Neural Representations for 3D Reconstruction

Hybrid-Surf: Neural RGB-D Surface Reconstruction using Hybrid Scene Representation

Hengyi Wang[1]

Master of Science in Computer Graphics, Vision, and Imaging

Supervisor: Prof. Lourdes Agapito

Submission date: 12/09/2022

[1]**Disclaimer:** This report is submitted as part requirement for the Master of Science in Computer Graphics, Vision, and Imaging at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

3D reconstructions of room-scale scene is a long-standing problem in computer vision. Recently, the advent of consumer-level RGB-D cameras as well as the research progress in neural representations have led to significant advances in the state-of-the-art of this research area. The existing works using neural representations for 3D room-scale scene reconstructions with RGB-D cameras can be divided into coordinate-based approaches and grid-based approaches. Coordinate-based methods achieve accurate, high-fidelity reconstructions with efficient memory usage at the cost of the slow training speed. Grid-based methods achieve fast optimisation, but sacrifice memory footprint and scene completion. In this thesis, we propose a hybrid sparse neural representation, Hybrid-Surf, which combines the advantages of coordinate-based approaches and grid-based approaches for 3D reconstructions. Specifically, we adopt a sparse hash-based feature grid with positional encoding to generate the feature vectors for each given position. The feature vectors are decoded by two shallow MLPs that output signed distance and RGB values respectively. The volume rendering techniques are used to provide additional supervision for signed distance via photometric loss. Experimental results show that Hybrid-Surf achieves on-par performance in comparison to other state-of-the-art methods while having a significantly faster training speed, efficient memory usage, and maintaining the ability of scene completion.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Lourdes Agapito, for her support and guidance throughout this project.

I would also like to thank Mr. Jingwen Wang for providing the implementation of GO-Surf along with the evaluation script, which served as an important baseline for my thesis.

Furthermore, I would like to acknowledge my fellow Master students for their insightful discussions, constructive feedback, and support.

Finally, I express my heartfelt gratitude to my family for their support and encouragement throughout the year.

Contents

1 Introduction **1**

1.1 3D reconstruction with RGB-D cameras 1

1.2 Aims and objectives 3

1.3 Contribution 4

1.4 Structure of the report 4

2 Preliminaries **5**

2.1 Camera models 5

2.2 3D shape representations 7

2.2.1 Explicit representation 8

2.2.2 Implicit representation 10

2.3 Volume rendering 11

2.3.1 Properties of Volumes 11

2.3.2 Volume rendering Equation 13

2.3.3 Volumetric formulation for NeRF 14

3 Literature Review **15**

3.1 Classic RGB-D 3D scene reconstruction 16

3.2 Coordinated-based neural representations 18

3.3 Grid-based neural representations 18

3.4 Neural representations for 3D reconstruction 19

4 Methodology **21**

4.1 Problem statement 21

4.2 Hybrid sparse scene representation 22

4.3 Volumetric rendering 25

4.4 Depth-guided sampling 26

4.5 Optimisation 27

5 Experimental Results **30**

5.1 Setup 30

5.2 Quantitative analysis 32

5.3 Qualitative analysis 36

5.4 Runtime and memory analysis 37

5.5 Rendering results 38

5.6 Ablation studies 39

5.6.1 Effect of the positional encoding 39

5.6.2 Effect of the depth measurement 40

5.6.3 Effect of the depth-guided sampling 41

6 Discussion and Future Works **42**

6.1 Tri-plane representation 42

6.2 Representation learning 43

7 Conclusion **44**

References **53**

A Derivation **54**

A.1 Depth to normal 54

A.2 Volumetric formulation 54

B More Experiments **55**

B.1 NeuS rendering 55

B.1.1 Qualitative analysis 55

B.1.2 Rendering normal 56

B.2 Online mapping 57

B.3 Separate decoder 58

List of Figures

1.1 Classic RGB-D scene reconstruction 2

2.1 Pinhole camera model 6
2.2 Shape representation 8
2.3 Properties of volumes 12

3.1 Research progress in neural representations 16
3.2 Pipeline of classic RGB-D scene reconstruction 17

4.1 Overview of Hybrid-Surf 22
4.2 Hybrid scene representation 23
4.3 Visualisation of our weight function 26
4.4 Illustration of depth-guided sampling 27
4.5 Illustration of our losses 28

5.1 Qualitative comparison 36
5.2 Rendering results 38
5.3 Performance change with different hash table size 39
5.4 Illustration of effect of depth measurements 41
5.5 Illustration of effect of depth-guided sampling 41

6.1 Illustration of representation learning 43

B.1 Qualitative results of using NeuS rendering 56
B.2 Visualisation of rendered normal 57
B.3 Qualitative results of online mapping 58
B.4 Visualisation of color mesh 59

List of Tables

5.1 Quantitative comparison 32
5.2 Per-scene quantitative comparison 35
5.3 Runtime and memory comparison 37
5.4 Comparison of memory usage 40

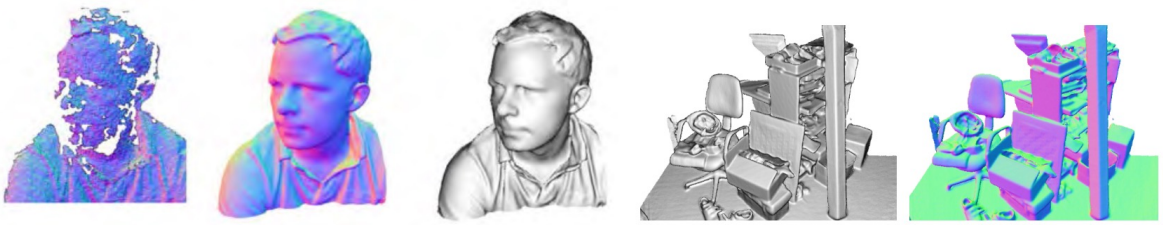
Chapter 1

Introduction

3D room-scale scene reconstruction from video sequences is one of the fundamental tasks in computer vision with many applications. In virtual reality (VR) for instance, to enable the immersive interaction between the user and the surrounding scene, an accurate, high-fidelity scene reconstruction is required. In recent years the advent of the consumer-level RGB-D camera has brought significant advances to this area. Many researchers have taken advantage of the additional range measurement from the RGB-D camera to design algorithms for 3D scene reconstructions. In this thesis, we aim at addressing the problem of 3D room-scale scene reconstruction with RGB-D cameras. In this chapter, we will give an introduction to our problem as well as the goal and contribution of this thesis.

1.1 3D reconstruction with RGB-D cameras

There have been decades of research for structured light or time-of-flight-based depth cameras. The recent advance in consumer-grade sensors has made such technology broadly accessible to users through massive productions. Nowadays we could easily find such cameras on many mobile devices (e.g., iPhone pro 13). Since these cameras not only have affordable prices but



(a) KinectFusion [New+11]

(b) Voxel Hashing [Nie+13a]

Figure 1.1: The static scene reconstruction using classic RGB-D reconstruction techniques. Images are taken from [New+11] and [Nie+13a]

also are able to simultaneously capture RGB and depth images with high resolution in real-time, researchers in vision and graphics instantly recognise the potential of such devices. In conjunction with the rising computing powers of the modern GPUs, many classic reconstruction algorithms [New+11; Iza+11; Nie+13b; CBI13; SKC13; CZK15; Whe+15; Dai+17] with RGB-D cameras have been proposed and show competitive reconstruction results (see Figure 1.1). These methods usually perform the fusion of the depth measurement using truncated signed distance function (TSDF). However, due to the noisy or missing measurement caused by the limitation of the depth sensor, these methods usually contain holes or incomplete geometry.

Recently, there is a trend of using neural implicit representation to represent a scene and learning such a representation from images with differentiable renders. NeRF [Mil+20] firstly proposes a novel view synthesis method that represents the scene using the weight of Multi-layer Perceptrons (MLPs) and trains it with the volumetric rendering. This method shows an unprecedented level of fidelity in scene representation and leads to many extensions of 3D reconstruction. Different from classic RGB-D scene reconstruction work that purely rely on depth information

during fusion, using NeRF style representation with additional depth measurement [Azi+22; WBA22] can take advantage of the color as well as the depth information to achieve an accurate scene reconstruction and complete the geometry with missing depth measurement.

Existing neural representations that are used for 3D reconstruction can be mostly divided into coordinate-based methods [Wan+21; Suc+21; Ort+22; Sun+22] and grid-based methods [Mur+20; Sun+21; WBA22]. Coordinate-based methods implicitly represent a scene using MLPs and show unprecedented performance in representing different scenes as well as scene completion at cost of slow training. In order to reduce the computational cost of such methods, grid-based methods, instead, represent the scene with a voxel grid and directly optimise the trilinear interpolated features. These methods significantly improve the time efficiency of reconstructing a scene at the cost of a large memory footprint. However, the grid-based methods are usually not good at scene completion as the feature grid would only update the parameters in the local region. Thus, achieving a fast 3D scene reconstruction with scene completion and efficient memory usage is still a challenging problem.

1.2 Aims and objectives

The goal of this thesis is to tackle the problem of 3D room-scale scene reconstruction with an RGB-D camera. We aim at proposing a neural RGB-D surface reconstruction system that satisfies:

- Accurate scene reconstructions with high fidelity
- Efficient in both speed and memory footprint
- Achieve scene completion for unobserved regions or regions with the missing depth

1.3 Contribution

In this thesis, we achieve our goal by proposing a neural RGB-D surface reconstruction system, Hybrid-Surf, which is a hybrid sparse neural representation that combines coordinate-based representations and grid-based representations to achieve an efficient, high-fidelity scene reconstruction with scene completion from RGB-D measurements. Our method has faster training speed in comparison to previous coordinate-based works [Azi+22; Suc+21; Ort+22] and grid-based works [WBA22] while achieving on par performance and require constant memory footprint. In terms of scene completion, GO-Surf [WBA22] proposes a first-order smoothness term that achieves the reconstruction with scene completion. However, such a smoothness term requires maintaining a second-order computational graph and introduces the bias that would also smooth the fine structures. In contrast, our method uses the continuity of the MLPs to naturally complete the unobserved regions. Thus, the contribution of this thesis can be summarised as:

- Propose a hybrid sparse neural representation for RGB-D surface reconstruction
- Achieve accurate scene reconstruction with fast optimisation and efficient memory usage
- Realise scene completion without any regularisation terms

1.4 Structure of the report

This thesis is mainly about neural representations for 3D reconstructions with RGB-D cameras, most of the techniques involved are recent advances in neural representations. In Chapter 2, we introduce preliminaries related to camera models, 3D shape representations, and volume rendering techniques. In Chapter 3, we review the current neural representations for 3D reconstruction as well as the classic 3D scene reconstruction with RGB-D cameras. Then, we describe the proposed method in Chapter 4, followed by our experimental results in Chapter 5 and discussions in Chapter 6.

Chapter 2

Preliminaries

Benefit from the recent advances in neural representations, research on 3D scene reconstructions has made impressive progress these days. In this Chapter, we introduce the preliminary knowledge for this thesis, including the camera models, 3D shape representations, and volume rendering techniques.

2.1 Camera models

A camera is the mapping between 3D coordinates in the world space and 2D coordinates in the image space. Consider a simple pinhole camera model as shown in Figure 2.1(b), $\mathbf{p}$ is the principal point in the image plane $z = f$, the point $\mathbf{X} = (X, Y, Z)^\top$ in the world space is mapped to the point $\mathbf{x}$, which is the intersection between the line from camera centre to $\mathbf{X}$ and the image plane. Thus, we can use similar triangles to find that

$$\mathbf{x} = (\frac{fX}{z}, \frac{fY}{z})^\top, \tag{2.1}$$

where the origin of the image plane is assumed at the principal point. However, in practice, the origin may be in another place (see Figure 2.1(c)). Thus, we need an offset $(p_x, p_y)^\top$ to formulate

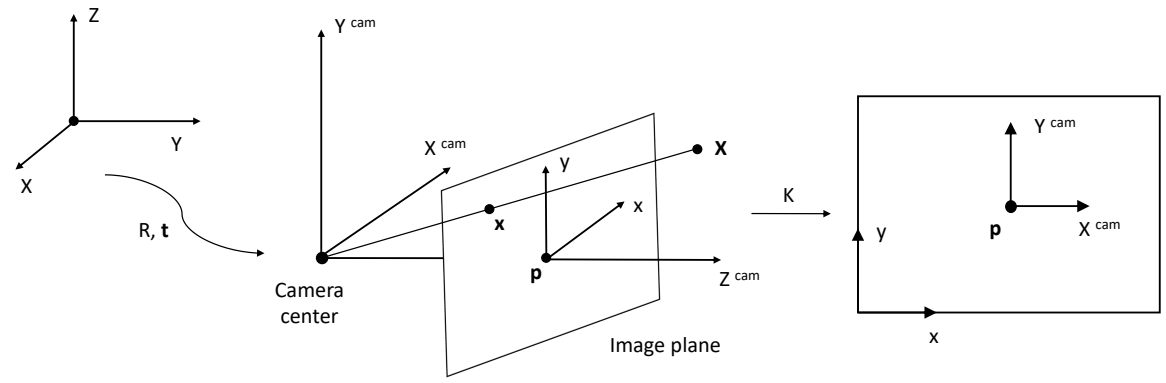


Figure 2.1: Illustration of the pinhole camera model. A projection from the point in the world space to the point in the image space requires the transformation from the world coordinate system to the camera coordinate system to the image coordinate system.

the mapping as

$$\mathbf{x} = (\frac{fX}{z} + p_x, \frac{fY}{z} + p_y)^\top. \tag{2.2}$$

By using the homogeneous coordinate system, we could write the mapping as

$$\lambda \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \\ 1 \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{bmatrix}, \tag{2.3}$$

where

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{2.4}$$

is the camera calibration matrix that depends on the intrinsic parameters of the camera. In general, points in space usually have another coordinate system that is known as the world coordinate

system (see Figure 2.1(a)). We need a rotation matrix R and a translation vector $\mathbf{t}$ to map the point in the world coordinate system into the camera coordinate system:

$$\lambda \begin{bmatrix} X^{cam} \\ Y^{cam} \\ Z^{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

where the R and t are called extrinsic parameters with each having 3 degrees of freedom (DoF). Thus, we can write the full equation of a pinhole camera that maps the points in the world coordinate system into the image coordinate system as follows:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.6}$$

For 3D reconstruction with RGB-D cameras, the intrinsic parameters of the camera are usually given. Thus, we only need to estimate the rotation matrix R and translation $\mathbf{t}$. This is usually called 6-DoF pose estimation.

2.2 3D shape representations

For the representation of the 2D images, we usually use a regular 2D grid structure that is known as pixels. However, for 3D geometry, there are various kinds of representations with different purposes and advantages. Figure 2.2 illustrates different 3D shape representations. Generally speaking, the 3D shape representation can be divided into explicit representations and implicit representations. In this section, we will go through these 3D shape representations in detail.

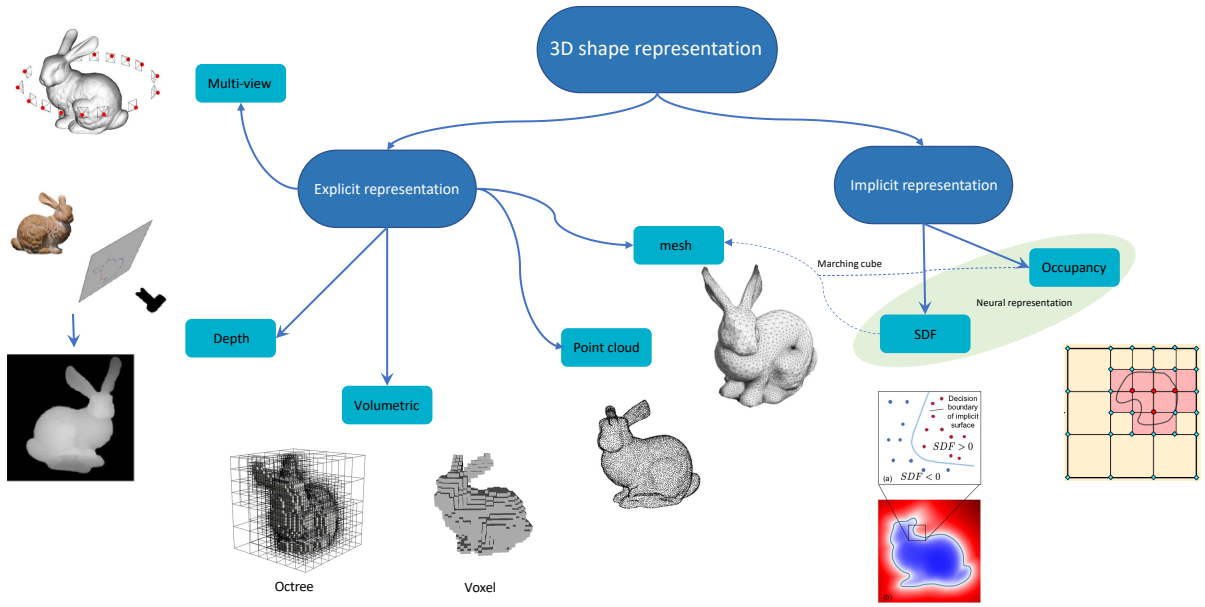


Figure 2.2: Illustration of different 3D shape representations using Stanford bunny [Tur00]. They are explicit representations (multi-view [Par+16], depth [Lah+22], volumetric [And13], point cloud, and mesh) and implicit representations (signed distance field [Par+19], occupancy [Mes+19], etc).

2.2.1 Explicit representation

Explicit representation is a kind of shape representation that explicitly contains the geometric information of the target shape. The existing explicit representations can be divided into multi-view, depth, volumetric, point cloud, and mesh.

Multi-view representation. For multi-view representation, the 3D shape is represented by a combination of multiple images that are captured from different viewpoints. Using such a representation for 3D shapes is relatively efficient as only 2D images are needed in comparison to other representations with 3 dimensions. Also, the acquisition of such data representation poses minimal requirements on the hardware. Specifically, RGB cameras are enough. However, since the multi-view measurement usually does not contain the range measurement, we need one extra step to estimate the 3D geometry using the relative positions of objects in different views.

Depth data. Thanks to the range measurement generated by 3D sensors (e.g. Microsoft Kinect),

depth image has been widely adopted as a representation of 3D objects. Usually, depth data are used with RGB images as RGB-D images. For each pixel, depth maps give the distance between the camera and the target object at that pixel in the world coordinate system. Strictly speaking, RGB-D data is a 2.5D representation since we cannot look behind the objects due to the existence of the occlusion. Given the depth map, we could also derive the surface normal, which is also a kind of 3D shape representation that gives the normal vector of the 3D object in the world.

Volumetric representation. Voxel is a typical volumetric representation that represents the shape with a $V \times V \times V$ grid with features stored in each voxel. This representation can be treated as a "3D pixel representation". Due to the structured nature of 3D voxels, we could naturally transfer the 2D processing techniques into 3D for processing such data (e.g. convolution). However, although voxels are conceptually similar to pixel representation, capturing fine details in 3D requires a high spatial resolution, which is a lot more computationally expensive in comparison to pixel representation. Thus, many sparse volumetric data structures, such as octree, and quadtree, are proposed.

Point cloud. The point cloud represents the shape as a set of points in 3D spaces. Such a lightweight representation can be provided by many sensors (i.e. LiDARs). Due to the fact that the point cloud is an unordered set, the methods used for processing such data should be permutation invariant. In comparison to voxel representation, a point cloud can represent fine structures without a huge number of points. However, the surface of the shape is not explicitly represented by point cloud. Thus, extracting the surface requires further post-processing.

Mesh. Mesh is one of the most classic shape representations in graphics. A mesh representation consists of a set of vertices that are connected with each other to form the polygons (i.e., triangles, quadrilaterals, etc). The set of the polygons will thus form the surfaces in the 3D space explicitly. Mesh is flexible to represent fine structures with more faces and flat surfaces with fewer faces. In comparison to a point cloud which only contains vertices locations, the mesh also includes the surface. However, as the processing of such surfaces with learning-based methods is not

intuitive, the mesh is usually adopted in graphics for the representation of 3D models.

2.2.2 Implicit representation

Instead of explicitly representing a 3D shape, implicit representation models the shape via a function that maps the 3D coordinate into different values. The surface is implicitly represented as the level set of such a function. Two typical implicit representations are the occupancy function and the signed distance function.

Occupancy function. Given the input point $\mathbf{x}$, an occupancy function describes the occupancy on such point:

$$o : \mathbb{R}^3 \to \{0, 1\}. \tag{2.7}$$

The surface of the target 3D object can thus be a level set of occupancy functions:

$$\{\mathbf{x} : o(\mathbf{x}) = \tau\}. \tag{2.8}$$

With such an implicit representation, we could fit a model that outputs the probability of occupancy and find a proper decision boundary as the surface.

Signed distance functions. The signed distance function encodes the signed distance of a given point $\mathbf{x}$ to the closest surface. The sign represents whether the point is inside (negative) or outside (positive) of a watertight surface:

$$SDF(\mathbf{x}) = d, \text{ where } \mathbf{x} \in \mathbb{R}^3, d \in \mathbb{R} \tag{2.9}$$

The surface is thus implicitly represented as the zero level set:

$$\{\mathbf{x} : SDF(\mathbf{x}) = 0\}. \tag{2.10}$$

As sometimes we only care about the point near the surface of the target scene, the truncated signed distance function (TSDF) is widely adopted. In comparison to occupancy, one great advantage of the signed distance function is that the surface is represented with a certain decision boundary, i.e. zero level set. Thus, many works of 3D reconstruction adopt such representation for implicit representation of the 3D shape.

2.3 Volume rendering

2.3.1 Properties of Volumes

Volumes, in our context, can be considered as a cloud of particles that ranges from atoms to any particle size. When a photon passes through such a volume, it has the possibility to collide with the particles inside the volume. Such collisions formulate the radiance distribution of the volume. Note that in order to satisfy the statistically independent collisions, the size of each particle should be small enough in comparison to their average distance.

The properties of volumes that impact the radiance, i.e., the radiant flux travel through a differential beam, can be divided into absorption, scattering, and emission (see Figure 2.3).

Absorption. Absorption means some particles can absorb the photons when a collision appears. Let us consider a radiance beam $L(\mathbf{x}, \omega)$, which starts at $\mathbf{x}$ with direction ω, passes through a segment of the volume. The radiance would be reduced, and the rate of such reduction depends on the absorption coefficient $\sigma_a(\mathbf{x})$, which determines the probability density of a photon being absorbed per unit distance dz:

$$\frac{dL}{dz} = -\sigma_a(\mathbf{x})L(\mathbf{x}, \omega). \tag{2.11}$$

Scattering. Scattering describes a collision which scatters the photons into different direc-

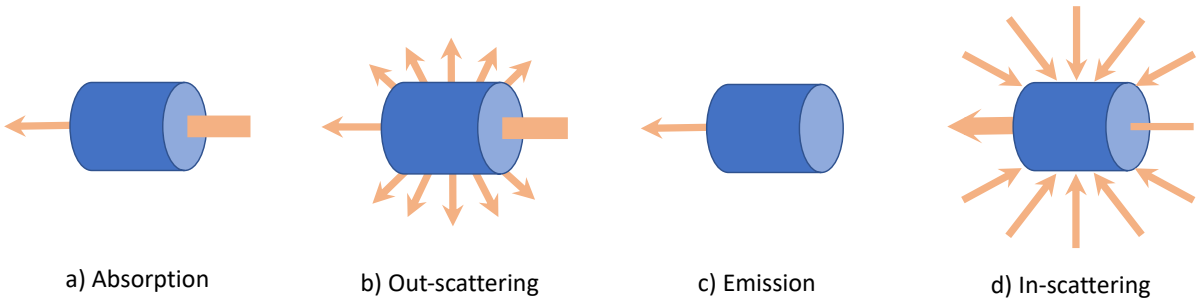


Figure 2.3: Illustration of properties of volumes that impact the radiance a) absorption b) out-scattering c) emission d) in-scattering.

tions that defined by phase function $f_p(\mathbf{x}, \omega, \bar{\omega})$ that is normalised over the sphere:

$$\int_{S^2} f_p(\mathbf{x}, \omega, \bar{\omega}) d\theta = 1. \tag{2.12}$$

When scattering appears, some of the photons will be scattered out, and cause the loss of the radiance. The rate of such loss is proportional to the scattering coefficient $\sigma_s(\mathbf{x})$, which determines the probability density of a photon being scattered per unit distance dz:

$$\frac{dL}{dz} = -\sigma_s(\mathbf{x}) L(\mathbf{x}, \omega). \tag{2.13}$$

Since the photons can out-scattering, there are also chances that the photons can scatter into the direction ω of the radiance beam:

$$\frac{dL}{dz} = \sigma_s(\mathbf{x}) L_s(\mathbf{x}, \omega), \tag{2.14}$$

where the in-scattering radiance $L_s(\mathbf{x}, \omega)$ is as follows:

$$L_s(\mathbf{x}, \omega) = \int_{S^2} f_p(\mathbf{x}, \omega, \bar{\omega}) L(\mathbf{x}, \bar{\omega}) d\bar{\omega}. \tag{2.15}$$

Emission. Emission describes that the radiance can also increase as a result of emission from the volume. Given the emitted radiance $L_e(\mathbf{x}, \omega)$, the emission should be the product of the

absorption coefficient:

$$\frac{dL}{dz} = \sigma_a(\mathbf{x})L_e(\mathbf{x}, \omega), \tag{2.16}$$

2.3.2 Volume rendering Equation

The radiance distribution within a volume can be defined by radiative transfer Equation [Cha13] (RTE) that combines absorption, scattering, and emission:

$$\frac{dL(\mathbf{x}, \omega)}{dz} = -\sigma_a(\mathbf{x})L(\mathbf{x}, \omega) - \sigma_s(\mathbf{x})L(\mathbf{x}, \omega) + \sigma_s(\mathbf{x})L_s(\mathbf{x}, \omega) + \sigma_a(\mathbf{x})L_e(\mathbf{x}, \omega). \tag{2.17}$$

As both absorption coefficient $\sigma_a(\mathbf{x})$ and out-scattering operates on the same radiance function $L(\mathbf{x}, \omega)$, we could merge them into one term:

$$\frac{dL(\mathbf{x}, \omega)}{dz} = -\sigma_t(\mathbf{x})L(\mathbf{x}, \omega) + \sigma_s(\mathbf{x})L_s(\mathbf{x}, \omega) + \sigma_a(\mathbf{x})L_e(\mathbf{x}, \omega), \tag{2.18}$$

where $\sigma_t(\mathbf{x}) = \sigma_a(\mathbf{x}) + \sigma_s(\mathbf{x})$. The RTE describes the change of the radiance beams during the traveling. By integrating both sides of the equation with finite length z, we could obtain the integral form of the RTE:

$$L(\mathbf{x}, \omega) = \int_{t=0}^{z} T(t)[\sigma_a(\mathbf{x}_t)L_e(\mathbf{x}_t, \omega) + \sigma_s(\mathbf{x}_t)L_s(\mathbf{x}_t\omega)]dt, \tag{2.19}$$

where transmittance $T(t) = exp(-\int_{s=0}^{t} \sigma_t(\mathbf{x}_s))ds$ (see Appendix A.2) is the probability of a photon that travels from $\mathbf{x}$ to $\mathbf{x}_t$ without hitting any particles. In order to model the radiance at the end of the beam, we could add $L_z(\mathbf{x}_z, \omega)$ to RTE and formulate the volume rendering equation (VRE):

$$L(\mathbf{x}, \omega) = \int_{t=0}^{z} T(t)[\sigma_a(\mathbf{x}_t)L_e(\mathbf{x}_t, \omega) + \sigma_s(\mathbf{x}_t)L_s(\mathbf{x}_t\omega)]dt + T(z)L_z(\mathbf{x}_z, \omega) \tag{2.20}$$

2.3.3 Volumetric formulation for NeRF

The exact interaction between light and the particle is quite a complex problem to solve. Thus, NeRF [Mil+20] adopts a volume density scattering model [Bli82; KV84] that assumes there is only one scattering of the radiation from the light source to eye. Volume density $\sigma(\mathbf{r}(t))$ can be considered as the probability of a photon stopping at a small interval around t. Given the color $c(\mathbf{r}(t))$ at each point of a radiance beam, the expected color $C(\mathbf{r})$ of a ray $\mathbf{r}$ can within near t_n and far t_f bound is the sum of the contribution of each volume element:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), \omega)dt, \text{ where } T(t) = exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds). \tag{2.21}$$

To solve such an integration, NeRF divides $[t_n, t_f]$ into N equal partitions and uniformly sample point t_i within each partition:

$$t_i \sim \mathcal{U}[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)]. \tag{2.22}$$

These samples are then adopted for the estimation of $C(\mathbf{r})$ using quadrature rule [Max95]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i(t_{i+1} - t_i)))\mathbf{c}_i, \quad \text{where} T_i = exp(-\sum_{j=1}^{i-1} \sigma_j(t_{j+1} - t_j)) \tag{2.23}$$

Chapter 3

Literature Review

Figure 3.1 shows an overview of the recent advances in neural representations for 3D reconstruction. Generally speaking, current neural representations can be mostly divided into coordinate-based approaches and grid-based approaches. By taking advantage of the continuity and compactness of the MLPs, coordinate-based approaches can achieve accurate, high-fidelity scene reconstruction as well as scene completion at the cost of slow training. Grid-based approaches, instead, store the feature in the discretized vertex of the feature grid and significantly reduce the optimisation time but sacrifice memory usage and scene completion. Thus, many follow-up works have been proposed to improve these methods. In the following sections, we will review the research progress in neural representations that includes coordinate-based neural representations, and grid-based neural representations as well as their applications for 3D reconstructions. Since the existing reconstruction pipelines using neural representations share lots of common features with the pipeline for classic RGB-D reconstruction, we will first review the classic 3D reconstructions with RGB-D cameras before we go through the neural representations.

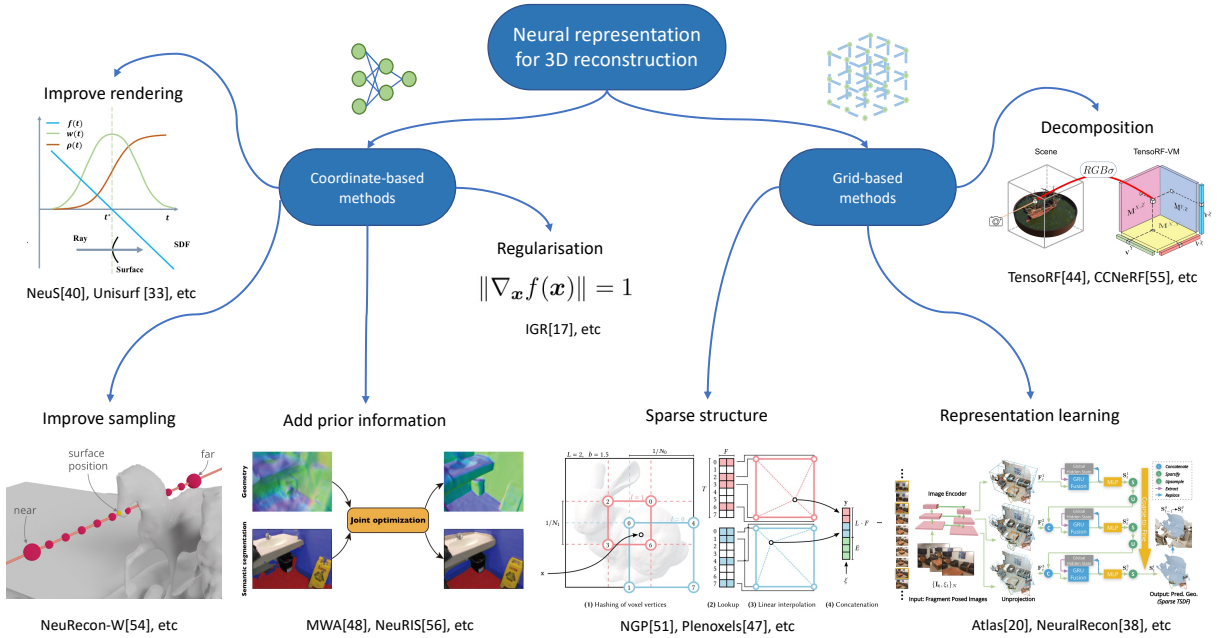


Figure 3.1: An overview of recent research advances in neural representations for 3D reconstructions. For coordinate-based approaches, existing methods mainly focus on improving rendering [Wan+21; OPG21], sampling [Sun+22], adding prior information [Guo+22; Wan+22], and regularisation [Gro+20]. For grid-based approach, existing works focus on investigating sparse structure [Mül+22; Fri+22], representation learning [Mur+20; Sun+21], and decomposition [Che+22; Tan+22]. Note that most of the advances in coordinate-based approach can be naturally adopted in grid-based approaches.

3.1 Classic RGB-D 3D scene reconstruction

The introduction of the consumer-level depth camera with the rising power of modern GPUs enables the development of classic 3D scene reconstruction with RGB-D cameras. Although there is a bunch of different algorithms for RGB-D reconstruction, most of them share a similar pipeline as presented in Figure 3.2.

In the first step, the input RGB-D data will be preprocessed to filter out the noise and remove the outliers. Generally speaking, the noise of a depth camera depends on various factors. In practice, bilateral filter [TM98] is the most commonly adopted approach. For different representations that have been used, some further processing [LWK15; Mai+11; Kel+13], such as estimation of the uncertainty of each individual range measurement, can be performed.

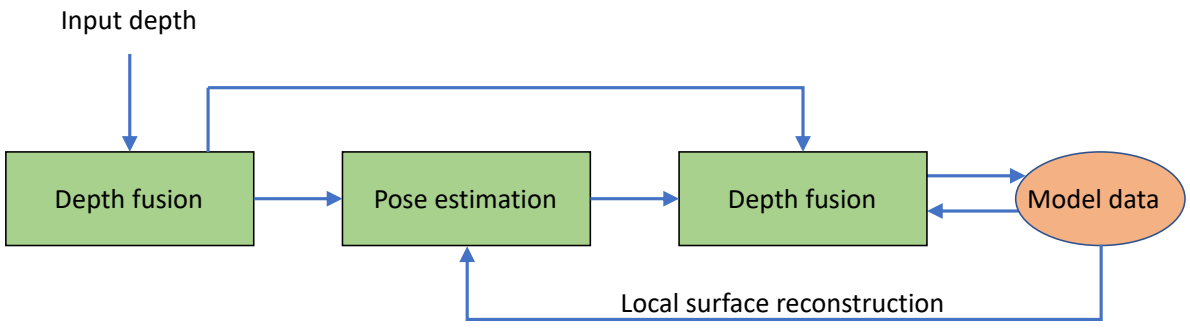


Figure 3.2: An overview of the typical pipeline for classic 3D reconstruction with RGB-D camera presented in [Zol+18].

In the second step, the pose estimation that computes the 6-DoF pose for every frame is performed. In order to address such issue, many methods adopt the variant of the Iterative Closest Point algorithm (ICP) [BM92; CM92; RL01] that use either point-to-point [BM92] or point-to-plane [CM92] error metrics. To address the drift issue, frame-to-model tracking methods that are based on point-to-plane ICP [Low04] have been proposed with many follow-up works that improve tracking by global optimisation [CZK15; Dai+17], relocalisation [Whe+15; Dai+17], etc. Since the camera pose estimation is not the focus of this thesis, we will not introduce these approaches in detail.

In the final step, the estimated camera pose for each frame is used to align the input data and merge it into a 3D model representation. There exist mainly 2 representations for classic 3D scene reconstruction, namely voxel-based representation and point-based representation. Voxel-based representation is first introduced in [CL96] and be extended to many real-time 3D scene reconstruction works that store the SDF or TSDF values in each voxel for implicitly representing the surface. Since the regular voxel-based representation requires a large memory footprint. Many follow-up works have been proposed to reduce the memory footprint of the voxel-based approach using octrees [FG11] or hashing [Tes+03]. Instead of storing the information in each voxel, the point-based representations [Pfi+00] store the range measurement as well as texture color, normal, and other information into each point attribute to represent the 3D scene.

3.2 Coordinated-based neural representations

Recently coordinate-based neural representations have become popular due to their flexibility and compactness. A coordinate-based representation for a given signal aims at mapping the input coordinate to the signal value at that coordinate. Benefiting from the continuity of MLPs, the coordinate-based neural representations are not limited to the fixed spatial resolution. Thus, it is a natural idea to exploit such a representation for implicit 3D scene representation. Depending on different problem settings, various works use coordinate-based neural representation to model volume density [Mil+20; Suc+21], occupancy [Mes+19; CAP20], signed distance field [Par+19; CAP20; Gro+20; Azi+22] or surface map [Mor+21]. Among these works, NeRF [Mil+20] first integrates coordinate-based neural representations with the volume rendering technique. Given only 2D images with corresponding poses for training, NeRF shows the unprecedented performance of novel view synthesis and lead to various extensions on different settings, such as dynamic scene [Pum+21; Gao+21; Li+21; Mar+21; Xia+21], deformation [Gaf+21; Nog+21; Par+21a; Par+21b], generative modelling [JA21; NG21; Sch+20]. Despite these coordinate-based neural representations can represent the complex scenes with low memory footprint and high fidelity, a significant drawback is that they are usually slow to train since it requires multiple times gradient descent of the entire weight of the model. Especially for 3D scene representation that requires multiple samples along each camera's ray for differentiable rendering, the training can take hours to days.

3.3 Grid-based neural representations

In order to reduce the computational cost of training coordinate-based neural representations, there is a trend of storing learnable features in the spatial grid and usually combined with a tiny MLP for decoding the tri-linear interpolated features in the space. This method is known as the grid-based approach [SSC22; Kar+22; Fri+22; Mül+22; Che+22; Tan+22; Cha+22]. Among

these works, DVGO [SSC22] and ReLU fields [Kar+22] directly optimise an explicit voxel grid with post-activation interpolation. Plenoxels [Fri+22] adopts a sparse grid structure to model the scene with spherical harmonics for view-dependent appearance. To reduce the huge cost of the grid-based neural representation, EG3D [Cha+22] proposes a tri-plane representation to reduce the memory cost. NGP [Mül+22] proposes a multi-resolution hashing scheme with a shallow MLP to decode the feature indexed by hashing. They also achieve a significantly faster training speed with the highly engineered CUDA implementation of the encoding and the MLP. Instead of modifying the spatial data structure, tensoRF [Che+22] and CCNeRF [Tan+22] propose a decomposition-based method that can reduce the computational cost of the grid-based methods by decomposing the scene tensor into low-rank tensor components. These grid-based neural representations show promising results for achieving fast 3D reconstruction by only optimising the local region of each sample. However, since the feature grid discretizes the spatial resolution, such scene representation lacks continuity in comparison to the coordinate-based approach. As a consequence, grid-based neural representations cannot well represent the scene with only a limited number of views since part of the grid may not be trained.

3.4 Neural representations for 3D reconstruction

The previously introduced neural representations are mostly focused on novel view synthesis using volume density. However, those density-based methods cannot enable a high-quality surface extraction as the volume density does not contain any surface constraint. Thus, in this section, we mainly focus on the neural representations that are used for 3D reconstruction. In order to reconstruct the 3D scene from a sequence of images, several methods [Mur+20; Sun+21; Boz+21] perform fusion on multiple frames and project the features into a volumetric representation with a decoder that outputs the SDF value. These methods learn a good neural representation without the need for the test time training for each specific scene at the cost of the scene fidelity.

Some other methods [Suc+21; Wan+21; Sun+22; Azi+22; Ort+22] adopt coordinate-based neural representation with volumetric rendering. Among these methods, IGR [Gro+20] proposes the eikonal regularisation for SDF. NeuS [Wan+21] and Unisurf [OPG21] propose volume rendering equations that integrate surface representation. NeuralRecon-W [Sun+22] improves the efficiency of the coordinate-based representation with voxel-guided and surface-guided sampling. MWA [Guo+22] and NeuRIS [Wan+22] add the prior information of the surface normal into the training. Different from those RGB-based methods, Neural RGB-D [Azi+22] takes advantage of a consumer-level depth camera and uses RGB-D image sequence to train MLP that output radiance and approximate SDF via volumetric rendering. Although they achieve competitive surface reconstruction results, the training of such coordinate-based neural representation is quite slow and usually takes several hours. To accelerate the training, iMAP [Suc+21] and iSDF [Ort+22] achieve the reconstruction in a live stream via key-frame selection and active sampling at the cost of sacrificing the details of the surface reconstruction. GO-Surf [WBA22] adopts grid-based neural representation and achieves around 60x times speed-up but with significantly larger memory usage in comparison to Neural RGB-D. In order to realise scene completion, GO-Surf proposes a first-order smoothness regularisation term. Different from these approaches, the proposed Hybrid-Surf uses hybrid neural representations that further reduce the training time and require less memory footprint in comparison to GO-Surf. Additionally, our method achieves scene completion without any regularisation term.

Chapter 4

Methodology

In this thesis, we aim at building an accurate, high-fidelity scene reconstruction system that is fast to optimise and efficient in memory footprint. Benefiting from the range measurement from the RGB-D cameras and recent advances in neural representations, we achieve our goal by proposing Hybrid-Surf, a hybrid sparse neural representation for RGB-D scene reconstructions. In this chapter, we will introduce the problem formulation as well as the details of the proposed method.

4.1 Problem statement

Figure 4.1 illustrates the proposed method for achieving a fast surface reconstruction system. Given a sequence of RGB-D images $I_i \in \mathbb{R}^{H \times W \times 4}$ with the corresponding intrinsic matrix $K_i \in \mathbb{R}^{3 \times 3}$ and extrinsic matrix $\hat{T}_i \in \mathbb{R}^{4 \times 4}$ that is initialised by BundleFusion [Dai+17], we aim at training a model f with weight θ that outputs the signed distance field (SDF) d and color $\mathbf{c} \in \mathbb{R}^3$ given position $\mathbf{x} \in \mathbb{R}^3$ and view direction $\boldsymbol{\omega} \in \mathbb{R}^3$:

$$\theta : f_\theta(\mathbf{x}, \boldsymbol{\omega}) \mapsto (\mathbf{c}, d), \tag{4.1}$$

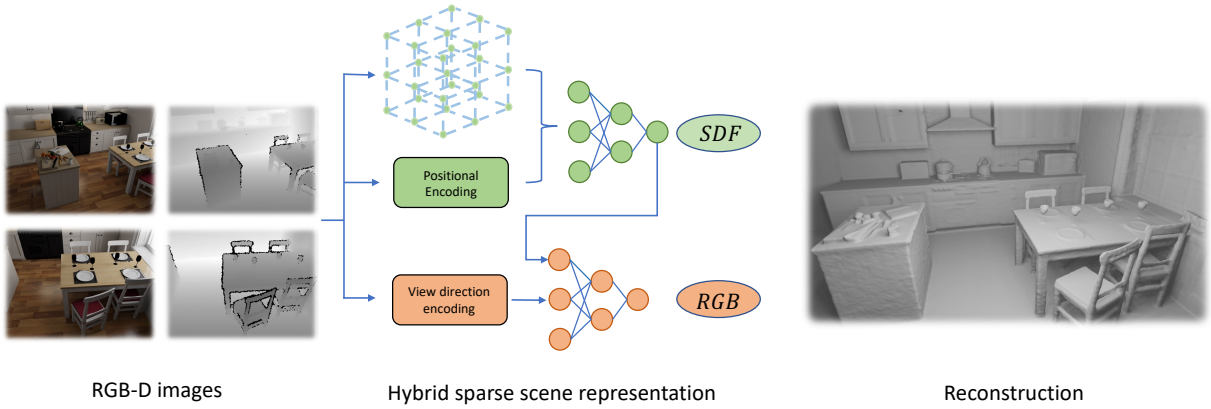


Figure 4.1: The proposed pipeline for achieving surface reconstruction given a sequence of RGB-D images. We use a hybrid sparse scene representation that combines coordinate-based and grid-based approaches together with volumetric rendering for achieving high-quality surface reconstruction.

where $\mathbf{x}$ and $\boldsymbol{\omega}$ are back-projected from the points and directions in the image coordinate system using intrinsic matrix K_i and the estimated extrinsic matrix $\hat{T}_i$. In order to compensate for the artifacts caused by the missing depth measurement, we adopt the volume rendering formulation in NeRF [Mil+20] that uses SDF to generate weights for color reproduction. The photometric loss is thus used for providing additional supervision to SDF. With such a method, we could recover the thin structure area or areas with missing depth measurements based on the RGB measurement. In the following sections, we will introduce our methods in detail.

4.2 Hybrid sparse scene representation

Benefiting from the continuity of the MLPs, the coordinate-based neural representations can achieve accurate scene reconstruction with completion at the cost of slow convergence. In contrast, grid-based neural representations have fast training speeds at the cost of memory footprint. However, these methods lack of ability for completing the unobserved regions as part of the grid has not been trained. Additionally, as the continuous positional information is implicitly encoded into the grid with discretized vertices, the continuity of the decoder has not been fully exploited

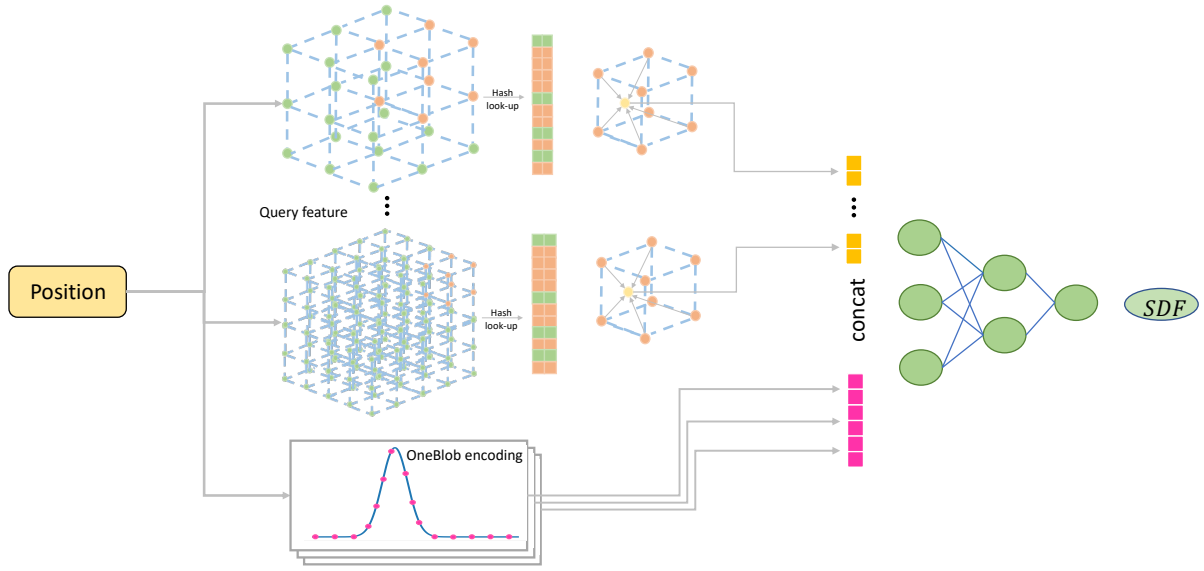


Figure 4.2: The proposed hybrid sparse scene representation contains a sparse hash-based grid and a positional encoding module. The features from these two modules are concatenated and put into an MLP to decode the SDF value for a given position.

for scene completion. In fact, the decoder for grid-based neural representations is a pure feature decoder that does not have any awareness of the position. Motivated by this observation, we propose a hybrid sparse scene representation that combines coordinate-based representations and grid-based representations.

Positional encoding. Instead of implicitly encoding the positional information into the discretized vertices of the grid representation, we explicitly add the positional information into the MLP after the feature grid via positional encoding. To achieve a robust scene completion, we adopt OneBlob encoding [Mül+19], which uses a kernel to activate multiple nodes of the MLP. Considering one scalar $x \in [0, 1]$ with k bins of the quantization, we could place a Gaussian kernel with $\mu = x$ and $\sigma = 1/k$, and quantize the value of this kernel into k bins. By using such an encoding strategy, only parts of the MLP would be activated for each position. Hence, the MLP is implicitly divided into multiple sub-networks for decoding the SDF values.

Sparse grid-based representations. In addition to the positional encoding, our model also contains a grid-based scene representation. The geometry representation is encoded into L levels

multi-resolution feature grid $\mathcal{G}_\theta = \{\mathcal{G}^l\}$ where $l = 1, 2, \dots L$. The spatial resolution of each level is set between the coarsest R_{min} and the finest resolution R_{max} in a progression manner:

$$R_l = R_{min} \cdot b^l, \quad \text{where } b = \exp(\frac{\ln R_{max} - \ln R_{min}}{L-1}). \tag{4.2}$$

In each vertex of $\mathcal{G}_l$, we would store a feature vector with size $F = 2$. For a regular multi-resolution grid-based representation, the number of parameters needed to represent is $\sum_{l=1}^{L} R_l^3 \cdot F$, which is a huge memory cost for scenes with large dimensions.

To address such an issue, we adopt a hash-based sparse feature grid. Specifically, in each level, we manage a feature embedding $\mathcal{E}_l$ with the size of $T \cdot F$. Given the input coordinate $\mathbf{x} \in \mathbb{R}^3$, we could get the corresponding feature by trilinear interpolation. In each level, the feature vectors at the 8 corners of the voxel that $\mathbf{x}$ is in would be queried from the $\mathcal{E}_l$. For a coarse level where $R_l^3 < T$, we could achieve a 1:1 mapping between the spatial coordinate and the feature embedding. For a fine level where $R_l^3 > T$, a hash function [Tes+03] $h : \mathbb{Z}^3 \to \mathbb{Z}_T$ is adopt to query the feature embedding:

$$h(\mathbf{x}) = \left(\bigoplus_{i=1}^{3} x_i \pi_i \right) \mod T, \tag{4.3}$$

where $\bigoplus$ is XOR while π_i denotes a large prime number. With such a hash-based representation, the number of parameters required for the grid would be reduced to $\mathcal{O}(T)$.

One remaining problem for this hash-based method is the hash collision at the fine level. Thus, a shallow MLP is adopted to decode the feature vector with dimensionality $F \cdot L + 3k$ that is generated by concatenating the trilinear interpolated feature vectors from each level and the positional encoding. With such a decoder, the hash collision would be implicitly handled by gradient-based optimisation.

View direction encoding. As the color also depends on the view direction, we adopt spherical harmonics to model the view direction. Since spherical harmonics form the orthogonal basis

over the sphere, it is natural to use this function for modeling view directions. In this thesis, we use spherical harmonics with 4 degrees for view direction encoding, the encoded vector will be concatenated with the feature from the MLP used for decoding SDF values, and input to an MLP for predicting the color.

4.3 Volumetric rendering

Our volumetric rendering method, inspired by the recent success of NeRF [Mil+20], renders both the depth and color using the sampled depth point and the predicted colors. Specifically, given the camera origin $\mathbf{o}$ and ray direction $\boldsymbol{\omega}$, we could sample N points $\mathbf{x_i} = \mathbf{o} + t_i\boldsymbol{\omega}$, $i \in \{1, \ldots, N\}$ with depth values $\{t_1, \ldots, t_N\}$ and predicted colors $\{c_1, \ldots, c_N\}$. Then, we could formulate our rendering equation as follows:

$$\hat{\mathbf{c}} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \mathbf{c}_i, \quad \hat{t} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i t_i, \tag{4.4}$$

where w is the weight function. Generally speaking, w can be any bell shape function that gives a higher weight around the surface. GO-Surf [WBA22] adopts NeuS [Wan+21] rendering equation that is unbiased and occlusion-aware. However, NeuS formulation requires first-order derivatives of the SDF function, which means we need to manage second-order derivatives in our differentiable computational graph at the cost of time complexity. In fact, given the range measurement provided by the depth sensor as supervision, a simple bell-shape function [Azi+22] is enough. Thus, our weight function is computed directly from the SDF using two sigmoid functions:

$$w_i = s\left(\frac{d_i}{tr}\right) s\left(-\frac{d_i}{tr}\right), \tag{4.5}$$

where $s(\cdot)$ is the sigmoid function and tr is the truncation distance. Figure 4.3 illustrates the

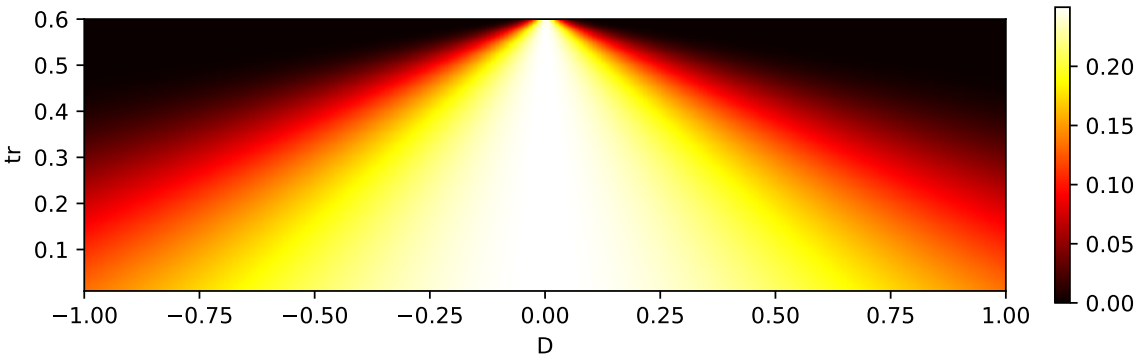


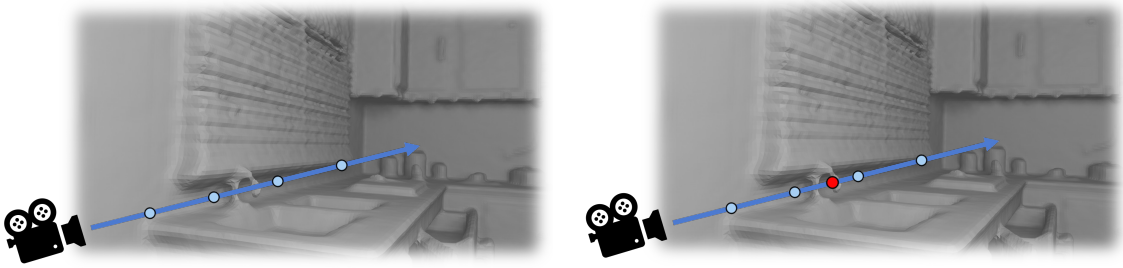
Figure 4.3: Visualisation of our weight function with respect to different SDF values (D) and truncation distance (tr). In our case, the truncation distance $tr = 0.05$.

adopted bell-shape function. This weight function achieves its peak at the surface point, i.e., the zero crossing point where $d = 0$. In order to compensate for the multiple intersections of the surfaces, the weight beyond the first truncated regions would set to be zero.

4.4 Depth-guided sampling

As aforementioned that the training of our model requires a number of sample points for each camera ray. Different from the novel view synthesis that only cares about the final rendering results, 3D reconstruction cares about the SDF value of each individual sample point. To obtain an accurate surface reconstruction without the noisy points floating around the free space, the sampling for 3D reconstruction needs to satisfy two criteria. Firstly, we need a sufficient number of sample points around the free space. Secondly, we need to make sure there exist sample points within the truncation region (i.e. around the surface).

To satisfy these criteria, Neural RGB-D [Azi+22] uses 256-640 sample points for scenes of different sizes. Instead of adopting different number of samples, GO-Surf [WBA22] takes advantage of NeuS [Wan+21] and uses 128 sample points for each scene with multiple iterations of importance sampling at the cost of maintaining a second-order computational graph. Although both of the works achieve good surface optimisation with their sampling strategy, they did not



(a) Regular sampling (b) Depth-guided sampling

Figure 4.4: Comparison of a) regular stratified sampling and b) the proposed depth-guided sampling. Using depth as a guidance to provide additional sampling can avoid missing the zero crossing point while maintain a reasonable computational cost.

take advantage of the range measurement provided by the depth sensor. Thus, we propose a simple yet effective sampling strategy (see Figure 4.4) that can be divided into three steps. In the first step, we perform general stratified sampling on the ray $\mathbf{r}_i$ between the near and far plane. In the second step, we perform an additional sampling that samples 11 additional points around $[-0.1, 0.1]$ of the depth point. In the third step, we perform the importance sampling strategy based on the weight function w obtained in previous steps. With such a simple approach, we could guarantee that we would not miss the zero-crossing points (see Figure 4.4) during the optimisation while maintaining a reasonable computational cost.

4.5 Optimisation

We jointly optimise our scene representation with the corresponding camera parameters for each frame. Specifically, for each iteration, we randomly sample N pixels from the whole training data. For each pixel i, a ray $\mathbf{r}_i$ would be generated based on the intrinsic and extrinsic parameters. Given the list of parameters $\mathcal{P}$ needed to be optimised, our global objective function can be formulated as:

$$L(\mathcal{P}) = \alpha_{rgb} L_{rgb} + \alpha_{depth} L_{depth} + \alpha_{sdf} L_{sdf} + \alpha_{fs} L_{fs}. \tag{4.6}$$

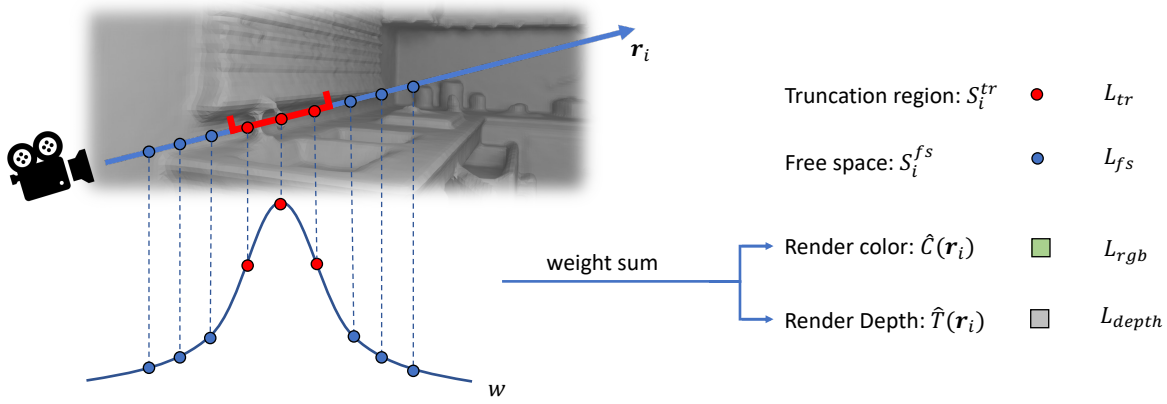


Figure 4.5: Illustration of each term in our global objective function L.

L_{rgb} and L_{depth} are the photometric loss and the depth loss respectively. The loss is generated by comparing the rendered color $\hat{C}(\mathbf{r})$ and depth $\hat{T}(\mathbf{r})$ with the ground truth color $C(\mathbf{r})$ and depth $T(\mathbf{r})$:

$$L_{rgb} = \frac{1}{N}\sum_{i=1}^{N}(\hat{C}(\mathbf{r}_i) - C(\mathbf{r}_i))^2, \quad L_{depth} = \frac{1}{N_d}\sum_{i=1}^{N_d}(\hat{T}(\mathbf{r}_i) - T(\mathbf{r}_i))^2. \tag{4.7}$$

Note that due to the incompleteness of the depth measurement, we only compute the depth loss at the N_d pixels that contain valid range measurements. L_{sdf} and L_{fs} loss are SDF loss which applied to the sample points within the truncation region $\mathbf{r}_i^{s_{tr}}$, and the sample points outside the truncation region $\mathbf{r}_i^{s_{fs}}$:

$$\mathcal{L}_{tr} = \frac{1}{N_d}\sum_{i=1}^{N_d}\frac{1}{|S_i^{tr}|}\sum_{s\in S_i^{tr}}(\hat{d}_i^s - d_i^s)^2, \quad \mathcal{L}_{fs} = \frac{1}{N_d}\sum_{i=1}^{N_d}\frac{1}{|S_i^{fs}|}\sum_{s\in S_i^{fs}}(\hat{d}_i^s - d_i^s)^2, \tag{4.8}$$

where $\hat{d}_i^s$ and d_i^s are the predicted signed distance and the approximated signed distance using the depth measurement for each point.

Geometric initialisation. To achieve fast convergence, we adopt the geometric initialisation [Gro+20] that initialises the scene into a hollow sphere that centers at the origin with a diameter equals to the smallest dimension.

Pose optimisation. The initial pose that contains a rotation matrix $\hat{\mathbf{R}} \in \mathbb{R}^{3\times3}$ and translation vector $\hat{\mathbf{t}} \in \mathbb{R}^3$ is initialised using bundlefusion [Dai+17]. To optimise the pose with our model, we define $\Delta\mathbf{R}$ using Euler angles and $\Delta\mathbf{t}$ for each frame and jointly optimise these parameters with our scene representation.

Chapter 5

Experimental Results

5.1 Setup

Datasets. We perform our experiments on 10 synthetic scenes proposed in [Azi+22]. Each scene contains an RGB-D video that goes through part of the scene, ground-truth mesh, pose, and depth images. The depth measurement of each RGB-D video has been manually modified to synthesize the inaccurate range measurement generated by a real depth camera. The training poses of each frame are generated by BundleFusion [Dai+17].

Evaluation metrics. We follow the evaluation metrics that have been adopted in [WBA22]. They are accuracy (Acc), completion (Comp), chamfer ℓ_1 distance (C-ℓ_1), normal consistency (NC), and F-score. Given the point cloud $\hat{P}$ and P generated by the predicted mesh and the ground truth mesh, Acc and Comp measure the average minimal ℓ_2 distance between the ground-truth points and predicted points:

$$Acc = \text{mean}_{\hat{p}\in\hat{P}}(\min_{p\in P} \|\hat{p} - p\|), \quad Comp = \text{mean}_{p\in P}(\min_{\hat{p}\in\hat{P}} \|\hat{p} - p\|). \tag{5.1}$$

The chamfer ℓ_1 distance can be formulated as the average of Acc and Comp:

$$C - \ell_1 = \frac{Acc + Comp}{2}. \tag{5.2}$$

The normal consistency measures the similarity between the normal vector of the predicted points $\mathbf{n}_{\hat{p}}$ and the ground truth point $\mathbf{n}_p$:

$$NC = \frac{\text{mean}_{\hat{p}\in\hat{P}}(\|\mathbf{n}_{\hat{p}} - \mathbf{n}_{p,\hat{p}}\|) + \text{mean}_{p\in P}(\|\mathbf{n}_{\hat{p},p} - \mathbf{n}_p\|)}{2}, \tag{5.3}$$

where $\mathbf{n}_{p,\hat{p}}$ is the normal vector of ground truth p that is the closet to the given prediction $\hat{p}$. F-score can be formulated as:

$$\text{F-score} = \frac{2 \times \text{ Precision } \times \text{ Recall}}{\text{Precision+Recall}}, \tag{5.4}$$

where

$$\text{Precision} = \text{mean}_{p\in P}\left(\min_{\hat{P}\in\hat{P}} \left\|p - \hat{P}\right\| < 0.05\right), \tag{5.5}$$

$$\text{Recall} = \text{mean}_{\hat{p}\in\hat{P}}\left(\min_{p\in P} \|p - \hat{p}\| < 0.05\right) \tag{5.6}$$

Baselines. We focus on the comparison of our methods with two state-of-the-art neural implicit representations for RGB-D surface reconstruction. They are NeuralRGB-D [Azi+22] and GO-Surf [WBA22]. Additionally, we also quantitatively compare our methods with some learning-based RGB-D reconstruction methods [Pen+20; Sit+20; Wed+20] as well as the traditional methods [SF16; Sch+16a; Sch+16b; Dai+17].

Implementation details. We run our experiments with a Tesla V100 GPU. The reconstruction method is implemented in PyTorch using Radam optimiser with a learning rate of $1e-2$ for grid and decoder, and $1e-3$ for camera pose optimisation. The weight for photometric loss, depth loss, free space loss, and SDF loss is set to be 0.1, 0.1, 10, and 6000 according to the scale

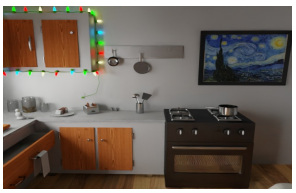
Method	Acc ↓	Com ↓	C-ℓ_1 ↓	NC ↑	F-score ↑	Run time
BundleFusion	0.0191	0.0581	0.0386	0.9027	0.8439	10min-40min
RoutedFusion	0.0223	0.0364	0.0293	0.8765	0.8736	40min-6h
COLMAP	0.0271	0.0322	0.0296	0.9134	0.8744	4h-5h
ConvOccNets	0.0498	0.0524	0.0511	0.8607	0.6822	120s
SIREN	0.0229	0.0412	0.0320	0.9049	0.8515	6h-12h
Neural RGBD	0.0151	0.0197	0.0174	0.9316	**0.9635**	15h-25h
GO-Surf	0.0158	0.0195	0.0177	**0.9317**	0.9591	15min-45min
Ours	**0.0149**	**0.0179**	**0.0164**	0.9292	0.9629	100s-500s

Table 5.1: Quantitative results of the reconstruction on 10 synthetic scenes [Azi+22]. The evaluation metrics are the same as the ones in Neural RGBD [Azi+22] and GO-Surf [WBA22]. We achieve on-par performance with the existing methods using neural implicit representation but our training is significantly faster.

of these losses. In terms of ray sampling, we use 128 sample points for stratified sampling, 11 sample points for depth-guided sampling, and 16 points for importance sampling.

5.2 Quantitative analysis

We perform the quantitative analysis on 10 synthetic scenes [Azi+22] with the ground truth mesh. The depth maps used for training are added noise to simulate the real range measurement from the depth sensor. The pose of each frame in the RGB-D video is initialised using Bundlefusion [Dai+17] and optimised with our model. Table 5.1 illustrates the quantitative results of the proposed method in comparison to various baselines. We can find that our method achieves on-par performance in comparison to NeuralRGBD [Azi+22] and GO-Surf [WBA22] but with a significantly faster training speed. In Table 5.2 we provide a detailed per-scene quantitative evaluation of the reconstruction. Usually, the performance of the hash-based feature grid would degrade with the increase of the scene size due to the hash collision. However, benefiting from the proposed hybrid scene representation, we could find that our method can still achieve competitive results in large scenes (e.g. complete kitchen, kitchen).

Scene	Method	Acc ↓	Comp ↓	C-ℓ_1 ↓	NC ↑	F-score ↑
Complete kitchen	BundleFusion	0.0303	0.1475	0.0889	0.8570	0.6943
	RoutedFusion	0.0270	0.0854	0.0562	0.8484	0.7939
	COLMAP	0.0365	0.0354	0.0360	0.9245	0.8248
	ConvOccNets	0.0502	0.0527	0.0514	0.8667	0.6610
	SIREN	0.0319	0.0700	0.0509	0.9031	0.7415
	NeuralRGBD	0.0224	0.0394	0.0309	0.9098	0.8962
	GO-Surf	0.0224	0.0258	0.0241	**0.9413**	0.8998
	Ours	**0.0200**	**0.0231**	**0.0215**	0.9073	**0.9251**
Kitchen	BundleFusion	0.0253	0.0578	0.0416	0.9112	0.7967
	RoutedFusion	0.0281	0.0362	0.0322	0.8553	0.8484
	COLMAP	0.0228	0.0282	0.0255	0.9332	0.9170
	ConvOccNets	0.0420	0.049	0.0455	0.8752	0.6253
	SIREN	0.0327	0.0575	0.0451	0.8996	0.7071
	NeuralRGBD	0.0218	0.0297	0.0257	0.9296	0.9005
	GO-Surf	0.0214	**0.0271**	**0.0243**	**0.9316**	**0.9379**
	Ours	**0.0211**	0.0274	**0.0243**	0.9302	0.9345
Breakfast room	BundleFusion	0.0129	0.0235	0.0182	0.9582	0.9606
	RoutedFusion	0.0181	0.0202	0.0191	0.9341	0.9758
	COLMAP	0.0191	0.0194	0.0192	0.9522	0.9533
	ConvOccNets	0.0311	0.0329	0.0320	0.8925	0.9602
	SIREN	0.0150	0.0454	0.0302	0.9371	0.9230
	NeuralRGBD	0.0145	0.0148	0.0146	**0.9657**	**0.9898**
	GO-Surf	**0.0144**	**0.0136**	**0.0139**	0.9629	0.9829
	Ours	0.0149	0.0144	0.0147	0.9644	0.9876
Morning apartment	BundleFusion	**0.0079**	0.0146	0.0112	0.8891	0.9740
	RoutedFusion	0.0100	0.0143	0.0121	0.8754	0.9795
	COLMAP	0.0133	0.0183	0.0158	0.8810	0.9666

	ConvOccNets	0.0408	0.0482	0.0445	0.8105	0.7912
	SIREN	0.0105	0.0146	0.0125	0.8765	0.9718
	NeuralRGBD	0.0087	0.0121	0.0104	**0.8918**	0.9866
	GO-Surf	0.0095	0.0129	0.0112	0.8874	0.9778
	Ours	0.0083	**0.0116**	**0.0099**	0.8896	**0.9868**
Grey white room	BundleFusion	0.0297	0.0456	0.0377	0.8612	0.7537
	RoutedFusion	0.0303	0.0347	0.0325	0.8531	0.7908
	COLMAP	0.0287	0.0293	0.0290	0.9013	0.9036
	ConvOccNets	0.0470	0.0488	0.0479	0.8434	0.6057
	SIREN	0.0323	0.0335	0.0329	0.8697	0.8142
	NeuralRGBD	**0.0132**	**0.0151**	**0.0142**	**0.9318**	0.9923
	GO-Surf	0.0140	0.0158	0.0149	0.9261	0.9895
	Ours	0.0150	0.0160	0.0155	0.9239	**0.9937**
White room	BundleFusion	0.0276	0.0918	0.0597	0.8788	0.7286
	RoutedFusion	0.0289	0.0430	0.0360	0.8280	0.8222
	COLMAP	0.0309	0.0342	0.0325	0.9188	0.8259
	ConvOccNets	0.0537	0.0583	0.0560	0.8653	0.5012
	SIREN	0.0276	0.0588	0.0432	0.8992	0.7788
	NeuralRGBD	0.0204	0.0256	0.0230	0.9297	**0.9551**
	GO-Surf	0.0210	0.0325	0.0268	0.9281	0.9233
	Ours	**0.0182**	**0.0231**	**0.0207**	**0.9307**	0.9542
Green room	BundleFusion	0.0118	0.0339	0.0228	0.9254	0.9314
	RoutedFusion	0.0156	0.0193	0.0174	0.9095	0.9735
	COLMAP	0.0159	0.0194	0.0177	0.9270	0.9712
	ConvOccNets	0.0548	0.0493	0.0521	0.8600	0.7434
	SIREN	0.0183	0.0253	0.0218	0.9143	0.9448
	NeuralRGBD	0.0106	0.0142	0.0124	**0.9348**	0.9913
	GO-Surf	0.0138	0.0169	0.0153	0.9256	0.9838
	Ours	**0.0096**	**0.0128**	**0.0112**	0.9335	**0.9921**

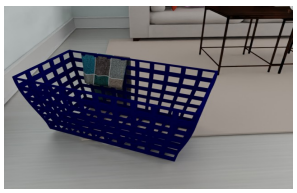
Staircase	BundleFusion	0.0257	0.1146	0.0701	0.8792	0.7108
	RoutedFusion	0.0411	0.0512	0.0461	0.8909	0.6896
	COLMAP	0.0454	0.058	0.0517	0.9253	0.6875
	ConvOccNets	0.0618	0.0562	0.059	0.8601	0.5646
	SIREN	0.0355	0.0514	0.0434	0.9117	0.7487
	NeuralRGBD	**0.0216**	**0.0254**	**0.0235**	0.9471	**0.9333**
	GO-Surf	0.0221	0.0257	0.024	**0.9496**	0.9235
	Ours	0.0229	0.0265	0.0247	0.9458	0.8911
Thin geometry	BundleFusion	0.0072	0.0305	0.0188	0.9063	0.9199
	RoutedFusion	**0.0070**	0.0396	0.0233	0.8243	0.8785
	COLMAP	0.0372	0.0558	0.0465	0.8181	0.7209
	ConvOccNets	0.0115	0.0329	0.0222	0.8800	0.9072
	SIREN	0.0086	0.0335	0.0210	0.8823	0.9115
	NeuralRGBD	0.0079	**0.0092**	**0.0086**	**0.9077**	**0.9956**
	GO-Surf	0.0093	0.0121	0.0107	0.8986	0.9817
	Ours	0.0093	0.0121	0.0107	0.8978	0.9712
ICL living room	BundleFusion	0.0129	0.0214	0.0172	0.9606	0.9694
	RoutedFusion	0.0168	0.0201	0.0185	0.9456	0.9841
	COLMAP	0.0209	0.0238	0.0224	0.9528	0.9730
	ConvOccNets	0.1049	0.0956	0.1003	0.8535	0.5619
	SIREN	0.0167	0.0219	0.0193	0.9555	0.9734
	NeuralRGBD	**0.0095**	**0.0115**	**0.0105**	**0.9689**	**0.9944**
	GO-Surf	0.0105	0.0127	0.0117	0.9661	0.9909
	Ours	0.0100	0.0125	0.0113	0.9685	0.9933

Table 5.2: We compare the reconstruction results on 10 synthetic scenes [Azi+22]. The evaluation metrics include accuracy, completion, chamfer ℓ_1 distance, normal consistency and F-score.

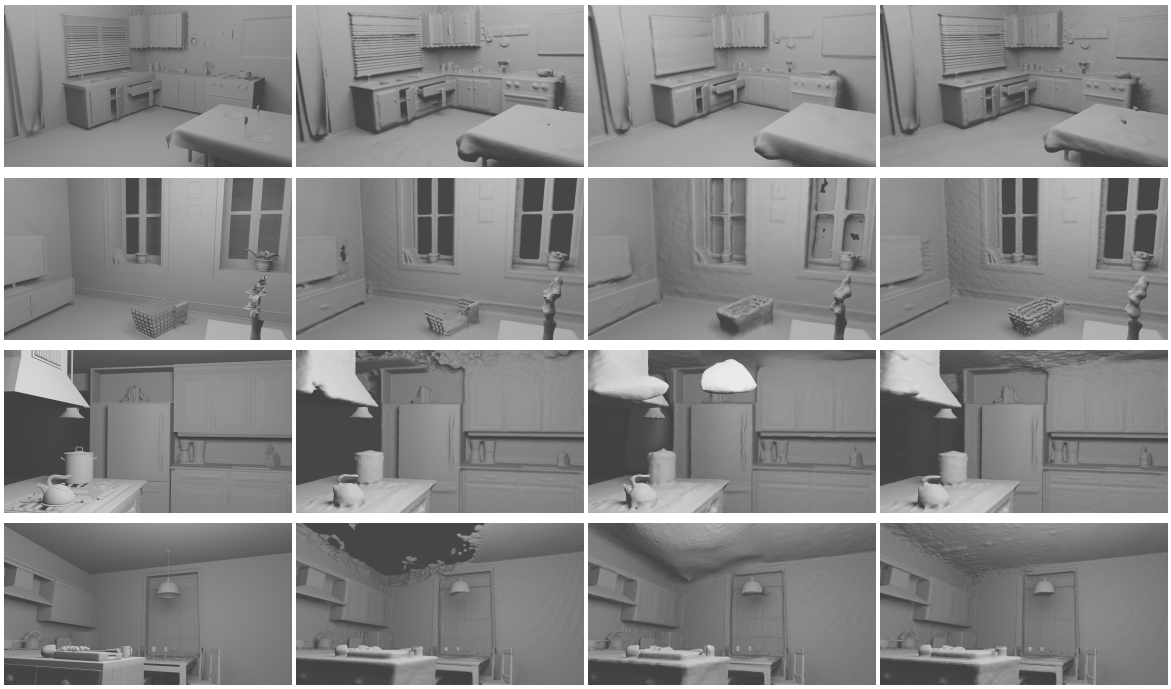


Figure 5.1: We compare the reconstruction results of the proposed method with neural-RGBD [Azi+22] and GO-Surf [WBA22]. Our method is able to achieve a good scene completion. At the mean time, the fine details are preserved.

5.3 Qualitative analysis

Figure 5.1 shows the reconstruction results of NeuralRGBD [Azi+22], GO-Surf [WBA22], and our method. In comparison to NeuralRGBD, GO-Surf and our method can achieve better scene completion that fills up the holes in floors and ceilings in row 1 and row 4. However, instead of relying on regularisation as GO-Surf [WBA22], our scene completion is achieved by the continuity of the MLP, and thus will not introduce bias that would smooth out the fine details like the shutters in the first scene and the basket in the second scene. Also, we surprisingly find that using OneBlob encoding with a feature grid can implicitly learn the cubic shape and complete the background surface that has not been observed in our scene. For instance, in 4-th row of Figure 5.1, part of the ceiling has not been observed. However, our method can still extrapolate the unobserved regions using the continuity of the MLPs.

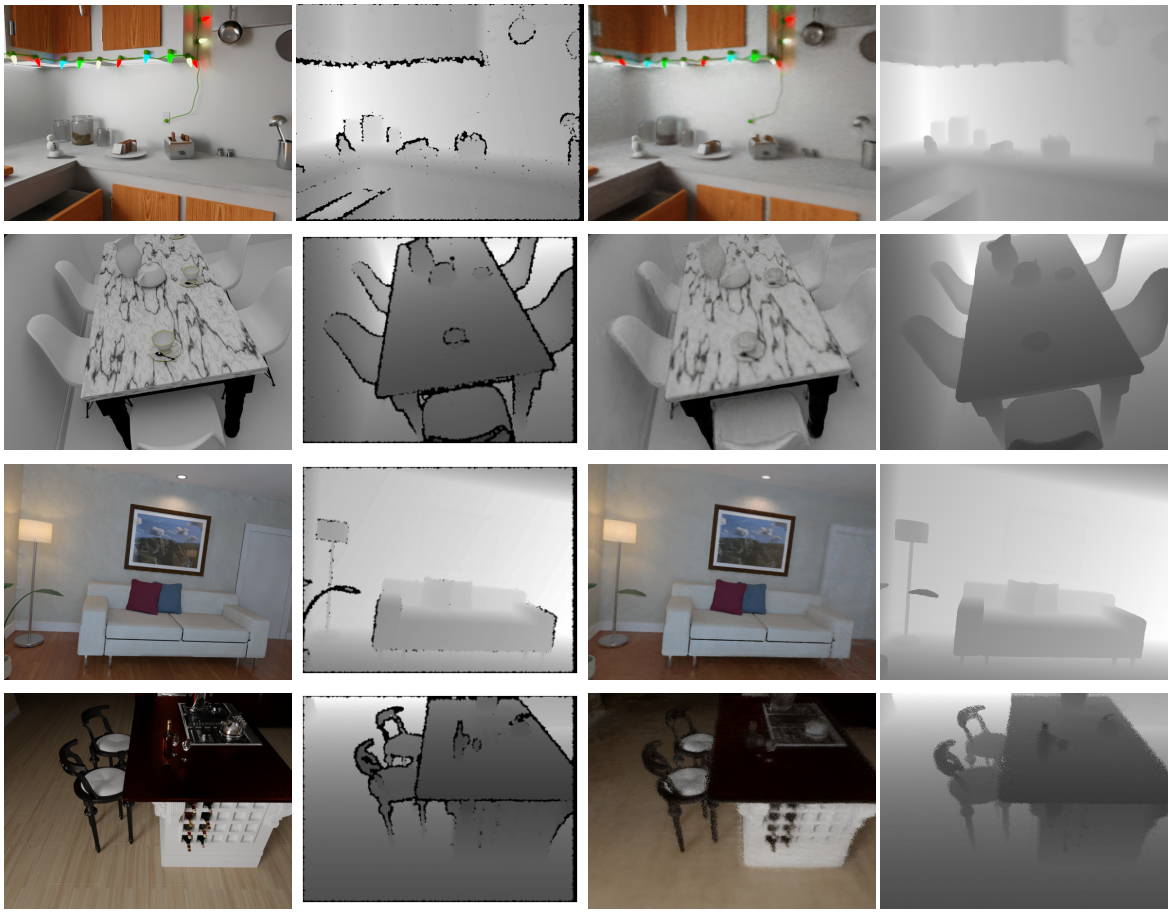
scene	scene dimension	num params (GO-Surf)	num params (Ours)	runtime (GO-Surf)	runtime (Ours)
breakfast room	$4.1 \times 4.7 \times 3.3$	20.1 M	11.5M	25min	300s
complete kitchen	$9.3 \times 10.0 \times 3.3$	73.1M	11.5M	45min	500s
green room	$8.0 \times 4.7 \times 3.1$	36.0M	11.5M	30min	300s
grey white room	$5.9 \times 4.4 \times 3.1$	23.4M	11.5M	24min	300s
ICL living room	$5.3 \times 2.9 \times 5.4$	24.1M	11.5M	24min	500s
kitchen	$7.0 \times 8.7 \times 3.4$	53.2M	11.5M	40min	200s
morning apartment	$3.5 \times 4.0 \times 2.3$	10.1M	11.5M	19min	100s
staircase	$6.8 \times 6.5 \times 3.7$	46.6M	11.5M	39min	500s
thin geometry	$3.4 \times 3.6 \times 1.2$	5.4M	11.5M	15min	400s
white room	$5.6 \times 7.9 \times 3.8$	44.9M	11.5M	39min	500s

Table 5.3: Comparison of the runtime and memory performance of GO-Surf [WBA22] and the proposed method on synthetic scenes. Our method achieve faster training speed with constant memory requirement for each scene.

5.4 Runtime and memory analysis

In Table 5.3, we compare the runtime and memory usage with GO-Surf, which is a similar approach to ours. In comparison to Go-Surf, our method requires a significantly smaller memory footprint using the feature grid with the same maximum resolution. The space complexity of our method is $\mathcal{O}(T)$ in comparison to $\mathcal{O}(N^3)$ of the GO-Surf. Additionally, since our scene representation is a hybrid approach, the memory usage can be further compressed without sacrificing too much accuracy (see Section 5.6.1).

In terms of computational cost, our method achieves a significantly faster optimisation speed while realising on-par performance with other state-of-the-art methods. According to our analysis, there are two main reasons for achieving faster convergence. Firstly, our method uses the hash function to index the feature. Secondly, thanks to the proposed hybrid scene representation with depth-guided sampling, our method requires neither the maintenance of any second-order computational graph nor regularisation over the entire feature grid.

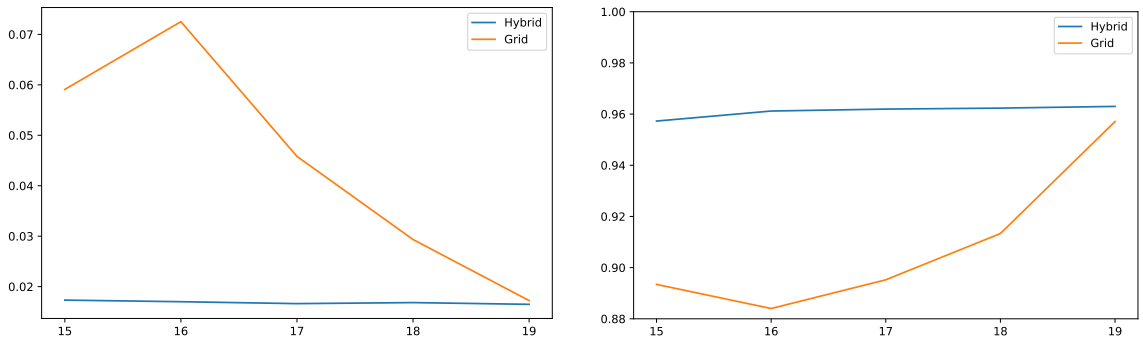


Input RGB | Input depth | Rendered RGB | Rendered depth

Figure 5.2: Visualisation of the rendered RGB and depth using the proposed Hybrid-Surf. The holes and noise in the input depth images are filled up with our Hybrid-Surf.

5.5 Rendering results

Figure 5.2 shows the rendered color and depth using the proposed Hybrid-Surf. We can find that input depth maps contain lots of regions with missing measurements. Thanks to the continuity of the MLPs, our method can fill up the incomplete depth maps and filter out the noise. In terms of color rendering, we can find that although our method produces rendering results with high quality, some thin structures may not be recovered. Additionally, we find that rendering requires accurate predictions on all sample points in both truncated regions and free spaces. Thus, a large scene usually requires a longer time to optimise the rendering results than to optimise the signed



(a) $C - \ell_1$ (b) F-score

Figure 5.3: The performance change of grid-based approach and our hybrid approach with respect to the size of the hash table on 10 synthetic scenes. We could find that using the proposed hybrid scene representation is helpful for MLPs to deal with the hash collision problem.

distance field. The 4th row shows the rendering results on a large scene, complete kitchen. We can observe that there is still some noise in the rendered color and depth maps while the mesh extracted by marching cubes [LC87] has already been optimal at that iteration. To address such an issue, using another feature grid for color reproduction may be helpful (see Appendix B.3 for details).

5.6 Ablation studies

In this section, we perform various ablation studies to validate the effectiveness of various components in our proposed method.

5.6.1 Effect of the positional encoding

Figure 5.3 illustrates the comparison of the proposed hybrid sparse scene representation with the sparse grid-based representation on 10 synthetic scenes. Generally speaking, with the decrease of the hash table size, the possibility of hash collision will increase, and thus degrade the quality of the reconstruction. However, we could find that with the proposed hybrid sparse representation, which has an additional positional encoding, the performance is rather stable with the decrease

Method	Acc ↓	Com ↓	C-ℓ_1 ↓	NC ↑	F-score ↑	Memory
Neural RGBD	0.0151	0.0197	0.0174	0.9316	**0.9635**	**0.6M**
GO-Surf	0.0158	0.0195	0.0177	**0.9317**	0.9591	5.4M-73.1M
Hybrid-Surf-15	0.0150	0.0195	0.0173	0.9304	0.9572	0.8M
Hybrid-Surf-16	0.0153	0.0186	0.0170	0.9296	0.9611	1.5M
Hybrid-Surf-17	0.0151	0.0181	0.0166	0.9296	0.9619	2.9M
Hybrid-Surf-18	0.0152	0.0183	0.0168	0.9305	0.9623	5.8M
Hybrid-Surf-19	**0.0149**	**0.0179**	**0.0164**	0.9292	0.9629	11.5M

Table 5.4: Comparison of memory usage of NeuralRGBD [Azi+22], GO-Surf [WBA22] and the proposed Hybrid-Surf with different hash table size on 10 synthetic scenes [Azi+22]. We can find that the memory usage of our method can be reduced to 0.8M while still achieve on par performance in comparison to NeuralRGBD [Azi+22] and GO-Surf [WBA22]. However, our method is significantly faster.

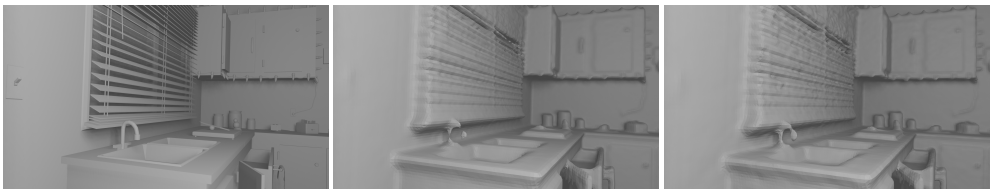
of the hash table size. The results show that the positional encoding is helpful to deal with the potential hash collision and thus can be also used for the compression of the model. As shown in Table 5.4, the memory usage of Hybrid-Surf can be compressed to 0.8M while still achieve on par performance in comparison NeuralRGBD [Azi+22] and GO-Surf [WBA22].

5.6.2 Effect of the depth measurement

Figure 5.4 visualises the comparison of the reconstruction results with and without the depth measurements. From which we could find that although using only RGB images can realise the scene reconstruction via minimising the photometric loss, the reconstruction results are noisy (see Figure 5.4(b)). Additionally, the small object or thin structure cannot be recovered very well. Given the depth measurement in addition to the RGB images, we could approximate the SDF values to supervise our model. By minimising our geometry loss as well as the photometric loss, we can achieve accurate, high-fidelity reconstructions. Figure 5.4(c) shows the reconstruction with depth measurements. We can observe that using depth measurements can significantly boost the reconstruction quality.

a) GT b) w/o depth c) w/ depth

Figure 5.4: Visualisation of the reconstruction result b) without depth measurement c) with depth measurement. Using depth measurements to supervise the model by approximating SDF values can significantly boost the reconstruction quality.



a) GT b) w/o guidance c) w/ guidance

Figure 5.5: Visualisation of the reconstruction result b) without depth-guided sampling c) with depth-guided sampling. Our depth-guided sampling is good at reconstruction of thin structures.

5.6.3 Effect of the depth-guided sampling

Regular stratified sampling is usually not good at thin structures as we may not have sample points within the truncation regions. Thus, we propose depth-guided sampling that provides additional sampling around the depth. With such a strategy, we could ensure that there exists sampling within the truncation region for each camera ray. Figure 5.5 illustrates the effectiveness of our method. We could observe that using the proposed depth-guided sampling, we are able to reconstruct the water tap while regular stratified sampling cannot deal with such a thin structure.

Chapter 6

Discussion and Future Works

In this thesis, a hybrid scene representation, that combines coordinate-based and grid-based representations, is proposed. While the proposed method achieves competitive reconstruction results with RGB-D cameras, there are also several potential directions that are worth to further exploring.

6.1 Tri-plane representation

Using hash-based representation can significantly reduce memory usage. However, since the grid feature is mapped to a compact representation, it becomes hard to conduct some further manipulations to the trained features. Thus, we could investigate the tri-plane representations that are proposed in EG3D [Cha+22]. This method uses three planes to represent the scene and shows competitive results in comparison to the regular feature grid with significantly less memory usage. In comparison to hash-based representation, tri-plane is more explainable and we can easily perform some post-processing on the learned feature representation.

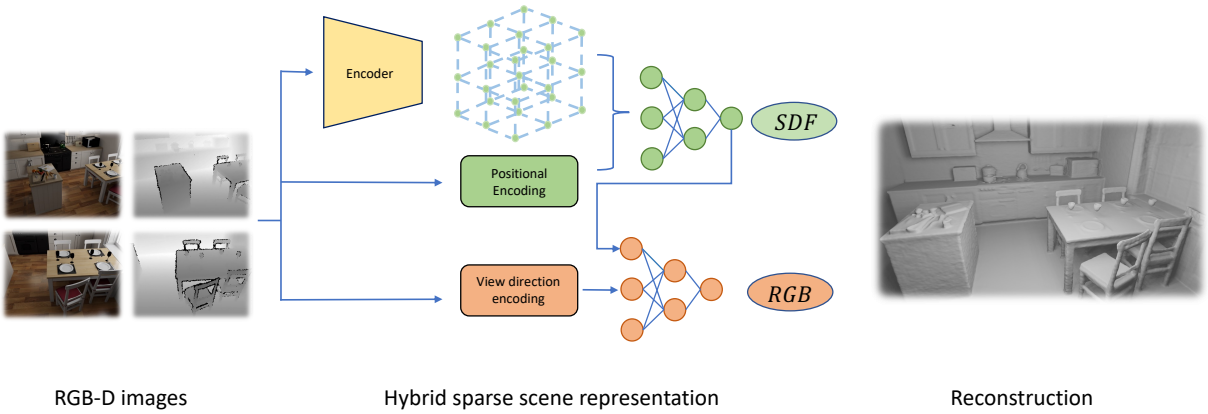


Figure 6.1: The potential directions for training an encoder that avoids the per-scene optimisation.

6.2 Representation learning

One downside of the proposed approach is that we need per-scene optimisation. That is to say, we have to train our model from scratch for each scene. However, we could use convolutional neural networks to learn a feature encoding from RGB-D images and map such encoding into 3D feature volume with the corresponding intrinsic and extrinsic parameters as shown in Figure 6.1. By doing so, we could avoid the test time training of the grid-based methods and learn some prior information, such as appearance, shape, and texture through large-scale offline training.

Chapter 7

Conclusion

In this thesis, we investigate the problem of 3D reconstruction with an RGB-D camera. To achieve an accurate and efficient RGB-D scene reconstruction system, we take advantage of the recent advances in neural representation and propose Hash-Surf, which is a hybrid neural representation that combines coordinate-based and grid-based representation. In order to prevent the cost of a large memory footprint and reduce the time for training, we adopt the sparse grid-based neural representation that is based on the Hash function. However, one downside of the grid-based methods is that they do not fully exploit the continuity of the MLPs for scene completion. In addition, due to the hash collision, we need to have a trade-off between memory and accuracy. To solve the above-mentioned problems, we propose a simple but effective method, that gives positional encoding in addition to the interpolated features for the decoder. With such a strategy, we could exploit the continuity of the MLP to complete the scene, and in the meantime, alleviate the problem caused by hash collision. As a result, the proposed method can achieve a further 16 times compression in comparison to the original hash-based representation with only minimal degradation in performance. Further works include investigating a. sparse feature grid that is more explainable and avoiding the per-scene optimisation by training an encoder that can encode the 3D feature volume from a set of posed RGB-D images.

References

[Bli82] James F. Blinn. “Light Reflection Functions for Simulation of Clouds and Dusty Surfaces”. In: *Acm Siggraph Computer Graphics* 16.3 (1982), pp. 21–29.

[KV84] James T. Kajiya and Brian P. Von Herzen. “Ray Tracing Volume Densities”. In: *ACM SIGGRAPH computer graphics* 18.3 (1984), pp. 165–174.

[LC87] William E. Lorensen and Harvey E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169.

[BM92] Paul J. Besl and Neil D. McKay. “Method for Registration of 3-D Shapes”. In: *Sensor Fusion IV: Control Paradigms and Data Structures*. Vol. 1611. Spie, 1992, pp. 586–606.

[CM92] Yang Chen and Gérard Medioni. “Object Modelling by Registration of Multiple Range Images”. In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[Max95] Nelson Max. “Optical Models for Direct Volume Rendering”. In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108.

[CL96] Brian Curless and Marc Levoy. “A Volumetric Method for Building Complex Models from Range Images”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. 1996, pp. 303–312.

[TM98] Carlo Tomasi and Roberto Manduchi. “Bilateral Filtering for Gray and Color Images”. In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998, pp. 839–846.

[Pfi+00] Hanspeter Pfister et al. “Surfels: Surface Elements as Rendering Primitives”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. 2000, pp. 335–342.

[Tur00] Greg Turk. *Stanford Bunny*. `https://faculty.cc.gatech.edu/~turk/bunny/bunny.html`. Accessed: 2022–8-26. 2000.

[RL01] Szymon Rusinkiewicz and Marc Levoy. “Efficient Variants of the ICP Algorithm”. In: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 145–152.

[Tes+03] Matthias Teschner et al. “Optimized Spatial Hashing for Collision Detection of Deformable Objects.” In: *Vmv*. Vol. 3. 2003, pp. 47–54.

[Low04] Kok-Lim Low. “Linear Least-Squares Optimization for Point-to-Plane Icp Surface Registration”. In: *Chapel Hill, University of North Carolina* 4.10 (2004), pp. 1–3.

[FG11] Simon Fuhrmann and Michael Goesele. “Fusion of Depth Maps with Multiple Scales”. In: *ACM Transactions on Graphics (TOG)* 30.6 (2011), pp. 1–8.

[Iza+11] Shahram Izadi et al. “KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. 2011, pp. 559–568.

[Mai+11] Lena Maier-Hein et al. “Convergent Iterative Closest-Point Algorithm to Accomodate Anisotropic and Inhomogenous Localization Error”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.8 (2011), pp. 1520–1532.

[New+11] Richard A. Newcombe et al. “Kinectfusion: Real-time Dense Surface Mapping and Tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2011, pp. 127–136.

[And13] Carlos Andujar. *Octrees*. `https://www.cs.upc.edu/~virtual/SGI/docs/1.%20Theory/Unit%2008.%20Octrees/Octrees.pdf`. Accessed: 2022–8-26. 2013.

[Cha13] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Courier Corporation, 2013.

[CBI13] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. “Scalable Real-Time Volumetric Surface Reconstruction”. In: *ACM Transactions on Graphics (ToG)* 32.4 (2013), pp. 1–16.

[Kel+13] Maik Keller et al. “Real-Time 3d Reconstruction in Dynamic Scenes Using Point-Based Fusion”. In: *2013 International Conference on 3D Vision-3DV 2013*. IEEE, 2013, pp. 1–8.

[Nie+13a] Matthias Nießner et al. “Real-Time 3D Reconstruction at Scale Using Voxel Hashing”. In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–11.

[Nie+13b] Matthias Nießner et al. “Real-Time 3D Reconstruction at Scale Using Voxel Hashing”. In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–11.

[SKC13] Frank Steinbrucker, Christian Kerl, and Daniel Cremers. “Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 3264–3271.

[CZK15] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. “Robust Reconstruction of Indoor Scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5556–5565.

[LWK15] Damien Lefloch, Tim Weyrich, and Andreas Kolb. “Anisotropic Point-Based Fusion”. In: *2015 18th International Conference on Information Fusion (Fusion)*. IEEE, 2015, pp. 2121–2128.

[Whe+15] Thomas Whelan et al. “ElasticFusion: Dense SLAM without a Pose Graph”. In: Robotics: Science and Systems, 2015.

[Par+16] Jaesik Park et al. “Robust Multiview Photometric Stereo Using Planar Mesh Parameterization”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.8 (2016), pp. 1591–1604.

[SF16] Johannes L. Schonberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113.

[Sch+16a] Johannes L. Schönberger et al. “A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval”. In: *Asian Conference on Computer Vision*. Springer, 2016, pp. 321–337.

[Sch+16b] Johannes L. Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision*. Springer, 2016, pp. 501–518.

[Dai+17] Angela Dai et al. “Bundlefusion: Real-time Globally Consistent 3d Reconstruction Using on-the-Fly Surface Reintegration”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), p. 1.

[Zol+18] Michael Zollhöfer et al. “State of the Art on 3D Reconstruction with RGB-D Cameras”. In: *Computer Graphics Forum*. Vol. 37. Wiley Online Library, 2018, pp. 625–652.

[Mes+19] Lars Mescheder et al. “Occupancy Networks: Learning 3d Reconstruction in Function Space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470.

[Mül+19] Thomas Müller et al. “Neural Importance Sampling”. In: *ACM Transactions on Graphics (TOG)* 38.5 (2019), pp. 1–19.

[Par+19] Jeong Joon Park et al. “Deepsdf: Learning Continuous Signed Distance Functions for Shape Representation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[CAP20] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. “Implicit Functions in Feature Space for 3d Shape Reconstruction and Completion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6970–6981.

[Gro+20] Amos Gropp et al. “Implicit Geometric Regularization for Learning Shapes”. In: *arXiv preprint arXiv:2002.10099* (2020). arXiv: 2002.10099.

[Mil+20] Ben Mildenhall et al. “Nerf: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *European Conference on Computer Vision*. Springer, 2020, pp. 405–421.

[Mur+20] Zak Murez et al. “Atlas: End-to-end 3d Scene Reconstruction from Posed Images”. In: *European Conference on Computer Vision*. Springer, 2020, pp. 414–431.

[Pen+20] Songyou Peng et al. “Convolutional Occupancy Networks”. In: *European Conference on Computer Vision*. Springer, 2020, pp. 523–540.

[Sch+20] Katja Schwarz et al. “Graf: Generative Radiance Fields for 3d-Aware Image Synthesis”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20154–20166.

[Sit+20] Vincent Sitzmann et al. “Implicit Neural Representations with Periodic Activation Functions”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7462–7473.

[Wed+20] Silvan Weder et al. “Routedfusion: Learning Real-Time Depth Map Fusion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4887–4897.

[Boz+21] Aljaz Bozic et al. “Transformerfusion: Monocular Rgb Scene Reconstruction Using Transformers”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 1403–1414.

[Gaf+21] Guy Gafni et al. “Dynamic Neural Radiance Fields for Monocular 4d Facial Avatar Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8649–8658.

[Gao+21] Chen Gao et al. “Dynamic View Synthesis from Dynamic Monocular Video”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5712–5721.

[JA21] Wonbong Jang and Lourdes Agapito. “Codenerf: Disentangled Neural Radiance Fields for Object Categories”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12949–12958.

[Li+21] Zhengqi Li et al. “Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6498–6508.

[Mar+21] Ricardo Martin-Brualla et al. “Nerf in the Wild: Neural Radiance Fields for Unconstrained Photo Collections”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7210–7219.

[Mor+21] Luca Morreale et al. “Neural Surface Maps”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4639–4648.

[NG21] Michael Niemeyer and Andreas Geiger. “Giraffe: Representing Scenes as Compositional Generative Neural Feature Fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11453–11464.

[Nog+21] Atsuhiro Noguchi et al. “Neural Articulated Radiance Field”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5762–5772.

[OPG21] Michael Oechsle, Songyou Peng, and Andreas Geiger. “Unisurf: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5589–5599.

[Par+21a] Keunhong Park et al. “Hypernerf: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields”. In: *arXiv preprint arXiv:2106.13228* (2021). arXiv: 2106.13228.

[Par+21b] Keunhong Park et al. “Nerfies: Deformable Neural Radiance Fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5865–5874.

[Pum+21] Albert Pumarola et al. “D-Nerf: Neural Radiance Fields for Dynamic Scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10318–10327.

[Suc+21] Edgar Sucar et al. “iMAP: Implicit Mapping and Positioning in Real-Time”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6229–6238.

[Sun+21] Jiaming Sun et al. “NeuralRecon: Real-time Coherent 3D Reconstruction from Monocular Video”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15598–15607.

[Wan+21] Peng Wang et al. “Neus: Learning Neural Implicit Surfaces by Volume Rendering for Multi-View Reconstruction”. In: *arXiv preprint arXiv:2106.10689* (2021). arXiv: 2106.10689.

[Xia+21] Wenqi Xian et al. “Space-Time Neural Irradiance Fields for Free-Viewpoint Video”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9421–9431.

[Azi+22] Dejan Azinović et al. “Neural RGB-D Surface Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6290–6301.

[Cha+22] Eric R. Chan et al. “Efficient Geometry-Aware 3D Generative Adversarial Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16123–16133.

[Che+22] Anpei Chen et al. “TensoRF: Tensorial Radiance Fields”. In: *arXiv preprint arXiv:2203.09517* (2022). arXiv: 2203.09517.

[Fri+22] Sara Fridovich-Keil et al. “Plenoxels: Radiance Fields Without Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5501–5510.

[Guo+22] Haoyu Guo et al. “Neural 3D Scene Reconstruction with the Manhattan-world Assumption”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5511–5520.

[Kar+22] Animesh Karnewar et al. “ReLU Fields: The Little Non-linearity That Could”. In: *arXiv preprint arXiv:2205.10824* (2022). arXiv: 2205.10824.

[Lah+22] Jean Lahoud et al. “3D Vision with Transformers: A Survey”. In: *arXiv preprint arXiv:2208.04309* (2022). arXiv: 2208.04309.

[Mül+22] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *arXiv preprint arXiv:2201.05989* (2022). arXiv: 2201.05989.

[Ort+22] Joseph Ortiz et al. “iSDF: Real-Time Neural Signed Distance Fields for Robot Perception”. In: *arXiv preprint arXiv:2204.02296* (2022). arXiv: 2204.02296.

[SSC22] Cheng Sun, Min Sun, and Hwann-Tzong Chen. “Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5459–5469.

[Sun+22] Jiaming Sun et al. “Neural 3D Reconstruction in the Wild”. In: *arXiv preprint arXiv:2205.12955* (2022). arXiv: 2205.12955.

[Tan+22] Jiaxiang Tang et al. “Compressible-Composable NeRF via Rank-residual Decomposition”. In: *arXiv preprint arXiv:2205.14870* (2022). arXiv: 2205.14870.

[Wan+22] Jiepeng Wang et al. “NeuRIS: Neural Reconstruction of Indoor Scenes Using Normal Priors”. In: *arXiv preprint arXiv:2206.13597* (2022). arXiv: 2206.13597.

[WBA22] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. “GO-Surf: Neural Feature Grid Optimization for Fast, High-Fidelity RGB-D Surface Reconstruction”. In: *arXiv preprint arXiv:2206.14735* (2022). arXiv: 2206.14735.

Appendix A

Derivation

A.1 Depth to normal

Let us consider a function f that gives the depth value $f(x, y)$ at the point $(x, y, f(x, y))$. Assume the normal vector of such point is $(a(x, y), b(x, y), c(x, y))$, we can obtain:

$$\frac{\partial f}{\partial x} = -\frac{a(x,y)}{c(x,y)}, \frac{\partial f}{\partial y} = -\frac{b(x,y)}{c(x,y)}. \tag{A.1}$$

A.2 Volumetric formulation

Transmittance $T(t)$ and volume density $\sigma(t)$ can be related as follows:

$$\begin{aligned} & P(\text{survive at } t+dt) = P(\text{survive before } t) \times P(\text{survive at } t) \\ & \Rightarrow T(t+dt) = T(t) \times (1 - \sigma(t)dt) \\ \Rightarrow \quad & T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt \\ & \Rightarrow \quad \frac{T'(t)}{T(t)}dt = -\sigma(t)dt \\ & \Rightarrow \quad \ln T(t) = -\int_{t_0}^{t} \sigma(s)ds \end{aligned} \tag{A.2}$$

Appendix B

More Experiments

B.1 NeuS rendering

B.1.1 Qualitative analysis

Figure B.1 illustrates the qualitative results of using NeuS rendering equation for reconstruction. We can observe that although the scene is successfully reconstructed, there are some noisy points around the free space. Based on our analysis, such a problem might be caused by hash collision. Since our model is based on a hash-based feature grid, the decoder needs to not only learn to decode the features but also address the Hash collision. However, using NeuS [Wan+21] rendering requires the eikonal regularisation that forces the unit gradient norm. By directly regularising the gradient with our prior knowledge, the decoder may not be able to perfectly handle the hash collision through the gradient descent. Thus, using NeuS rendering with hash-based feature grid may require careful tuning of various hyper-parameters.

Figure B.1: Qualitative results of using NeuS [Wan+21] rendering. We can find that many noisy points are around the free space.

B.1.2 Rendering normal

During our experiment with NeuS [Wan+21], we observe that eikonal regularisation only considers the magnitude of the gradient instead of the direction. Thus, using such regularisation may degrade the performance of the reconstruction. Thus, we, instead, supervise the gradient by directly rendering the normal. Recall from the previous sections that we already have the weight for rendering the color and depth. Thus, it is a natural idea to use such weight to render the normal $\hat{\mathbf{n}}$:

$$\hat{\mathbf{n}} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \hat{\mathbf{n}}_i. \tag{B.1}$$

Figure B.2 illustrates results of the rendered normal. We can find that although the rendered normal contains some noise, the main structure of the scene is retained. This method may not be suitable for hash-based feature grid, however, replacing the eikonal term with normal rendering might be helpful for other grid-based or coordinate-based methods that use NeuS [Wan+21]

a) rendered image b) rendered depth c) rendered normal

Figure B.2: Visualisation of the rendered a) RGB image b) depth image c) normal image.

rendering.

B.2 Online mapping

We have also done experiments of online mapping that achieve the reconstruction in an online fashion, i.e. processing the video frame by frame. Specifically, for each frame, we sample 1024 rays for training. Additionally, we also maintain a keyframe database. The 1024 rays for training would be selected from the current frame and our keyframe database. Figure B.3 illustrates the reconstruction with our online mapping strategy. We can find that although the main structure of the scene is reconstructed, the results are not very smooth. There are two reasons for such effect. Firstly, we do not have a sufficient number of training rays in order to make our online mapping operate in real time. Secondly, we do not have any pose optimisation strategy that fits the online mapping. In order to achieve a perfect reconstruction that is accurate and smooth, we could investigate the pose estimation strategy that operates in online fashion with our neural

Figure B.3: Qualitative results of online mapping. Due to the absence of pose optimisation and insufficient number of training rays, the reconstruction is not very smooth.

representations. Also, we can also investigate some regularisation methods. For instance, the smoothness term proposed in GO-Surf [WBA22] is extremely useful in such case.

B.3 Separate decoder

We find that although using the proposed Hybrid-Surf would achieve good results on both reconstruction and color rendering, the color prediction on the surface points without volume rendering may not be very accurate. Thus, we suggest predicting the color of the surface point using the results generated by volume rendering. Otherwise, we should separate the decoder for color and SDF. Specifically, for each feature vector in our grid, we use half of them for the SDF decoder and half of them for the color decoder. By doing so, we would double the memory usage but can achieve a more accurate color prediction for each surface point. Figure B.4 illustrates the comparison of the color mesh using the color prediction on the surface points generated by Hybrid-Surf and Hybrid-Surf with separate decoders. We can find that with separate decoders,

(a) w/o separate decoder (b) w/ separate decoder

Figure B.4: The comparison of color mesh generated by Hybrid-Surf a) w/o separate decoder b) with separate decoder. Note that we cannot obtain the groundtruth color mesh, so we do not have ground truth here.

we can obtain a clean color mesh that does not contain noisy colors on the table. Note that without using volume rendering for color reproduction, we may not model the effect on lights and shadows with the change of the view directions. Thus, in practice, we recommend the usage of volume rendering for color reproduction on the mesh.