for the problem of disparity estimation because of the ability of Convolutional Neural Networks (CNNs) to extract image features. Most of these approaches train their network using KITTI [GLU12, MG15] and Scene Flow [NEP+16] datasets. We are particularly interested in approaches trained on KITTI and check the leaderboard of KITTI 2015. We choose Pyramid Stereo Matching Network (PSMNet) [CC18] because even though it not the leader currently, it is used as a backbone network by multiple papers that tackle disparity estimation.

4.4.1 Pyramid Stereo Matching Network

Finding dense correspondences is a challenging task for the stereo matching pipeline and PSMNet utilize CNNs to design their network. The highlight of their network is the use of spatial pyramid pooling and dilated convolutions to obtain a larger receptive field. They also stack 3D CNNs in an hourglass fashion and process cost

volume in a top-down/bottom up manner to propagate global contextual information. Thus, we use the network architecture of PSMNet without any modifications and fine-tune their network on each scene individually. Since our ground truth disparity (the projected depth for each view) is very sparse, i.e 10% of the total pixels for LiDAR, we rely on pretrained weights. We fine-tune the network for few epochs to ensure that the network learns to fit the metric depth values of the given scene but not enough to predict random noisy artifacts in pixels without ground truth information. We use smooth L_1 loss following [CC18] due to its low sensitivity to outliers in comparison to L_2 loss and it is defined follows:

$$\begin{aligned} L(z', z_{sparse}) &= \frac{1}{N}\sum_{i=1}^{N} \beta\left(z'(i) - z_{sparse}(i)\right) \\ \beta(x) &= \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases}, \end{aligned} \tag{4.8}$$

where z', z_{sparse} are the predicted and ground truth depth values respectively.

Network Architecture

The network architecture of PSMNet [CC18] is visualized in Figure 4.8. We initialize the network with their pretrained weights that is trained on KITTI [GLU12, MG15].

4.5 Optimization Objective

Following NeRF, we optimize the radiance field f with mean squared error between the predicted color and the ground truth image. For a set of input camera rays $\mathcal{R}$ where $\mathbf{c}_{gt}$ is the ground-truth color value of the respective pixel .

$$\mathcal{L}_{MSE}(\theta, \mathcal{R}) = \sum_{r \in \mathcal{R}} \|\hat{\mathbf{c}}_\theta(\mathbf{r}) - \mathbf{c}_{gt}(\mathbf{r})\|^2 \tag{4.9}$$

Additionally, we experiment with various regularization techniques that optimize the radiance field over the expected depth $\hat{z}_\theta$ computed by (3.7). We follow several works from the literature and incorporate their strategies to improve the underlying geometry and decrease artifacts. We list them in detail below.

4.5.1 Depth Loss

Initialization at the start of the training is problematic because the differential density is uniformly allocated across the volume. Even though our sampled points are restricted to be concentrated around the approximate depth points, when the uncertainty of pixels are very high, the sampling interval is similar to NeRF [MST+20]

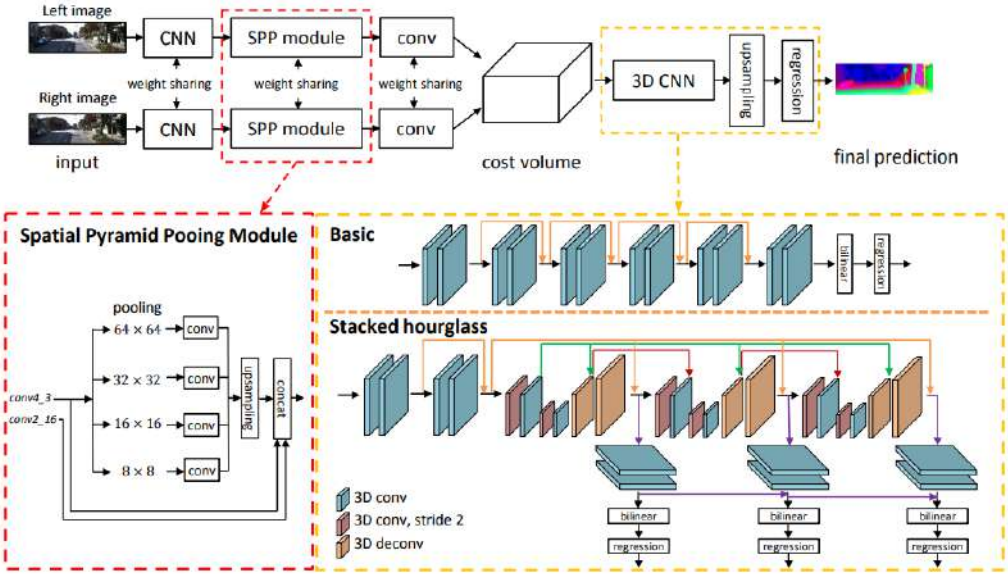

Figure 4.8: PSMNet architecture: The figure is taken from the paper [CC18]. The left and right images are directly fed to two separate CNN pipelines which share weights, this then outputs feature maps. A convolutional layer is used to fuse the features to form a 4D cost volume. This cost volume is further fed into a 3D CNN for regularization and regressing disparity.

formulation. This is mostly observed for regions that are further than 30m where the prediction from the depth network is blurry. Moreover, we have access to per-view sparse depth maps (LiDAR in case of KITTI-360 and partial point cloud for ShapeNet) with reliable depth information. In order to provide a stronger initialization, we optimize the radiance field f additionally by supervising the expected depth $\hat{\mathbf{z}}_\theta$. For a given input ray batch $\mathcal{R}$, we sample the subset of rays $\mathcal{R}_d \subset \mathcal{R}$ that have an associated depth value, i.e

$$\mathcal{R}_d = \{\, \mathbf{r} \,|\, \forall\, \mathbf{r} \in \mathcal{R},\, \mathbf{d}_{gt}(r) > 0 \,\}$$

Similar to [DLZR22] we use the mean squared error:

$$\mathcal{L}_{\text{depth}}(\theta, \mathcal{R}_d) = \sum_{r \in \mathcal{R}} \|\hat{\mathbf{z}}_\theta(\mathbf{r}) - \mathbf{d}_{gt}(\mathbf{r})\|^2 \tag{4.10}$$

Simultaneously applying (4.9) and (4.10) reinforces the network to learn a volumetric representation in tandem with respect to color and depth values.

4.5.2 Patch-Based Regularization

RegNeRF[NBM+22] propose a patch-based depth smoothness loss to regularize the geometry. The key to their regularization technique is the supervision of expected depth of rays sampled from a defined set of camera poses that are independent of the training poses. This acts as a self-supervised loss to enforce smoothness for unobserved viewpoints. However, we do not directly adapt their strategy because then the additional points queried from rays belonging to these defined camera poses will occupy the entire sampling interval along the bounds (t_n, t_f). This will deteriorate our training procedure because the points queried for the input views are restricted to lie within the bounds that we define close to the surface and with this strategy additional points belonging to free-space are also queried. Hence, we define an additional ray batch $\mathcal{R}_{\text{patch}}$ that samples random patches of size S_{patch} from input camera poses $P \in R^{M\times 3\times 4}$. Let r_{ij} be the ray through pixel (i, j) of a patch centered at $\mathbf{r}$, then the depth smoothness loss is defined as follows:

$$\mathcal{L}_{\text{DS}}(\theta, \mathcal{R}_{\text{patch}}) = \sum_{\mathbf{r}\in\mathcal{R}_{\text{patch}}} \sum_{i,j=1}^{S_{\text{patch}}-1} \left(\hat{\mathbf{z}}_\theta(\mathbf{r}_{ij}) - \hat{\mathbf{z}}_\theta(\mathbf{r}_{i+1j})\right)^2 + \left(\hat{\mathbf{z}}_\theta(\mathbf{r}_{ij}) - \hat{\mathbf{z}}_\theta(\mathbf{r}_{ij+1})\right)^2, \tag{4.11}$$

where $\hat{\mathbf{z}}_\theta$ is the expected depth.

4.5.3 Hard Loss

Despite querying points close to the surface, our method is still based on volume rendering and does not inherit any properties of surface rendering. When we consider scenes from KITTI-360 where the camera moves forward, the number of images containing overlapping sequences is very less (3-10 images per object). With insufficient views for each object in a scene, it is difficult for the network to enforce multi-view consistency on the geometry. Moreover, NeRF[MST+20] does not model hard transitions from free-space to the real geometry explicitly like other hybrid methods [OPG21, WLL+21]. Thus, we consider a regularization strategy proposed by [RMY+21] where they impose a prior on the probability of the integration weights $w(t)$ (defined in equation (3.4)) as a mixture of Laplacian distributions:

$$\mathbb{P}(w(t)) \propto e^{-|w(t)|} + e^{-|1-w(t)|}, \tag{4.12}$$

where one of the modes is around weight zero and the other around one. This leads to a peaky distribution that encourages a sparse solution where any solution in the interval (0, 1) is discouraged. For a given set of input camera rays $\mathcal{R}$, the prior is converted to a loss function as defined below:

$$\mathcal{L}_h(\theta, \mathcal{R}) = \sum_{r\in\mathcal{R}} -\log(\mathbb{P}(w(t)) \tag{4.13}$$

Final Objective Function

We ablate both the the regularizaton losses and use the depth smoothness loss as our final regularizer. Our optimization objective is given by:

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda_d \mathcal{L}_{\text{depth}} + \lambda_{ds} \mathcal{L}_{\text{DS}} \tag{4.14}$$

5 Datasets

5.1 ShapeNet

To conduct extensive experiments with ground-truth depth values we choose the ShapeNet [CFG+15]. ShapeNet dataset contains thousands of computer-aided models (CAD) of common objects that are water-tight and available in standard formats (e.g. .obj, .ply, .binvox). Like most of the 3D reconstruction methods [MON+19] , we choose the "chair" category because it is highly varied and many models contain high-frequency details. The "chair" category is further divided into multiple classes like arm chair, lawn chair, lounge chair, etc. We choose 10 chair models by picking one from each subclass and ensure that all the 10 chairs are unique with respect to their geometry. Figure 5.1 visualizes the 10 chair models that we use for evaluation. In Section 5.1.1 we discuss the details of rendering these chair models and 5.1.2 discusses the process of generating sparse depth for each camera pose.

Figure 5.1: 10 chair models used for evaluation.

5.1.1 Rendered Images

We use Blender to generate our own dataset that consists of 10 chair models. We render images of each object by sampling camera views on the upper hemisphere of the object. They are rendered at a resolution of 320×320 where 40 images are used for training and 10 randomly sampled images are used for testing.

5.1.2 Projected Depth Maps

To obtain incomplete point clouds for each model, we use the partial and complete training pairs generated by [YKH+18]. They generate ground-truth point cloud by uniformly sampling 16384 points from the surface of each mesh. To obtain points clouds that are "incomplete", they render depth maps for 8 viewpoints and block certain regions, i.e legs, armrest, headrest of the chairs. Next, they backproject these depth maps to obtain a point cloud that is both sparse and "incomplete". This partial point cloud is further downsampled to contain 2048 points per model. We use each of these partial point clouds as our sparse input depth.

To obtain per-view depth maps of the partial point cloud, we project the partial point cloud to each input camera view. However, the disadvantage is that, the sparse depth maps contain only 0.5% of the total pixels of the rendered chair. This makes each depth map very sparse and we cannot expect a fair prediction from the monocular depth network from the supervision of such a sparse depth map. Figure 5.2 demonstrates the *sparsity* and "incomplete" nature of the partial point cloud. Further, Figure 5.2 also shows an example of the projected depth map overlaid on the RGB image of the particular camera view.

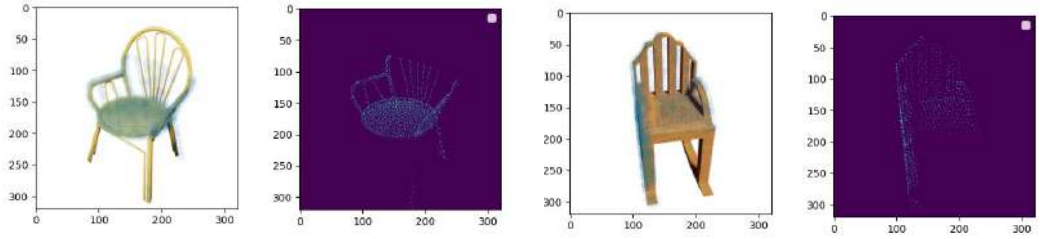

Figure 5.2: Left image contains the projected point cloud overlaid on the RGB image and the right one contains the partial point cloud to demonstrate its "incomplete" and sparse nature. Both chair models are missing two of their legs and half of the headrest.

5.2 KITTI-360

KITTI-360 [LXG21] is a large dataset of outdoor urban environments that was collected by mounting perspective cameras, fisheyed cameras, velodyne and SICK sensors and a GPS system. Relevant for us are the perspective cameras used to capture images, GPS system that captures the camera poses, velodyne and SICK sensors that scans the location with a time-of-flight concept to capture scans of the environment. The dataset is divided into 11 driving sequences corresponding to individual driving trajectories. For our experiments, we randomly choose scenes

Scene	Sequence	Frame Start	Frame End
1	2013_05_28_drive_0000_sync	250	310
2	2013_05_28_drive_0002_sync	4450	4510
3	2013_05_28_drive_0004_sync	3368	3428
4	2013_05_28_drive_0005_sync	4600	4660
5	2013_05_28_drive_0009_sync	12448	12508
6	2013_05_28_drive_0010_sync	43	103
7	2013_05_28_drive_0009_sync	12550	12610
8	2013_05_28_drive_0009_sync	12375	12435
9	2013_05_28_drive_0004_sync	2897	2957
10	2013_05_28_drive_0000_sync	149	209

Table 5.1: **Information of KITTI-360 dataset**. This table contains information of each scenes. We keep a standard limit of 60 frames and sample the start frame from each sequence. We keep dynamic sequences in mind and avoid those.

from drive sequences that do not contain dynamic objects. While sampling each scene, we consider the following factors:

- We choose scenes that contain wide streets where each image has a coverage of approximately 40m at a single glance. In total, each of our scene spans atleast 60-80m in length.
- We prioritise scenes containing multiple objects like: cars, trees, buildings, signboards, etc.
- We keep the lighting condition in mind and try to avoid scenes that are very dimly lit, i.e since the dataset is captured outdoors, it is dependent on sunlight. In some sequences when the weather is cloudy, each image is poorly lit and the texture information is barely visible. We have a single scene similar to this and most of the methods struggle on this.

Table 5.1 contains the drive sequences and its respective frames for each scene. We choose 60 frames from the start frame. For training, we drop alternative frames and each scene contains 30 frames for training. The other 30 frames are used for evaluation.

5.2.1 Accumulated Point Clouds

LiDAR operates in a continuous time-of-flight operation and image frames are not synchronised with respect to the point cloud, i.e a LiDAR scan of a particular camera pose contains information behind and in front of the camera. Thus to accumulate points between two sets of camera poses, we utilise the accumulation script provided by KITTI-360 [LXG21].

Thus, for training we accumulate only LiDAR scans belonging to the train camera poses. For evaluation we obtain a separate point cloud by accumulating scans belonging to LiDAR and 2 SICK scanners for all the camera poses (60 frames), to obtain a dense point cloud.

5.2.2 Projected Depth Maps

The next step is to obtain the depth map corresponding to each camera pose. Since each scene is very large and point clouds do not define a continuous surface, we can easily see through objects. This can be problematic when we project points to obtain a depth map. Hence, to perform an occlusion test, we convert the accumulated point clouds (train and test) to a mesh using Poisson Surface Reconstruction [KBH06].

After obtaining the respective meshes, we perform the following steps separately to train and test camera poses (with their respective accumulated point clouds) to obtain the final projected depth maps:

1. Project the accumulated point cloud to each camera view i, to obtain a sparse depth map $D^i_{proj} \in \mathbb{R}^{H \times W}$
2. Render the respective mesh at the same camera view to obtain depth map: $D^i_{mesh} \in \mathbb{R}^{H \times W}$
3. Now, in order to perform an occlusion test, we want to ensure that the depth values of the projected depth does not deviate from the depth values rendered from the mesh. This is because, mesh is solid 3D representation and the depth values rendered will not contain occluded objects.
4. We set a threshold of $\epsilon = 0.5$cm and compute the difference between the depth maps from 2. and 3., and consider only pixels from D^i_{proj} that pass this test.

$$|D^i_{mesh} - D^i_{proj}| < \epsilon$$

We repeat the above procedure to obtain the projected depth maps for all the camera views. Figure 5.3 visualizes the projected depth map overlaid on RGB image. Note that, the process of converting point clouds to mesh is non-trivial and the final mesh contains noisy artifacts at random regions and this results in a sparse projected depth map (e.g., notice the holes present in the road region of all the scenes in 5.3). Thus for each scene, we have 30 input images of resolution $H = 192, W = 704$, with their corresponding camera views and view-specific projected depth maps.

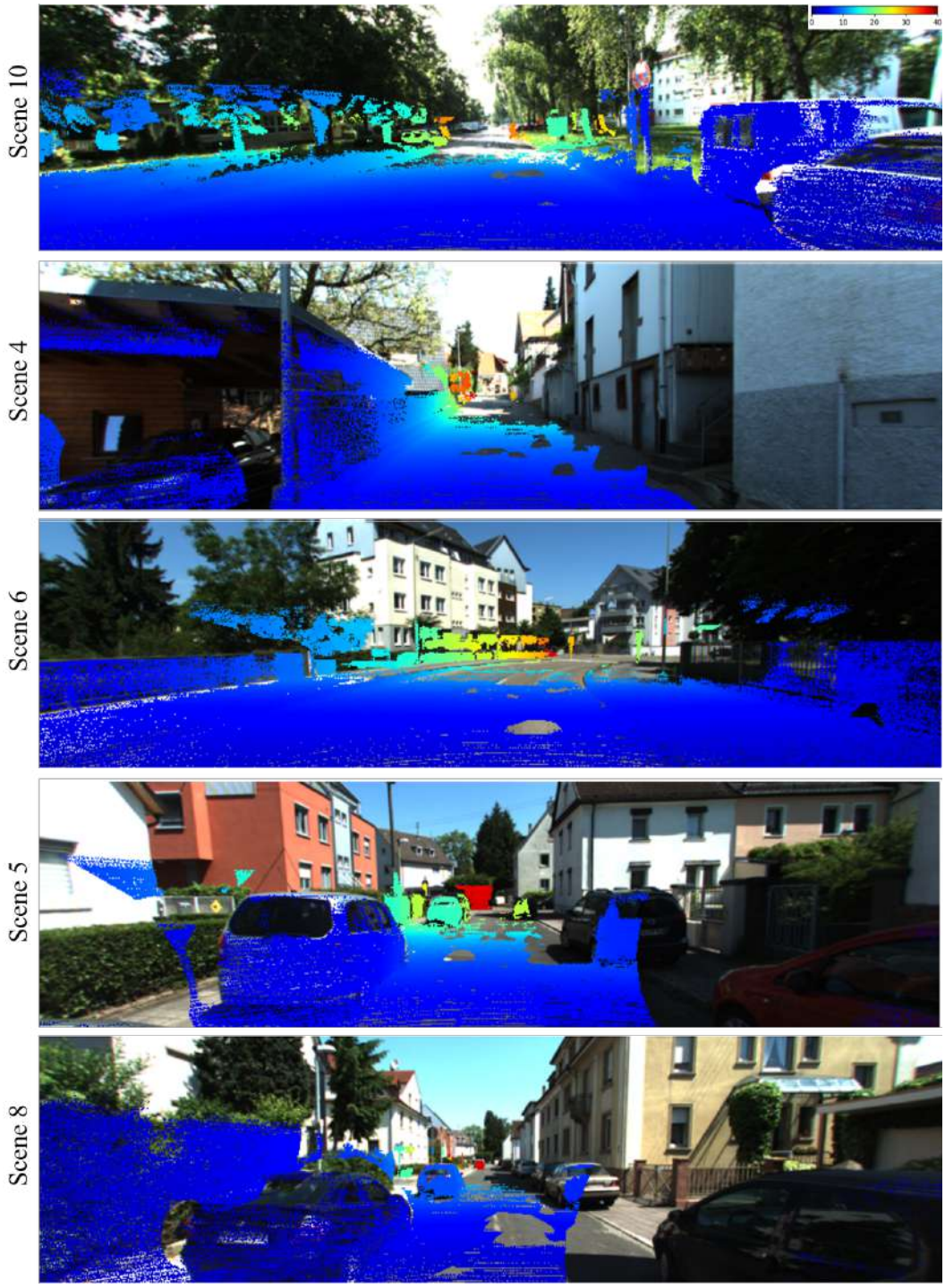

Figure 5.3: **Projected Depth Maps of KITTI-360**. This figure contains projected depth maps overlaid on RGB image of the respective camera view.

6 Experiments

6.1 Evaluation Protocol

In this section, we explain our evaluation process. First, we describe the evaluation metrics in Section 6.1.1 and then explain our protocol for extracting the point cloud from the radiance field in Section 6.1.3

6.1.1 Evalutation Metrics

We evaluate the predicted point clouds using Chamfer Distance and F-score following [MON+19, PFS+19, CZ18, FSG16]. They are defined as follows:

- **Chamfer Distance**:

$$d_{\mathrm{CD}}(S_{\mathrm{pred}}, S_{\mathrm{gt}}) = \underbrace{\sum_{x \in S_{\mathrm{pred}}} \min_{y \in S_{\mathrm{gt}}} \|x - y\|^2}_{\text{Completeness}} + \underbrace{\sum_{y \in S_{\mathrm{gt}}} \min_{x \in S_{\mathrm{pred}}} \|x - y\|^2}_{\text{Accuracy}}$$

  Chamfer Distance is the linear sum of Completeness and Accuracy, and we report both separately to analyse our results in detail. Completeness measures the distance between each predicted point compared to the ground-truth points and intuitively, a lower completeness score is obtained when predicted points lie very close to the ground truth, i.e this value increases when the point cloud is noisy or contains floating artifacts. Accuracy measures the distance between each ground-truth point in comparison to the predicted point cloud. When the ground-truth point cloud is incomplete (in case of KITTI-360), measuring the accuracy can be misleading because a high score does not necessarily indicate better geometry. Rather, a high score in this case can be obtained if the network predicts all the points in a cluster around the ground-truth. We observe this behaviour in the results section of KITTI-360 and analyse it in detail.

- **F-Score:**
  F-score was proposed by [TRR+19] and calculates the harmonic mean between precision (Accuracy) and recall (Completeness) which is an established metric in the community. Precision is measured by counting the percentage of predicted points that lie within a certain distance (Γ) to the ground-truth points. Recall is calculated by counting the percentage of ground-truth points that lie within a

certain distance (Γ) to the predicted points. The F-score is controlled by varying the distance value Γ. Following [TRR+19, MON+19], we set Γ = 0.1m for all the experiments. Intuitively, the metric measures the percentage of points reconstructed correctly.

6.1.2 Evaluation Dataset

In this section we describe the evaluation dataset (ground-truth) against which we compare the predicted point clouds.

- **ShapeNet**: We use the completed point cloud that contains 16384 points generated by [YKH+18].
- **KITTI-360**: We use the fused point cloud that contains points accumulated from one LiDAR sensor and two SICK sensors. Note that, we evaluate on test camera views and the projected LiDAR depth associated with the test camera views is not used in the training process. Thus, all the points in the ground-truth test point cloud are unseen. Additionally, we use pretrained weights for stereo priors that were trained on KITTI[GLU12, MG15], that contains information only from a single LiDAR sensor. Moreover, KITTI[GLU12, MG15] and KITTI-360[LXG21] dataset were collected separately.

6.1.3 3D Reconstruction Details

The underlying geometry can be extracted from the predicted expected depth values or the MLP network of NeRF can be evaluated to obtain density predictions. We explain the procedure for both methods and argue about our choice for evaluation.

Expected Depth

The expected depth is calculated in equation (3.7) for each camera pose. For a given ray origin o_i along a direction d_i with the expected depth value z_i, we calculate the corresponding 3D point as $\mathbf{x}_i = o_i + z_i d_i$. We iterate over all the relevant camera views and accumulate the corresponding 3D points of each test ray to obtain the predicted point cloud. This method ensures that we have the same number of points in the predicted point cloud of each baseline method because we use the same test rays for all the evaluations. This enables us to evaluate the baselines in a fair manner because Chamfer Distance does not restrict the number of points for the test and ground truth point clouds and it is possible to get a lower score (low is better) by providing fewer points [TRR+19]. Moreover, the radiance field is optimized to learn integration weights that predict depth and color estimates. Thus, for all our quantitative evaluations we use this method to extract the predicted point cloud.

Queried Points

Another method for obtaining the predicted point cloud is by directly querying the neural network for the density prediction σ_θ. The entire sampling space is divided into a voxelgrid, where a single point is queried per voxel and this grid of sampled points is evaluated using the neural network to obtain the density value of each point. Once the density prediction for each point is obtained, we can threshold the values to extract the geometry via Marching Cubes. Marching Cubes extracts isosurfaces from the voxelgrid containing density predictions.

For NerfingMVS and all our studies, we evaluate the MLP network explicitly with points that lie within the defined sampling interval concentrated around the depth priors. Thus, we want to ensure that the points queried for evaluating the MLP is already a seen point (i.e previously trained with the MLP network). After dividing the sampling space into a voxelgrid, we perform an additional check to see if the points sampled lie within the defined sampling interval and then evaluate the neural network with only these points.

We set a voxelgrid of size 0.005 for all the experiments, where the space between each point is 0.5 cm. We choose this value because the distance between each point in the fused point cloud of KITTI-360 is also 0.5 cm and this will enable a fair qualitative comparison. The threshold for the density field is set as 25 for all the evaluations.

Extracting point clouds with this method is challenging in comparison to the expected depth as the density field does not explicitly model hard surfaces. Querying points along the entire space also leads to floating artifacts. But, this procedure is advantageous for all our experiments because we explicitly restrict the sampling ranges to match the adapted depth priors and this helps in the prevention of floating artifacts. Moreover, for all our experiments, we repeatedly sample points close to the approximate surface. Thus, when we query points with this procedure, we obtain higher resolution point clouds with fine geometric details because the network was trained with points that were densely sampled close to the approximate surface.

6.2 Implementation details

The training protocol of methods using depth priors (NerfingMVS and Our experiments) is a two-step procedure. First, we train the Depth Network (Monocular/Stereo) on each scene separately to obtain the respective depth priors. Next, we use the depth priors to calculate the Uncertainty maps for each depth map. Finally, we use the depth priors and the Uncertainty maps to train the NeRF framework with ray-wise adaptive sampling ranges. In this section, we describe the training protocol

utilised to train NeRF and the Depth Networks separately in individual sections. We use the same training protocol of NeRF for all the experiments (baseline and ours).

6.2.1 Monocular Depth Network

We fine-tune the monocular depth network for 5 epochs for ShapeNet scenes due to the sparse nature of the projected depth maps. For KITTI-360, since the scenes have a higher resolution, we fine-tune the network for 15 epochs. We set the learning rate of all the experiments to 4×10^5 and use the Adam optimizer [KB14]. We use a batch size of 4 for ShapeNet experiments and 2 for KITTI-360.

6.2.2 Stereo Depth Network

The stereo network is used to train KITTI-360 and we fine-tune each scene for only 5 epochs. This is because, when we fine-tune the network for a longer duration, the network introduces random artifacts in regions without ground truth depth. Thus, it was a trade-off between over-fitting on the sparse depth map or ensure that the depth maps have low artifacts. Following [CC18] we set a learning rate of 1×10^{-3} and batch size of 4. We use the Adam optimizer [KB14] similar to the original paper and directly utilize their PyTorch implementation. The maximum disparity is set to 192 following [CC18].

6.2.3 Neural Radiance Fields

A separate model is used to train each scene and all baselines and a similar training protocol for all the scenes is implemented as follows. We sample 64 points along each ray with a batch size of 1024. For experiments where we ablate the bi-directional depth reprojection error (ShapeNet experiments), we do not adapt hierarchical sampling to ensure fair comparison with NerfingMVS. For KITTI-360 experiments, we adapt hierarchical sampling for all our experiments due to the complex nature of each scene. For the experiments with hierarchical sampling, we sample 48 points for the coarse network and 16 points for the fine network. This way, we ensure that the total points sampled are 64 with hierarchical sampling, for a fair comparison. The same network architecture described above is employed to evaluate fine and coarse networks. To regularize the network further, we follow NerfingMVS [WLR+21] and add random Gaussian noise with zero mean and unit variance to the density σ to train all our scenes and baselines. We follow NeRF [MST+20] and add their positional encoding with $L = 4$ for the direction, and $L = 10$ for the position. The Adam optimizer [KB14] is used with an initial learning rate of 5×10^{-4} and is decayed exponentially to 5×10^{-5}. Our implementation is based on the PyTorch implementation of NerfingMVS where some parts are borrowed from the PyTorch implementation of NeRF by [YC20]. In case of KITTI-360 scenes, the network is

trained for 300k iterations and for the ShapeNet chairs, it is trained for 200k iterations. We follow the same number of iterations for all baselines and ablation studies.

6.3 Object-Level Reconstruction

The goal of conducting experiments on ShapeNet is to compare the bi-directional reprojection error with the depth reprojection error of NerfingMVS. ShapeNet is a synthetic dataset with watertight meshes from which perfect ground-truth depth values can be rendered and this is advantegous as we can empirically quantify the Uncertainty maps. In this section, we describe the baseline methods that we consider for our experiments on ShapeNet. Additionally, we explain the evaluation protocol of the Uncertainty maps in this section as the metrics we use are intuitive and it is easier to interpret the results. Note that we perform studies on all the methods (including ours) using color supervision only, because the goal of these experiments is to explicitly study the effects of the Uncertainty map. Since NerfingMVS utilise only color supervision (4.9), we adapt their framework and only modify a single component for a fair comparison.

6.3.1 Baseline Methods

1. **NeRF**: We use the same training protocol described in 6.2.3. We train for 200k iterations following [MST$^+$20, WLR$^+$21] and (t_n, t_f) is set as (0,3) as the maximum depth of every chair from each camera view is 2m.

2. **NerfingMVS**: We use the protocol described in 6.2.1 to train the monocular depth network first and then train the NeRF-MLP network for 200k iterations. Each scene of the chair contains images rendered from the upper hemisphere of the object with a white background. The monocular depth network predicts a depth value for the entire image including the background, but the depth prediction of the background contains artifacts because of its textureless nature. Moreover, when we calculate the depth reprojection error of each input view, we get uneven depth predictions for the background. Since, NerfingMVS propose their framework for a real-world scene containing multiple objects in each frame with coherent background, for a fair evaluation we mask the background while calculating the depth reprojection error. We render per-view masks while we prepare the dataset (i.e directly from the mesh). The depth value of the background is set to zero.

3. **+bi-dir. reproj.**: To ensure fair comparison and ablate the improvements made by the bi-directional depth reprojection error (bi-dir. reproj.), we use the same network and training protocol used to train NerfingMVS as described above. Thus, the only difference is the use of bi-directional depth reprojection error to calculate the Uncertainty map.

6.3.2 Evaluation: Uncertainty Maps

One of our main contributions in this thesis is to measure uncertainty of depth estimates using the bi-directional depth reprojection error described in 4.3. We establish the consequences of underestimating the uncertainty of depth estimates in 4.2.3. In this section, we demonstrate qualitatively and quantitatively that Nerfing-MVS underestimates the uncertainty and further show that the bi-directional depth reprojection error yields better estimates for the uncertainty of the depth prior.

Figure 6.1 shows a visualization comparing the following: (1) NerfingMVS: Uncertainty map (2) +bi-dir reproj.: Uncertainty Map (3) Depth prediction from the Monocular Depth Network (4) Ground-truth depth. The boxes marked in red in 6.1 highlight regions where the depth prior prediction is significantly different from the ground truth depth, e.g. the depth prior prediction in red boxes of all three chairs is around 0.2 m from the camera, whereas GT depth is around 1.2-1.4m from the camera. For these regions NerfingMVS underestimates the uncertainty value (<0.2), whereas +bi-dir. reproj is able to a high uncertainty (around 1.0) which is adequate for these regions. However, in some cases, +bi-dir. reproj also overestimates the uncertainty and this is due to the formulation of +bi-dir. reproj where we consider the largest errors from each Uncertainty map before averaging the maps from all camera views (details in Section 3.3). It is interesting to note the regions marked in orange boxes, where the depth inaccuracies of the depth prior are very subtle. For these regions, +bi-dir reproj is able to detect these regions and assign a slightly higher uncertainty. This is because of the reprojection term in the backward direction which captures the inaccuracies occuring within the depth prior compared to all the other projected views. NerfingMVS is unable to detect these subtle regions and assigns a very low uncertainty (<0.2) since they calculate reprojection only in the forward direction. Moreover, +bi-dir reproj is designed such that overestimation of the uncertainty is acceptable, but underestimation is not. This is to ensure that the sampling interval is large enough to contain the true surface within, if not points queried from multiple camera views of the same 3D world point will not coincide. As a sanity check, we directly evaluate our sampling interval by computing the percentage of pixels for which the real depth lies outside the sampling interval. This evaluation is described in the next section.

Quantitative Evaluation

The goal of this evaluation is to estimate the following:

1. Estimate whether the predicted uncertainty faithfully captures the inaccuracies of the depth priors.
2. Evaluate if the surface depth lies inside the sampling interval and further estimate the width of the sampling interval.

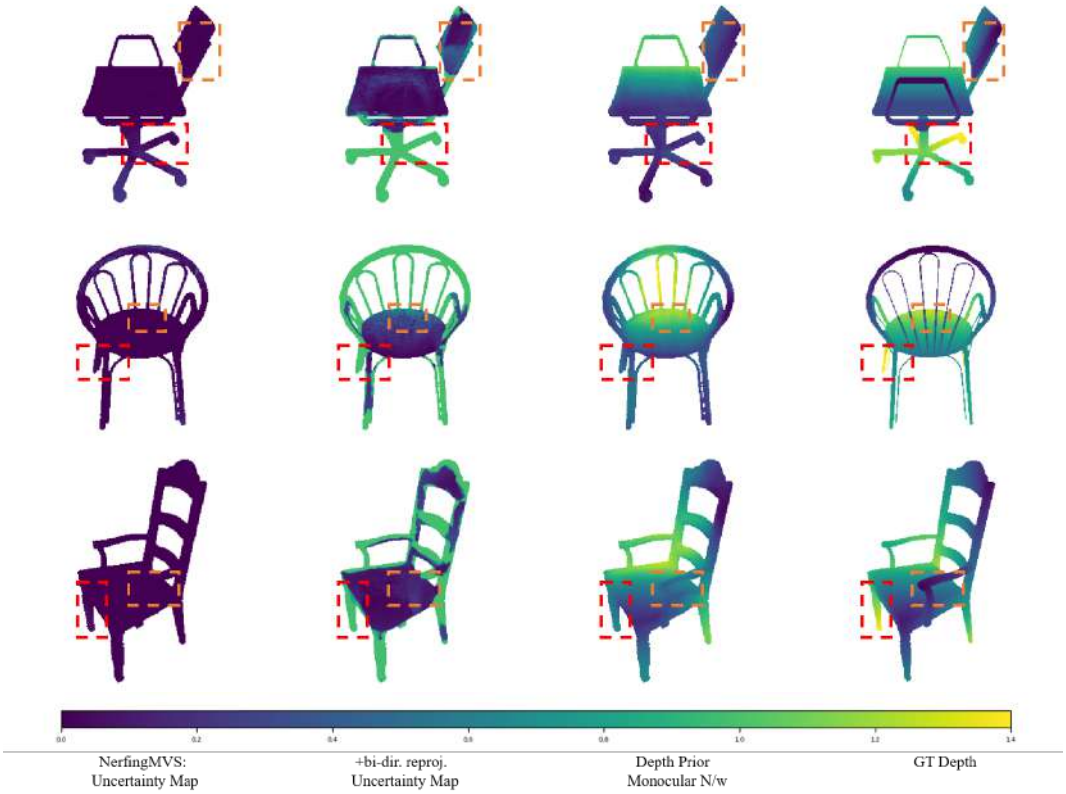

Figure 6.1: **Qualitative Comparison of Uncertainty Map on ShapeNet** This figure visualizes the uncertainty maps where purple color indicates a low uncertainty value (<0.2) and green (value of 1) indicates a higher uncertainty. Regions marked with red boxes shows the inaccurate depth prediction of the depth prior compared to GT depth; and the same red boxes marked in NerfingMVS shows that the uncertainty is underestimated (< 0.2), whereas +bi-dir. reproj is able to predict a high uncertainty (close to 1). It is challenging to adequately capture depth inconsistencies, hence we formulate the +bi-dir. reproj to ensure that the uncertainty is overestimated than underestimated. Since, overestimated uncertainty ensures that the real depth values are contained within the sampling interval while underestimating it will result in the real depth values lying outside the interval.

For the evaluation metric we consider depth estimation methods that additionally compute the uncertainty of each depth map [PATM20, GDS20, QLT21, IÇG+18]. For completeness, we describe each evaluation metric in detail below.

	Abs Rel		RMSE		$\mathcal{L}_1$	$\mathcal{L}_2$
Method	AUSE↓	AURG↑	AUSE↓	AURG↑	$\lvert\Delta_p - \Delta_r\rvert \geq 0$↓	$\lvert\Delta_p - \Delta_r\rvert < 0$↓
NerfingMVS	0.0419	0.0093	0.0618	0.0142	1.2584	1.3853
+ bi-dir. reproj.	**0.0316**	**0.0197**	**0.0485**	**0.0277**	**1.0697**	**0.9343**

Table 6.1: **Quantitative Comparison of Uncertainty Map on ShapeNet.** The results in this table are averaged over 10 chairs. AUSE and AURG are computed for two depth estimation error metrics defined in (6.1), (6.2). A lower AUSE can be obtained if the uncertainty map encodes the inaccuracies present in a depth map. AURG quantifies the difference between modeling the uncertainty map in comparison to no modeling. $\mathcal{L}_1$ and $\mathcal{L}_2$ quantifies the width of the sampling interval where the former penalizes the sampling interval linearly for the case where the real depth lies inside the sampling interval. While, the latter penalises the sampling interval exponential because the real depth lies outside the sampling interval.

1. Area Under the Sparsification Error (AUSE)

This metric was originally proposed to estimate the uncertainty of an optical flow model by [IÇG+18], but is widely adapted as a standard for estimating the uncertainty estimates for depth prediction as well [PATM20, GDS20, QLT21, IÇG+18]. To calculate AUSE, we require a depth estimation error metric and we choose the Root Mean Squared Error (RMSE) and Absolute Relative Error (Abs Rel) following [PATM20, GDS20, QLT21, IÇG+18]. They are defined as follows:
Absolute Relative Error (Abs Rel):

$$\frac{1}{|D^i|} \sum_{z' \in D^i, z_{gt} \in D^i_{gt}} \frac{| z' - z_{gt} |}{z_{gt}} \tag{6.1}$$

Root Mean Sqaured Error (RMSE):

$$\sqrt{\frac{1}{|D^i|} \sum_{z' \in D^i, z_{gt} \in D^i_{gt}} || z' - z_{gt} ||^2} \tag{6.2}$$

where z' indicates the depth value of the depth prior (prediction from the monocular depth value) for each pixel of depth prior $D^i \in \mathbb{R}^{H \times W}$ from view i. z_{gt} is the ground-truth depth value per pixel from $D^i_{gt} \in \mathbb{R}^{H \times W}$.

A lower AUSE score is obtained when the Uncertainty map is able to encode most of the inaccuracies present in the depth map as the area under the sparsification curve is lower. This is explained in detail below. We use the excellent implementation provided by [PATM20] and the steps to calculate this error metric are as follows:

1. The first step is to sort all the pixels of the Uncertainty map in the descending order of uncertainty.

2. A subset of pixels (2% in our experiments following [PATM20]) are extracted and the error is computed following (6.1), (6.2). Similarly, all the pixels are extracted and their corresponding error values are calculated. Recall that the pixels removed here are arranged according to the *descending order of uncertainty*.

3. An ideal sparsification curve (*oracle*) is obtained by removing the pixels (using the same procedure as 2.), in the *descending order of the error*.

To get a single number that quantifies the error, the difference between both the curves is calculated and the area under that curve (Area Under the Sparsification Error: **AUSE**) is estimated. If the uncertainty encodes the errors correctly, the estimated sparsification must be very close to the oracle. Thus, a lower AUSE implies that both the curves are very close to each other and that the uncertainty encodes most of the inaccuracies of the depth prior. The Area Under the Random Gain (**AURG**) is an additional metric used by [PATM20, GDS20, QLT21] to quantify how much better the uncertainty term is in comparison to no modeling at all. A random uncertainty is modeled as a constant giving no information regarding the error, thus, a flat curve. AURG can be modeled by subtracting this flat curve with the estimated sparsification.

Table 6.1 shows the evaluation of NerfingMVS and +bi-dir reproj on AUSE and AURG for both error metrics (6.1), (6.2). We can see that our method has a lower AUSE score for both the error metrics (6.1), (6.2) because +bi-dir reproj is able to detect most of the inaccuracies present in the depth prior, validating the qualitative evaluation. NerfingMVS has a higher difference between AUSE-RMSE and AUSE-Abs Rel indicating that there are outlier values where the encoded uncertainty is very far from the oracle. This is because of the regions indicated in red boxes in Figure 6.1 where the estimated uncertainty is underestimated (<0.2) for a large depth error (difference between the depth prior and GT depth is approximately 1m) where, the uncertainty predicted is very low for a high error. Since, +bi-dir reproj. does not underestimate the uncertainty, the difference between AUSE-RMSE and AUSE-Abs Rel is not that high in comparison to NerfingMVS.

A higher AURG score implies better modeling of the uncertainty map in comparison to no modeling at all. +bi-dir. reproj. has a higher value compared to NerfingMVS indicating that the uncertainty modeled by +bi-dir. reproj is not random, and contains meaningful information regarding the errors present in the depth prior. We outperform NerfingMVS in both the evaluation metrics and this valides our qualitative evaluation demonstrating that NerfingMVS underestimates the uncertainty.

2. Evaluating the bounds

Firstly, as a sanity check, we compute the percentage of pixels for which the real depth lies outside the sampling interval. For the ground-truth depth from the i^{th} camera

Method	$z_{gt} < [t_n, t_f] \downarrow$ (%)										
	01	02	03	04	05	06	07	08	09	10	Avg.
NerfingMVS	33.75	40.88	35.23	38.65	35.19	34.58	34.12	47.33	43.75	61.58	40.50
+bi-dir. reproj.	0.27	1.43	1.542	1.91	1.27	1.38	1.91	2.61	0.239	1.95	1.45

Table 6.2: **Quantitative Comparison of Uncertainty Map on ShapeNet**. Percentage of samples where the real depth samples lies outside the defined sampling interval. We evaluate this for each chair and on an average 40.50% of the queried points do not contain the real depth values for NerfingMVS, whereas, with our method this number drops to an average of 1.45%.

view $D^i_{gt} \in \mathbb{R}^{H \times W}$, where (T^i_n, T^i_f) are derived from their respective Uncertainty map, we compute per-pixel bounds, i.e $\forall z_{gt} \in D^i_{gt}, (t_n, t_f) \in (T^i_n, T^i_f)$. Next, we simply count the number of pixels for which the real depth is not contained within the sampling interval: $z_{gt} < [t_n, t_f]$. This is evaluated for all the input views for each chair and given in Table 6.2. The percentages are averaged across all the input views. For NerfingMVS, an average of 40% of the queried points do *not* contain the real depth whereas using the bi-directional reprojection error reduces this number to an average of 1.45% for 10 objects. This table shows that the direct consequence of underestimating the error is that the real depth lies outside the sampling interval. This affects the entire learning process and the network learns to predict a point cloud with multiple surface parts that do not form a single surface in 3D (discussed in point cloud evaluation).

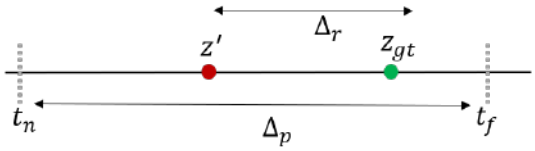

Figure 6.2: Let z' be the depth value of single pixel of the depth prior and z_{gt} be the ground-truth depth of respective pixel from the ground truth depth map. Δ_p is the width of the sampling interval set according to the uncertainty maps and Δ_r is the real distance between the depth prior and the ground-truth depth.

We propose a metric combining $\mathcal{L}_1$ and $\mathcal{L}_2$ norms to quantify the width of the sampling interval, i.e to estimate whether the sampling bounds are wide/tight. Consider Figure 6.2, where the sampling interval (t_n, t_f) is calculated using the respective Uncertainty map described in 4.3.1 for ray $\mathbf{r}$. The true distance Δ_r defines the absolute distance between the ground-truth depth z_{gt} and the depth prior at that pixel z' for ray $\mathbf{r}$. The predicted distance Δ_p is the absolute distance between the nearest sampled point t_n and the farthest point t_f, i.e Δ_p is the width of the sampling

interval for the same ray. The loss is defined for a single ray $\mathbf{r}$ as follows:

$$\mathcal{L}_{\text{bds}} = \begin{cases} \mathcal{L}_1(\Delta_p, \Delta_r), & |\Delta_p - \Delta_r| \geq 0 \\ \mathcal{L}_2(\Delta_p, \Delta_r), & |\Delta_p - \Delta_r| < 0 \end{cases} \tag{6.3}$$

Analysis of $\mathcal{L}_1$:

$|\Delta_p - \Delta_r| \geq 0$ implies that the sampling interval is large enough to contain Δ_r within and can tell if the real depth lies inside the sampling interval. In case it does, we penalize the distances linearly because the uncertainty map has encoded the depth inaccuracies of the prior correctly and as a result the sampling interval is wide enough to contain the real depth values within. Table 6.1 contains the results of evaluating on $\mathcal{L}_1$. Since the metric directly evaluates distances, the difference between our method and NerfingMVS is not very high when we linearly penalize the bounds because +bi-dir. reproj. overestimates the error and this will simply lead to larger distance values. When the error is overestimated, even though the bounds are not tight, it ensures that the real depth lies inside the sampling interval. Thus, while training NeRF with these adapted sampling ranges, the network can learn a consistent geometry.

Analysis of $\mathcal{L}_2$:

But for the second case, when $\Delta_r > \Delta_p$, it implies that the sampling interval is too tight and does not contain the real depth within. In this case, we penalize these distances exponentially because it implies that the sampling intervals belonging to different camera views of the same 3D world point does not coincide. When NeRF is trained on these queried points, the network will learn to predict density values tailored to each camera view, independent of the original surface. From Table 6.1, the results of $\mathcal{L}_2$ shows that NerfingMVS has a higher distance value compared to +bi-dir. reproj. because NerfingMVS has a higher percentage of samples (refer to 6.2) that lie outside the sampling interval and, thus, $\mathcal{L}_2$ is higher for NerfingMVS. Thus, this metric gives a higher loss value to all the depth estimates that lie outside the interval while simultaneously penalizing the width of the bounds.

6.3.3 Ablation: Hyperparameters

In this section, we first show ablation studies on hyperparameters that directly impact the width of the sampling interval (Multiplicative factor α) and the constant K, that decides the *number* of Uncertainty maps that are averaged to obtain the final map for each view. The impact of these hyperparameters are described in detail in Section 4.3.1.

		Abs Rel		RMSE		$\mathcal{L}_1$	$\mathcal{L}_2$
k	α	AUSE↓	AURG↑	AUSE↓	AURG↑	$\lvert\Delta_p - \Delta_r\rvert \geq 0$↓	$\lvert\Delta_p - \Delta_r\rvert < 0$↓
4	1	**0.0316**	**0.0197**	**0.0485**	**0.0277**	**1.0697**	**0.9343**
	2	0.0322	0.0190	0.0506	0.0176	1.0984	1.4825
8	1	0.0325	0.0106	0.0496	0.0191	1.0602	1.1197
	2	0.0396	0.0116	0.0488	0.0201	1.1531	2.0452
12	1	0.0326	0.0183	0.0515	0.0260	1.0861	2.5751
	2	0.0325	0.0197	0.0503	0.0270	1.1750	2.5012
16	1	0.0342	0.0158	0.0493	0.0273	1.8321	2.1655
	2	0.0386	0.0144	0.0492	0.0261	1.992	2.1122

Table 6.3: **Ablation Study on ShapeNet**. In this table, we ablate the multiplicative factor α of the sampling interval described in Section 4.3.1. The results in this table are averaged over all 10 chairs. We set $\alpha = 1$ for all our experiments on ShapeNet. We ablate the number of uncertainty maps k, and set $k = 4$ for all the experiments.

Multiplicative factor

The hyperparameter α, decides the scale of the sampling interval based on the uncertainty for a certain depth estimate. We conducted all our experiments on ShapeNet with $\alpha = 1$ after performing an ablation reported in Table 6.3. The scale of each chair is small and the maximum depth of each chair from the camera is 3m. Hence, this experiment does not require a large sampling width and $\alpha = 1$ is sufficient to contain the object within.

Number of Uncertainty Maps

The number of Uncertainty maps that are averaged to get the final map is decided by the hyperparameter k. Using a k that is large degrades the performance, as evidence by $k = 16$ from Table 6.3. In case of large k the number of Uncertainty maps that are averaged increases and due to a combination of low and large uncertainty values (distributed across k), the final uncertainty prediction is not high enough. We set $K = 4$ for all the experiments (NerfingMVS and our studies). From the table, we notice a gradual increase in AUSE as k increases implying that the uncertainty maps are not encoding the inaccuracies of the depth prior.

6.3.4 Evaluation: 3D Reconstruction

The point clouds that we evaluate in this section are extracted from test views, i.e we query points in a voxel grid of voxel size 0.005, and then consider only the points that lie within the defined sampling interval of the specific camera view as described in Section 6.1.3. We compare 3D reconstruction results from NerfingMVS, +bi-dir. reproj.

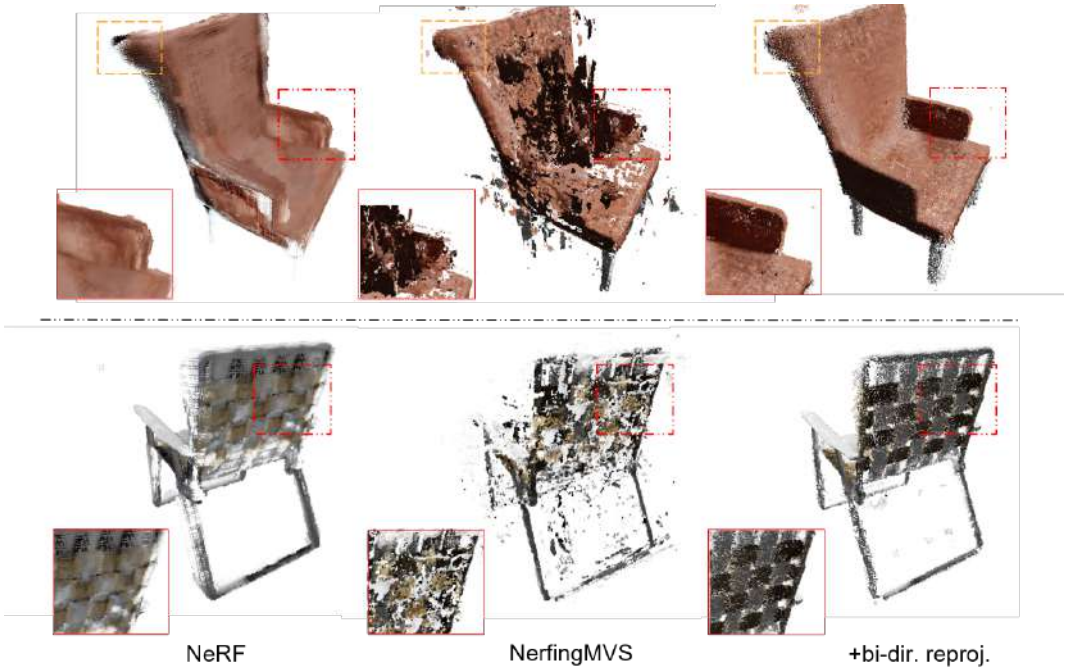

Figure 6.3: **Qualitative Comparison of Reconstruction on ShapeNet**. This figure demonstrates point clouds extracted by querying the network described in Section 6.1.3. For NerfingMVS and +bi-dir. reproj we use the test views to get the bounds that satisfy each queried point. All three methods are queried at a voxel size of 0.005 (resolution of 0.5cm) and at a threshold of 10.

and NeRF aswell. We consider NeRF as an additional baseline because NerfingMVS extends NeRF and increases the efficiency of sampling of the NeRF framework. Hence, NerfingMVS and +bi-dir. reproj. must produce better reconstruction in comparison to NeRF. We consider sparse number of input views for each scene, i.e 40 images, whereas NeRF is trained on 100 input images to produce high quality results. The reason we consider sparse input views is because we study these methods on KITTI-360 in the next section, where the number of overlapping view containing a certain object is very less (around 8-20). Hence, we create a similar setting in this evaluation to study the effects of bi-dir. reproj error in comparison to NerfingMVS and understand the importance of the Uncertainty map. Further, we analyse the effectiveness of the sampling strategy on the entire NeRF framework.

Effectiveness of bi-Directional Depth Reprojection Error

Table 6.4 contains the quantitative results of the reconstructed point cloud evaluated on Chamfer-L1, Accuracy and F-score and Figure 6.3 contains qualitative

Method	Chamfer-L1 ↓	Accuracy↓	F1↑
NeRF	0.0082	0.0157	0.8661
NerfingMVS	0.0126	0.0248	0.8715
+bi-dir. reproj.	**0.0078**	**0.0149**	**0.8802**

Table 6.4: **Quantitative Comparison on ShapeNet**. Averaged on 10 scenes from the ShapeNet dataset. From this evaluation, we quantify the effect of introducing the bi-directional depth reprojection error and additionally study the improvements made by the sampling strategy over NeRF.

comparisons. From the qualitative results, it is evident that NerfingMVS produces a point cloud that is unable to reconstruct the global structure of the object. From Figure 6.3, the top chair has a hole in the seating area and the backside of the bottom chair is noisy. This is because, the Uncertainty map of NerfingMVS underestimates the error as evidenced by our evaluation in Section 6.3.2. As a result the network learns to predict multiple surface parts that is view dependent and does not form a single coherent surface in 3D. To elaborate this further, we consider an additional example in Figure 6.4. The left column contains the extracted point cloud while the right column has points clouds that are obtained by backprojecting the expected depth from three different camera views. We rotate the reprojected point clouds to face forward to illustrate potential inconsistencies and validate our hypothesis. The left back rod of the chair (marked in red box), is reconstructed consistently by +bi-dir. reproj. in the extracted point cloud and the reprojected point cloud of different camera views. But, NerfingMVS does not reconstruct the left back rod in view 1. In view 2 the backrod is reconstructed, but the middle groves are noisy. Thus, due to underestimation of the error, the network learns a higher density prediction that is view-dependent (e.g. view 1, view 2) but does not form a consistent 3D representation. The noisy in the backrest of NerfingMVS's extracted point cloud is because of the incorrect density prediction for each training view that is accumulated. Since, +bi-dir. reproj is able to capture the depth inaccuracies, the model is able to learn coherent geometry that is consistent with respect to all the camera views. This further reinforces the importance of the role played by the uncertainty map for the entire learning process.

Effectiveness of Sampling Strategy

Qualitatively, from Figure 6.3, the point cloud reconstructed by NeRF learns the global shape of the chair. This validated by the quantitative results where the Accuracy of NeRF (0.0157) is higher than NerfingMVS (0.0248) implying that the ground truth points are closer to the predicted points. But the point clouds produced by NerfingMVS and +bi-dir. reproj is denser and can reconstruct sharp edges. For instance, consider the square boxes marked in orange color in Figure 6.3 (top). NerfingMVS reconstructs this region correctly and here we notice that the curved

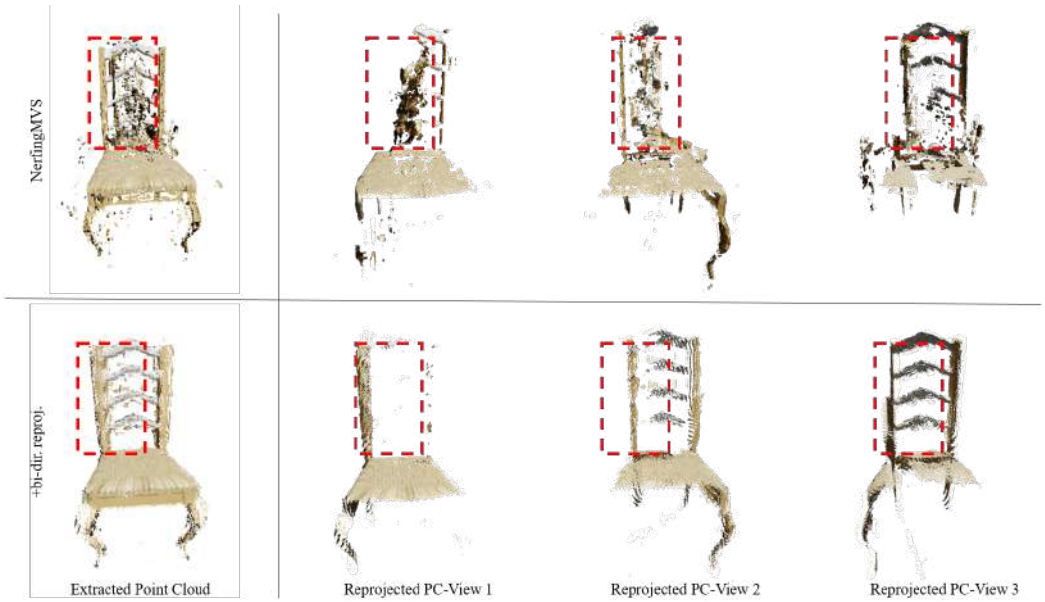

Figure 6.4: **Qualitative comparison on ShapeNet:** This figure demonstrates point clouds extracted from NerfingMVS and +bi-dir. reproj. The point cloud on the left demonstrates the extracted point cloud by querying points and evaluating the MLP network (Section 6.1.3). The right column contains points clouds obtained by reprojecting expected depth from three different camera views. The reprojected point clouds are rotated to a single camera view facing forward.

end of the chair's backrest is captured with high detail in comparison to NeRF and even +bi-dir. reproj. This is because of the sampling strategy where points are densely sampled very close to the depth prior. As a consequence, the number of points queried within a very small interval is large and as a result the network is able to capture local details with high accuracy. In other words, the resolution of the predicted point cloud is high. In comparison, NeRF produces a point cloud with a blunt edge where the circular part of the chair (orange sqaure) is not visible. +bi-dir. reproj is able to capture the small curve of the top of the chair similar to NerfingMVS. However, NerfingMVS produces noisy point clouds that does not adhere to the global shape of the object. In conclusion, the bi-directional depth reprojection error is advantageous and our solution combines the desirable properties of NerfingMVS (dense sampling) while ensuring the network learns a 3D consistent representation. The output point cloud that we obtain has sharp boundaries and is able to capture fine details in each chair. Notice the second chair in Figure 6.3, where +bi-dir reproj is able to capture the small squared patterns of the chair's back while NeRF produces a blurry backside. Our quantitative evaluation corroborates the qualitative observations where +bi-dir. reproj outperforms NerfingMVS by a large margin for both Chamfer-L1 and Accuracy.

6.4 Scene-level Reconstruction

In this section, we first describe all the baseline methods that we compare with respect to our experiments on KITTI-360. Next section discusses our ablation studies and contains all the experiments that we investigate. Finally, we demonstrate results in comparison to the baseline methods.

6.4.1 Baseline Methods

1. **Monocular and Stereo Depth Network:** This is an additional baseline that we verify because we directly use their prediction as a prior to our experiments. Hence, our experiments must improve in comparison to these two baselines. To get a point cloud for the evaluation, we backproject the depth maps and accumulate points for the test camera view using the procedure described in Section 6.1.3.

2. **NeRF:** We use the training protocol described in section 6.2.3. We use hierarchical sampling for training NeRF, whereas NerfingMVS do not use hierarchical sampling to report their metrics on NeRF in their paper.

3. **NeRF+Depth:** In this baseline, we use the NeRF framework and training protocol described in 6.2.3, but additionally supervise the MLP network with depth values obtained from LiDAR. We consider this baseline to investigate the effect of depth supervision and further evaluate the geometry in comparison

to using depth priors. This is based on DS-NeRF [DLZR22], but modified to accept LiDAR input instead of depth points from COLMAP. [DLZR22] also supervise their network with a weighted depth loss, but we use mean squared error without any weights. Hence, we do not refer to this baseline as DS-NeRF, as we do not incorporate all their contributions.

4. **NerfingMVS:**We use the same training protocol described in 6.2.3 without any modifications to the original framework.

Our Experiments:

1. **+Mono:** This baseline contains NerfingMVS's framework with bi-directional depth reprojection error to measure the uncertainty. We indicate it as +Mono implying the depth priors we use are from the monocular depth network.
2. **+Stereo:** In this baseline, we replace the monocular depth network with a stereo depth network and follow the training protocol described in 6.2.3. This baseline contains two extensions from NerfingMVS: (1) Bi-directional depth reprojection error (2) Stereo Depth Network.
3. **+Stereo-DS:** We build on the baseline described above and additionally incorporate depth supervision discussed in Section 4.5.
4. **+Stereo-DS-Reg:** This is our final method that contains all the three changes that we investigate in this thesis: (1) Bi-direction depth reprojection error (2) Stereo depth network (3) Depth smoothness regularization.

6.4.2 Ablation Studies

We evaluate our method on 10 scenes from KITTI-360, where each scene spans 60-80 m. Point clouds are extracted by querying points at a resolution of 0.5 cm and the neural network is evaluated to obtain density values (details in section 6.1.3).

We first ablate the two depth priors (Monocular and Stereo) by comparing the results of the network trained with both priors. Next, we compare the optimization objective and discuss their benefits individually. All the evaluations in this section are trained with bi-directional depth projection error.

Ablation of Depth Priors

Stereo depth acts as a powerful prior by capturing thin structures mainly because depth is estimated via disparity, where the stereo network is designed to find matching features between corresponding image pairs to estimate the horizontal displacement between pairs of pixels. Moreover, we utilise pretrained weights (pretrained on [MG15, GLU12]) that contains rich prior information of the 3D space

Depth Prior	$\mathcal{L}_{MSE}$	$\mathcal{L}_{\text{depth}}$	$\mathcal{L}_h$	$\mathcal{L}_{\text{DS}}$	Chamfer-L1 ↓	Completeness ↓	Accuracy ↓	F1 ↑
Mono	✓				0.0181	0.0051	0.0311	0.8260
Stereo	✓				0.0339	0.0020	0.0657	0.8400
	✓	✓			0.0264	**0.0008**	0.0520	0.8544
	✓	✓	✓		**0.0158**	0.0030	**0.0286**	0.8606
	✓	✓		✓	0.0166	0.0026	0.0302	**0.8889**

Table 6.5: **Ablation Study on KITTI-360**. This table compares individual optimization objectives (row 2-4) that we study and compares monocular prior with stereo priors (row 1-2). The column with $\mathcal{L}_{depth}$ indicates Depth Supervision (-DS). $\mathcal{L}_h$ and $\mathcal{L}_{DS}$ are two regularizers (-Reg) we ablate individually.

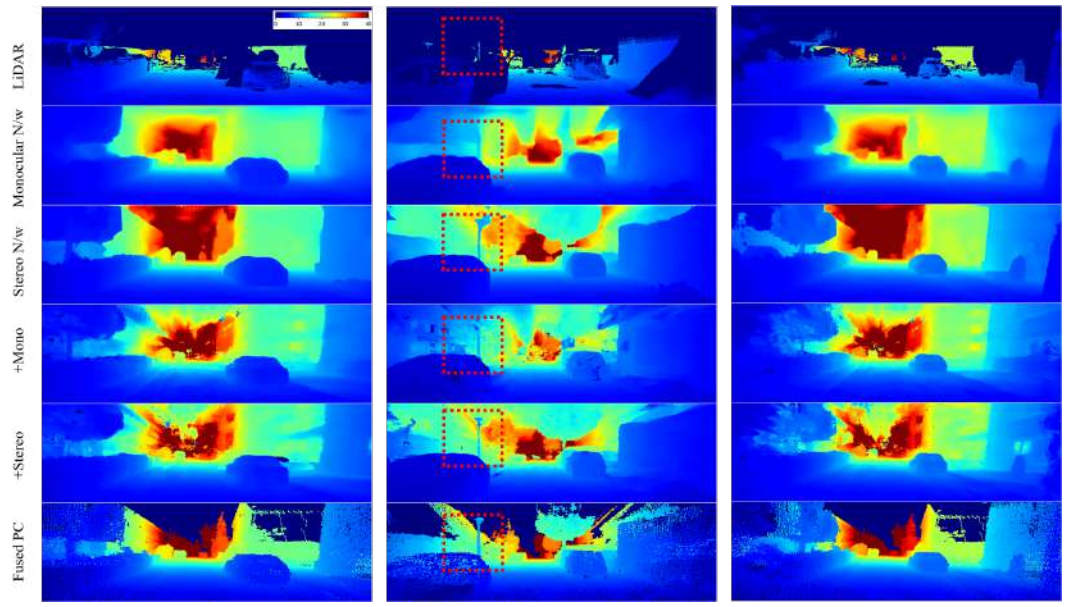

Figure 6.5: **Qualitative Comparison on KITTI-360**. This figure compares the two depth priors (Monocular and Stereo) predicted from the depth network to demonstrate that stereo depth acts as a stronger depth prior and the network trained with stereo priors is able to predict depth maps with sharp boundaries and capture thin structures. This visualization contains the expected depth predicted from +Mono and +Stereo of Scene 3 and 1 from test camera view. The depth map is visualized up to 40 m.

of urban environments. For the monocular depth network, following [WLR+21] we initialize it with the weights of mannequin challenge which contains scenes of dynamic people on streets. Since, the stereo network is trained on [GLU12, MG15], it contains prior information that is directly relevant to our scene. Moreover, the stereo network contains a feature-matching module that helps in capturing local details.

An important aspect to consider, is the scale of the monocular depth prior. The

Method	Chamfer-L1 ↓	Completeness ↓	Accuracy ↓	F1 ↑
+Stereo	0.0339	0.0020	0.0657	0.8400
+Depth	0.0325	0.0037	0.0614	0.8403
+Both	**0.0264**	**0.0008**	**0.0520**	**0.8544**

Table 6.6: **Ablation Study on KITTI-360**. This table compares the performance of networks trained with depth prior only (+Stereo), depth supervision only (+Depth) and finally network trained with both depth prior and depth supervision (+Both). This table contains results from Table 6.5 and 6.7. But we made a separate table with these specific results for the sake of clarity and ease of reference.

monocular depth network is trained with the scale-invariant loss to retain the global scale of the each depth map. But, we have a dataset of a single scene with multi-view images and this implies that the relative scale of depth for each camera pose must correspond to the other. To further calibrate the relative scale of the monocular prior with respect to LiDAR depth of each camera view, we follow [WLR+21, LDC+19] and obtain a scale factor. The details of computing the scale factor is provided in Section 3.2.1. Since each depth map is manually scaled by the scale factor, it leads to an ambiguity in the prediction of individual objects within each scene and decreases the quality of the depth prediction (evident from Figure 6.5). With stereo depth, adjusting the global scale will not be necessary as we train with $\mathcal{L}_1$ loss and depth map of each camera view is directly supervised by LiDAR depth values. This is advantageous because the quality of the depth prediction is retained.

Figure 6.5 compares the depth map prediction of the Monocular and Stereo Depth Network and the expected depth maps predicted by +Monocular and +Stereo. From the results, we can see that the network that utilizes stereo priors has better performance compared to the network that uses the monocular prior. For e.g. if we consider the middle row of Fig 6.5 marked in a square, the entire pole is not part of the LiDAR prediction, but the stereo depth manages to capture it. The network trained on stereo priors densely samples points close to the pole and is able to learn its geometry. Thus, this is one of the reasons we obtain better reconstruction results compared to our baseline methods because the stereo depth acts as a strong prior that aids the entire learning process.

Comparison of Depth Input (Prior + Supervision)

We compare incorporating only depth supervision (+Depth), only depth prior (+Stereo) and finally combining depth supervision and depth prior in the same framework (+Both). The quantitative results can be found in Table 6.6. Quantitative evaluation shows that +Both outperforms +Depth and +Stereo, where +Stereo has a better score compared to +Depth. Qualitative results are in Figure 6.6. We can

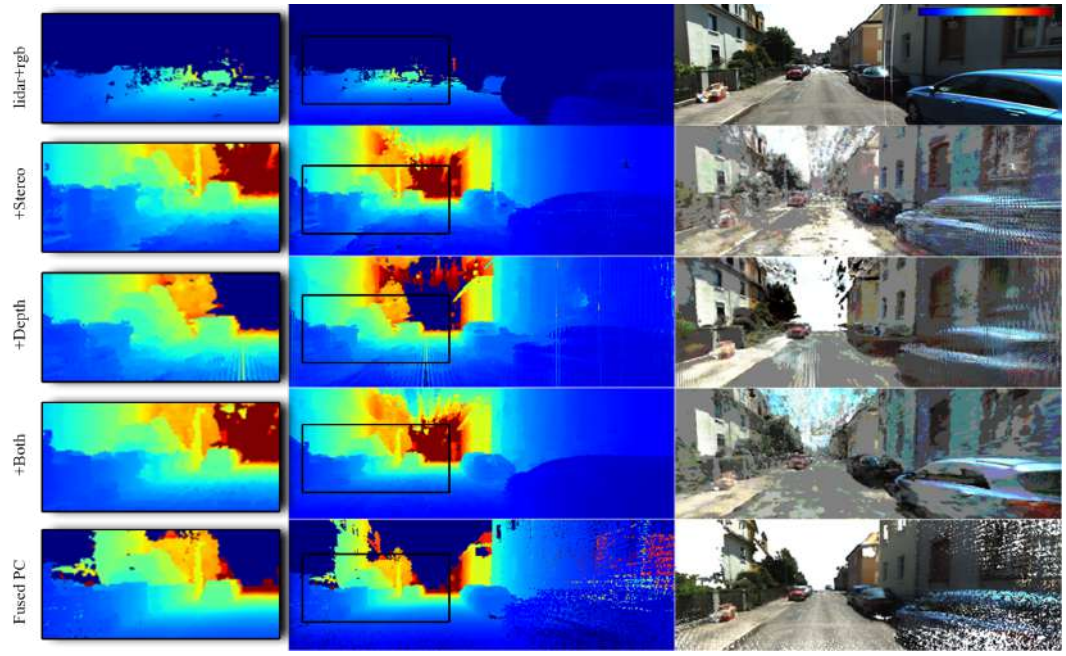

Figure 6.6: **Qualitative Comparison on KITTI-360**. This figure compares qualitative results when (a) +Depth: Only Depth supervision is added to NeRF framework (b) +Stereo: Stereo priors are added to NerfingMVS's framework (c) +Both: Both depth prior and depth supervision is incorporated to the framework. Left side: depth visualization of point cloud up to 40m, Right side: colored point cloud. This result is taken from scene 7.

notice that the stereo prior is able to detect thin structures (e.g. the pole, gate on the left) of the scene that is not captured by the LiDAR scanner. This is because of the pretrained stereo network [CC18] that is trained on [GLU12, MG15] and has encoded rich prior information of the 3D space very similar to our scene. This is advantageous for +Stereo framework because the role of the prior is to detect these distinct regions even if there is a slight compromise in accuracy of depth estimates.

+Depth improves the NeRF framework by incorporating sparse point cloud information in the form of depth supervision. This enables NeRF to learn better geometry in regions with depth information. In other words, for all the pixels with depth information the integration weights of queried points are encouraged to learn a peaky distribution at the depth value. For instance, from Figure 6.6 we can see that the road region of +Depth is smoother in terms of depth prediction despite the hole in the geometry. But, it is unable to detect thin regions (e.g. pole) when they are not present in lidar depth. +Stereo has a different sampling strategy where queried points are densely sampled close to the depth prior. To elaborate this, for all pixels with low uncertainty values, the queried points contain fewer free-space

samples and most of the sampled points are close to the depth prior values. This is advantageous because the neural network is seldom evaluated with free-space points. When combined, +Both is able to perform better because in addition to efficient sampling, depth supervision improves the geometry of queried points. It is likely that the regions containing sparse point cloud information has a lower uncertainty value (as we fine-tune the depth network with sparse point cloud). This implies that for these regions, the queried points contain less free-space samples and in addition the integration weights are directly supervised with the depth values. Therefore, we notice an increase in the sharpness of object boundaries in +Both, specifically in regions containing sparse point cloud information. Therefore, +Both inherits the advantages characteristics of both +Stereo (detect thin structures) and +Depth (refined object boundaries).

Ablation of Regularization Function

In this section, we study the effects of two different regularizers individually and state their advantages with our adapted framework.

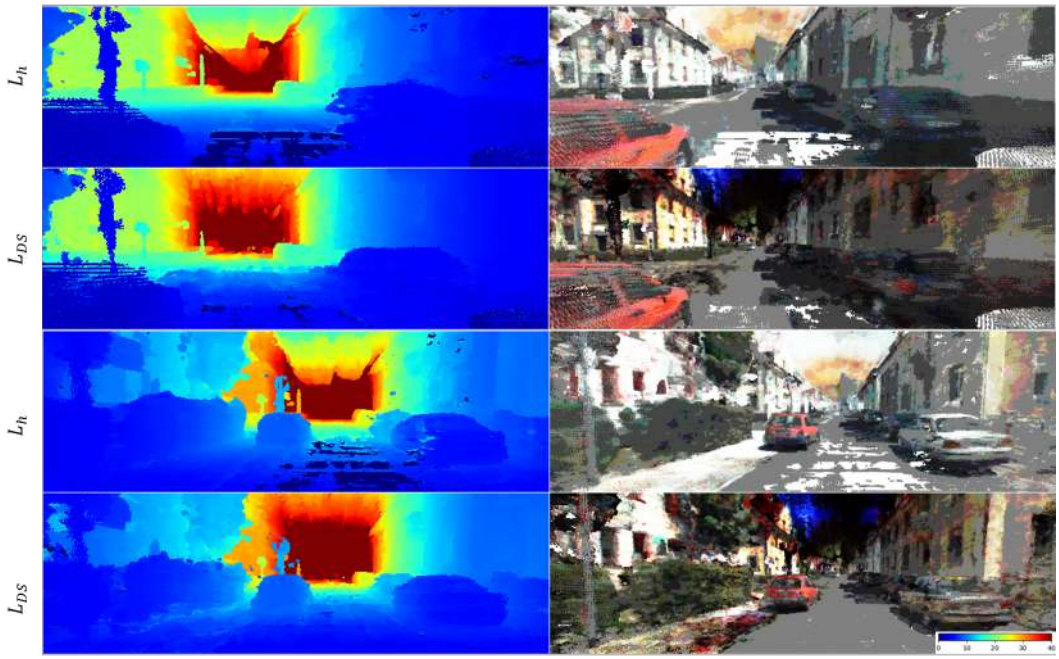

Figure 6.7: **Ablation Study on KITTI-360**. This figure contains a comparison of the extracted point cloud between the two regularization strategies that we investigate. It is taken from Scene 8 where the left column visualizes the depth value of the point cloud upto 40 m from the camera. The right column has the colored point cloud.

Method	LiDAR Supervision	Chamfer-L1 ↓	Completeness ↓	Accuracy ↓	F1 ↑
Monocular Depth N/w	☑	0.1225	0.1831	0.1531	0.5594
Stereo Depth N/w	☑	0.1647	0.1624	0.1831	0.5420
NeRF		0.0731	0.0027	0.1435	0.7826
NeRF+Depth	☑	0.0325	0.0037	0.0614	0.8403
NerfingMVS		0.0277	0.0037	0.0518	0.8269
+Stereo-DS	☑	0.0264	**0.0008**	0.0520	0.8544
+Stereo-DS-Reg	☑	**0.0166**	0.0026	**0.0302**	**0.8889**

Table 6.7: **Quantitative Comparison on KITTI-360**. This table contains quantitative results that are averaged on 10 scenes. The method trained with +Stereo-DS-Reg which combines bi-directional depth reprojection error, depth supervision, stereo prior and depth smoothness regularizer achieves the best results overall.

Table 6.5 (row 4-5) quantitatively compares two regularization methods: (1) Hardloss $\mathcal{L}_h$ and (2) Depth Smoothness Regularization $\mathcal{L}_{DS}$. We show a qualitative comparison in Figure 6.7. We notice that the method trained with $\mathcal{L}_h$ has a lower chamfer distance, but the high F-score indicates that the reconstructed point cloud with $\mathcal{L}_{DS}$ is closer to the fused point cloud (F-score calculates the harmonic mean between Accuracy and Completeness explained in Section 6.1.1). From Fig 6.7, we can observe that the holes in the predicted point cloud of $\mathcal{L}_h$ is mostly along the road region. Road regions are visible in LiDAR and the stereo depth priors are able to predict them with a fairly good accuracy, implying the uncertainty in these regions are low. When the uncertainty is low, the corresponding sampling interval is not wide and close to the depth values of the prior. But in these small regions, points sampled are very dense. It is possible that the strength of the regularizer $\mathcal{L}_h$ maybe too strong as it is formulated to enforce a bimodal Laplacian prior directly on the integration weights. $\mathcal{L}_{DS}$ applies a smoothness constraint to small patches and the density of the sampled points does not directly impact it. $\mathcal{L}_{DS}$ is beneficial to decrease floating artifacts and obtain smoother depth prediction for objects like buildings that spatially cover a large area of the image. Hence, $\mathcal{L}_{DS}$ is well suited for our studies and fits the framework better.

6.4.3 Baseline Comparison

First, we discuss the comparison of the best model from the ablation studies with the baselines, i.e the method trained with stereo priors including Depth Supervision (+Stereo-DS) and the method including patch-based depth smoothness regularization (+Stereo-DS-Reg). Furthermore, we provide an analysis of Chamfer-L1 and compare the methods with high Chamfer-L1 scores.

+Stereo-DS-Reg empirically performs better when compared to all the baselines and this is evidenced by the qualitative results shown in Figure 6.8; which con-

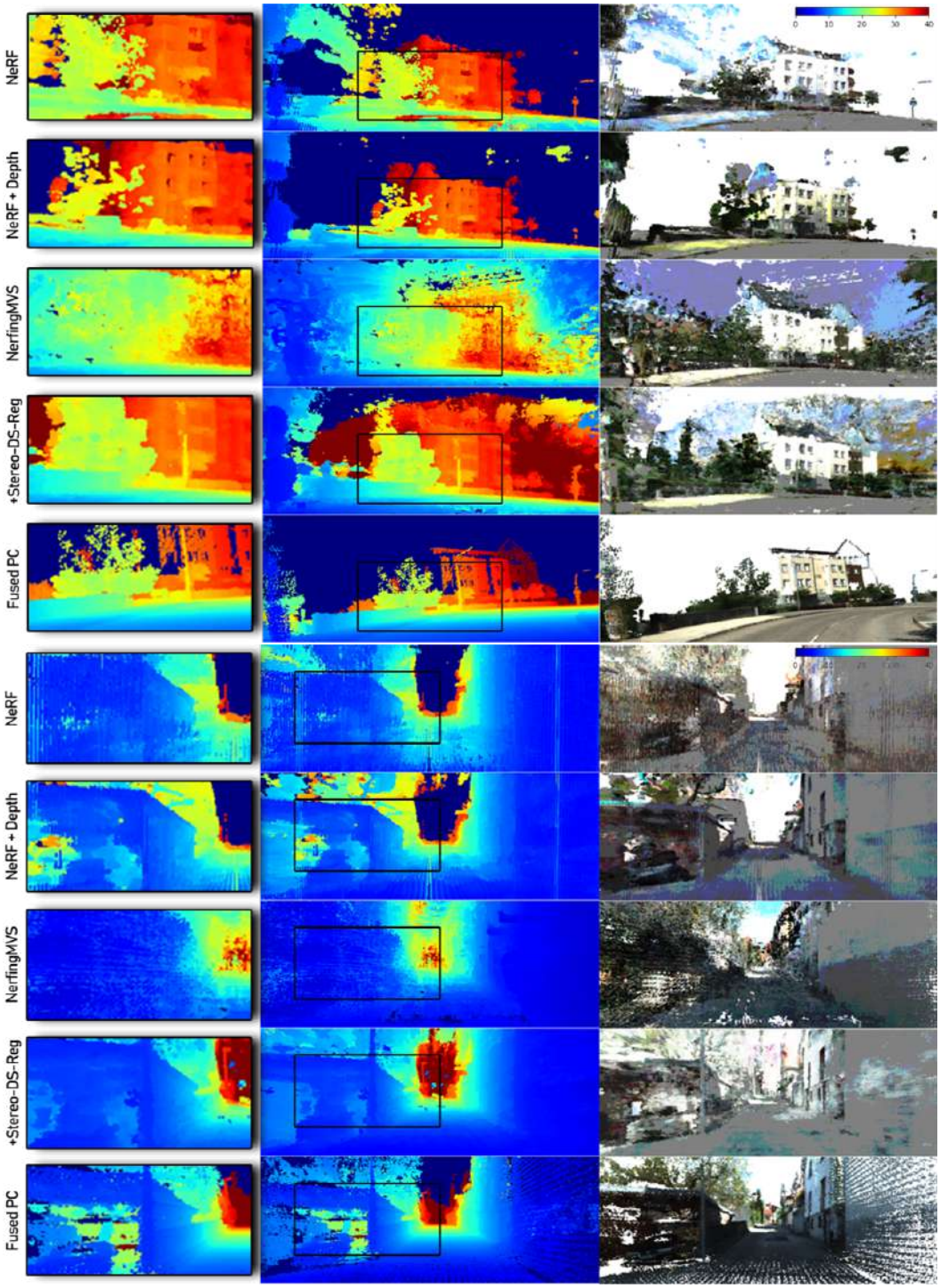

Figure 6.8: **Qualitative Comparison on KITTI-360**. This figure compares the extracted point cloud of our baseline methods. The top scene (a) is viewed at 35° along azimuth and the second scene (b) at 0°. The left column contains the depth values of each point (up to 40m from the camera) and the right column has the colored points taken from Scene 6 and 7. The point clouds are queried at a spatial resolution of 0.5 cm.

tains a comparison between NeRF [MST+20], NeRF+Depth, NerfingMVS [WLR+21] and the fused point cloud of KITTI-360 [LXG21]. NerfingMVS achieve a higher Chamfer-L1 score compared to NeRF and a close score with NeRF+Depth. But, the qualitative results of NeRF and NeRF+Depth is better than NerfingMVS and does not validate the quantitative evaluation. We analyse the reason for obtaining higher Chamfer-L1 score in the next section (6.4.3) by comparing the results of NerfingMVS and +Stereo-DS as they have very close Chamfer-L1 but very different qualitative results.

From the visualization of point clouds colored with depth values, we can see that NerfingMVS reconstruct a noisy point cloud where most of the points lie close to each other without forming meaningful objects. This is due to their sampling strategy, particularly the step that calculates the final sampling ranges. NerfingMVS restrict the sampling interval to lie very close to the monocular depth prior (to avoid floating artifacts). But this is disadvantageous when the depth prior contains high inaccuracies and also restrictive for the network to learn geometry via color supervision. But this does not occur in +Stereo-DS as we introduce an additional scale factor that increases the width of the sampling interval (Details in Section 4.3.1). Moreover, the bi-directional depth reprojection error is able to predict regions with high depth inaccuracies and the network can avoid sampling points close to inaccurate priors.

NeRF predicts a point cloud that retains the global structure of the objects from the scene but is unable to learn detailed geometry. Incorporating point cloud information to NeRF improves the geometry by smoothing out irregular patches and learning distinct objects in the scene. This is noticeable in Fig 6.8 (b), where the additional depth supervision allows NeRF to learn the car shed which is unrecognizable from NeRF's point cloud. From 6.8 (a) we can notice that the road region is smoother in NeRF+Depth. Stereo-DS-Reg outperforms all the baselines because it combines the advantages of using a prior in addition to depth supervision, i.e we are incorporating the same point cloud information in two different ways (Prior and Supervision). We study the individual benefits of training the network with prior-only and supervision-only in the next section. But this formulation enables the network to capture fine geometric details. For instance, in Fig 6.8 (a), it can reconstruct the electric pole, the fence, and the tiny plants in front of the building while learning a smooth street. The qualitative results are validated by the quantitative evaluation and this shows the advantages of combining depth priors in addition to supervision for reconstruction. When we compare the results of NeRF (which is image-based reconstruction) with +Stereo-DS, we can see the extent by which sparse point cloud information is beneficial for reconstruction.

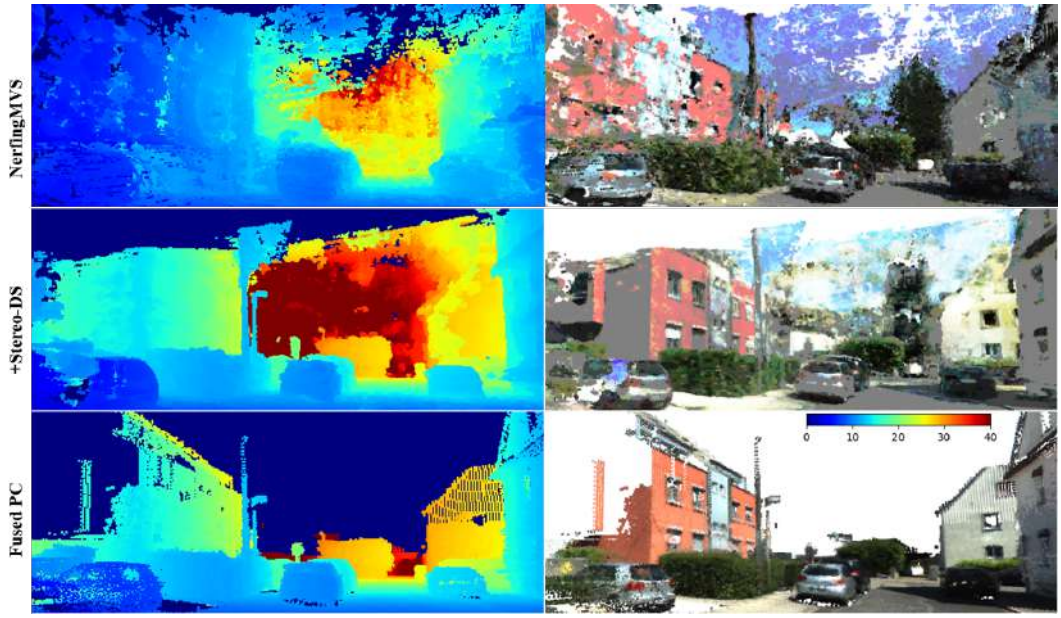

Figure 6.9: **Qualitative Comparison on KITTI-360**. This figure compares qualitative results between methods with highest Chamfer Distance (CD): (1) NerfingMVS: 0.0277 (2) +Stereo-DS: 0.0264. Left side: depth visualization of point cloud up to 40m, Right side: colored point cloud. This result is taken from scene 5 and viewed at 35° along azimuth.

Analysis of Chamfer Distance

Chamfer Distance is the linear sum of Completeness and Accuracy, where completeness measures how close the predicted points are to the ground-truth points while accuracy measures how close the ground-truth points are to the predictions. We utilize the fused point cloud as the ground truth which is incomplete and does not contain dense points for each object within the scene (this is noticeable from Fig 6.9, e.g. the building on the left is not dense and the scanners have captured only a trace of it). [TRR+19] systematically study and analyse the behaviour of chamfer distance and conclude that the measure has high sensitivity to outliers. In case of the building, since the fused point cloud only contains a trace of the building, a higher accuracy score can be obtained when predicted points like very close to the trace (measured distance between each ground-truth point and predicted point is lower). This behavior can be observed from the extracted point cloud of NerfingMVS, where most of the predicted points are clustered around the shape of each object without sharp boundaries. The Completeness score of NerfingMVS (0.037) is higher in comparison to +Stereo-DS (0.0008) and this is caused by the floating artifacts in the predicted point cloud of NerfingMVS is higher (the distance between the predicted point cloud and ground-truth increases). +Stereo-DS has significantly lower floating artifacts and the predicted point cloud has distinct object boundaries.

α	Chamfer-L1 ↓	Accuracy ↓	F1 ↑
1	0.0646	0.0923	0.8197
2	**0.0218**	**0.0431**	**0.8356**
3	0.1724	0.3396	0.7242

Table 6.8: **Ablation Study on KITTI-360**. In this table, we ablate the multiplicative factor α of the sampling interval described in Section 4.3.1. We conduct this experiment on three scenes from KITTI-360: 0, 1, 2 and average the results. We set $\alpha = 2$ for all the other experiments. This point cloud is taken from scene 0.

6.4.4 Appendix

Sampling Interval

In this section, we first show ablation studies on hyperparameters that directly impact the width of the sampling interval (Multiplicative factor α). Next, we conduct an ablation on the constant K, that decides the *number* of poses with which the uncertainty map is calculated. They are described in detail in Section 4.3.1.

Multiplicative factor

This hyperparameter α, decides the scale of the sampling interval based on the uncertainty for a certain depth estimate.
KITTI-360: We have already discussed the role played by this hyperparameter for a large scene like KITTI-360. Table 6.8 contains a quantitative analysis and Figure 6.10 contains the qualitative results on point clouds. From the figure, we can see that an α value of 1 introduces plenty of floating artifacts. These artifacts arise because camera rays belonging to the same 3D world point fail to coincide because the width of the sampling interval is not wide enough. This occurs even when the uncertainty prediction is maximum (i.e. 1) because the sampling interval is only doubled in this case. Whereas, with $\alpha = 2$, the sampling interval is quadrupled. This gives the network some freedom to learn geometry that is multi-view consistent. We do not report qualitative results on $\alpha = 3$, because this is very close to NeRF formulation and decreases the performance of our network. Thus, we set $\alpha = 2$ for all our experiments based on this ablation.

Number of Uncertainty Maps

The number of uncertainty maps that are averaged to get the final uncertainty is decided by the hyperparameter K. Using a K that is too large degrades the performance of the network because in that case average the uncertainty values and get a very low final uncertainty. Moreover, by setting a large K value, we will also include depth projections that are invalid (from far views) because the camera has a

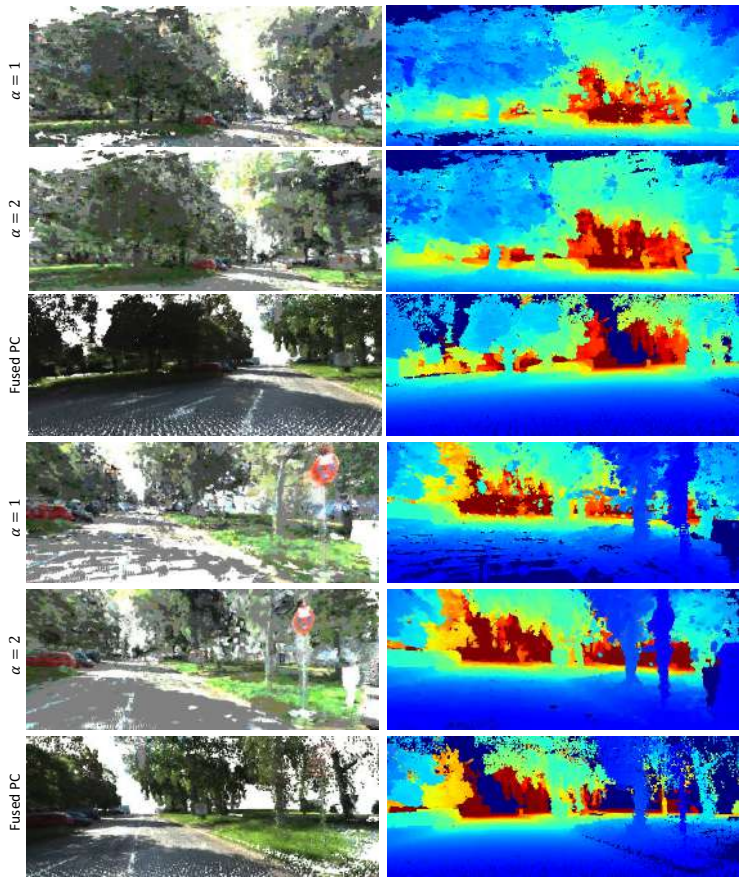

Figure 6.10: **Ablation Study on KITTI-360**. In this figure, we ablate the multiplicative factor α of the sampling interval described in Section 4.3.1. From the qualitative results, it is evident that a higher alpha value helps the network to reduce floating artifacts because the rays from different camera views belonging to same 3D world point intersect to learn consistent geometry. This figure visualizes scene 10.

K	Chamfer-L1 ↓	Accuracy ↓	F1 ↑
4	**0.0352**	**0.0699**	0.8399
12	0.0534	0.0879	**0.8412**
24	0.1129	0.224	0.7914

Table 6.9: **Ablation Study on KITTI-360**. This experiment was conducted on four scenes: 0, 2, 4, 5 and the results were averaged. K decides the number of uncertainty maps that are averaged to obtain the final map. In case of KITTI-360, the camera is moving forward and the number of images containing overlapping objects are about 8-10. As we skip consecutive frames, the number of overlapping images are 4-10. Hence, we did not conduct this ablation on $K < 4$ and instead considered 12 and 24. For all our experiments, we set $K = 4$.

forward motion in KITTI-360. We ablate this parameter in Table 6.9 and set $K = 4$ for all ours and NerfingMVS's experiments.

7 Conclusion

In this thesis, we investigate a framework for reconstructing dense 3D point clouds with 2D images and sparse point cloud supervision. Our key objective is to fuse sparse point cloud information along with images to improve the quality of reconstruction. Point clouds acquired by sensors contain real-world depth measurements that can be encoded as a rich prior to guide the reconstruction process. We adopt the framework proposed by NerfingMVS [WLR+21] and observe that the extracted geometry is noisy due to incorrect sampling of points which misguides the optimization process. We investigate the cause of this behavior on a toy-setting consisting of 10 chairs from the ShapeNet dataset and empirically show that the estimated uncertainty map of NerfingMVS underestimates the inaccuracies of the depth prior. As a direct consequence, an average of 40.50% of the sampled points (ShapeNet chair dataset) do not contain the real surface points within and the network learns an inconsistent 3D representation by predicting multiple portions of the surface that do not form coherent geometry. To overcome this problem, we formulate a bi-directional depth reprojection (bi-dir. reproj.) error to estimate the uncertainty map and capture the inaccuracies of the depth prior by reprojecting the depth maps in both forward and backward directions. Quantitative evaluations show that the uncertainty map obtained from the bi-dir. reproj. error faithfully encodes the depth inaccuracies in comparison to NerfingMVS, and reduces the percentage of incorrectly sampled points to an average of 1.27%. As a result, bi-dir. reproj. learns a consistent representation for each scene and the reconstructed point clouds are compact.

Next, we compare 3D reconstruction results on the chair setting by considering NeRF, NerfingMVS and the framework with bi-dir. reproj.. In order to study the advantages of the sampling strategy in comparison to NeRF. Quantitatively, NeRF has similar results to +bi-dir. reproj. and qualitatively both methods capture the global structure of the scene, however, +bi-dir. reproj is also able to reconstruct detailed geometry. Thus, the sampling strategy of +bi-dir. reproj is beneficial as it enables the network to capture fine geometric details by densely sampling points close to the depth prior.

Furthermore, we investigate the framework for 10 real-world scenes by considering the KITTI-360 dataset where each scene contains outdoor environments with complex geometric details. Therefore, to enforce a stronger prior on the underlying geometry, we exploit the availability of rectified stereo images in the dataset. We study the

benefits of stereo depth priors in comparison to monocular depth priors, where quantitatively the framework trained with monocular priors is better. Qualitatively, the framework trained with stereo priors specifically improve the results of the reconstructed point clouds by capturing local structures. The stereo depth priors are able to capture the depth values of thin structures which are present in the scene and this, in turn, helps the network reconstruct fine geometric details in regions that are not captured by the sensors. Thus, utilising the stereo priors over the monocular priors shows a remarkable improvement in geometry.

Finally, to further improve the quality of reconstruction we directly supervise the expected depth with the sparse point cloud. We observe that depth supervision refines the extracted point cloud and results in sharp object boundaries with defined discontinuities. Our evaluations demonstrate that we obtain a significantly lower Completeness score (distance between predicted points and ground truth) because the network reconstructs a compact point cloud with fewer floating artifacts in comparison to our baselines. To prevent "bleeding" edges (referred to as flying pixels), we adapt the patch-based regularizer proposed by [NBM+22]. The regularizer empirically improves the performance of the model and qualitatively produces point clouds with smoother geometry in textureless (flat) regions. Quantitatively and qualitatively, our evaluations demonstrate that combining the advantages of depth supervision, improve reconstruction quality, while depth priors, capture thin structures, and together improve the reconstruction of NeRF framework to a large extent. Therefore, fusing sensor information with images is beneficial for the reconstruction pipeline.

7.0.1 Limitations and future work

Representation: Our method reconstructs dense point clouds that are able to capture individual objects of a large scene. Our sampling strategy is beneficial for regions with low uncertainty where the sampling interval is tightly bound, but for regions with higher uncertainty, the sampling interval is wider and our reconstructions lack high frequency details in those regions. This is because, the framework does not implicitly reconstruct a surface. Thus, we believe that following [OPG21, WLL+21, AMBG+22] and combining implicit surface representation with the advantages of our efficient sampling strategy (for volume rendering) is a promising future direction to improve geometry.
Modeling Sky: Modeling the sky is crucial for outdoor environments because the sky forms a large region that is often textureless. Our qualitative results show that for the sky regions, the network learns geometry at a finite distance depending on the depth priors. An alternative solution is to obtain segmentation masks for the sky region with a pretrained semantic segmentation network. But this can be problematic for our formulation where the sampling ranges are adapted per pixel, and obtaining a perfect segmentation map can be non-trivial. An elegant solution

would be to explicitly model the sky, similar to [RLS+21] as promising direction to address this limitation.

Depth Prior: The quality of the depth prior impacts the reconstruction to a great extent because the sampling ranges are dependent on the depth prior. We demonstrate the degree to which stereo priors improve reconstruction quality compared to the monocular priors, but rectified stereo pairs may not be available in all case. Thus, to improve the quality of monocular depth priors, off-the shelf filtering strategies like the bilateral filtering [VKB+18] can be experimented.

Bibliography

[AMBG+22] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[BRR11] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In *BMVC*, 2011.

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[BTS+14] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. State of the Art in Surface Reconstruction from Point Clouds. In *Eurographics 2014 - State of the Art Reports*, volume 1 of *EUROGRAPHICS star report*, pages 161–185, Strasbourg, France, April 2014.

[CAP20] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. *CoRR*, abs/2003.01456, 2020.

[CBLPM21] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2021.

[CC18] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018.

[CFG+15] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[CFYD16] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *CoRR*, abs/1604.03901, 2016.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[CWY18] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–119, 2018.

[CZ18] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *CoRR*, abs/1812.02822, 2018.

[DCS+17] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[DLZR22] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283, 2014.

[FHT17] Mohammed E. Fathy, Ashraf Saad Hussein, and Mohamed F. Tolba. Fundamental matrix estimation: A study of error criteria. *CoRR*, abs/1706.07886, 2017.

[FSG16] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *CoRR*, abs/1612.00603, 2016.

[GDS20] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schön. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[GFK+18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CoRR*, abs/1802.05384, 2018.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[HGH+17] Wilfried Hartmann, S. Galliani, Michal Havlena, Luc Van Gool, and Konrad Schindler. Learned multi-patch similarity. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1595–1603, 2017.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[IÇG+18] E. Ilg, Ö. Çiçek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *European Conference on Computer Vision (ECCV)*, 2018. https://arxiv.org/abs/1802.07095.

[JJHZ19] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. *CoRR*, abs/1912.07109, 2019.

[JLY20] Peng Wang Yu Liu Jie Li, Kai Han and Xia Yuan. Anisotropic convolutional networks for 3d semantic scene completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2006.

[LCOZ+11] Jinjie Lin, Daniel Cohen-Or, Hao Zhang, Cheng Liang, Andrei Sharf, Oliver Deussen, and Baoquan Chen. Structure-preserving retargeting of irregular 3d architecture. *ACM Trans. Graph.*, 30(6):1–10, dec 2011.

[LDC+19] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T. Freeman. Learning the depths of moving people by watching frozen people, 2019.

[LFB18] Vincent Leroy, Jean-Sébastien Franco, and Edmond Boyer. Shape reconstruction using volume sweeping and learned photoconsistency. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 796–811, Cham, 2018. Springer International Publishing.

[LHZ+18] Shice Liu, Yu Hu, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and think: Disentangling semantic scene completion. In *Advances in Neural Information Processing Systems*, pages 261–272, 2018.

[LLG+19] Jie Li, Yu Liu, Dong Gong, Qinfeng Shi, Xia Yuan, Chunxia Zhao, and Ian Reid. Rgbd based dimensional decomposition residual network

for 3d semantic scene completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7693–7702, June 2019.

[LPZR18] Kejie Li, Trung T. Pham, Huangying Zhan, and Ian D. Reid. Efficient dense point cloud object reconstruction using deformation vector fields. In *ECCV*, 2018.

[LSU16] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5695–5703, 2016.

[LXG21] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv preprint arXiv:2109.13410*, 2021.

[LZL+20] Siqi Li, Changqing Zou, Yipeng Li, Xibin Zhao, and Yue Gao. Attention-based multi-modal fusion network for semantic scene completion. In *AAAI*, 2020.

[Max95] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[MG15] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[MON+19] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[MST+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[MWA+13] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Gool, and W. Purgathofer. A survey of urban reconstruction. *Comput. Graph. Forum*, 32(6):146–177, sep 2013.

[NBM+22] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[NEP+16] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[NMOG20] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[OMN+19] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *International Conference on Computer Vision*, October 2019.

[OPG21] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021.

[PATM20] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[PFS+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019.

[PNM+20] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *CoRR*, abs/2003.04618, 2020.

[QLT21] Chao Qu, Wenxin Liu, and Camillo Jose Taylor. Bayesian deep basis fitting for depth completion with uncertainty. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16127–16137, 2021.

[QSMG16] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[RBM+22] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[RLS+21] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Thomas A. Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CoRR*, abs/2111.14643, 2021.

[RLS+22] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CVPR*, 2022.

[RMBF21] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari.

Sharf: Shape-conditioned radiance fields from a single view. In *ICML*, 2021.

[RMY+21] Daniel Rebain, Mark Matthews, Kwang Moo Yi, Dmitry Lagun, and Andrea Tagliasacchi. Lolnerf: Learn from one look. *CoRR*, abs/2111.09996, 2021.

[SF16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[SKB18] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1120–11208, 2018.

[SL20] Songyou Peng Boxin Shi Marc Pollefeys Zhaopeng Cui Shaohui Liu, Yinda Zhang. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[SSZ01] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001.

[SYZ+16] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *arXiv preprint arXiv:1611.08974*, 2016.

[SZPF16] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[SZW19] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *CoRR*, abs/1906.01618, 2019.

[TRR+19] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? *CoRR*, abs/1905.03678, 2019.

[TSM+20] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[VKB+18] Julien Valentin, Adarsh Kowdle, Jonathan T. Barron, Neal Wadhwa, Max

Dzitsiuk, Michael John Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Lee Turner, Ivan Dryanovski, Joao Afonso, Jose Pascoal, Konstantine Nicholas John Tsotsos, Mira Angela Leung, Mirko Schmidt, Onur Gonen Guleryuz, Sameh Khamis, Vladimir Tankovich, Sean Fanello, Shahram Izadi, and Christoph Rhemann. Depth from motion for smartphone ar. *ACM Transactions on Graphics*, 2018.

[vL16] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17(1):2287–2318, jan 2016.

[WLL+21] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[WLR+21] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*, 2021.

[XHG+21] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Synlidar: Learning from synthetic lidar sequential point cloud for semantic segmentation. *arXiv preprint arXiv:2107.05399*, 2021.

[YC20] Lin Yen-Chen. Nerf-pytorch. `https://github.com/yenchenlin/nerf-pytorch/`, 2020.

[YKH+18] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737, 2018.

[YKM+20] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

[YLL+18a] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *CoRR*, abs/1804.02505, 2018.

[YLL+18b] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *European Conference on Computer Vision (ECCV)*, 2018.

[YLL+19] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[YMHR19] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *The*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[YYTK20] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images, 2020.

[ZRSK20] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *CoRR*, abs/2010.07492, 2020.