UTS UNIVERSITY OF TECHNOLOGY SYDNEY

3D Scene Reconstruction with Explicit Geometric Modeling

by Ancheng Lin

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of
Dr. Jun Li,
Prof. Paul Kennedy

University of Technology Sydney
Faculty of Engineering and Information Technology

June 2025

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Ancheng Lin*, declare that this thesis is submitted in fulfilment of the requirements for the award of *Doctor of Philosophy*, in the *School of Computer Science, Faculty of Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

SIGNATURE: Production Note:
Signature removed prior to publication.

Ancheng Lin

DATE: 18th August, 2025

PLACE: Sydney, Australia

ABSTRACT

3D scene reconstruction aims to recover the geometric and visual structure of real-world environments from sensor data such as images or LiDAR point clouds. Existing methods generally fall into two categories: explicit approaches, which represent geometry using discrete primitives like points and meshes, and implicit approaches, which encode scenes as continuous functions. While recent advances have led to impressive performance in both appearance modeling and view synthesis, many existing methods lack explicit geometric modeling, which limits their utility in physically grounded applications such as simulation, interaction, and control. This thesis addresses this limitation by improving explicit geometric modeling in 3D reconstruction from two complementary perspectives: *data enhancement* and *representation design*, with a particular focus on learning-based approaches.

From the data enhancement perspective, we enrich the explicit geometric information of raw LiDAR point clouds by estimating per-point surface normals, aided by visual semantics from images. Specifically, we propose the Hybrid Geometric Transformer (HGT), a transformer-based neural network that fuses visual semantic and 3D geometric information while capturing long-range dependencies to improve global consistency. Experimental results demonstrate its superior performance over existing methods, not only in terms of normal estimation accuracy but also in significantly improving subsequent surface reconstruction. Trained in a simulated 3D traffic environment, the model learns transferable geometric knowledge that generalizes well to real-world 3D scenes, such as those in the KITTI dataset.

While enhancing input data acts as a foundational step for geometric accuracy, robust reconstruction also necessitates scene representations that intrinsically enforce structural consistency. To address this challenge in static environments, we present a hybrid representation that integrates 3D Gaussian Splatting (3DGS) with a mesh, enhancing the surface accuracy of the originally unstructured Gaussians in 3DGS. Our model jointly learns geometry and appearance in an end-to-end manner by binding 3D Gaussians to mesh faces and employing differentiable rendering for photometric supervision. This creates an effective information pathway for the joint optimization of 3DGS and mesh learning. Experimental results show that the learned scene model achieves state-of-the-art efficiency and rendering quality while enabling scene manipulation via the explicit mesh.

Extending the scope of explicit geometry beyond static scenes, we further tackle the complexity of dynamic environments, where most existing methods rely on implicit deformation fields that are often inefficient and physically ambiguous. We propose the Dynamic Appearance Particle Neural Radiance Field (DAP-NeRF), which introduces explicit appearance particles into the implicit field framework. This approach targets dynamic scenes, which pose greater challenges than static 3D reconstruction. DAP-NeRF consists of a static field and a dynamic field, with the latter represented as a collection of appearance particles that carry visual information and are governed by learnable motion models. All components, including the static field, visual features, and particle motion models, are learned from monocular videos without prior geometric knowledge. We further develop an efficient computational framework for this particle-based model and construct a new dataset to evaluate motion modeling. Experimental results show that DAP-NeRF effectively captures both the appearance and physically meaningful motion in dynamic 3D scenes.

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisors, Dr. Jun Li and Prof. Paul Kennedy, for their unwavering support and mentorship over the past eight years. From recognizing my potential during my undergraduate studies to guiding me throughout my doctoral journey, they have been both invaluable mentors and trusted friends. Their wisdom, encouragement, and generosity have profoundly shaped my academic path, and I am forever grateful for the confidence they have placed in me.

I would like to express my deepest appreciation to my parents and my sister for their unwavering belief in me. Their love, understanding, and faith have been my greatest sources of motivation, inspiring me to persevere even in the face of adversity.

Finally, I owe my deepest gratitude to my wife, Xifan Yang, for her boundless love, patience, and encouragement, despite the great physical distance between us. Her steadfast support has been a constant source of strength, and I could not have reached this milestone without her.

TABLE OF CONTENTS

Certificate of Original Authorship ii

Abstract iii

Acknowledgment v

List of Figures xi

List of Tables xvii

List of Publications xix

1 Introduction 1

1.1 Motivation 1

1.1.1 Background: 3D Scene Reconstruction 1

1.1.2 Existing Methods and Limitations 2

1.2 Research Objectives 4

1.2.1 Objective 1: Surface Normal Estimation via LiDAR-Image Fusion . . . 4

1.2.2 Objective 2: Surface Constraints in Scene Representation 5

1.2.3 Objective 3: Dynamic Scene Modeling with Explicit Geometry 6

1.3 Contributions 7

1.4 Thesis Outline 8

2 Literature Review **11**

2.1 3D Scene Reconstruction . . . 11

2.1.1 Point Cloud . . . 14

2.1.2 Voxels . . . 17

2.1.3 Mesh . . . 18

2.1.4 3D Gaussian Splatting (3DGS) . . . 21

2.1.5 Occupancy Field . . . 26

2.1.6 Signed Distance Function (SDF) . . . 27

2.1.7 Neural Radiance Field (NeRF) . . . 29

2.2 Dynamic Neural Radiance Fields . . . 30

2.3 Surface Normal Estimation . . . 31

2.4 Summary . . . 33

3 Surface Normal Estimation Enhanced by Semantics **35**

3.1 Background . . . 35

3.2 Method . . . 39

3.2.1 Problem Formulation . . . 40

3.2.2 Overview of Hybrid Geometric Transformer (HGT) . . . 40

3.2.3 Low-level features . . . 41

3.2.4 Transformer Encoder and Prediction Head . . . 43

3.2.5 Loss function . . . 44

3.2.6 Effective Training of the Attentional Transformer . . . 44

3.3 Experiments . . . 46

3.3.1 Data Synthesis . . . 47

3.3.2 Implementation Details . . . 49

3.3.3 Test on Synthetic Data . . . 50

3.3.4 generalization Test on KITTI . . . 52

3.3.5 Application on 3D Reconstruction 54
3.3.6 Additional Experiments 54
3.4 Summary 56

4 3D Gaussian Splatting with Explicit Surface Modeling 57
4.1 Background 57
4.2 Preliminary 60
4.2.1 3D Gaussian Splatting 60
4.2.2 Signed Distance Function 61
4.3 Method 62
4.3.1 Problem Formulation 64
4.3.2 Signed Distance Function and Differentiable Mesh Computation . . 64
4.3.3 Gaussians Consistent with Scene Geometry 66
4.3.4 Appearance Model 69
4.3.5 Rendering and Optimization 71
4.4 Experiments 71
4.4.1 Implementation details 72
4.4.2 Datasets and Metrics 73
4.4.3 Novel View Synthesis 75
4.4.4 Surface Reconstruction 77
4.4.5 Object Deformation by Hybrid Representation 78
4.4.6 Method Analysis 78
4.5 Summary 81

5 Dynamic Appearance Particles Neural Radiance Field 83
5.1 Background 83
5.2 Preliminaries on Neural Radiance Fields (NeRFs) 86
5.3 Method 87

5.3.1 Particle-based Dynamic Model 90
5.3.2 Appearance Particle Model 91
5.3.3 Efficient Computation 93
5.3.4 Optimization 94
5.4 Experiments 96
5.4.1 Experimental Settings 100
5.4.2 Novel View Synthesis 101
5.4.3 User Study 104
5.4.4 Evaluation of Efficiency 105
5.4.5 Evaluation of Motion modeling 105
5.4.6 Method Analysis 107
5.4.7 Practical Application 111
5.5 Summary 112

6 Conclusion and Future Directions **113**
6.1 Conclusion 113
6.2 Future Directions 114

Bibliography **117**

List of Figures

Figure **Page**

1.1 Overview of the research framework. The thesis is organized into three complementary works that progressively address the lack of explicit geometry: enhancing sensory data (Chapter 3), enforcing surface constraints in static representations (Chapter 4), and modeling explicit motion in dynamic scenes (Chapter 5). 9

2.1 Structure of this Literature Review. This review covers the most commonly used explicit representations (sub-figure (b)) and implicit representations (sub-figure (c)), which capture 3D information from raw data sources (sub-figure (a)). The sections are organized based on the used 3D representations, including Point Cloud (Sec. 2.1.1), Voxels (Sec. 2.1.2), Mesh (Sec. 2.1.3), 3DGS (Sec. 2.1.4), Occupancy Field (Sec. 2.1.5), SDF (Sec. 2.1.6), and NeRFs (Sec. 2.1.7). This figure also highlights some representative works under each specific sub-category. 12

3.1 Semantic information in related regions can help estimate surface normals. The figure shows estimation of normal in two regions **R1**: faraway, **R2**: near. R1 is more challenging due to sparse 3D points. The estimation can be assisted by visual information in close/related regions (light green boxes). This figure is best viewed in color. 36

3.2 Normal estimation on KITTI data. **(top)** LiDAR scans projected to images. **(bottom)** Surface normal estimated by Hybrid Geometry Transformer. 36

3.3 Mechanism of representing geometric information in an attention block. **i)** Tokens from multi-modal inputs **(specifically, fused features from RGB images and LiDAR point clouds)** are transformed into an association matrix **A**, where each row determines how a specific position collects information from others. See Section 3.2 for details. **ii)** Estimating the normal for a single position (denoted in blue), another position (denoted in red) shows high value in the corresponding row in **A**. **iii)** Although the blue position has sparse neighbors, its normal vector can be accurately estimated with auxiliary information from the red position, which has sufficient neighbors. 39

3.4 The pipeline of estimating surface normal from image and LiDAR points. 39

3.5 Architecture of U-Net [165]. 41

3.6 Transformer encoder with three self-attention blocks. 43

3.7 Making small sample by partitioning point clouds. The LiDAR points are grouped by the projections on the image into 4 panes. Note the entire image is associated with each group of the points to make the input sample, not only the visual contents in the corresponding pane. 45

3.8 Scene and movement setup for data collection. 47

3.9 Observations rendered by shaders in the 3D rendering engine. 47

3.10 LiDAR points' projection of KITTI and our synthetic data. 47

3.11 Frequency polygon of errors on the synthetic dataset. Each pane corresponds to varying noise levels. The horizontal axis denotes the range of errors, while the vertical axis indicates the number of points with specific estimation errors. The vertical axis is truncated between 0-12K to emphasize the 'long tail' effect, which differentiates the performance of various methods. 49

3.12 Qualitative comparison on synthetic dataset. 50

3.13 Qualitative results on the KITTI dataset. 52

3.14 Evaluation on 3D reconstruction task. Alpha Shapes represents a point-only reconstruction method, while the others perform Screened Poisson Surface Reconstruction (SPSR) [89] utilizing their respective estimated surface normals. . . 53

3.15 Visualization of attention weights produced by HGT using the synthetic dataset. White cross markers indicate the positions for normal estimation. Note that we are testing six distinct positions on the same frame. 54

3.16 Visualization of attention weights produced by HGT using the KITTI dataset. In the left subfigures, red cross markers indicate the positions for normal estimation. In the right subfigures, regions associated with high attention weights are depicted with greater opacity. 55

4.1 The Gaussians and the underlying surface did not align well in the original 3DGS [91], whereas our hybrid representation explicitly restricts the Gaussians to the mesh faces. Our method benefits from the high-quality rendering of 3DGS and the controllability of a mesh. 58

4.2 Method Overview. The mesh is derived from a learnable SDF grid using a differentiable marching algorithm. Gaussians are created from the mesh faces, ensuring their alignment with the surface. A neural appearance model determines colors for Gaussians, which are then used to render an image. 62

4.3 Parametric signed distance function (SDF) representation and its differentiable influence on mesh geometry. The SDF values at the eight grid nodes (colored squares) serve as learnable parameters. For any location inside the cell, its SDF value is obtained by trilinear interpolation from these parameters. Increasing a node's value (blue, top) shifts the interpolated zero-level set, thus moving the extracted mesh vertex (black) and altering the generated face. 63

4.4 Sub-figure (a) shows using $K = 1, 3, 6$ Gaussians to represent the appearance of a triangle face, (b) defines a local coordinate frame, and (c) illustrates applying linear transformation $\boldsymbol{M}$ to let Gaussians adapt to the irregular triangle. See Sec. 4.3.3 for more details. 67

4.5 Comparison between two approaches to learn appearance. 69

4.6 Qualitative comparisons with baseline methods (SuGaR [62], NeRF2Mesh [188]) on the NeRF-Synthetic [128] and Mip-NeRF360 dataset [4]. 74

4.7 Qualitative comparisons of reconstructed meshes with SuGaR [62] on the NeRF-Synthetic [128] and Mip-NeRF360 [4] datasets. 76

4.8 Object deformation by hybrid representation of mesh and Gaussians. We employ four Blender modifiers to manipulate the mesh, subsequently render perspectives using the bound Gaussians. 78

4.9 Adaptation to scene updates. (a) Models initially trained on the original scene. (b) and (c): Models adapted to the updated scene for 500 and 5000 steps, respectively. (d): The meshes learned on the updated scene. 79

5.1 Method Overview. **(a)** Particles represent observed movements. Each particle $\boldsymbol{p}$ corresponds to a small volume of material that represents a semantically meaningful part of a moving object. The position $\boldsymbol{p}(t)$ forms an explicit dynamic model of the object part. $\boldsymbol{v}$ denotes the visual feature of the particle (See Sec. 5.3.1). **(b)** Particles are integrated into a grid-based feature field, enabling efficient computation and superposition with the static feature field (See Sec. 5.3.3). **(c)** Colors of rays are computed by querying sampled points on the superpositional radiance field and performing volume rendering. Photometric loss is then calculated to optimize the model (see Sec. 5.3.4). 87

5.2 Commonly adopted structure of dynamic NeRFs. 88

5.3 Overview of the proposed hybrid representation. **(a)** Superposition of dynamic and static fields modeled by particle-based and Eulerian (voxel-grid-based) representations. We explicitly decouple the reconstruction tasks: the particles represent the moving parts to handle motion modeling, while the static field represents the stationary environment for background reconstruction. **(b)** Two main attributes represented by particles. The particle trajectory $\boldsymbol{p}^i(t)$ is time-varying, while the appearance feature $\boldsymbol{v}^i$ is time invariant. 88

5.4 Computational structure for time-varying position of a particle. γ is the positional encoding function [128], ϕ_m and ϕ_t are 3 and 2-layer MLP, 'CAT' is concatenate operation and 'ADD' is element-wise addition. 91

5.5 Training process of particles at $t = 0$. Panes show the particles at $t = 0$ during different stages of training. (See Sec. 5.3.2) 91

5.6 Comparative analysis with ParticleNeRF [1] on the D-NeRF dataset [154]. Although both methods utilize particle-based representations, ParticleNeRF struggles with monocular video data. 96

5.7 Qualitative comparisons on D-NeRF dataset [154]. 97

5.8 Synthesized novel views for D-NeRF dataset [154] and particle-based dynamic models. **(a)** displays the images rendered by the learned DAP-NeRF models, along with the trajectories of 100 sampled particles drawn onto the images. **(b)** shows the particles at time $t = 0$. To enhance visual clarity, we only render 2k randomly sampled particles. **(c)** shows the corresponding learned trajectories of particles, where only 200 randomly sampled particles are rendered. 97

5.9 Particle positions learned by DAP-NeRF at different frames. The color of each point is determined by its y-axis value. 98

5.10 Qualitative comparisons on NHR dataset [213]. 98

5.11 This figure displays the novel views rendered by the learned DAP-NeRF models, along with the trajectories of 200 randomly sampled particles. 99

5.12 Novel views rendered by DAP-NeRF on the HyperNeRF dataset [147]. For each scene, we render images from two perspectives at various times. 99

5.13 Network Architectures. **(a)** shows the implementation of Fig. 5.2, which is the NeRF architecture used in our method. **(b)** details Fig. 5.4, which is the motion model for time-varying position of a particle. In this figure, ϕ_t, ϕ_m, ϕ_v, MLP_σ and $\text{MLP}_\mathbf{c}$ are neural networks. The superscript of $\gamma^{(\cdot)}$ is the frequency number of positional encoding. Yellow boxes denote linear layers, grey boxes denote activation functions. 100

5.14 User study on rendering quality. This boxplot records the ratio of user preferences between DAP-NeRF (ours) and TiNeuVox [45]. Participants could also select 'Same' to indicate no significant difference between the methods. 104

5.15 The dataset for evaluating motion modeling. 106

5.16 Comparison in dealing with dynamics. We add artificial grids (in red) aligned with the coord-axes at the canonical/first frame for TiNeuVox and our method. . 107

5.17 Decomposition of scene components. The 'full' image is rendered using the final superpositional feature field. The other two images are rendered using only static and dynamic field respectively. 108

5.18 Application on editing dynamic scene. We reverse the original trajectories of particles that are responsible for the shape's right hand. 108

5.19 This figure demonstrates the effect of particle removal and re-sampling scheme. 109

5.20 Impact of Motion Regularization Term $\mathscr{L}_{\text{tvm}}$. To enhance visual clarity, only 500 particles randomly sampled from around the hand area are rendered. 109

5.21 Visualization of collision detection. The red plane represents a virtual wall, while the collided surfaces of the ball are marked in green. 112

List of Tables

Table **Page**

3.1 Average angle error on synthetic dataset. This table also includes an ablation study showing results when two key components are removed from our final HGT model. 51

3.2 Comparison of time and space efficiency. This table shows the average time required to process one frame in KITTI dataset. 51

4.1 Per-scene quantitative comparisons on NeRF-Synthetic dataset [128]. 73

4.2 Quantitative comparisons on Mip-NeRF360 dataset [4]. 75

4.3 Per-scene quantitative results on Mip-NeRF360 dataset [4]. 75

4.4 Quantitative Evaluation of Mesh Quality and Efficiency on the NeRF-Synthetic [128] dataset. 77

4.5 Evaluation on Efficiency. We report the training time and PSNR metric across different methods. 80

4.6 Ablation studies on adaptive covariance and refinement stage. We report the average PSNR across all scenes of the NeRF-Synthetic [128] and Mip-NeRF360 [4] datasets. 80

4.7 Ablation Study on the Number of Gaussians per Face. This table shows the PSNR results on the Mip-NeRF360 [4] dataset. 80

5.1 Per-scene quantitative comparisons on D-NeRF dataset [154]. 102

5.2 Per-scene quantitative comparisons on NHR dataset [213]. 102
5.3 Per-scene quantitative comparisons on HyperNeRF dataset [147]. **Red** text indicates the best result while **blue** text indicate the second best. 102
5.4 Comparisons in terms of training and rendering times, and parameter counts on the D-NeRF Dataset [154]. 105
5.5 Motion Field Error (MFE) for evaluating motion modeling. The definition of MFE can be found in (5.18). 105
5.6 Ablation study of particle quantity. 108
5.7 Ablation studies of Loss Components. We conduct evaluations on the D-NeRF dataset [154] and report the average PSNR across all scenes. 110
5.8 Tuning of weights for $\mathscr{L}_{\text{tvf}}$ and $\mathscr{L}_{\text{tvm}}$. 111

LIST OF PUBLICATIONS

This thesis is primarily built upon three major publications [1] [2] [3], each of which forms the foundation of one of the main technical chapters.

In addition, several other publications [4] [5] [6] [7] are also listed below. While they are not the main focus of the thesis, they are conceptually or methodologically related to the research. Their connections to the present work are discussed where relevant in the technical chapters.

[1] **Ancheng Lin**, Jun Li, Yusheng Xiang, Wei Bian and Mukesh Prasad, "Normal Transformer: Extracting Surface Geometry from LiDAR Points Enhanced by Visual Semantics," in *IEEE Transactions on Intelligent Vehicles (T-IV)*, vol. 9, no. 10, pp. 6172-6182, Oct. 2024, doi: 10.1109/TIV.2024.3363174.

[2] **Ancheng Lin**, Yusheng Xiang, Jun Li and Mukesh Prasad, "Dynamic Appearance Particle Neural Radiance Field," in *IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT)*, vol. 35, no. 7, pp. 6853-6866, July 2025, doi: 10.1109/TCSVT.2025.3540792.

[3] **Ancheng Lin**, Yusheng Xiang, Paul Kennedy and Jun Li, "Direct Learning of Mesh and Appearance via 3D Gaussian Splatting," (under review, arXiv: 2405.06945).

[4] **Ancheng Lin**, Jun Li and Zhenyuan Ma, "On Learning and Learned Data Representation by Capsule Networks," in *IEEE Access*, vol. 7, pp. 50808-50822, 2019, doi: 10.1109/ACCESS.2019.2911622.

[5] **Ancheng Lin**, Jun Li, Lujuan Zhang, Zhenyuan Ma and Weiqi Luo, "Multiple-Task Learning and Knowledge Transfer Using Generative Adversarial Capsule Nets," in *AI 2018: Advances in Artificial Intelligence*, doi: 10.1007/978-3-030-03991-2_60. (Publiched date: 10 November 2018)

[6] **Ancheng Lin**, Jun Li, Lujuan Zhang, Lei Shi, Zhenyuan Ma, "A New Family of Generative Adversarial Nets Using Heterogeneous Noise to Model Complex Distributions," in *AI 2018: Advances in Artificial Intelligence*, doi: 10.1007/978-3-030-03991-2_63. (Publiched date: 10 November 2018)

[7] Wanrong Hong, **Ancheng Lin**, Mukesh Prasad and Jun Li, "Surf-Style: Surface-Based Scene Stylization," (under review).

CHAPTER 1

INTRODUCTION

1.1 Motivation

1.1.1 Background: 3D Scene Reconstruction

3D reconstruction is the process of recovering the shape and appearance of objects from sensor data, such as images or LiDAR point clouds. When applied to dynamic scenes, the task also involves capturing how the scene evolves over time. The output of a reconstruction system is a structured representation that may take various forms, such as polygonal meshes or function-based encodings that allow querying of object attributes. This technology supports a wide range of applications, including augmented and virtual reality [186], autonomous driving [115], and medical imaging [131].

The need for 3D reconstruction arises from the inherent limitations of raw sensor data, which are often incomplete and view-restricted. For instance, a camera image provides only a 2D projection of surface appearance, without conveying any direct 3D geometry. Similarly, LiDAR sensors offer sparse samples of distances along visible surfaces, which do

not form a complete or continuous surface model. To overcome these limitations, 3D reconstruction aims to estimate or infer the underlying geometry and appearance of objects that are only partially observed, using multiple frames or multi-sensor data.

A central question in this field is: **What information needs to be reconstructed?**

While both geometry and appearance contribute to comprehensive scene understanding, this thesis focuses on the aspect more critical for practical system: **explicit geometry**. This refers to structured representations that provide direct access to either the occupancy state of any spatial location [38, 74] or surface-level attributes such as normals, curvatures, and boundaries [73, 88]. Such geometric information plays a central role in the computation of physical interactions, simulation, and motion planning.

In practice, 3D reconstruction serves as a foundational tool that enables intelligent agents to perceive and interact with their environments. Explicit geometric representations are particularly effective as they directly encode geometric attributes for efficient access and computation. These representations are also readily compatible with existing robotics and simulation frameworks, facilitating high performance and low-latency in interactive applications. For example, polygonal meshes and point clouds are widely used in virtual reality (VR) applications to enable real-time rendering and interaction.

1.1.2 Existing Methods and Limitations

Existing methods for 3D reconstruction can be broadly grouped based on the type of representation they use: explicit or implicit.

Explicit methods use geometric primitives, such as *voxels* [80], *point clouds* [91], and *meshes* [200], to parameterize scene geometry. These primitives are assigned visual attributes (e.g., RGB colors) to model appearance. Explicit methods offer clear geometric interpretability and compatibility with existing graphics pipelines. They have evolved from traditional Multi-View Stereo (MVS), voxelization, and triangulation techniques [88, 171,

238] to learning-based models [134, 217, 225]. Notably, 3D Gaussian Splatting (3DGS) [91] has emerged as an efficient alternative, representing scenes with millions of 3D Gaussians and adopting a high-fidelity differentiable rendering.

However, accurately reconstructing object surfaces using explicit primitives remains challenging, due to limitations such as imperfect real-world observations and insufficient structural constraints on the primitives themselves. Specifically, the surfaces (typically represented as meshes) extracted from sparse and noisy raw point clouds often lose geometric details, as real-world sensors suffer from measurement errors, environmental interference, and limited resolution. Furthermore, the recently proposed 3D Gaussian primitives [91], while producing visually appealing renderings, may be geometrically inaccurate — they can be inconsistent with any 2D manifold since the model imposes no explicit structural constraints on the primitives.

Implicit methods, by contrast, employ continuous functions such as *signed distance functions (SDFs)* [145], *occupancy fields* [152], and *neural radiance fields (NeRFs)* [128], to encode 3D scenes as learnable functions queried at arbitrary positions. These methods achieve high flexibility and smooth interpolation by utilizing neural networks to model continuous functions within a learning-based framework. Early approaches involve learning a conditional occupancy field [20] or SDF [145] to represent a class of shapes. Differentiable rendering techniques [96, 226] enable direct multi-view supervision, making implicit methods practical for real-world scene reconstruction. NeRFs further extend this paradigm by optimizing volumetric radiance fields, allowing photorealistic novel view synthesis from sparse images.

When extending this paradigm to dynamic scenes, existing methods predominantly rely on deformation fields to map time-variant observations to a canonical space. While effective for view synthesis, these approaches generally adopt an Eulerian formulation, where motion is implicitly encoded as continuous coordinate mappings within the volu-

metric field. This indirect modeling often leads to physical ambiguities, such as implausible deformations in empty space, and lacks the explicit geometric structure required for tasks involving object manipulation or physical interaction.

However, implicit reconstruction methods represent geometry indirectly, which can limit efficiency and physical interpretability, particularly when modeling dynamic scenes. In addition, field-based models such as NeRFs take 3D Euclidean coordinates as input to query physical quantities. While effective for modeling properties distributed throughout the entire space, this approach becomes inefficient when the target quantities are confined to specific regions or lie on sub-manifolds, such as within solid objects or along surfaces.

Therefore, a limitation of current methods is **the lack of explicit geometric modeling**. This gap motivates this thesis to explore new approaches that can recover high-quality and explicit geometry, while ensuring efficiency and compatibility in practical applications.

1.2 Research Objectives

To narrow the research gap, this thesis improves explicit geometric modeling in 3D reconstruction from two complementary perspectives: *data enhancement* and *representation design*, with a particular focus on learning-based approaches. Specifically, the research is organized around the following three objectives.

1.2.1 Objective 1: Surface Normal Estimation via LiDAR-Image Fusion

From the data enhancement perspective, we aim to improve the explicit geometric information of raw LiDAR point clouds by estimating per-point surface normals. These normals reflect the orientation of local surface patches, thereby enriching the interpretability and reliability of the sensor data. Moreover, surface normals provide an essential geometric prior that supports more accurate and robust surface reconstruction in subsequent processes.

However, recovering geometric properties is challenging due to the limited information captured from a single viewpoint. On one hand, image pixels lack depth information and are subject to perspective ambiguity. On the other hand, individual LiDAR points provide 3D distance measurements but are unstructured and sparse. The topological and geometric information of the surfaces is not directly available. Traditional regression-based methods [73] formulate surface normal estimation as a least-squares optimisation problem. However, they often fail in real-world settings due to their sensitivity to noise and reliance on carefully chosen parameters. More recent learning-based approaches [64, 101, 239] have shown promise in outperforming traditional methods, but they are mostly designed for dense, uniformly sampled point clouds from scanned or synthetic data.

To address these challenges, we propose to leverage the semantic information present in images to assist with surface normal recovery. To effectively estimate surface normals from images and point clouds, we design a neural network with attentional mechanisms that automatically learn cross-modal correlations.

1.2.2 Objective 2: Surface Constraints in Scene Representation

Building on the idea of using surface priors, we further investigate how surface modeling can be effectively integrated into scene representations. To this end, we incorporate surface constraints into explicit representations to enhance the geometric accuracy.

3D Gaussian Splatting (3DGS) employs an explicit representation called *Gaussians*, which represent anisotropic ellipsoids in 3D space. Despite the advantages in rendering quality and speed, it is not straightforward to reconstruct object surfaces using traditional Gaussians. This is because the 3DGS model imposes no explicit constraints on the structure of the Gaussians. While visually compelling, the resulting geometry may not correspond to a coherent 2D surface, which limits its usefulness in applications that require accurate geometric modeling.

Several recent methods attempt to address this issue by incorporating surface information into the learning process. For instance, NeuSG [15] uses normals estimated by a separate NeuS model [201] to refine the Gaussians, which significantly increases training time. SuGaR [62] aligns Gaussians with surfaces and converts them to a point cloud for meshing, but then requires an additional training phase to re-learn appearance. Such multi-stage pipelines are inefficient and fragile. It is worth noting that similar issues are encountered in downstream applications [46, 56, 83, 159], which typically perform meshing and Gaussian training separately. These challenges motivate us to design a more effective information pathway to supervise the learning of scene representations while achieving more accurate surface modeling.

To achieve this objective, we explicitly attach 3D Gaussians to mesh faces and jointly reconstruct geometry and appearance within an end-to-end learning framework. Specifically, the geometry is represented as triangle meshes, while the view-dependent appearance is modeled by 3D Gaussians. This direct correspondence between Gaussians and mesh faces imposes strong surface constraints, allowing photometric supervision to effectively guide both geometry and appearance learning, thereby improving rendering quality and geometric accuracy.

1.2.3 Objective 3: Dynamic Scene Modeling with Explicit Geometry

Also situated within the representation design domain, this objective aims to introduce explicit geometric modeling into the reconstruction of dynamic scenes — a challenging setting where most existing methods rely entirely on implicit representations.

In the reconstruction of dynamic scenes, most existing methods use deformation-field models, where motion and appearance are encoded in learnable fields over 3D space. These models adopt an Eulerian formulation: neural networks take 3D coordinates as input and return physical quantities such as deformation or color. This formulation is effective when

the region of interest covers the entire space, but becomes inefficient if the quantities of interest are confined to specific regions, such as the surface of a moving object. This inefficiency can lead to increased computational overhead and reduced interpretability. Moreover, current dynamic NeRF-based methods [113, 154] often require extra preprocessing, such as shape extraction [160] and inverse deformation computation [16], to be usable in practical applications involving physical interactions. These limitations motivate us to employ explicit geometric primitives for moving or deforming objects within dynamic scenes.

To achieve this objective, we propose a particle-based representation that offers an explicit and physically interpretable description of dynamic elements in a scene. Each particle carries both spatial and appearance information and evolves over time according to a learnable model. We also develop an efficient computational framework that integrates this particle representation with neural radiance fields, allowing the hybrid model to reconstruct and render dynamic scenes more accurately and efficiently.

1.3 Contributions

The research presented in this thesis is based on the following three main contributions, each of which is supported by a published or under-review paper (see 'List of Publications').

- We propose the Hybrid Geometric Transformer (HGT), a novel transformer-based network architecture to effectively estimate surface normals from multi-modal inputs. Unlike traditional estimation methods that rely on local neighborhoods, HGT employs self-attention mechanisms to capture long-range dependencies, improving global consistency. Experiments on traffic scene datasets demonstrate that our method achieves state-of-the-art normal estimation accuracy, significantly benefiting downstream surface reconstruction tasks.

- We develop a hybrid representation that explicitly attaches primitives to a mesh model

of the object surfaces, thereby introducing local structural constraints. Specifically, we adopt the latest explicit 3D Gaussians [91] to model the scene appearance and enable high-fidelity and efficient rendering. The mesh is extracted from a learnable signed distance field (SDF). A set of Gaussians is then generated and attached to each mesh face, aligning with the position and shape of the corresponding triangle. We demonstrate that this hybrid of Gaussians and a mesh achieves both high geometric accuracy and superior rendering quality.

- We propose Dynamic Appearance Particle Neural Radiance Field (DAP-NeRF), which introduces a novel representation called *appearance particles*. Each particle encodes a small finite volume of a scene's visible content through a 3D position and a visual descriptor. To model dynamic scenes from monocular videos, we adopt a Lagrangian formulation where dynamic elements are represented by particles moving through space over time. This complements the widely adopted Eulerian dynamic NeRFs, which specify the static elements of the scene in our framework. We demonstrate that appearance particles can be constructed without prior geometric knowledge and achieve state-of-the-art reconstruction quality. Compared to existing dynamic NeRF methods, our approach offers a more physically meaningful and efficient representation for modeling dynamic scenes.

1.4 Thesis Outline

The contribution of works in this thesis has been outlined in Figure 1.1, which illustrates the research questions addressed in this research and how they are connected to the generic field of data-driven 3D environment reconstruction.

Following this framework, the detailed structure of this thesis is organized as follows:

- **Chapter 2** provides a comprehensive review of existing 3D scene representations and

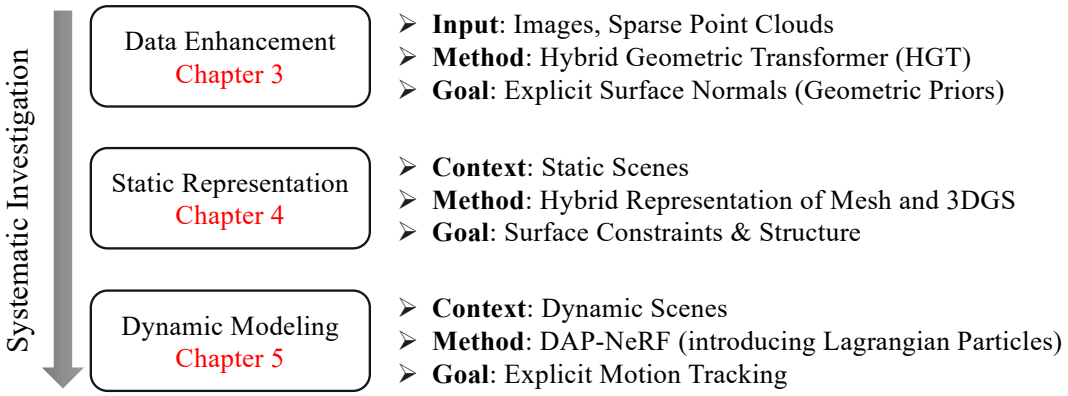

Figure 1.1: Overview of the research framework. The thesis is organized into three complementary works that progressively address the lack of explicit geometry: enhancing sensory data (Chapter 3), enforcing surface constraints in static representations (Chapter 4), and modeling explicit motion in dynamic scenes (Chapter 5).

reconstruction techniques. It also includes advancements in surface normal estimation and dynamic NeRFs, which are closely related to this thesis.

- **Chapter 3** introduces a transformer-based architecture that effectively fuses visual semantic and geometric information for high-quality surface normal estimation.
- **Chapter 4** proposes a hybrid model that jointly learns 3D Gaussians and a mesh, improving geometric accuracy through explicit surface modeling.
- **Chapter 5** presents an explicit representation called appearance particles. This novel representation is integrated into the NeRF framework to model dynamic scenes, achieving superior rendering quality and interpretability.
- **Chapter 6** concludes the thesis and discusses potential future research directions.

CHAPTER 2

LITERATURE REVIEW

2.1 3D Scene Reconstruction

Modeling and reconstructing 3D scenes is a fundamental area of research in computer vision and computer graphics. Techniques from this field have found extensive applications in virtual and augmented reality (AR/VR), autonomous driving, and film production.

As illustrated in Fig. 2.1, the goal of 3D scene reconstruction is to capture the geometry and appearance of a scene from sensor data such as images and point clouds.

Problem Formulation

Formally, this process aims to recover a scene representation $\mathscr{S}$ from a set of sensor observations $\mathscr{O} = \{o_1, \ldots, o_N\}$, captured from known or estimated poses $\mathscr{P} = \{T_1, \ldots, T_N\}$. The representation $\mathscr{S}$ typically encodes both geometry (e.g., occupancy, density, or signed distance) and appearance (e.g., color or radiance). The reconstruction is generally formulated as an optimization problem:

$$\mathscr{S}^* = \arg\min_{\mathscr{S}} \sum_{i=1}^{N} \mathscr{L}(\mathscr{R}(\mathscr{S}, T_i), o_i) \tag{2.1}$$

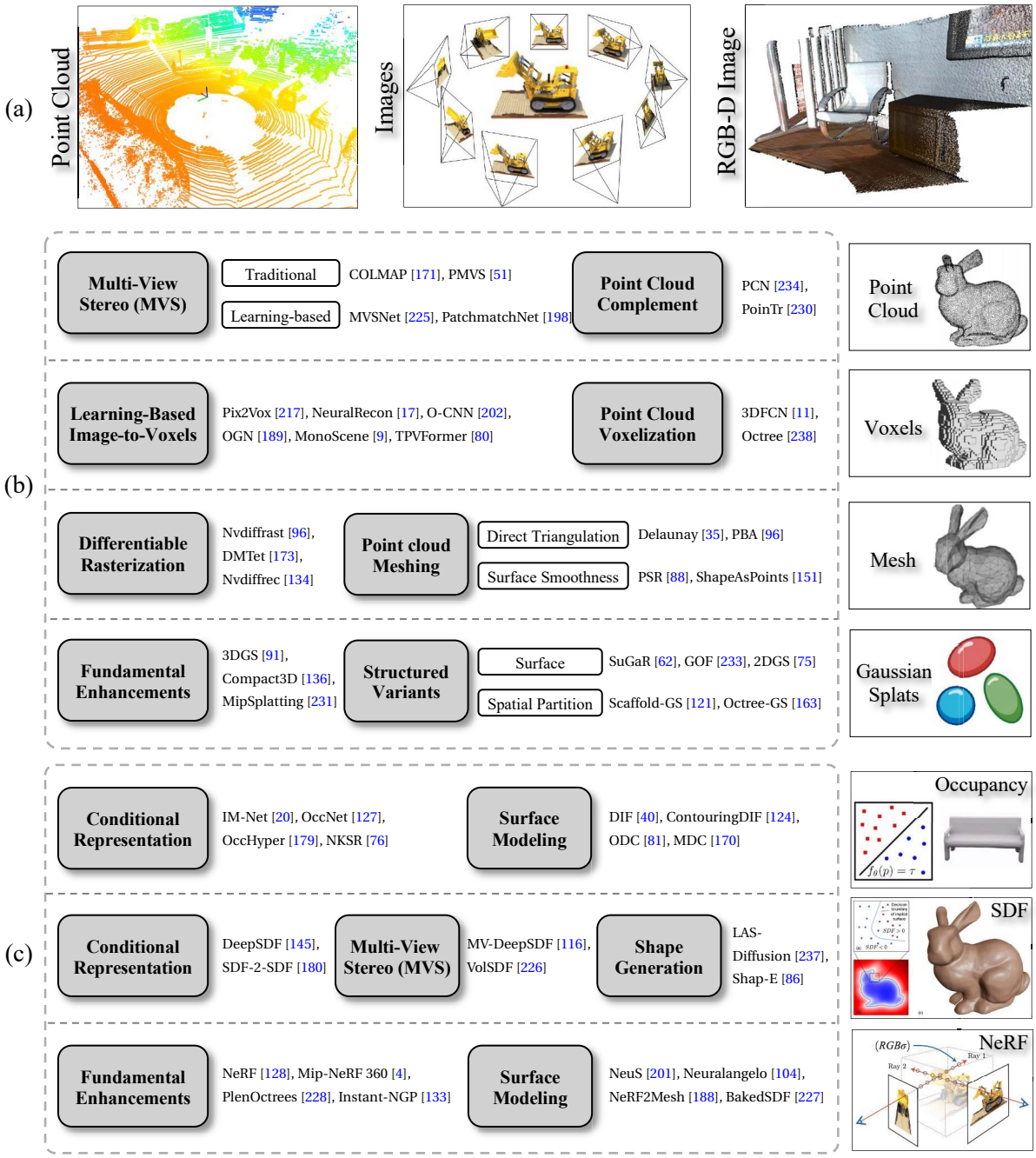

Figure 2.1: Structure of this Literature Review. This review covers the most commonly used explicit representations (sub-figure (b)) and implicit representations (sub-figure (c)), which capture 3D information from raw data sources (sub-figure (a)). The sections are organized based on the used 3D representations, including Point Cloud (Sec. 2.1.1), Voxels (Sec. 2.1.2), Mesh (Sec. 2.1.3), 3DGS (Sec. 2.1.4), Occupancy Field (Sec. 2.1.5), SDF (Sec. 2.1.6), and NeRFs (Sec. 2.1.7). This figure also highlights some representative works under each specific sub-category.

where $\mathscr{R}$ is a projection or rendering operator mapping the 3D representation to the sensor domain, $\mathscr{L}$ measures the consistency between synthesized and observed data.

Explicit vs. Implicit Representations

Based on the mathematical nature of $\mathscr{S}$, existing approaches can be classified into two main categories: **explicit** and **implicit** methods.

Explicit approaches include geometry entities in the representation of the 3D target, such as discretized surfaces or volumes. Hence the approach is convenient for downstream tasks such as computer graphics, physics simulation or robotics, where collision, deformation, projective geometry are readily managed using existing information in the models. On the other hand, the discrete nature of such representations makes such models cumbersome to deal with topology change, such as removing a hole, merge or split existing volumes. Specifically, in the scenario of learning the model from sensory data, one usually does NOT have full information of the topology, hence flexibility is welcomed, which makes space for explicit models to improve.

On the other hand, implicit approaches represent the target of interest as point set in the embedding space that meet the condition specified by a function. For example, a surface can be considered as the 2D boundary of a 3D volume, where the volume includes points meeting a condition. The representation is inherently continuous and admissible to rich mathematical tools. There is no effective limit on the topology or connectivity of the 3D target. However, implicit models lack a direct access to the geometry concepts that are commonly used in existing workflows in many application areas. The extract of useful information, e.g. the zero-level set, is often an iterative procedure costing significant resources on its own stand. This makes it expensive to be used in applications.

In this section, we review current 3D reconstruction techniques, organizing the literature into subsections based on the types of 3D representations employed. Specifically, we cover the commonly used explicit representations in this field. Additionally, we discuss im-

plicit neural representations, which have gained significant attention in recent years due to the rapid development of deep learning. Note that our research scope differs from traditional 3D reconstruction approaches, which typically focus on recovering depth maps from images followed by mesh extraction. This review takes a broader perspective, incorporating recent advancements in the field.

2.1.1 Point Cloud

A point cloud is a set of unorganized 3D points in world or device space, describing the small finite volumes in a scene. Structure-from-Motion (SfM) and Multi-View Stereo (MVS) are two well-known methods for recovering point clouds from camera images. In contrast, LiDAR or Time-of-Flight (ToF) cameras directly capture data in the form of point clouds. However, the obtained point clouds can sometimes be sparse, incomplete, and potentially problematic in practice. Therefore, the point cloud completion technique is often employed to enhance the data before implementing downstream applications.

Structure-from-Motion (SfM). This technique aims to estimate the sparse point cloud of a scene and the camera poses from uncalibrated images, which is the foundational process in MVS. This method generally involves feature detection and matching across images, then using multi-view geometry [70] (e.g. five-point technique and triangulation technique) and bundle adjustment (BA) [192] to solve for 3D structure and camera pose simultaneously. According to the specific processing pipeline, SfM can be classified into Global, Incremental, and Hybrid SfM. Global SfM [25] attempts to solve for all camera poses and 3D points simultaneously in a global optimization framework. While this might lead to more accurate results, it consumes a lot of computational resource, especially for large image sets. Incremental SfM [171] is more common in practice. It starts with a small set of images (often just two), estimates the camera poses and 3D points for this initial set, and then incrementally adds more images at a time. Incremental manner is more efficient for large im-

age sets, but can sometimes lead to drift errors that accumulate over time. Hybrid SfM [26] combines the best aspects of both global and incremental pipeline. It use the incremental SfM for quick initial recovering, and then apply global optimization to reduce accumulated drift errors.

There has been a vast body of literatures that introduce learning based models in the core components of SfM, including feature extraction [30], feature matching [112, 169, 212]. Recently, detector-free matchers [71] aim to avoid explicit feature matching which suffers from matching issues in texture-poor regions. Furthermore, some end-to-end SfM methods are proposed, such as direct poses regressing [144] and differential bundle adjustment [61].

Multi-View Stereo (MVS). This technique aims to generate a dense 3D model, which can be represented as a dense point cloud. In the MVS pipeline, the inputs include a set of calibrated images and the sparse point cloud obtained from the previously mentioned Structure from Motion (SfM). Traditional MVS methods can be categorized into volumetric methods [194], surface-based methods [106], patch-based methods [51, 118], and depth map-based methods [171, 220]. Depth map-based methods, which decouple the 3D model estimation problem into depth map estimation and fusion, have shown superior flexibility and accuracy. COLMAP [171] is currently one of the most commonly used algorithms, which jointly estimates pixel-wise view selection, depth map, and surface normal. However, traditional methods often suffer from high computational requirements and poor quality in challenging scenes.

Deep learning-based methods [122, 225, 232] have shown significant improvements in MVS performance. A representative work in this area is MVSNet [225], whose pipeline has been widely adopted by subsequent deep learning-based MVS systems. MVSNet constructs a 3D cost volume that is regularized with a 3D CNN and then regresses the depth map from the probability volume. However, due to the high memory usage and runtime consump-

tion, it can only handle images with low resolution. Recently, several variants of MVSNet have been proposed to improve efficiency [232] and robustness [122]

By fusing the estimated depth maps of different perspectives, a dense point cloud can be obtained. To further improve accuracy, offline MVS methods [60, 198, 225] perform data filtering before fusing, including photometric consistency filtering and geometric consistency filtering. Optionally, some applications adopt TSDF (Truncated Signed Distance Function) [27, 137] to fuse the depth maps into a TSDF volume and then extract a mesh [120].

Point Cloud Completion. This technique aims to estimate the complete point cloud from raw point clouds, which are often noisy, sparse, and incomplete. This field has rapidly developed since the pioneering work of PCN [234] is proposed. Recent methods for point cloud completion can be roughly categorized into Point-based [155, 224], Voxel-based [204, 219], Graph-based [158, 175], and Transformer-based methods [216, 230]. Point-based methods mainly use PointNet [155] and its variants as the feature encoder, focusing on the decoding process to produce complete point clouds. Voxel-based methods map the point cloud to a voxel grid, and then use 3D convolution to extract features. In graph-based methods, points or local regions are regarded as vertices in a graph. The edges of the graph can be established according to the vertices' adjacency in 3D space or feature space. Compared to point-based methods, graph-based methods are better at capturing regional geometric details. Transformer-based methods use self-attention mechanisms to capture the relationships between points in a point cloud. However, despite their effectiveness, they have a drawback in their high computational complexity.

Methods above mainly work on indoor scenes or small objects. In outdoor LiDAR point completion (often referred to as scene completion), the point cloud is sparser and has varying density, which poses greater challenges. This task is often transformed into depth completion [22, 37], where LiDAR points are projected onto the image plane to construct a sparse depth map. Therefore, advances in depth completion based on CNN can be applied.

2.1.2 Voxels

Voxel representation discretizes three-dimensional space into a uniform grid of cubic cells, each of which has two states: empty or occupied. As a fundamental and intuitive structure, voxel grids have been widely used in 3D vision and graphics tasks such as shape modeling and collision detection. However, the memory requirements for a dense voxel grid grow cubically as voxel resolution increases. As an extension, Octree [125] is a more efficient spatial representation than standard voxels, as it represents details by recursively subdividing a voxel into eight child voxels instead of changing the total resolution.

Learning-based Image-to-Voxels. This technique predicts the the scene geometry from single or multi-view images. One of the pioneering methods in this field is 3D-R2N2 [23], which leverages an end-to-end model based on 3D Convolutional Neural Network (3D CNN) and Long Short-Term Memory (LSTM) [72] to reconstruct voxel representations. As a variant of recurrent neural networks (RNNs), LSTM introduces challenges such as sensitivity to the order of input images and high computational complexity. Subsequent works [79, 148, 217, 218] have been proposed to mitigate these limitations. Notably, Pix2Vox [217] and Pix2Vox++ [218] utilize an encoder-decoder structure to simultaneously process images into coarse voxels. These coarse voxels are then fused and refined to yield high-resolution reconstructed voxels. Given posed images, Atlas [135] and NeuralRecon [17] back-project image features into a 3D feature volume, which is then processed by a 3D CNN to regress the TSDF volume. In addition, there are some methods [164, 189, 202, 203] adopting a hierarchical Octree representation to increase voxel resolution while reducing memory consumption.

Outdoor occupancy prediction [9, 80, 190] which uses voxel-based representation has become a common fundamental task in visual autonomous driving. The desired output is an occupancy descriptor, where voxels within a scene are assigned an occupancy state and a semantic label. Various driving tasks, such as semantic scene completion and ob-

ject detection, can be performed based on the occupancy descriptor. MonoScene [9] first uses 3D U-Net to infer 3D occupancy with semantic labels from a single monocular RGB image of an outdoor scene. TPVFormer [80] is a transformer-based model that obtains tri-perspective view representations from multiple images. Using LiDAR-based sparse occupancy as supervision, it achieved superior performance in occupancy prediction.

Point Cloud Voxelization. The simplest voxelization from a point cloud involves dividing 3D space into a voxel grid and then assigning each point to the one voxel based on its coordinates. A voxel is labeled as occupied when it contains one or more points, and as empty otherwise. This technique is often seen as a preprocessing step for various downstream tasks. Unlike unorganized point clouds, voxel representation has a regular structure that can leverage the benefits of 3D convolutional networks.

A body of research [11, 117] has introduced a coarse-to-fine approach to enhance the quality of voxelization when dealing with noisy and incomplete scan inputs. In addition, Octree-based voxelization [238] has been widely adopted to address the memory efficiency issue of regular voxel grids. By recursively subdividing the space into octants and only storing occupied regions, Octree structures can achieve adaptive resolutions while maintaining a compact representation. This approach is particularly beneficial for large-scale scenes where detailed geometry is only needed in specific regions.

2.1.3 Mesh

A surface mesh approximates a 2D manifold in 3D Euclidean space, providing a fundamental explicit representation of object geometry. It consists of vertices, edges, and faces (typically triangles) that form a polyhedral shape. Thanks to its flexibility and efficient computation of surface properties, meshes excel in rendering and physical simulations.

Recent advances enable direct mesh creation from images or point clouds. For image-based approaches, we first discuss end-to-end solutions that recover meshes directly from

images, including template deformation using neural network and multi-view supervision using differentiable rasterization. Methods that extract meshes from intermediate representations (e.g., point cloud, SDF, NeRF, and 3DGS) are discussed in later sections. For point cloud-based approaches, we cover direct triangulation methods and surface regularization techniques.

Reconstruction by Deforming Mesh Template. One representative of these methods is Pixel2Mesh [200], which extracts features from a single image and deforms an ellipsoid mesh into the desired shape using a Graph Convolutional Network (GCN). Follow-up works include Pixel2Mesh++ [210], which improved reconstruction accuracy by supporting multi-view inputs, and [143], which proposed removing error faces using an error pruning network. [176] introduced a key-point detector to regularize the deformed mesh.

Differentiable Iso-surface Extraction and Rasterization. Differentiable rasterizers take a mesh with vertex colors as input and allow gradients from the image-space loss to be back-propagated to vertex positions, enabling iterative optimization of the mesh.

A notable early contribution is Neural 3D Mesh Renderer [87], which introduce a differentiable rendering pipeline that supports end-to-end neural network training with mesh-based representations. More recent techniques, such as the nvdiffrast library [96], have enhanced the scalability and efficiency of differentiable rendering by leveraging hardware-accelerated rasterization primitives. This approach enables high-resolution rendering of large triangle meshes with accurate gradient computation and robust occlusion handling. However, these methods assume a fixed mesh topology and focus solely on optimizing vertex positions.

To address the challenge of optimizing mesh topology, differentiable iso-surface extraction methods are particularly effective. DMTet [173] directly optimizes meshes by employing a differentiable marching tetrahedra layer. Similarly, FlexiCubes [174] extends Dual

Marching Cubes [139] by introducing additional parameters for grid interpolation, vertex deformation, and triangle splitting, thereby increasing flexibility while maintaining robustness and ease of optimization. These methods can seamlessly integrate with differentiable rasterization and learn complex topology reconstruction even from random initialization.

The recent nvdiffrec [134] incorporates differentiable marching tetrahedra into the joint optimization of mesh topology, spatially-varying materials, and lighting from multi-view images. This integration improves the ability to reconstruct intricate shapes while producing explicit and editable 3D representations.

Point Cloud Meshing. Depth scanning sensors, such as structured light and time-of-flight (ToF) devices, produce point clouds as their primary form of data acquisition. Surface reconstruction from these point clouds is frequently required in applications like virtual and augmented reality, computer animation, and robotics. Additionally, classical multi-view stereo (MVS) methods, such as COLMAP [171], estimate depth maps and reconstruct meshes from back-projected point clouds, ultimately addressing the point cloud meshing problem.

Direct point triangulation is the most classical and intuitive approaches to meshing. Delaunay triangulation [35] subdivides the convex hull of a point cloud into triangles, ensuring that the circumcircle of each triangle does not contain any other points. Extensions to this method introduce regularization, such as removing undesirable triangle shapes [123] or applying a greedy selection of valid triangles based on topological constraints [24]. Another popular method, the Ball-Pivoting Algorithm (BPA) [7], uses rolling balls of varying radii to generate triangles by connecting sets of three points touched by a ball.

To achieve smoother surfaces, Poisson Surface Reconstruction (PSR) [88] takes oriented points as input and solves a 3D Poisson equation to compute the signed distance function (SDF) whose gradient best aligns with the surface normals. Screened Poisson Surface Reconstruction (SPSR) [89] extends this approach by incorporating regularization terms to mitigate the over-smoothing issues commonly encountered in PSR. More recently, differ-

entiable Poisson solvers [151] have been introduced, enabling end-to-end learning with neural networks.

Existing methods often require accurate surface normals at each point in the cloud. Therefore, in Sec. 2.3, we further review techniques for estimating surface normals, which motivates one of our research topics.

2.1.4 3D Gaussian Splatting (3DGS)

3D Gaussian Splatting (3DGS) [91] employs anisotropic 3D Gaussian primitives to explicitly represent volumetric density, which determines how light is scattered in a 3D scene. 3DGS achieves real-time rendering with state-of-the-art visual quality, making it suitable for various domains such as Simultaneous Localization and Mapping (SLAM) [90] and Artificial Intelligence Generated Content (AIGC) [44].

Since 3DGS is closely related to one of the projects presented in this thesis, we first introduce the technical principles of the original 3DGS in this section. Subsequently, we review the 3DGS literature from two perspectives: **Fundamental Enhancements**, which focus on improving the efficiency and photorealism of the original 3DGS framework; and **Structured Variants**, which incorporate geometric constraints or structural representations (e.g., grids or hierarchical models) to extend the capability of 3DGS.

Technical Principles. 3DGS represents the 3D scene using a collection of Gaussians, each defined as a density function in $\mathbb{R}^3$:

$$G(\boldsymbol{x}) = \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \propto \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \tag{2.2}$$

where $\boldsymbol{\mu}$ is a center position (mean), and $\boldsymbol{\Sigma}$ denotes a 3D covariance matrix. To model scene appearance, each Gaussian is associated with an opacity $\alpha \in [0,1]$, and a view-dependent color $\boldsymbol{c}$, which is parameterized with a set of Spherical Harmonics coefficients (SHs) $\{\boldsymbol{y}_l^m \in \mathbb{R}^3\}_{0\le l\le l_{\max}}^{-l\le m\le l}$ [50].

The interaction of a ray with a Gaussian can be characterized by integrating the Gaussian's density function $G(\boldsymbol{x})$. This leads to an efficient imaging model that projects Gaussians into 2D image space, resulting in 2D Gaussians with covariance calculated as:

$$\boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T\boldsymbol{J}^T, \tag{2.3}$$

where $\boldsymbol{W}$ denotes the transformation from world space to camera space, and $\boldsymbol{J}$ is the Jacobian for an affine approximation to the projective transformation [241].

To query the color of a pixel at location $\boldsymbol{x}' \in \mathbb{R}^2$ on image $\mathbf{P}$, a sorted list $\mathbf{L}$ of contributing Gaussians is determined based on their depths. The color of this pixel is then computed as follows:

$$\mathbf{P}(\boldsymbol{x}') = \sum_{i\in\mathbf{L}} \boldsymbol{c}_i \alpha'_i \prod_{j=1}^{i-1}(1-\alpha'_j), \tag{2.4}$$

where $\boldsymbol{c}_i$ is the color of the i-th Gaussian, and α'_i is the opacity contributed by the Gaussian, computed by:

$$\alpha'_i = \alpha_i \exp\left\{-\frac{1}{2}(\boldsymbol{x}'-\boldsymbol{\mu}'_i)^T\boldsymbol{\Sigma}'^{-1}_i(\boldsymbol{x}'-\boldsymbol{\mu}'_i)\right\}, \tag{2.5}$$

where α_i and $\boldsymbol{\mu}'_i$ denote the learnable opacity and the projected center of the i-th Gaussian, respectively.

Using images from known camera poses, the parameters of the Gaussians are adjusted to align rendered and observed images $\mathbf{P}_{\mathrm{GT}}$ by minimizing the photometric loss:

$$\underset{\alpha,\boldsymbol{\mu},\boldsymbol{c},\boldsymbol{\Sigma}}{\text{minimize}}\ \mathscr{L}_{\text{photo}}(\mathbf{P},\mathbf{P}_{\mathrm{GT}}). \tag{2.6}$$

The covariance matrix $\boldsymbol{\Sigma}$ is parameterized by a quaternion $\boldsymbol{q}$ and a 3D vector $\boldsymbol{s}$, through a change of variables defined as:

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T, \tag{2.7}$$

where $\boldsymbol{R}$ and $\boldsymbol{S}$ are the rotation and scaling matrices derived from $\boldsymbol{q}$ and $\boldsymbol{s}$.

Fundamental Enhancements on Storage and Computational Efficiency. Improving storage and computational efficiency is a key focus in the development of Gaussian Splatting (GS) techniques. In 3DGS, millions of Gaussian primitives are often required to represent a scene. This typically demands gigabyte-level storage for a single unbounded scene, and while 3DGS is computationally faster than Neural Radiance Fields (NeRFs), it still faces efficiency challenges. To address these, several works have introduced methods such as vector quantization (VQ) [39] to compress Gaussian parameters effectively.

One foundational work, Compact3D (C3D) [136], applies K-means quantization to create codebooks, where Gaussian parameters are replaced by corresponding entries in the codebook. This method significantly reduces storage memory, achieving a compression rate of 10–20 times. C3DGS [138] builds on this by using sensitivity-aware K-means [172] and the DEFLATE compression algorithm [31], further improving the compression ratio to up to 31 times with minimal visual quality degradation.

Another line of work focuses on compressing the color parameters, specifically the spherical harmonic (SH) coefficients. LightGaussian [42] reduces SH complexity through model distillation and pseudo-view augmentation, lowering the degree of SH representation. Furthermore, subsequent research [84] introduces offline clustering to pre-identify and remove redundant Gaussian primitives based on their spatial proximity and contribution to the rendered image, achieving enhanced efficiency.

For rendering pipeline acceleration, DISTWAR [33] identifies performance bottlenecks caused by atomic operations during rasterization and proposes optimizations to mitigate them. Additionally, GSCore [97] introduces a hardware acceleration unit tailored to Gaussian-based rendering, streamlining the sorting and rasterization processes, which are typically the most time-consuming stages.

Fundamental Enhancements on Photorealism. Photorealism in 3DGS has also been significantly improved through advancements in filtering and rendering techniques. Mip-

splatting [231] addresses sampling rate limitations by incorporating a Gaussian low-pass filter. This design follows Nyquist's theorem and dynamically adjusts the frequency of 3D Gaussians based on the maximum observed sampling rate. It also replaces traditional 2D dilation filters with a 2D Mip filter, approximating a box filter to reduce aliasing and dilation artifacts. Unlike Mip-splatting, which focuses on training-time modifications, SA-GS [182] introduces a 2D scale-adaptive filter during the test phase. This filter is compatible with any pre-trained 3DGS model, making it a flexible post-processing enhancement.

Further research addresses challenges in rendering complex scenes. For instance, Relightable 3D Gaussian (R3DG) [54] employs bidirectional reflectance distribution function (BRDF) decomposition and ray tracing to enable realistic and efficient relighting. DeblurGS [141] tackles the problem of blurred 3D scenes caused by camera pose inaccuracies. It estimates 6-DoF camera motion and synthesizes corresponding blurry renderings to optimize sharpness in 3D scenes.

Structured Variants. The original 3D Gaussian Splatting (3DGS) represents scenes using unorganized Gaussian primitives, which poses challenges for efficient storage and acceleration. To address this, recent work has explored structured variants of 3DGS that incorporate spatial organization to improve both storage efficiency and computational performance. Scaffold-GS [121] introduces a sparse grid of anchor points to organize local 3D Gaussians. Attributes stored on these anchor points are dynamically converted into Gaussians based on the viewer's pose, enabling efficient storage. This approach also employs multi-resolution voxel grids with strategies for growing and pruning anchors, thereby improving structural adaptability to varying scene complexities.

Level-of-Detail (LOD) techniques, widely used in computer graphics, balance visual quality and computational efficiency by varying scene detail based on factors such as viewing distance. Several studies have adapted LOD principles for 3DGS: CityGaussian [114] partitions scenes into cubic blocks, each compressed and represented at varying levels of

detail. The appropriate level of detail is selected dynamically for each block depending on the viewer's distance. Hierarchical-GS [92] organizes 3D Gaussians into a tree-based structure, allowing independent optimization of hierarchical levels. Octree-GS [163] utilizes an explicit Octree structure combined with an accumulative LOD strategy, accelerating rendering and reducing storage requirements.

Surface Modeling. A large body of literature focuses on the integration of surface representations, typically with the subsequent goal of extracting surfaces (e.g., meshes) or directly constructing hybrid representations. This is because mesh-based representations remain the preferred choice in many workflows due to their compatibility with robust tools for editing, animation, and relighting.

DreamGaussian [187] evaluates density values at grid nodes using a mixture of neighboring Gaussians and extracts isosurfaces via the Marching Cubes algorithm [120]. However, the resulting density fields often fail to capture accurate surface geometry, particularly in complex scenes. NeuSG [15] combines NeuS [201] and 3DGS models with a regularization term that aligns Gaussian orientations with surface normals predicted by NeuS. This joint optimization improves surface quality but requires significantly extended training times. SuGaR [62] incorporates signed distance function (SDF) values to regularize Gaussian positions, aligning them with surfaces. The resulting Gaussians are treated as point clouds for Poisson Surface Reconstruction.

Recent methods improve surface reconstruction by reformulating the representation of Gaussians to better encode surface-aware geometry and facilitate mesh extraction. GaussianSurfels [28] flatten each 3D Gaussian into a 2D ellipse by setting its z-scale to zero, so that the local z-axis consistently acts as the surface normal direction. It introduces a self-supervised normal-depth consistency loss, applies volumetric cutting to remove outlier depth samples, and fuses depth maps via screened Poisson reconstruction to produce a mesh. Gaussian Opacity Fields (GOF) [233] define a continuous opacity field based on

explicit ray-Gaussian intersections, allowing direct identification of the surface level set without requiring Poisson surface reconstruction or TSDF fusion. GOF approximates normals using the orientation of the ray-Gaussian intersection plane and extracts mesh geometry via Marching Tetrahedra [173]. 2D Gaussian Splatting (2DGS) [75] replaces volumetric Gaussians with oriented planar Gaussian disks that serve as view-consistent surface elements. It renders them using perspective-accurate ray-splat intersections instead of affine approximations, and incorporates depth distortion and normal consistency regularization to produce fast, noise-free mesh reconstruction.

2.1.5 Occupancy Field

Neural implicit representation (INR) has emerged as a powerful paradigm for modeling 3D geometry, where a neural network is treated as a continuous function mapping spatial queries to shape attributes. Due to its differentiable and compact nature, INR can be seamlessly integrated into learning-based frameworks.

One classic form of INR is the occupancy field, which represents a scene using a continuous neural function that takes 3D coordinates as input and outputs a scalar occupancy probability, indicating whether a point lies inside or outside a shape. IM-Net [20] and Occupancy Networks (OccNet) [127] are two pioneering works that represent 3D shapes as occupancy functions using neural networks. Both approaches achieve generalization across the dataset "ShapeNet" [12] by conditioning the MLPs with an additional global shape latent code, which can be extracted using a CNN with image input [20] or a PointNet with point cloud input [127]. Many works have focused on improving latent code construction, such as using hypernetworks [177–179] or gradient-based meta-learning methods [18]. While most approaches rely on supervised settings, some methods extend occupancy models to unsupervised learning. For example, [142] infers occupancy fields from sparse points by employing a margin-based uncertainty measure to sample the decision boundary and encourage

alignment between the boundary and the input point cloud. Recently, [211] and [76] employed a data-dependent kernel to model the occupancy field, which is then predicted as a linear combination of this kernel. These methods can learn a good inductive bias for general 3D reconstruction instead of catering to a specific dataset.

Although implicit representations are well-suited for learning-based frameworks, explicit triangle meshes remain the preferred format in many practical applications such as rendering and simulation. However, extracting explicit meshes from occupancy fields is challenging. Applying Marching Cubes [120] to a binary occupancy function often leads to staircase-like artifacts due to the discrete nature of the representation. One straightforward way to mitigate such artifacts is to smooth the occupancy values, as suggested in works like [21, 98]. Discrete Indicator Function (DIF) and Contouring DIF [40, 82, 124] attempt to relax binary occupancy by representing it as the fraction of each cell contained within the object. Multiresolution Isosurface Extraction (MISE) [127] improves mesh extraction by iteratively refining mesh vertices toward the surface using gradient descent. Recently, Occupancy-Based Dual Contouring (ODC) [81] introduces a more advanced approach by leveraging continuous occupancy information to better capture sharp features. Building on the Manifold Dual Contouring framework [170], ODC uses auxiliary 2D points along with grid edge points to compute local surface normals, which help identify 3D grid cell points through the quadric error function. This method avoids reliance on distance fields while effectively preserving sharp features and improving surface quality.

2.1.6 Signed Distance Function (SDF)

The Signed Distance Function (SDF) is a widely used representation for watertight shapes and surfaces of 3D objects. Similar to an occupancy field, SDF takes 3D coordinates as input but outputs the distance to the nearest surface, where the sign indicates whether the point lies inside (negative) or outside (positive) the object.

DISN [221] and PIFu [166] are pioneering approaches that leverage image features to construct neural implicit representations. These methods associate 3D query points with image features through projection, predicting the signed distance of the query points using MLPs. DeepSDF [145] extends the SDF framework to generalize across datasets such as ShapeNet [12] by incorporating a global shape latent code, enabling interpolation and reconstruction of diverse object shapes. SDF-2-SDF [180] builds on DeepSDF by learning direct mappings between SDF representations of different shapes, allowing for high-quality shape interpolation and editing.

MV-DeepSDF [116] combines multi-view supervision with the SDF representation, achieving improved generalization and high-quality 3D reconstruction from multi-view images. VolSDF [226] integrates signed distance functions with volumetric rendering to produce high-quality reconstructions under multi-view settings. SDF is often used as an intermediate representation for multi-view reconstruction pipelines, where techniques like differentiable isosurface extraction [174] and differentiable rendering [134] enable end-to-end learning of meshes and underlying SDFs. As radiance fields gain popularity, methods such as NeuS [104, 201, 205] adopt SDF as an intermediate representation, transforming it into density fields within the NeRF (Neural Radiance Field) pipeline. This approach allows the extraction of meshes while also modeling view-dependent appearance for high-quality reconstruction and rendering.

In shape generation, LAS-Diffusion [237] introduces a diffusion-based framework enhanced by a view-aware local attention mechanism, improving the controllability and generalizability of generated 3D shapes. By taking 2D sketch images as input, LAS-Diffusion facilitates image-conditioned generation, meeting detailed design and artistic requirements. Shap-E [86] directly generates implicit function parameters, including those for SDFs and NeRFs. Despite the higher dimensional complexity of its output space, Shap-E achieves faster convergence and produces high-quality results, expanding the scope of implicit rep-

resentation techniques.

For articulated shapes, Articulated SDFs (A-SDFs) [132] introduce disentangled latent spaces for shape and articulation, allowing adjustments at test time and expanding SDF applications from static to dynamic models. AutoSDF [130] further enhances SDF capabilities by using autoregressive shape priors for 3D shape completion, reconstruction, and generation, demonstrating robustness in multi-modal 3D tasks.

2.1.7 Neural Radiance Field (NeRF)

Neural Radiance Field (NeRF) [128] is a significant advancement in 3D reconstruction. Thanks to the differentiable volume rendering initially proposed by Neural Volumes [119], NeRF can achieve photorealistic novel view synthesis using only calibrated images as supervision. The technical details of NeRF will be further elaborated in the preliminary section (see Sec. 5.2), as it forms the foundation of one of our proposed methods.

Because of NeRF's rendering quality and flexibility, NeRF has inspired a large collection of follow-up work. Mip-NeRF [3] extends the original NeRF by rendering conical frustums instead of rays, reducing aliasing and improving the fine details of the rendered results. Mip-NeRF 360 [4] further uses a non-linear scene parameterization to improve performance on unbounded scenes. Another stream of work aims to reduce the reliance on accurate poses of images [109, 193, 208], and even to conduct pose estimation given an unseen image [110]. These works have boosted recent NeRF-based SLAM systems [183, 240], which contain real-time pose-free NeRFs. Furthermore, there are some few-shot NeRFs that conduct novel view synthesis from sparse input. The challenges induced by sparse input are addressed by transfer learning from large-scale datasets [14, 229], depth supervision [29], or incorporating regularizations during training [140, 223]. A notable research stream bridges Neural Radiance Fields (NeRF) to Signed Distance Functions (SDFs) [104, 188, 201, 206]. Representative work such as NeuS [201] models the scene as an SDF and performs unbi-

ased approximation of the weight function used in volume rendering, thus enabling the training process similarly to NeRF.

One major challenge of NeRF is its efficiency in rendering, mainly due to the time-consuming MLP forward passes for a large number of query points along the rays. Existing methods for improving rendering and training speed can be mainly divided into three categories. The first category focuses on reducing the sampling points by incorporating a learnable module to determine suitable sample locations [95, 111, 153]. The second category replaces the MLP with explicit or hybrid representations, which are distributed parameterizations of a scene. Specifically, Plenoxels [50] and PlenOctrees [228] adopt fully explicit voxels to store the scene's information, where querying can be done via trilinear interpolation. DVGO [185] uses a hybrid representation consisting of a dense voxel grid and a shallow MLP. While being efficient to render and optimize, these grid-based representations might lead to large model sizes. Therefore, Instant-NGP [133] uses a hashing technique to reduce the storage costs of voxels. TensoRF [13] decomposes the feature grid into vector and matrix factors to achieve greater compactness. Inspired by TensoRF, other low-rank representations such as tri-plane [47] and Sparse Tri-Vector [57] have also been explored recently. The third category of methods [19, 227] represents NeRF using triangle meshes, which enjoy the efficiency of standard polygon rasterization and rendering pipelines.

Another promising extension of NeRF is Dynamic NeRFs, which aim to build a time-varying radiance field to reconstruct dynamic scenes. As this scope is more relevant to our research project, we review this direction separately in Sec. 2.2.

2.2 Dynamic Neural Radiance Fields

There has been a growing interest in developing dynamic NeRFs for scenes where objects are moving or undergoing deformation. Advancements in this field successfully model dynamic scenes from synchronized multi-view videos [113, 119, 199] and even monocular

videos [45, 154, 191].

The first category of methods learns a time-conditional radiance field by adding a temporal input to the static radiance network or constructing a 4D volume [52, 53, 105, 215]. Methods that employ a time-conditional neural network produce completely different radiance and density fields at different time steps, may lead to a severely under-constrained problem. These methods typically require additional supervision or regularization, such as depth and optical flow (as in Video-NeRF [215] and NSFF [105]), or by incorporating multi-view observations [103]. Recently, the factorization of 4D volumes [10, 48] has yielded more efficient and less redundant representations, enabling more efficient training.

Another series of methods, known as deformable NeRFs [45, 67, 113, 146, 154, 191], decouple the model into a deformation field and a static canonical radiance field. The most commonly used technique in this category is backward deformation, which maps current 3D points back to a canonical space. While straightforward and efficient, this technique can struggle with motion inconsistencies in topologically complex scenes. Subsequent works such as [147, 181] have sought to address these challenges. On the other hand, [68] recognizes the discontinuity issues of the backward deformation and proposes using forward deformation instead. Nonetheless, challenges such as mapping ambiguity [195] in the deformation field still persist, especially when modeling empty space.

2.3 Surface Normal Estimation

Normal estimation is an essential task in 3D scene understanding. The simplest and best-known approach is to fit a least-squares local plane using principal component analysis (PCA) [73]. Although this method is efficient and well-understood, it is sensitive to noise and density variations and tends to smoothen sharp details [129]. To address these challenges, some techniques were proposed to achieve more robust estimation, such as Voronoi cells [126], distance-weighted approaches [150] and algebraic spheres' fitting [63]. 3F2N [41]

is a fast and accurate normal estimator performing three filtering operations on an inverse depth or a disparity image. However, both these traditional methods still depend heavily on the hyper-parameters like neighbourhood size.

Researchers have applied data-driven deep neural networks widely in 3D shape analysis. In particular, we can group existing models for normal estimation into three categories depending on the data sources.

Based on Image. This kind of work takes a single depth/color image or both as input and then performs the pixel-wise normal estimation. For a single RGB image, Eigen et al. [36] designed a three-scale convolution network to produce better results than traditional methods. Bansal et al. [2] and Zhang et al.[236] adopted the skip-connected structure and U-Net structure to improve performance. Recently, GeoNet++ [157] uses two-stream CNNs to predict both depth and surface normal maps from a single RGB image. Zeng et al. [235] presented a hierarchical fusion scheme for RGB-D data in a deep learning model and reached state-of-the-art performance.

The common challenge for these image-based methods is the low data efficiency, which means the training procedure requires a large amount of labelled surface normals [100]. In contrast, the point-based approach can improve data efficiency due to the explicit use of intrinsic 3D structure in supervised learning.

Based on Point Cloud. Another group of methods perform 3D scene understanding with point cloud only. Those methods usually use PointNet backbone [155] or Graph neural networks (GNNs) [214] to handle unstructured 3D points. Earlier attempts on normal estimation task came from [64]. They use PointNet architecture to extract local 3D shape properties from multi-scale patches of a point cloud. Nesti-Net [6] designed an extra module that learns to decide the optimal scale and hence improves performance. IterNet [99] uses graph neural network to model an adaptive anisotropic kernel that iteratively produces point weights for weighted least-squares plane fitting. Recently, a series of geometry-

guided methods [5, 101, 239] achieve state-of-the-art performance by combining traditional methods with deep learning based layers like Cascaded Scale Aggregation [239] and graph-convolutional layers [101].

It is worth mentioning that the point cloud data used above is mainly gathered from small CAD objects or indoor depth cameras. These point clouds have an even and relatively high density. But In outdoor scenes like urban environments, The heavy interference of the passive illumination [43] usually makes those active depth sensing solutions fail. In distant areas with low resolutions and small triangulation, stereo methods become less accurate. Hence, LiDAR has become the dominating reliable depth solution in the outdoor environment. However, the point cloud gathered from LiDAR is generally sparse, with noise and a nonuniform point density. Due to the limitations of LiDAR data and the complexity in an outdoor environment, approaches suitable for a dense indoor dataset are often incapable of estimating accurate normals for outdoor data.

Based on Multimodal Fusion. Various works combine image and LiDAR data to complement each other, ensuring reliable perception in challenging environments. The majority of these studies focus on 3D object detection and segmentation tasks [94, 167]. Nonetheless, few investigations have focused on estimating normals from images and sparse point clouds. Closely related to this field, DeepLiDAR [161] projects LiDAR scans onto the image plane to construct a sparse depth map, and then feed both the depth map and color image to a CNN-based encoder-decoder for normal estimation.

2.4 Summary

This chapter provides a review of existing 3D scene representations and reconstruction techniques, including explicit methods (e.g., point clouds, voxels, meshes, and 3D Gaussian Splatting) and implicit methods (e.g., occupancy fields, signed distance functions, and Neural Radiance Fields). We examined the strengths and limitations of each representa-

tion, particularly their ability to model complex surfaces and dynamic scenes. Additionally, we explored advancements in surface normal estimation and dynamic NeRFs, which are closely related to this thesis.

We identify a critical gap in surface reconstruction from LiDAR scans: the inherent sparsity and noise in raw point clouds make it difficult to estimate reliable surface normals, which are essential for downstream mesh reconstruction. While existing methods rely on normal estimation as an intermediate step, they often fail due to missing or ambiguous local geometry. This motivates our first research work: to leverage multi-modal sensor fusion for improving normal estimation, an area that remains largely underexplored in the literature (see Section 2.3).

We identify another key limitation in the recent explicit method 3D Gaussian Splatting (3DGS): despite its strength in radiance modeling, it lacks structural organization and struggles to produce accurate and consistent surfaces. Prior attempts to extract meshes or enforce geometric constraints often rely on post-processing or auxiliary modules, which limit end-to-end training and efficiency. This motivates our second research work: to design a more effective information pathway that directly supervises the learning of Gaussians and meshes (see Section 2.1.4).

We further identify challenges in dynamic scene reconstruction using implicit representations: although they support flexible and high-quality modeling, their lack of explicit geometric structure hinders interpretability and computational efficiency. These limitations motivate our third research work: to incorporate explicit geometric priors into implicit representations, enabling more structured and physically meaningful scene modeling (see Section 2.2).

In the following chapters, we elaborate on these three research works, providing detailed discussions of our proposed methodologies and their technical implementations.

CHAPTER 3

SURFACE NORMAL ESTIMATION ENHANCED BY SEMANTICS

3.1 Background

Decoding interesting 3D geometric information of an environment using observations from a specific viewpoint is challenging. The primary difficulty arises from the inherent insufficiency of information to capture a 3D scene from one viewpoint. For example, image pixels lack depth information and are subject to perspective ambiguity. On the other hand, individual points in a point cloud contain the distance to the ego-device in 3D space. However, the 3D points are unorganized. The topological and geometrical information of the surfaces is not directly available.

Surface normal estimation has proven valuable across a variety of computer vision and robotic applications such as odometry and mapping [65, 196], and 3D reconstruction [77, 89]. In the context of autonomous driving, point clouds with precise normal vectors enrich the understanding of urban environments' spatial relationships. Mid-level surface normal estimation enhances tasks' interpretability and reliability compared to raw point clouds. For instance, decisions for collision avoidance become more comprehensible when based

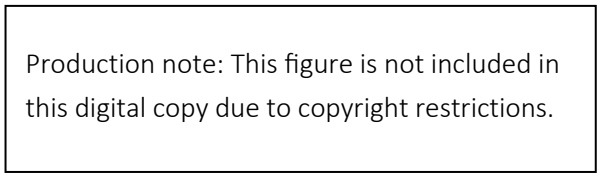

Figure 3.1: Semantic information in related regions can help estimate surface normals. The figure shows estimation of normal in two regions **R1**: faraway, **R2**: near. R1 is more challenging due to sparse 3D points. The estimation can be assisted by visual information in close/related regions (light green boxes). This figure is best viewed in color.

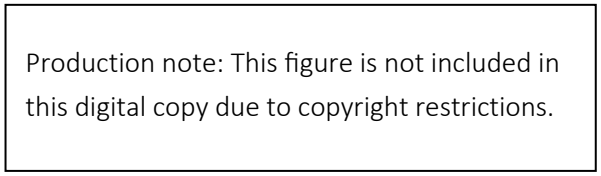

Figure 3.2: Normal estimation on KITTI data. **(top)** LiDAR scans projected to images. **(bottom)** Surface normal estimated by Hybrid Geometry Transformer.

on predicted distances to object surfaces. We consider the problem of estimating the normal vectors of object surface from point clouds. The classical regression-based methods [73] formulate surface normal estimation as a least-squares optimization problem. However, traditional methods could easily become fragile in real scenarios because they are sensitive to noise and manually selected parameters. Recently, learning-based techniques have demonstrated promises to outperform traditional methods [64, 101, 239]. However, these methods are primarily designed for uniform, dense point clouds derived from scanned or synthetic objects. As a result, their applicability to more realistic scenarios, such as those

involving LiDAR scans obtained by vehicles, is often limited.

In real-life LiDAR scans, points are not uniformly distributed, leading to significant challenges in geometry reconstruction. Specifically, scan points in areas of interest are often scarce, particularly in mid-to-long range fields. As the distance from the device increases, the scan becomes sparser, often resulting in incomplete or missing surfaces. On the other hand, an image provides a high-resolution, dense pixel array representation of the scene, inherently containing rich geometric information. Such images also imply the scene geometry – biological and machine vision can perform effective geometric inference from images. Therefore, our primary motivation is to leverage this rich semantic information inherent in images to assist in the recovery of surface normal vectors from point clouds.

The normal of a point on a smooth surface is directly determined by a small neighboring area. Theoretically, one only needs to collect sufficient points of a small region around a point $\boldsymbol{x}$ to recover the corresponding normal vector $\boldsymbol{n}_{\boldsymbol{x}}$. However, there can be few measured 3D points close to $\boldsymbol{x}$. One may turn to image information and fusion-based techniques [94]. This is useful when the local texture is sufficient to extract desirable scene information from the observation, but it has limited use for reconstructing detailed geometry of surfaces lacking salient visual cues. To overcome this limitation, regions not necessarily in immediate proximity to $\boldsymbol{x}$ can also contribute valuable information.

Fig. 3.1 illustrates an example. The masked regions R1 and R2 are both on road surfaces. R2 is closer to the device, containing more LiDAR points. The surface geometry could be directly computed using these points in R2. In contrast, a small neighborhood may contain only one or a few points in R1. No reliable geometric information can be directly extracted from the small set of scan points. However, if one considers the *image* regions marked by green boxes, the geometry in R1 could be implied by the visual information:

> Q: “What could be a flat surface lying between two buildings and supporting a vehicle beside it?"

A: “Perhaps a patch of road."

A naive approach would be adopting convolutional operators to extract local features from the sensory data to represent relevant geometric information, including variants of sparse or graph-based local operators [222]. However, determining the scale of a “local” area at every point on a manifold is a non-trivial task. And manually crafted procedures also tend to be fragile to deal with complex geometric structures.

Prior to this work, we conducted an in-depth investigation into how capsule networks (CapsNet) encode structured representations of data and how their routing mechanisms influence model fitting and generalization [107]. This study highlighted the advantage of preserving part-whole relationships and disentangled local structures within data representations. The insight that richer intermediate representations can improve downstream interpretability and adaptation inspires our integration of semantic cues into surface normal estimation. By combining multi-modal features and attentional reasoning, the current method extends the structural modeling ideas of CapsNet into the 3D geometric domain.

To automatically identify and utilize relevant information in the sensory data, this work presents the following insights and innovations:

- We present a transformer neural network-based model, Hybrid Geometry Transformer (HGT), to extract geometric information and estimate normal vectors from hybrid data (Fig. 3.2).
- We propose an effective training technique for self-attention nets on large-scale outdoor data.
- We release a data collection toolkit for the Urban environment built on the Unity 3D platform.
- The evaluation results show that HGT outperforms previous methods. Furthermore, the tests on KITTI benchmark reveal that our method has excellent generalization.

3.2 Method

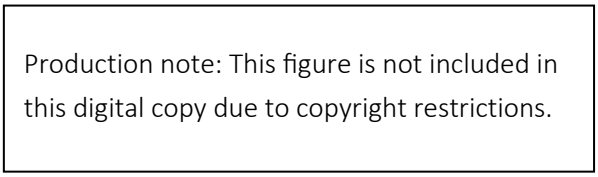

Figure 3.3: Mechanism of representing geometric information in an attention block. **i)** Tokens from multi-modal inputs **(specifically, fused features from RGB images and LiDAR point clouds)** are transformed into an association matrix **A**, where each row determines how a specific position collects information from others. See Section 3.2 for details. **ii)** Estimating the normal for a single position (denoted in blue), another position (denoted in red) shows high value in the corresponding row in **A**. **iii)** Although the blue position has sparse neighbors, its normal vector can be accurately estimated with auxiliary information from the red position, which has sufficient neighbors.

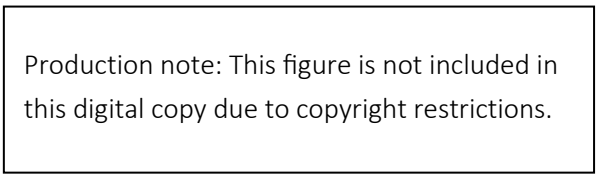

Figure 3.4: The pipeline of estimating surface normal from image and LiDAR points.

This section presents the proposed framework to estimate normal vectors on 3D points from hybrid observations. Fig. 3.4 shows the workflow of the framework. The pipeline takes two modalities as input: a sparse LiDAR point cloud and an aligned RGB image. It processes these inputs to recover the geometric details, specifically the surface normals, which are otherwise difficult to infer from sparse data alone.

3.2.1 Problem Formulation

Formally, we define the input point cloud as $\mathbf{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^{N}$, which is an $[N \times 3]$ array representing the coordinates of 3D points projected onto the image plane. The associated image is denoted as $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$.

The desired output of the model is the surface normal vector $\boldsymbol{n}_i \in \mathbb{S}^2$ for each point $\boldsymbol{x}_i$. While traditional methods estimate n_i using only the local neighborhood $\mathcal{N}_i \subset \mathbf{X}$, the sparsity of LiDAR scans often makes $\mathcal{N}_i$ insufficient. Therefore, we formulate the task as learning a mapping function $\mathscr{F}_\theta$:

$$n_i^{est} = \mathscr{F}_\theta(\boldsymbol{x}_i, \mathcal{N}_i, \mathbf{I}) \tag{3.1}$$

where the function aggregates both geometric structure and visual semantics to predict the normal vector.

3.2.2 Overview of Hybrid Geometric Transformer (HGT)

As illustrated in Fig. 3.4, the **Hybrid Geometric Transformer (HGT)** architecture is structured into three distinct stages:

- **Stage 1: Low-level Feature Extraction.** This stage processes the raw inputs independently to extract rich feature representations. We employ a U-Net to extract semantic features f_{img} from the image, a PointNet-based Geometric Module to capture local 3D structures f_{geo} from the point cloud, and a Positional Embedding module for spatial encoding f_{pos}.
- **Stage 2: Token Construction.** This stage fuses the multi-modal features from Stage 1. The image, geometric, and positional features are concatenated and projected to construct a unified sequence of tokens, which serves as the input for the subsequent attention mechanism.

- **Stage 3: Surface Normal Estimation.** This is the core computation stage. It utilizes a Transformer Encoder with self-attention mechanisms to capture long-range dependencies among the tokens. The transformer allows the model to collect information from semantically related regions, which is complimentary to the local data. Finally, a prediction head regresses the unit normal vector from the encoded tokens.

3.2.3 Low-level features

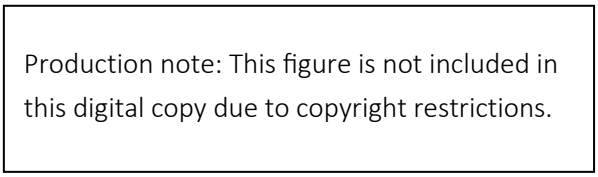

Figure 3.5: Architecture of U-Net [165].

As shown in Fig. 3.4, low-level features are descriptors of the 3D points, carrying three pieces of information: local image content, local point cloud geometry and location.

For each 3D point $\boldsymbol{x}_i$, a neighborhood $\mathcal{N}_i$ is extracted based on a Euclidean distance. The following image and geometric features at the neighbor points will be aggregated to produce a descriptor $\phi(\boldsymbol{x}_i)$ at $\boldsymbol{x}_i$.

$$\phi(\boldsymbol{x}_i) \leftarrow R\left(\{\boldsymbol{f}_j | j \in \mathcal{N}_i\}\right) \tag{3.2}$$

$$\boldsymbol{f}_j \leftarrow \phi_{fuse}\left(\boldsymbol{f}_{img}(j) \oplus \boldsymbol{f}_{geo}(j) \oplus \boldsymbol{f}_{pos}(j)\right) \tag{3.3}$$

where the operator R performs a reduction. In this work, we employ the Max operation. In (3.3), a feature vector $\boldsymbol{f}_j$ is the result of fusing (e.g. using a MLP ϕ_{fuse}) three-component

features: i) image feature $\boldsymbol{f}_{img}$, ii) geometric feature $\boldsymbol{f}_{geo}$ and iii) encoded positions $\boldsymbol{f}_{pos}$. The image and geometric features are derived from widely applied techniques as follows.

The *image features* are computed at each image pixel using a U-Net fully convolutional structure [165] as shown in Fig. 3.5. The U-Net is known for its capability to extract locally semantic information from an image. The hourglass structure enables the net to utilize the information at multiple scales to make predictions. The attribute is desirable for the motivation of utilizing semantics to make the normal estimation more robust. An additional advantage is the flexibility regarding input size, making it more generalizable.

It is worth noting that our framework does not require explicit semantic segmentation labels or pre-trained models. The U-Net is initialized from scratch and optimized in an end-to-end manner, guided solely by the supervision of the final surface normal estimation. This allows the network to automatically learn latent visual features, such as edges, textures, and object parts. The features are relevant for inferring the underlying 3D geometry.

Specifically, the U-Net encodes the raw image $\mathscr{I} \in \mathbb{R}^{H\times W\times 3}$ into feature map $\boldsymbol{f}_{img} \in \mathbb{R}^{H\times W\times C_{img}}$. Then we obtain $\boldsymbol{f}_{img}(j) \in \mathbb{R}^{C_{img}}$ by taking the feature vector from the 2D position where the 3D point $\boldsymbol{x}_j$ is projected, utilizing the known Camera-LiDAR calibration.

The *geometric features* at 3D points are computed using an MLP structure derived from the PointNet++ [156]. At each 3D point $\boldsymbol{x}_j$, $j \in \mathscr{N}_i$, we first apply a coordinate normalization

$$\hat{\boldsymbol{x}}_j = \boldsymbol{x}_j - \boldsymbol{x}_i$$

and the geometric feature vector is computed by an MLP ϕ_{geo}: $\mathbb{R}^3 \mapsto \mathbb{R}^{C_{geo}}$ shared by all points.

The *location* of a point is encoded by *positional embedding* layers ϕ_{pos}, which employ MLP structures as well. Following the common practice of encoding 2D coordinates in image processing [32], we encode every point's 3D spatial information in the global coordinate system into $\mathbb{R}^{C_{pos}}$.

3.2.4 Transformer Encoder and Prediction Head

Production note: This figure is not included in this digital copy due to copyright restrictions.

Figure 3.6: Transformer encoder with three self-attention blocks.

The multi-modal feature extraction and aggregation process described in (3.2) produces a comprehensive feature vector $\phi(\boldsymbol{x}_i) \in \mathbb{R}^C$ for each 3D point $\boldsymbol{x}_i$. All the N points can result in a matrix of tokens $\mathbf{T} : [N \times C]$. The transformer encoder comprises three self-attention blocks. The first self-attention block takes $\mathbf{T}$ as input and outputs new tokens $\mathbf{T}'$, which then serve as the subsequent self-attention block's input.

Fig. 3.6 provides a detailed illustration of the pipeline. In each block, $\mathbf{T}$ is linearly transformed into three tensors $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{N \times D}$, and $\mathbf{V} \in \mathbb{R}^{N \times E}$. Then $\mathbf{Q}$ and $\mathbf{K}$ are used to represent the relationship between the tokens. One commonly adopted computational model of the relation is to take the inner product:

$$\mathbf{A}' = \left[a_{i,j}\right]_{N \times N} = \mathbf{Q}\mathbf{K}^T \tag{3.4}$$

$$\mathbf{A} = \left[\frac{\hat{a}_{i,j}}{\sum_k \hat{a}_{i,k}}\right]_{N \times N}, \quad \hat{a}_{i,j} = \frac{\exp(a_{i,j})}{\sum_k \exp(a_{k,j})} \tag{3.5}$$

where the normalization operation in (3.5) is following the practice of [66]. The entry $a_{i,j}$ represents how token i in $\mathbf{Q}$ is related to the reference token j in $\mathbf{K}$ and $\mathbf{V}$.

The model then conducts a matrix multiplication $\mathbf{AV}$ for internal feature combinations between tokens. The combined tokens are passed into a Feed Forward layer followed by

batch normalization and residual connection. Finally, the layer outputs new tokens $\mathbf{T}'$ serving as the next self-attention block's input.

Following the self-attention blocks, an MLP prediction head ϕ_{pred} transforms each encoded token into a 3-dimensional vector representing three components of the surface normal. Finally, we regularise the output to unit 3D vectors to ensure a reasonable surface normal.

3.2.5 Loss function

The loss function for optimising the neural network model is defined as the mean squared error (MSE) between the directions of the estimated and the ground-truth normal vectors. The MSE of the estimations at N points is

$$\mathscr{L} = \frac{1}{N}\sum_{j}^{N} \| \boldsymbol{n}_{(j)}^{\text{est}} - \boldsymbol{n}_{(j)}^{\text{gt}} \|_2^2 \tag{3.6}$$

where $\boldsymbol{n}_{(j)}^{\text{est}}$ is an estimation by the model and $\boldsymbol{n}_{(j)}^{\text{gt}}$ is the corresponding ground-truth normal.

3.2.6 Effective Training of the Attentional Transformer

Batching. The transformer normal estimator is designed to utilize semantic information across the scene by employing large attentional fields. However, the attention model is expensive in terms of computation and storage. The attention weight matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ in (3.4) grows quadratically with the fused features, which is prohibitive for a complex scene.

More specifically, fully executing the model on one typical training sample (400×400 image and 10K points) costs about 6 Gigabytes of GPU memory. The cost is about 4x higher for a single frame in the real traffic scene in the KITTI dataset. Although straightforward implementation is viable on modern hardware configurations, the sample number in a batch

Figure 3.7: Making small sample by partitioning point clouds. The LiDAR points are grouped by the projections on the image into 4 panes. Note the entire image is associated with each group of the points to make the input sample, not only the visual contents in the corresponding pane.

during training must be compromised. At the same time, the batch size can affect the effectiveness of training [168].

For effective training, we partition the point cloud in one frame by grouping the points according to their projections on the image. For example, Fig. 3.7 (a) shows partitioning the points into 4 parts. The points of each part make one training sample. Since the input tokens of the transformer net correspond to individual points, the partitioned samples make the size of the weight matrix $\mathbf{A}$ decreases quadratically, e.g. from $[N \times N]$ to $[\frac{1}{4}N \times \frac{1}{4}N]$ in the example shown in Fig. 3.7. Within the storage limit, the batch size can be increased correspondingly, see the right pane of the figure. Increased batch is helpful during the training, especially when batch normalization is employed.

In this paper, we use $\frac{1}{16}$ of each frame's point cloud, and the batch size is 8. The experimental results in Fig. 3.15 show that our training technique will not lead to a loss of global sensibility during the testing phase. Furthermore, this technique can also be used in the testing phase if the range of scene observation is too extensive.

Synthetic data. Measuring normal vectors is difficult in practical applications. And most real-world datasets do not contain annotations of the ground-truth normal vectors. A toolkit is developed to produce synthetic data by simulating a city scene in a realistic 3D renderer environment. Artificial cameras and LiDAR devices are planted in the scene. We construct a customized shader to synthesize the normal vectors at points on object surfaces. Data samples are collected in the same form as the actual sensors. More importantly, labels (colors, depth, surface normals) are easily acquired from the simulator. Details about how data is synthesized can be found in the next section. We use this dataset to train and evaluate our model.

Experiments have shown that the model trained only on synthetic data has been able to generalize to practical datasets and obtain impressive results. Further improvement could be achieved if we use real-world data to fine-turn our model.

3.3 Experiments

This section presents the experimental results of the proposed technique. We first introduce how the synthetic data has been constructed. We then evaluate our model on this dataset and compare it with other existing works. We add different levels of noise into the data to test whether our model is robust. The model trained on the synthetic dataset is also applied to the real-life KITTI dataset [58]. The KITTI dataset does not contain ground-truth normal vectors at the LiDAR points. Hence the estimated normal vectors are used for 3D reconstruction to assess the estimation.

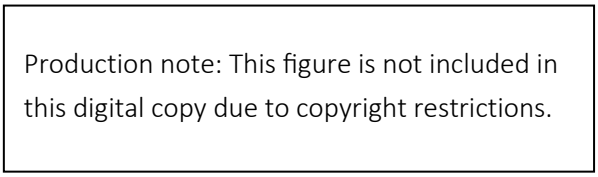

Figure 3.8: Scene and movement setup for data collection.

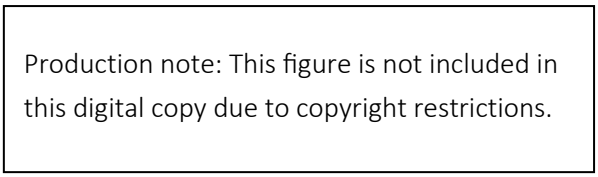

Figure 3.9: Observations rendered by shaders in the 3D rendering engine.

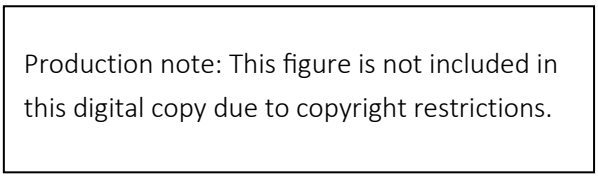

Figure 3.10: LiDAR points' projection of KITTI and our synthetic data.

3.3.1 Data Synthesis

We exploit Unity 3D [69] as a simulator and rendering engine to collect data. We first build a high-quality outdoor scene based on the package "POLYGON CITY"[1] on Unity 3D Assert Store. It contains vehicles, cyclists, pedestrians, and necessary lighting configurations. The

[1]https://assetstore.unity.com/packages/3d/environments/urban/polygon-city-low-poly-3d-art-by-synty-95214

scene can be easily changed to others according to practical needs. Our efforts mainly focus on the data collection after setting up a scene. More specifically, we develop in the following aspects.

Sensors and Shaders: The sensors we place in the simulator include a color camera, depth camera and normal sensor. The depth sensor is utilized to produce LiDAR points in the latter process. As shown in Fig. 3.8 (a), we place these sensors on the roof of the car. We can render the sensor observations by developing engine shaders for the three sensors, as shown in Fig. 3.9.

LiDAR acquisition: We find that the LiDAR projections to the image are mostly uniform. Therefore we take the lower rectangular region of the image and evenly sample rays passing through the image pixels to simulate projected LiDAR points. Note the points are distributed evenly on the image, but not in the 3D scene. The density has been made comparable to that in the KITTI dataset. Fig. 3.10 compares the sampling of the LiDAR points between the synthetic and KITTI datasets.

We then apply the designed mask to the given depth map from the sensor. Finally, the synthetic LiDAR scans can be obtained via inverse projection using camera intrinsic. Unlike other simulators, which spend a lot of time and computational cost on simulating real LiDAR using ray tracing, we simulate LiDAR directly from the depth map.

Dataset generation: With the communication interfaces provided by ML-Agent [85], we deploy scripts in Unity and an external Python-based controller, respectively.

We follow the preset path shown in Fig. 3.8 (b) and collect 151 frames' observations (121 for training, 30 for testing), which contain images with size 400×400, sparse LiDAR points in view, and points' surface normal ground truth. For convenience, we export the projection relationship arrays between LiDAR points and pixel index and the camera projection matrix.

To evaluate the robustness of the model, we also add different levels of noises to the

Production note: This figure is not included in this digital copy due to copyright restrictions.

Figure 3.11: Frequency polygon of errors on the synthetic dataset. Each pane corresponds to varying noise levels. The horizontal axis denotes the range of errors, while the vertical axis indicates the number of points with specific estimation errors. The vertical axis is truncated between 0-12K to emphasize the 'long tail' effect, which differentiates the performance of various methods.

LiDAR data. Note that we only add a numerical drift to the z-coordinate (in front of the camera) of the LiDAR point cloud. The noises follow the settings in [6, 64], see Fig. 3.11.

3.3.2 Implementation Details

We implement our model using PyTorch [149] on a NVIDIA RTX 5000 GPU. We adopts Adam [93] as optimizer with learning rate $1e^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. All models are trained for 200 epochs from randomly initialized parameters.

In the low-level features extraction, for each point, we randomly select 60 neighbors, and the radius of the spherical query is 0.75. If there are fewer points within the sphere, we will pad the results with the querying point itself. We have implemented a lightweight network architecture to enhance computational and storage efficiency. Specifically, we utilize a 2-layer MLP for both geometric feature extraction ϕ_{geo}, positional embedding ϕ_{pos}, multi-modal feature fusion ϕ_{fuse}, and prediction head ϕ_{pred}. The transformer encoder in our model comprises three self-attention blocks.

Production note: This figure is not included in this digital copy due to copyright restrictions.

Figure 3.12: Qualitative comparison on synthetic dataset.

3.3.3 Test on Synthetic Data

3.3.3.1 Evaluation Metric

To compare two directions, we consider directly measure the angles between the vectors. The following trigonometric evaluation is computed.

$$d_{\text{angle}} = \frac{1}{N}\sum_{j}^{N} \arccos\left(\frac{|\boldsymbol{n}_{(j)}^{\text{est}} \cdot \boldsymbol{n}_{(j)}^{\text{gt}}|}{|\boldsymbol{n}_{(j)}^{\text{est}}||\boldsymbol{n}_{(j)}^{\text{gt}}|}\right) \tag{3.7}$$

where $\boldsymbol{n}_{(j)}^{\text{est}}$ is an estimation by the model and $\boldsymbol{n}_{(j)}^{\text{gt}}$ is the corresponding ground-truth normal vector. This angle difference is consistent with MSE in (3.6).

Table 3.1: Average angle error on synthetic dataset. This table also includes an ablation study showing results when two key components are removed from our final HGT model.

Method	Noise Level			
	0	0.0025	0.012	0.024
PCA [73]	39.27	40.10	42.80	44.06
PCPNet [64]	10.50	15.94	19.50	20.83
AdaFit [239]	7.20	11.12	16.07	17.89
GraphFit [101]	**6.91**	10.41	14.90	17.01
HSurf-Net [102]	7.34	11.05	16.18	18.10
Ours (w/o image)	10.56	12.54	13.60	14.67
Ours (w/o transformer)	8.49	10.42	11.31	11.41
Ours (full)	8.18	**9.61**	**10.36**	**11.38**

Table 3.2: Comparison of time and space efficiency. This table shows the average time required to process one frame in KITTI dataset.

Method	Time	#.params
PCPNet [64]	354ms	3.47M
AdaFit [239]	1167ms	3.53M
GraphFit [101]	6401ms	4.26M
HSurf-Net [102]	1231ms	2.16M
HGT (Ours)	**86ms**	**1.83M**

3.3.3.2 Performance

We compare our method with other baseline methods [64, 101, 102, 239], each trained using the same train/test data (121 frames for training, 30 frames for testing; 4 noise levels). Additional settings including the number of training epochs (200) and the number of points in local patches (60), are aligned with those used in HGT to ensure a fair comparison.

Tab. 3.1 lists the average angular errors for a numerical comparison. In zero-noise data, the performance of HGT is comparable to that of state-of-the-art methods, with a minor deviation of less than 1.5°. The superiority of HGT becomes significant as noise levels escalate. This distinction in robustness will impact performance in real-world urban data, as demonstrated in subsequent tests.

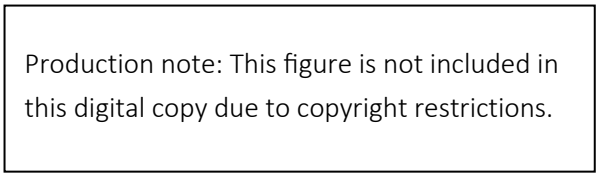

Figure 3.13: Qualitative results on the KITTI dataset.

In Fig. 3.11, we analyze the distribution of estimation errors across all LiDAR points. Specifically, we segment the error range (0-90°) into 60 bins and count the number of points falling within each bin. The distributions reveal that as noise levels increase, the estimations of other methods shift towards the tail (indicating a high error), whereas our method maintains a relatively low and flat tail. Fig. 3.12 further visualizes some of the estimated surface normal maps for qualitative comparison.

The report does not include the results of DeepLiDAR [161] since its pure CNN-Based architecture can not converges well on our small dataset (only 121 images and point clouds). This also reveals that our hybrid structure is more effective in handling multi-modal data.

3.3.4 generalization Test on KITTI

In this experiment, we use the real-world dataset KITTI [58] to investigate the model generalization. Given a frame of LiDAR scans and camera observations of KITTI, we first compute and save the point-to-pixel projection mapping with calibration parameters provided by the dataset. Then we directly feed the KITTI frames into HGT for surface normal estimation.

For comparison, we also illustrate the surface normal maps produced by two recent

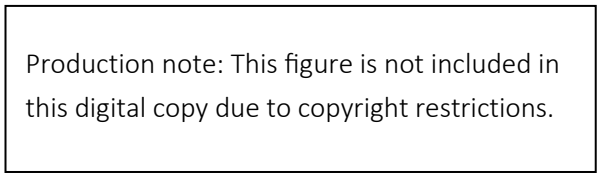

Figure 3.14: Evaluation on 3D reconstruction task. Alpha Shapes represents a point-only reconstruction method, while the others perform Screened Poisson Surface Reconstruction (SPSR) [89] utilizing their respective estimated surface normals.

Production note: This figure is not included in this digital copy due to copyright restrictions.

Figure 3.15: Visualization of attention weights produced by HGT using the synthetic dataset. White cross markers indicate the positions for normal estimation. Note that we are testing six distinct positions on the same frame.

methods and the classical PCA as in Tab. 3.1. Both methods are trained on the same synthetic dataset from Section 3.3.3 without additional fine-tuning.

The results are shown in Fig. 3.13. HGT has generalized surprisingly well, given that no fine-tuning has been conducted on the KITTI dataset. Compare the estimated normal vectors of the road surface in Fig. 3.12 for a specific example.

3.3.5 Application on 3D Reconstruction

Given the estimated normal vectors at the one-frame LiDAR points from the KITTI dataset, we reconstruct the 3D scene using Screened Poisson Surface Reconstruction (SPSR) [89]. The 3D reconstruction could serve as an intuitive criterion to compare the normal estimation methods. We also include comparisons to reconstruction w/o normals.

As shown in Fig. 3.14, the point-only method can lead to many mesh holes and is unsuitable for handling practical sparse LiDAR data. The Poisson Reconstruction technique can recover a continuous surface with the help of surface normals. By comparison, we can find that our model outperforms in 3D reconstruction as the quality of the reconstructed model significantly relies on the accuracy of the surface normals.

3.3.6 Additional Experiments

Ablation studies. To verify the effectiveness of multi-modal information, we train our model using only point cloud data by disabling the U-Net branch. The results in Tab. 3.1

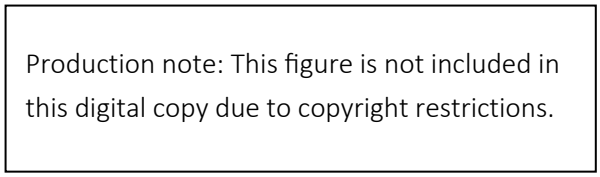

Figure 3.16: Visualization of attention weights produced by HGT using the KITTI dataset. In the left subfigures, red cross markers indicate the positions for normal estimation. In the right subfigures, regions associated with high attention weights are depicted with greater opacity.

demonstrate that incorporating image input reduces average angle error by approximately 3°.

To investigate how the transformer structure helps to recover the surface normal, we replace self-attention blocks with a PointNet structure. Specifically, an MLP and max pooling operation is applied to tokens and then generates a global feature. We concatenate this global feature to all tokens and pass them to the estimation head. The ablation results can also be found in Tab. 3.1, revealing that the transformer does bring a performance improvement.

Efficiency evaluation. To assess the efficiency of our method and verify its practical applicability, we count the number of model parameters and measure the average inference time required to process a single frame in the KITTI dataset. We conduct the same evaluation with other methods for comparison. Due to the out-of-memory issue encountered with AdaFit and GraphFit, we divided a single frame into 16 batches for all methods. As shown in Tab. 3.2, our method demonstrates a significant advantage in terms of both efficiency and model size.

Attention behavior visualization. A key aspect of our design involves establishing connections between semantically related regions, even if they are far away in space. This capability can be confirmed by visualizing runtime attention maps generated by our trained model. Specifically, we create heatmaps utilizing rows of matrix $\mathbf{A}$ in (3.5) derived from the model's inference process.

Fig. 3.15 is produced using the synthetic dataset, showing that areas of interest are comprehensively considered for normal estimation at an individual position. Furthermore, this characteristic has been successfully generalized to the KITTI dataset without additional fine-tuning, as shown in Fig. 3.16. For instance, in Fig. 3.16 (a), the distant points on the road aggregate information from numerous points across the entire road for geometric recovery, which is originally challenging due to the sparsity of the neighboring LiDAR scans.

3.4 Summary

This chapter presents a model that employs an attention mechanism for information fusion, with a specific application in estimating surface normals. We have also developed a comprehensive toolkit for data collection and a novel batching technique, both of which enable efficient and effective model training. Multiple evaluations have been conducted, revealing that our model consistently delivers superior performance on both synthetic and real-world data. A limitation of HGT is that the estimation is at where the LiDAR point is available.

CHAPTER 4

3D GAUSSIAN SPLATTING WITH EXPLICIT SURFACE MODELING

4.1 Background

Mesh is often the preferred choice in numerous geometry processing scenarios, as it carry explicit surface information. When building a 3D scene model that includes a mesh, it is challenging to directly learn from visual observation due to the complicated pipeline required to produce the synthetic views. Recently, neural surface reconstruction methods [201, 227] have been developed to represent scene geometry as an implicit signed distance function (SDF). These methods establish a connection with Neural Radiance Fields (NeRFs) [128], enabling effective image-based supervision for geometry learning. However, the geometry is learned indirectly, as it involves converting the SDF to a radiance field. Additionally, NeRF-based learning requires expensive ray-marching in the volumetric rendering. These factors lead to efficiency challenges in learning geometry and appearance.

3D Gaussian Splatting (3DGS) [91] has recently gained popularity due to its ability to render photorealistic images significantly faster than NeRFs. 3DGS employs an explicit representation called *Gaussians*, which represent anisotropic ellipsoids in 3D space. Gaussians

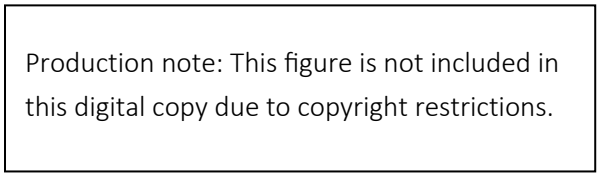

Figure 4.1: The Gaussians and the underlying surface did not align well in the original 3DGS [91], whereas our hybrid representation explicitly restricts the Gaussians to the mesh faces. Our method benefits from the high-quality rendering of 3DGS and the controllability of a mesh.

enable efficient rendering due to their explicit nature and the highly parallelized splatting procedure. Despite the advantages in rendering quality and speed, it is not straightforward to reconstruct object surfaces in the traditional form of Gaussians. This is because the 3DGS model imposes no explicit constraints on the structure of the Gaussians. As illustrated in Fig. 4.1, Gaussians that result in visually appealing rendering may be geometrically inaccurate – they can be inconsistent with any 2D manifold. Recently, some methods have incorporated regularization into the learning process of 3DGS. NeuSG [15] refines Gaussians using surface normals estimated by a jointly trained NeuS model [201], which notably increases the training time. SuGaR [62] encourages Gaussians to align with the surface and subsequently converts them into a point cloud for Poisson Surface Reconstruction. Once a coarse mesh is extracted, an additional stage is required to learn surface appearance from scratch using a new set of Gaussians. This two-stage pipeline may introduce efficiency problems. It is worth noting that similar issues are encountered in existing downstream applications [46, 56, 83, 159], which perform meshing and Gaussian training separately. These challenges motivate us to design a more effective information pathway to supervise the learning of scene representations.

In this work, we present a technique to explicitly bind Gaussians to mesh faces, enabling

their simultaneous learning. Specifically, the mesh is extracted from a learnable SDF of objects using differentiable marching techniques. The 3D Gaussians are then constructed as disks that are regularly attached to the mesh faces, as shown in Fig. 4.1 (b). Since the 3DGS model is bound to the zero-level set of the SDF representing object surfaces, the photometric supervision applied to the 3DGS model simultaneously guides the reconstruction of both the appearance and geometry of a scene. In addition, we model the background elements using the original 3DGS, which is jointly learned with our proposed surface-bound Gaussians. This approach makes our method more practical compared to existing methods that require inconvenient foreground masks [134, 174]. As the mesh faces vary during training, the Gaussians dynamically bound to them struggle to learn per-Gaussian colors as effectively as in the original 3DGS. To address this issue, we employ a neural network to learn the scene's appearance and predict Gaussian colors for rendering.

Extensive experiments show that our method achieves state-of-the-art rendering accuracy and produces high-quality meshes in indoor and outdoor scenes. Furthermore, it reduces total reconstruction time by 50% compared to recent methods and is over 10x faster than methods that rely on Neural SDFs and NeRFs [104, 201]. We also demonstrate that our method has a unique advantage in adapting to scene updates compared to other two-stage methods [62, 197].

In summary, our contributions are as follows:

- We present a framework for directly learning scene representations with mesh and Gaussian-based appearance in an end-to-end manner, while ensuring explicit alignment and correspondence between Gaussians and mesh faces.

- We propose a technique to model background appearance, extending the applicability of differentiable mesh extraction methods to outdoor scenes without the need for foreground masks.

- We introduce neural appearance model that predicts Gaussians colors, enhancing the robustness of the optimization process.

4.2 Preliminary

4.2.1 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [91] represents the 3D scene using a collection of Gaussians, each defined as a density function in $\mathbb{R}^3$:

$$G(\boldsymbol{x}) = \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \tag{4.1}$$

where $\boldsymbol{\mu}$ is a center position (mean), and $\boldsymbol{\Sigma}$ denotes a 3D covariance matrix. To model scene appearance, each Gaussian is associated with an opacity $\alpha \in [0,1]$, and a view-dependent color $\boldsymbol{c}$, which is parameterized with a set of Spherical Harmonics coefficients (SHs) $\{\boldsymbol{y}_l^m \in \mathbb{R}^3\}_{0\le l\le l_{\max}}^{-l\le m\le l}$ [50].

The interaction of a ray with a Gaussian can be characterized by integrating the Gaussian's density function $G(\boldsymbol{x})$. This leads to an efficient imaging model that projects Gaussians into 2D image space, resulting in 2D Gaussians with covariance calculated as:

$$\boldsymbol{\Sigma}' = \boldsymbol{J}\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T\boldsymbol{J}^T, \tag{4.2}$$

where $\boldsymbol{W}$ represents the current viewing transformation, and $\boldsymbol{J}$ is the Jacobian for an affine approximation to the projective transformation [241].

To query the color of a pixel at location $\boldsymbol{x}' \in \mathbb{R}^2$ on image $\mathbf{I}$, a sorted list $\mathcal{S}$ of contributing Gaussians is determined based on their depths. The color of this pixel is then computed as follows:

$$\mathbf{I}(\boldsymbol{x}') = \sum_{i\in\mathcal{S}} \boldsymbol{c}_i \alpha_i' \prod_{j=1}^{i-1}(1-\alpha_j'), \tag{4.3}$$

where $\boldsymbol{c}_i$ is the color of the i-th Gaussian, and α'_i is the opacity contributed by the Gaussian, computed by:

$$\alpha'_i = \alpha_i \exp\left\{-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu}'_i)^T \boldsymbol{\Sigma}'^{-1}_i (\boldsymbol{x}' - \boldsymbol{\mu}'_i)\right\}, \tag{4.4}$$

where α_i and $\boldsymbol{\mu}'_i$ denote the learnable opacity and the projected center of the i-th Gaussian, respectively.

Using images from known camera poses, the parameters of the Gaussians are adjusted to align rendered and observed images $\mathbf{I}_{\text{GT}}$ by minimizing the photometric loss:

$$\underset{\alpha,\boldsymbol{\mu},\boldsymbol{c},\boldsymbol{\Sigma}}{\text{minimize}}\ \mathscr{L}_{\text{photo}}(\mathbf{I}, \mathbf{I}_{\text{GT}}). \tag{4.5}$$

The optimization variables for covariance matrix $\boldsymbol{\Sigma}$ include a quaternion $\boldsymbol{q}$ and a 3D vector $\boldsymbol{s}$, which relate to the objective through a change of variables:

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T, \tag{4.6}$$

where $\boldsymbol{R}$ and $\boldsymbol{S}$ are the rotation and scaling matrices derived from $\boldsymbol{q}$ and $\boldsymbol{s}$.

4.2.2 Signed Distance Function

To incorporate geometry into a learning process, one of the major challenges is to design an appropriate computational procedure that produces observables (e.g., images) from the geometry model. The procedure should be efficient and allow the passing of supervision information so that the geometry model to be learned can be optimized with respect to the observables. For example, in traditional computer graphics, the rendering process involves a rasterization step that sorts the geometry primitives by depth, which is not differentiable. At pixels on the occluded sides, changes in vertex locations lead to discontinuities in pixel values and photometric loss, making geometry learning difficult.

Various intermediate representations have been proposed to address this issue. The Signed Distance Function (SDF) is a scalar field that defines the distance from any point

Figure 4.2: Method Overview. The mesh is derived from a learnable SDF grid using a differentiable marching algorithm. Gaussians are created from the mesh faces, ensuring their alignment with the surface. A neural appearance model determines colors for Gaussians, which are then used to render an image.

in space to the nearest surface of an object. The sign of the distance indicates whether the point is inside or outside the object. Formally, given a point $\boldsymbol{x}$ in 3D space, the SDF value is defined as:

$$D(\boldsymbol{x}) = \begin{cases} +d(\boldsymbol{x}, \partial\Omega) & \text{if } \boldsymbol{x} \in \mathbb{R}^3 \setminus \Omega \\ -d(\boldsymbol{x}, \partial\Omega) & \text{if } \boldsymbol{x} \in \Omega \end{cases} \tag{4.7}$$

where $d(\boldsymbol{x}, \partial\Omega)$ is the Euclidean distance from $\boldsymbol{x}$ to the surface $\partial\Omega$ of the object Ω.

Directly learning a mesh from images is challenging. A common paradigm is to optimize an parameterized SDF and then extract a triangle mesh approximating the level set of that function. Notable examples include recent works that use differentiable marching techniques [134, 174, 209] and the conversion from SDF to neural radiance fields [104, 201, 205].

4.3 Method

The proposed method aims to obtain a set of Gaussians that strictly align with object surfaces. To achieve this, we introduce a hybrid representation that fuses the strengths of two explicit modalities. The triangle mesh acts as the explicit geometry backbone. The explicit representation enables useful geometric operations such as topology or manifold check, or

Figure 4.3: Parametric signed distance function (SDF) representation and its differentiable influence on mesh geometry. The SDF values at the eight grid nodes (colored squares) serve as learnable parameters. For any location inside the cell, its SDF value is obtained by trilinear interpolation from these parameters. Increasing a node's value (blue, top) shifts the interpolated zero-level set, thus moving the extracted mesh vertex (black) and altering the generated face.

physics applications such as collision detection. The 3D Gaussians function as a volumetric appearance skin attached to the mesh surface. Unlike flat textures, this volumetric skin captures complex near-surface visual effects.

As illustrated in Fig. 4.2, our method includes modular differentiable steps, allowing the model to learn from images as in the original 3DGS. Let the 3D scene of interest be within a region $U \subset \mathbb{R}^3$. The model consists of:

1. a scalar field of signed distance function $D : U \mapsto \mathbb{R}$, which relates locations to the presence of objects;
2. a triangle mesh $G = (\mathcal{V}, \mathcal{F})$ derived from D, where vertices $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_V\} \subset \mathbb{R}^3$ and faces $\mathcal{F} = \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_F\}$, with each $\boldsymbol{f}$ denoting a triplet of indices within $[1, \ldots, V]$;
3. a constrained 3DGS model where the Gaussians are attached to the faces in $\mathcal{F}$;
4. a neural appearance model, which maps a Gaussian center to spherical harmonics coefficients.

4.3.1 Problem Formulation

Given a set of multi-view images, our goal is to jointly reconstruct the explicit geometry and view-dependent appearance. To enforce structural consistency, we propose a constrained 3DGS model where the Gaussians are explicitly attached to the faces in $\mathscr{F}$. Specifically, the triangle faces correspond to the zero-level set of D. We strictly constrain the center $\boldsymbol{\mu}_k$ of each Gaussian to reside on this surface and parameterize its covariance to ensure the primitive lies within the geometry of the corresponding triangle face. The appearance is parameterized by a neural model Φ that maps the center $\boldsymbol{\mu}_k$ to spherical harmonics coefficients. The reconstruction is formulated as optimizing the parameters of D and Φ to minimize photometric error:

$$\min_{D,\Phi} \sum \mathscr{L}_{photo}(\mathscr{R}(\{g_k\}), I_{gt}) \quad \text{s.t.} \quad \boldsymbol{\mu}_k \in \partial\Omega, \tag{4.8}$$

where $\mathscr{R}$ denotes the differentiable splatting rasterizer and $\partial\Omega = \{x|D(x) = 0\}$ represents the implicit surface. This formulation ensures that the photometric supervision simultaneously optimizes the explicit surface and the appearance model.

4.3.2 Signed Distance Function and Differentiable Mesh Computation

The geometry of the scene consists of object surfaces, represented as the zero-level sets of the signed distance function, denoted by $\{\boldsymbol{x}|D(\boldsymbol{x}) = 0\}$. We represent D using an explicit format: for a set of points $\{\boldsymbol{x}_n\}_{n=1}^N$, which correspond to the nodes of a regular 3D grid, the function values $\{s_n\}_{n=1}^N$ are specified. For a generic location $\boldsymbol{x}$, $D(\boldsymbol{x})$ is computed by interpolation from the eight nodes of the grid cell enclosing $\boldsymbol{x}$. Therefore, the value set $\{s_n\}_{n=1}^N$ forms a parametric representation of D. Fig. 4.3 illustrates how a grid cell determines a mesh face through a parametric representation

Given the SDF, the object surfaces are implicitly defined by its zero-level set. However, a practical 3D scene model requires explicit geometry to be incorporated into the com-

putational pipelines for rendering and learning. A traditional method to recover a discrete surface representation *for a given SDF* is the Marching Cubes algorithm [120], which results in a triangle mesh consistent with a 2D manifold. In contrast, this work involves not just recovering a surface but also *learning the SDF*. This requires the model to adjust the SDF parameters $\{s_n\}_{n=1}^{N}$ to make the resultant surfaces vary according to desired updates. Therefore, we adopt the recently proposed FlexiCubes [174] as the differentiable surface extractor.

To initialize the SDF, we train the original 3DGS model for only 20,000 steps to obtain a set of Gaussians, whose center positions are treated as a point cloud. We then employ Alpha Shapes [34] to extract a coarse mesh and make it watertight [78], which is subsequently used to construct an SDF grid. Our approach differs from others [134, 174] that randomly initialize the SDF, which can cause an OOM issue when the scene becomes large. Thanks to the fast optimization and the discrete nature of 3DGS, we are able to capture the coarse geometry in only 5–20 minutes. This initialization naturally preserves the background Gaussians for later joint optimization.

Additionally, we address two substantial challenges of modeling outdoor and realistic scenes when learning the SDF: (1) For real-world scenes, especially unbounded outdoor environments, the scale of the background ($\mathbb{R}^3 \setminus U$) is significantly large and costly to represent using an explicit mesh unified with the foreground U. Existing methods often manually exclude the background using known foreground segmentation masks [134, 174]. This approach is inconvenient in practice, and visual modeling of the background is still necessary for tasks such as scene image synthesis. (2) As the scale of U increase, the memory required to execute differentiable marching algorithms grows significantly.

We address challenge (1) by incorporating an additional set of Gaussians to handle the visual components from the background. Specifically, we construct background Gaussians $\mathcal{GS}_{\text{bg}} = \{\alpha_k, \boldsymbol{\mu}_k, \boldsymbol{c}_k, \boldsymbol{\Sigma}_k\}$ using the original form of 3DGS. Due to their explicit nature, $\mathcal{GS}_{\text{bg}}$

can be directly combined with the foreground Gaussians $\mathcal{GS}_{\text{fg}}$ attached to the mesh (detailed in the next section), enabling rendering and training.

For challenge (2), we apply two schemes in the use of the SDF grid to mitigate the storage and computational demands, thus enabling a larger scale of U:

Visible Node Optimization During each training iteration, only the scene elements within a single camera's viewing frustum are learning. Therefore, we extract mesh faces only from the visible grid nodes in $\{\boldsymbol{x}_n\}_{n=1}^{N}$ using the camera's intrinsic and extrinsic parameters. Specifically, the visible subset is calculated by:

$$\{\hat{\boldsymbol{x}}_{\boldsymbol{m}}\} := \left\{\boldsymbol{x}_{\boldsymbol{n}} \middle| \text{NDC}(\boldsymbol{x}_{\boldsymbol{n}}) \in [-1,1]^3\right\}, \tag{4.9}$$

where $\text{NDC}(\boldsymbol{x}_{\boldsymbol{n}})$ is the transformation from world-space coordinates $\boldsymbol{x}_{\boldsymbol{n}}$ to their normalized device coordinates (NDC) via the projection matrix. The marching algorithm is only applied to cells whose eight corner nodes are within $\{\hat{\boldsymbol{x}}_{\boldsymbol{m}}\}$.

Coarse-to-fine Grid Starting with a high-resolution grid can lead to redundant faces, causing unstable learning and memory issues. To address this, we begin with a low-resolution grid and progressively increase the resolution during training. The resolution is enhanced by linearly interpolating between the nodes of the current grid and the refined grid.

Given the bounding box (BBox) with size (L_x, L_y, L_z) that tightly encloses the foreground region U, and an expected cell count C, the cubic cell size of the SDF grid is set to $(L_x/s) \times (L_y/s) \times (L_z/s)$, where $s = \sqrt{[}3]{L_x \cdot L_y \cdot L_z / C}$. During optimization, we perform 4 coarse-to-fine operations, incrementally increasing the cell count by $1.5\times$ each step until the target count C is reached.

4.3.3 Gaussians Consistent with Scene Geometry

As shown in Fig. 4.4 (a), we employ K Gaussians to represent the surface appearance within a triangle $(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)$, where K depends on desirable level of detail.

Production note: This figure is not included in this digital copy due to copyright restrictions.

Figure 4.4: Sub-figure (a) shows using $K = 1, 3, 6$ Gaussians to represent the appearance of a triangle face, (b) defines a local coordinate frame, and (c) illustrates applying linear transformation $\boldsymbol{M}$ to let Gaussians adapt to the irregular triangle. See Sec. 4.3.3 for more details.

The centers of the Gaussians are placed at specific barycentric coordinates within the triangle. Let these barycentric coordinates be $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K \in \mathbf{S}^2$, where $\mathbf{S}^2$ is the 2-simplex: $\mathbf{S}^2 = \{(x_1, x_2, x_3) | x_n \geq 0, x_1 + x_2 + x_3 = 1\}$. In the world coordinate system, the center of a Gaussian is computed as:

$$\boldsymbol{\mu}_k = [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3] \cdot \boldsymbol{\xi}_k. \tag{4.10}$$

Note that although the barycentric coordinates $\{\boldsymbol{\xi}_k\}$ are fixed, the centers $\{\boldsymbol{\mu}_k\}$ depend on the mesh vertices $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$, which in turn depend on the SDF field. The choice of K influences the barycentric coordinates of the Gaussian centers. For $K = 3$, we use

$$\begin{aligned}
\boldsymbol{\xi}_1 &= \left[(3 - \sqrt{3})/6, (3 - \sqrt{3})/6, \sqrt{3}/3\right], \\
\boldsymbol{\xi}_2 &= \left[(3 - \sqrt{3})/6, \sqrt{3}/3, (3 - \sqrt{3})/6\right], \\
\boldsymbol{\xi}_3 &= \left[\sqrt{3}/3, (3 - \sqrt{3})/6, (3 - \sqrt{3})/6\right].
\end{aligned}$$

For $K = 6$, we use

$$\begin{aligned}
\boldsymbol{\xi}_1 &= \left[(3-2\sqrt{3})/6, (3-2\sqrt{3})/6, 2\sqrt{3}/3\right], \\
\boldsymbol{\xi}_2 &= \left[(3+2\sqrt{3})/12, (3-\sqrt{3})/6, (3+2\sqrt{3})/12\right], \\
\boldsymbol{\xi}_3 &= \left[(3-\sqrt{3})/6, (3+2\sqrt{3})/12, (3+2\sqrt{3})/12\right], \\
\boldsymbol{\xi}_4 &= \left[2\sqrt{3}/3, (3-2\sqrt{3})/6, (3-2\sqrt{3})/6\right], \\
\boldsymbol{\xi}_5 &= \left[(3+2\sqrt{3})/12, (3+2\sqrt{3})/12, (3-\sqrt{3})/6\right], \\
\boldsymbol{\xi}_6 &= \left[(3-2\sqrt{3})/6, 2\sqrt{3}/3, (3-2\sqrt{3})/6\right].
\end{aligned}$$

Before determining the Gaussian covariances, we introduce a local coordinate frame as illustrated in Fig. 4.4 (b). This frame has its origin at $\boldsymbol{v}_1$. The axes of the frame are denoted as $\boldsymbol{t}_i = \tilde{\boldsymbol{t}}_i / \|\tilde{\boldsymbol{t}}_i\|$, $i \in \{1,2,3\}$, where

$$\tilde{\boldsymbol{t}}_1 = \boldsymbol{a} \times \boldsymbol{b}, \quad \tilde{\boldsymbol{t}}_2 = \boldsymbol{a}, \quad \tilde{\boldsymbol{t}}_3 = (\boldsymbol{a} \times \boldsymbol{b}) \times \boldsymbol{a}, \tag{4.11}$$

where $\boldsymbol{a} \equiv \boldsymbol{v}_2 - \boldsymbol{v}_1$, $\boldsymbol{b} \equiv \boldsymbol{v}_3 - \boldsymbol{v}_1$, and $\times$ denotes the cross product. The rotation matrix from local space to world space is then $\boldsymbol{R}_{t2w} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3]$.

If the angles of a triangle are all close to $\pi/3$, the Gaussian covariances in the local coordinate frame can be naturally defined as $\boldsymbol{\Sigma}_e = \mathrm{diag}(\epsilon, r^2, r^2)$, representing a thin layer of Gaussian surrounding a small area of the surface. The choice of r is empirical: we set $r = \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|/(2\sqrt{3}+2)$ for $K = 3$, and $r = \|\boldsymbol{v}_1 - \boldsymbol{v}_2\|/(2\sqrt{3}+4)$ for $K = 6$. For triangles with a more irregular aspect ratio, where the area is challenging to approximate by a disc, it is beneficial to apply a linear transformation $\boldsymbol{M}$ on $\boldsymbol{\Sigma}_e$ as visualized in Fig. 4.4 (c). Hence, the covariance matrix in world coordinate system is given by:

$$\boldsymbol{\Sigma} = \boldsymbol{R}_{t2w} \boldsymbol{M} \boldsymbol{\Sigma}_e \boldsymbol{M}^T \boldsymbol{R}_{t2w}^T. \tag{4.12}$$

The derivation of matrix $\boldsymbol{M}$ is as follows.

In the local coordinate system defined in (4.11), the target triangle is denoted as $\boldsymbol{T} = [\boldsymbol{v}_1', \boldsymbol{v}_2', \boldsymbol{v}_3']$, and the reference equilateral triangle as $\boldsymbol{E} = [\boldsymbol{v}_1', \boldsymbol{v}_2', \hat{\boldsymbol{v}}_3']$, where $\hat{\boldsymbol{v}}_3' = l \cdot [0, \frac{1}{2}, \frac{\sqrt{3}}{2}]^T$,

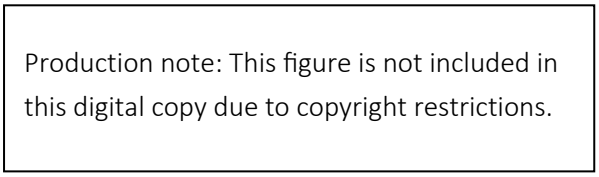

Figure 4.5: Comparison between two approaches to learn appearance.

with $l = \|\boldsymbol{v}_2 - \boldsymbol{v}_1\|$. By solving the equation $\boldsymbol{ME} = \boldsymbol{T}$, we obtain:

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{2\boldsymbol{v}_3'^{(2)} - l}{\sqrt{3}\,l} \\ 0 & 0 & \frac{2\boldsymbol{v}_3'^{(3)}}{\sqrt{3}\,l} \end{bmatrix}, \tag{4.13}$$

where $\boldsymbol{v}_3'^{(i)}$ denotes the i-th component of $\boldsymbol{v}_3'$.

4.3.4 Appearance Model

In the original 3DGS model, the per-Gaussian colors $\{\boldsymbol{c}_k\}$ are optimized directly because the Gaussians remain consistent throughout the training process. However, in our framework, the Gaussians are dynamically constructed from the evolving SDF. Unlike scene geometry, where gradients from the Gaussian centers $\{\boldsymbol{\mu}_k\}$ flow back through the same parametric representation $\{s_n\}_{n=1}^N$ of the SDF, this dynamic construction makes continuous color optimization difficult.

Specifically, the key challenges we face is the lack of direct connections between Gaussians derived from $(\mathcal{V}_t, \mathcal{F}_t)$ and those from $(\mathcal{V}_{t-1}, \mathcal{F}_{t-1})$, where t denotes the t-th training step. A straightforward approach to overcome this would be to compute colors based on their neighboring Gaussians from the previous step, formulated as $\boldsymbol{c}_{i,t} = \frac{1}{|\mathcal{N}|}\sum_{j\in\mathcal{N}} \boldsymbol{c}_{j,t-1}$, where $\boldsymbol{c}_{i,t}$ denotes the color of the i-th Gaussian at step t, and $\mathcal{N}$ is the set of neighbors

based on the Gaussian centers. However, this approach presents two significant drawbacks: it requires substantial computation due to the neighbor search between two large sets of Gaussians, averaging colors within a local region might lead to low effectiveness in learning scene details, as shown in Fig. 4.5 (b).

To address these challenges, we introduce a neural appearance model, $\mathscr{A} : \mathbb{R}^3 \mapsto \mathbb{R}^d$, with d representing the number of the used Spherical Harmonics coefficients. At each step, we determine the color as $\boldsymbol{c} = \mathscr{A}(\boldsymbol{\mu})$ for the Gaussian centered at $\boldsymbol{\mu}$. Thus, the appearance of Gaussians depends on the learnable parameters within $\mathscr{A}$, which remain consistent during training.

We implement $\mathscr{A}$ using hash-grid positional encoding [133] and a lightweight MLP to enhance efficiency and achieve a compact and continuous representation, inspired by NvdiffRec [134]. This neural appearance model can be robustly learned, even as the mesh faces changes from $(\mathscr{V}_{t-1}, \mathscr{F}_{t-1})$ to $(\mathscr{V}_t, \mathscr{F}_t)$. Fig. 4.5 illustrates the experimental comparison between the two approaches. The neural appearance model can achieve nearly 40x speed-up and better learned details than the straightforward nearest neighbor searching approach.

It is worth noting that this coordinate-based neural representation inherently enforces spatial consistency. Since the MLP acts as a continuous function mapping spatial coordinates to visual attributes, Gaussians attached to spatially neighboring mesh faces naturally acquire similar appearance features without requiring explicit adjacency constraints. Simultaneously, the multi-resolution hash-grid encoding enables the model to capture high-frequency variations, allowing for sharp transitions at object edges (e.g., boundaries between different semantic regions) driven purely by photometric supervision. This design eliminates the need for auxiliary semantic inputs, as the explicit mesh topology and the implicit continuity of the neural field essentially provide sufficient regularization.

Additionally, we set the opacity to $\alpha = 1$ to avoid semi-transparent faces.

4.3.5 Rendering and Optimization

As the Gaussians we construct retain the same form as those in the original 3DGS, we employ the same method defined in Eq. (4.3) for rendering an image. For scenes with a background, the rendered Gaussians include both the Gaussians derived from the mesh and the additional background Gaussians. Our method relies only on photometric supervision and utilizes the loss function:

$$\mathscr{L} = (1-\lambda)\mathscr{L}_1 + \lambda\mathscr{L}_{D-SSIM}, \tag{4.14}$$

where λ is a weighting factor, and $\mathscr{L}_1$, $\mathscr{L}_{D-SSIM}$ represent standard L1 and D-SSIM metrics for comparing the rendered image against the ground truth.

Upon the convergence of the mesh and appearance model, we fix the topology defined by faces $\mathscr{F}$ and proceed to jointly refine the vertices and the associated 3D Gaussians. During this phase, Gaussian centers continue to be derived from the face vertices using established barycentric coordinates. However, for covariance and color, we adopt a different approach since we can optimize a consistent set of Gaussian attributes during this phase. Specifically, we sample the spherical harmonics (SHs) from the appearance model $\mathscr{A}$ and make them a learnable, per-Gaussian variable as in the original 3DGS. Moreover, the Gaussian covariance is parameterized into learnable 2D scaling $\hat{\boldsymbol{s}} \in \mathbb{R}^2$ and a complex number $\boldsymbol{a}i + \boldsymbol{b}$ representing rotation, which are similar to the process in SuGaR [62]. It should be noted that this process is optional, as in many cases, the quality of the learned geometry and appearance without refinement is already satisfactory.

4.4 Experiments

In this section, we first detail the implementation across different stages and describe the datasets used for evaluation.

To assess the effectiveness of our method, we compare it with models that specialize in novel view synthesis and surface reconstruction tasks, respectively. Note that the mesh is not used in the rendering of our method. Instead, it serves more as a constraint, ensuring that the Gaussians are aligned with the mesh to faithfully represent the geometry while also reproducing the scene's appearance. We will demonstrate that, even with this constraint, our model can approximate or even surpass the performance of models without these constraints.

Furthermore, we showcase potential applications in object manipulation, and then discuss the adaptability to scene updates and ablation studies in the method analysis section.

4.4.1 Implementation details

The proposed method is implemented using PyTorch and the rasterization toolkit from 3DGS [91]. Experiments are conducted on an RTX3090 GPU with 24GB memory.

4.4.1.1 Joint Learning of Mesh and Gaussians

The resolution of the used SDF grid depends on the scene. For Mip-NeRF360 scenes, we use 150^3 to 250^3 grid cells for the foreground, attaching 3 Gaussians to every face. For NeRF-Synthetic scenes, we use 100^3 grid cells and attach 3 Gaussians to every face. For the appearance model, we adopt an MLP with 2 hidden layers, each comprising 32 neurons. This stage requires 10k iterations for optimization, totally using 15-30 minutes.

4.4.1.2 Refinement

Before the refinement stage, we perform subdivision and decimation on the mesh to unify the number of faces (0.1M for single objects, 0.5M for real-world scenes). We attach 6 Gaussians to each face and compute their covariances and colors using the adaptive Gaussian

Table 4.1: Per-scene quantitative comparisons on NeRF-Synthetic dataset [128].

Method	Chair			Drums			Ficus			Hotdog		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
without mesh												
NeRF [128]	33.00	0.967	0.046	25.01	0.925	0.091	30.13	0.964	0.044	36.18	0.974	0.121
Plenoxels [50]	33.98	0.977	0.031	25.35	0.933	0.067	31.83	0.976	0.026	36.43	0.980	0.037
3DGS [91]	35.82	0.987	0.012	26.17	0.954	0.037	34.83	0.987	0.012	37.67	0.985	0.020
with mesh												
NVdiffRec [134]	31.60	0.969	0.034	24.10	0.916	0.065	30.88	0.970	0.041	33.04	0.973	0.033
NeuManifold [209]	34.39	0.981	0.014	25.39	0.939	0.072	31.91	0.978	0.028	35.69	0.979	0.036
NeRF2Mesh [188]	34.25	0.978	0.031	25.04	0.926	0.084	30.08	0.967	0.046	35.70	0.974	0.058
SuGaR-15K [62]	35.13	0.983	0.014	25.44	0.943	0.057	32.75	0.982	0.018	36.67	0.980	0.021
Ours	**35.40**	**0.986**	**0.013**	**25.75**	**0.952**	**0.041**	**34.01**	**0.986**	**0.014**	**36.80**	**0.985**	**0.018**

Method	Lego			Materials			Mic			Ship		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
without mesh												
NeRF [128]	32.54	0.961	0.050	29.62	0.949	0.063	32.91	0.980	0.028	28.65	0.856	0.206
Plenoxels [50]	34.10	0.975	0.028	29.14	0.949	0.057	33.26	0.985	0.015	29.62	0.890	0.134
3DGS [91]	35.69	0.983	0.016	30.00	0.960	0.034	35.34	0.991	0.006	30.87	0.907	0.106
with mesh												
NVdiffRec [134]	29.14	0.949	0.042	26.74	0.923	0.060	30.78	0.977	0.024	26.12	0.833	**0.080**
NeuManifold [209]	34.00	0.977	0.024	26.69	0.924	0.115	33.40	0.986	0.012	28.63	0.875	0.168
NeRF2Mesh [188]	34.90	0.977	0.025	26.26	0.906	0.111	32.63	0.979	0.038	**29.47**	0.875	0.138
SuGaR-15K [62]	34.87	0.981	0.015	27.86	0.944	0.046	34.81	0.989	0.008	29.22	0.880	0.103
Ours	**34.94**	**0.982**	**0.016**	**27.93**	**0.950**	**0.043**	**34.89**	**0.991**	**0.007**	28.32	**0.883**	0.121

construction and appearance model, with minimal impact on model performance. The refinement process consists of 5-10k iterations and requires 10-20 minutes.

4.4.2 Datasets and Metrics

The NeRF-Synthetic dataset [128] includes 8 scenes with 360° viewpoint settings of images. Besides testing novel view synthesis on this dataset, we quantitatively evaluate reconstructed surfaces using the provided ground-truth mesh. We also evaluate our method on 7 of the 9 real-world scenes in the Mip-NeRF360 dataset [4], excluding *Flowers* and *Treehill* due to licensing restrictions. The strategy for splitting subsets for training and testing follows that of 3DGS [91].

For novel view synthesis, we employ the standard PSNR, L-PIPS, and SSIM metrics. In the surface reconstruction task, we compute the Chamfer Distance (CD) between the extracted mesh and the ground truth.

Production note: This figure is not included in this digital copy due to copyright restrictions.

Figure 4.6: Qualitative comparisons with baseline methods (SuGaR [62], NeRF2Mesh [188]) on the NeRF-Synthetic [128] and Mip-NeRF360 dataset [4].

Table 4.2: Quantitative comparisons on Mip-NeRF360 dataset [4].

Method	Indoor scenes			Outdoor scenes			All scenes		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
without mesh									
Plenoxels [50]	24.83	0.766	0.426	22.02	0.542	0.465	23.62	0.670	0.443
INGP-Base [133]	28.65	0.840	0.281	23.47	0.571	0.416	26.43	0.725	0.339
3DGS [91]	30.41	0.920	0.189	26.40	0.805	0.173	28.69	0.870	0.182
with mesh									
NeRFMeshing [162]	23.83	-	-	22.23	-	-	23.15	-	-
BakedSDF [227]	27.06	0.836	0.258	-	-	-	-	-	-
NeuManifold [209]	27.16	0.813	0.316	-	-	-	-	-	-
NeRF2Mesh [188]	-	-	-	22.74	0.523	0.457	-	-	-
SuGaR-15K [62]	**29.43**	**0.910**	0.216	24.40	0.699	0.301	27.27	0.820	0.253
Ours	29.33	**0.910**	**0.193**	**25.17**	**0.754**	**0.237**	**27.54**	**0.843**	**0.212**

Table 4.3: Per-scene quantitative results on Mip-NeRF360 dataset [4].

Metric	Bicycle	Bonsai	Counter	Garden	Kitchen	Room	Stump	Mean
PSNR	24.18	31.47	27.16	26.41	28.96	29.71	24.91	27.54
SSIM	0.722	0.945	0.888	0.829	0.898	0.908	0.712	0.843
LPIPS	0.260	0.175	0.212	0.161	0.162	0.221	0.291	0.212

4.4.3 Novel View Synthesis

We first evaluate our method on the novel view synthesis task. Tabs. 4.1 and 4.2 present the quantitative results on NeRF-Synthetic and Mip-NeRF360 datasets. Tab. 4.3 includes the per-scene results on Mip-NeRF360 dataset [4].

To ensure a fair comparison, we employ a white background for the NeRF-Synthetic dataset. Our approach is compared against various recent methods, including those with reconstructed mesh and those without.

For the NeRF-Synthetic dataset, our method achieves state-of-the-art rendering quality among methods reconstructing a mesh, and even outperforms some models that solely focus on rendering quality. Furthermore, there is only a slight decrease in performance compared to the original 3DGS [91]. For the Mip-NeRF360 dataset, our method is comparable to or surpasses the state-of-the-art methods [62, 188]. Fig. 4.6 includes qualitative compar-

isons between images rendered by our method and the baselines, demonstrating that our method captures better scene details.

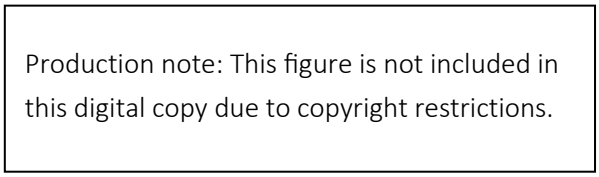

Figure 4.7: Qualitative comparisons of reconstructed meshes with SuGaR [62] on the NeRF-Synthetic [128] and Mip-NeRF360 [4] datasets.

Table 4.4: Quantitative Evaluation of Mesh Quality and Efficiency on the NeRF-Synthetic [128] dataset.

Method	CD (10^{-3})	#Faces	PSNR	Train Time
NVdiffRec [134]	8.15	**80k**	29.05	52 mins
NeRF2Mesh [188]	**5.06**	192k	31.04	46 mins
SuGaR-15K [62]	8.62	1000k	32.09	103 mins
Ours (w/o refine)	8.50	141k	27.68	**16 mins**
Ours	7.27	100k	**32.26**	28 mins

'CD': Chamfer Distance between the ground truth and reconstructed meshes.

4.4.4 Surface Reconstruction

Although our focus is on efficiently learning a hybrid representation that ensures alignment between Gaussians and mesh faces, the surfaces reconstructed by our method are of high quality. In Fig. 4.7, we visualize the reconstructed meshes and their normal maps, comparing our method against the baseline. The results demonstrate that our method effectively captures the surface details of a scene, whereas SuGaR [62] may produce floating patches and holes in the mesh.

For a quantitative evaluation of the meshes, we compare the reconstructed mesh to the ground truth from the NeRF-Synthetic dataset using the Chamfer Distance (CD) metric. The computation method follows NeRF2Mesh [188], where 2.5M points are sampled from the surfaces of both the ground truth and the reconstructed object through ray casting. For a more comprehensive assessment, Tab. 4.4 presents the CD results, the number of faces, the rendering PSNR, and the training time. The results demonstrate that our reconstructed meshes achieve both high quality and training efficiency. Additionally, our Gaussian-based appearance model enables high-quality rendering even with fewer faces compared to NeRF2Mesh.

Furthermore, the learned mesh without refinement is already of satisfactory quality. This indicates that for applications requiring only an untextured mesh, our method can produce excellent results very efficiently, with a training time of just 16 minutes.

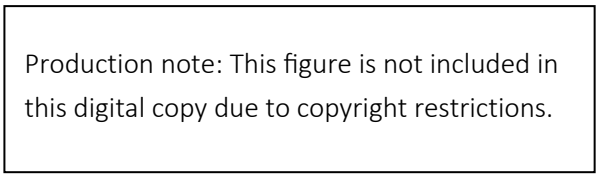

Figure 4.8: Object deformation by hybrid representation of mesh and Gaussians. We employ four Blender modifiers to manipulate the mesh, subsequently render perspectives using the bound Gaussians.

4.4.5 Object Deformation by Hybrid Representation

As discussed in our introduction section, the hybrid representation not only benefits from the high-quality and efficient rendering of Gaussians but also offers the convenience of manipulation through the mesh. To demonstrate the potential applications of this work, we selected object deformation tasks for testing. Specifically, we deform the learned mesh using four common mesh modifiers (Twist, Stretch, Bend, and Taper) in Blender and correspondingly adjust the bound Gaussians using the technique described in Sec. 4.3.3. We then use the Gaussian-based rasterization to render images of the deformed object, as shown in Fig. 4.8. More advanced operations, as explored in [55, 83], are also compatible with the representation we have obtained.

4.4.6 Method Analysis

4.4.6.1 Efficiency

We assess our method's efficiency by comparing the average time taken to reconstruct scenes from the NeRF-Synthetic and Mip-NeRF360 datasets. Both our method and the baseline methods are tested under identical hardware conditions. The results in Tab. 4.5

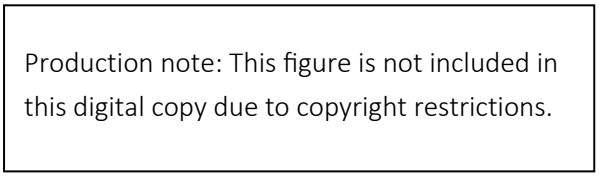

Figure 4.9: Adaptation to scene updates. (a) Models initially trained on the original scene. (b) and (c): Models adapted to the updated scene for 500 and 5000 steps, respectively. (d): The meshes learned on the updated scene.

reveal that our method is not only faster but also provides superior rendering quality.

4.4.6.2 Adaptability in Scene Updates

One unique feature of our method is the simultaneous learning of topology (defined by mesh faces) and Gaussians. In contrast, the existing method, SuGaR [55, 62, 197], separates the learning of topology and Gaussians, as the mesh extraction process (Poisson Surface Reconstruction) is non-differentiable for Gaussian rasterization. In practice, when part of the scene requires updating, our model can directly adapt to these changes without re-learning from scratch, whereas the existing method cannot. Fig. 4.9 presents the results of handling scene updates using our method and SuGaR. Our method successfully updates the geometry and appearances of the modified regions, while SuGaR struggles to adapt to new scene geometries with its fixed topology.

4.4.6.3 Ablation Studies

In this section, we examine the impact of three factors: 1) the adaptive Gaussian covariance, 2) the refinement stage, and 3) the number of Gaussians per face. To evaluate the effect of adaptive Gaussian covariance, we set $\boldsymbol{M}$ in Eq. (4.12) to be an identity matrix. Tab. 4.6

Table 4.5: Evaluation on Efficiency. We report the training time and PSNR metric across different methods.

Method	NeRF-Synthetic [128]		Mip-NeRF360 [4]	
	PSNR↑	Train Time↓	PSNR↑	Train Time↓
NVdiffRec [134]	29.05	52 mins	-	-
NeRF2Mesh [188]	31.04	46 mins	-	-
SuGaR-15K [62]	32.09	103 mins	27.27	132 mins
Ours	**32.26**	**28 mins**	**27.54**	**61 mins**

'-' indicates that the method does not have results reported in their paper and fails to reconstruct all scenes of the dataset using their published codes.

Table 4.6: Ablation studies on adaptive covariance and refinement stage. We report the average PSNR across all scenes of the NeRF-Synthetic [128] and Mip-NeRF360 [4] datasets.

Adaptive Cov.	Refine	NeRF-Synthetic [128]	Mip-NeRF360 [4]
x	✓	32.02	27.19
✓	x	27.78	24.44
✓	✓	**32.26**	**27.54**

Table 4.7: Ablation Study on the Number of Gaussians per Face. This table shows the PSNR results on the Mip-NeRF360 [4] dataset.

PSNR		Stage-2		
		#.GS/face=1	#.GS/face=3	#.GS/face=6
Stage-1	#.GS/face=1	26.80	27.12	27.22
	#.GS/face=3	26.89	27.21	**27.54**

'Stage-1': Joint learning of mesh and Gaussian appearance.
'Stage-2': Refinement stage that fixes mesh faces.
Note: The table does not include #.GS/face≥6 for Stage-1 due to OOM issues in some scenes.

shows that both adaptive Gaussian covariance and the refinement stage significantly enhance rendering quality.

The choice of the number of Gaussians per face involves a trade-off between computational consumption and performance. As shown in Tab. 4.7, increasing the number of Gaussians per face benefits rendering quality. To control the memory footprint and train-

ing time within an acceptable range while still achieving excellent rendering quality, we choose 3 Gaussians per face during the joint learning of mesh and Gaussian appearance, but use 6 Gaussians for the refinement stage.

4.5 Summary

We have proposed a novel learning method to capture comprehensive 3D scene information from multiple views. The method simultaneously extracts both the geometry and the physical properties affecting the observed appearance. The geometry is extracted in the explicit form of triangle meshes. The appearance properties are encoded in 3D Gaussians that are bound to the mesh faces. Thanks to the 3DGS-based differentiable rendering, we are able to establish an effective and efficient learning procedure by directly optimizing the photometric loss. Experiments verify that the resulting representation enjoys both high-quality rendering and controllability.

CHAPTER 5

DYNAMIC APPEARANCE PARTICLES NEURAL RADIANCE FIELD

5.1 Background

Physical world is inherently dynamic, where objects undergo continuous motion, complex deformations, and topological changes. The ability to reconstruct and render dynamic scenes is essential for modeling physical reality. This capability has a wide range of applications, from immersive virtual reality (VR) and cinematic visual effects to the simulation of autonomous agents interacting with changing environments. Unlike static reconstruction, which focuses solely on geometry and appearance, dynamic scene modeling requires capturing the temporal evolution of these attributes. It is challenging to maintain temporal consistency and physical plausibility.

Recent advancements in neural radiance fields (NeRFs) [128] have shown remarkable success in modeling 3D scenes with a continuous representation. These models facilitate high-fidelity rendering from novel viewpoints without requiring explicit geometry. One of the most promising extensions is Dynamic NeRFs [113, 154], which aims to model dynamic scenes by incorporating a time-varying radiance field. There are two primary schemes: di-

rectly adding a time dimension or introducing a deformation field before the canonical radiance field. The deformation-field scheme separately represents the motion and appearance of a dynamic scene and has gained popularity due to its effectiveness.

Existing models employ an Eulerian formulation for both the deformation and appearance fields. A learnable field model, such as a group of neural networks, uses the 3D Euclidean coordinates as its input query and outputs the desired physical quantities (field variables). This scheme is effective when the field needs to be defined throughout the entire Euclidean space, such as when considering the light scattering characteristics over a three-dimensional scene. However, the Eulerian field representation can be problematic when the quantities of interest are confined to specific regions or supported on a sub-manifold within the Euclidean space, for instance, within a solid object or on the surface. It might waste model capacity and make the subsequent use of the model inconvenient. For example, existing methods that apply dynamic NeRF to explore the interaction between objects and the environment require manual preprocessing like shape extraction [160] and inverse deformation estimation [16].

To address these limitations, we propose to employ particle-based representation for moving or deforming objects within dynamic scenes. In our method, particles represent a finite approximation of the distribution of physical quantities that determine appearance. More specifically, in a volumetric rendering scheme, if the light properties, color, and scattering probability at a location $\boldsymbol{x} \in \mathscr{X}$ are determined by a view-independent physical feature $\boldsymbol{f}$ and the viewpoint, then the spatial distribution of $\boldsymbol{f}$ over $\mathscr{X}$ becomes the primary focus of modeling. For this purpose, particles are used to quantize the space $\mathscr{X}$, with the advantage that the distribution of $\boldsymbol{f}$ can be made time-varying by using a movement model for the particles. This particle-based representation can be integrated with an existing Eulerian appearance field that represents static elements. Therefore, we achieve a hybrid (static-Eulerian, dynamic-Lagrangian) NeRF model that can learn from monocular

videos using only photometric supervision.

We propose a dynamic modeling approach for generic fields. This methodology was developed through a case study of light-scattering fields and demonstrated with computer vision applications. However, the proposed Lagrangian framework is generic and can be applied to other fields. This strength has also been discussed in a preliminary task on modeling physical motion via visual observations, which remains challenging for traditional computer vision methods.

This design choice is not only motivated by the physical and computational advantages of particle-based modeling, but also builds upon our earlier work on structured representation learning [108]. In the previous study, we proposed a mechanism to disentangle task-relevant and task-irrelevant factors using capsule-based encoders trained under adversarial supervision. The goal was to enable semantic separation of variation sources within a shared representation space, facilitating robust transfer across tasks. In the present work, we adopt a similar philosophy by explicitly modeling the scene's dynamic and static components through distinct mechanisms. This separation enhances interpretability, supports modular learning, and enables editing and physical reasoning in dynamic scenes.

Moreover, the technique is developed in combination with the NeRF model due to the consistency in their computational frameworks. With more sophisticated adaptive models, it is possible to extend the dynamic modeling to the more efficient Gaussian splatting, as discussed in the previous chapter.

In summary, this work introduces the Dynamic Appearance Particle Neural Radiance Field (**DAP-NeRF**). The major contributions are as follows:

- We have designed a hybrid framework of radiance field models. The dynamic elements of the fields are described using a particle-based representation, which corresponds to a Lagrangian approach to field models. The introduction of the Lagrangian model complements the widely adopted Eulerian dynamic NeRFs, which specify the

static elements of the scene in our framework. The hybrid framework serves as more than an adequate appearance model of dynamic scenes. The employed particles also provide an explicitly interpretable and physically meaningful description of the motions.

- We have developed an efficient and effective computational structure for the proposed hybrid framework. This structure has the capability to automatically identify dynamic and static elements.

- We have constructed a dataset and introduced a metric to evaluate the motion modeling of dynamic NeRFs.

Empirical studies have shown that our framework achieves state-of-the-art performance in novel view synthesis tasks. We've demonstrated that our framework produces particles that effectively capture the dynamics of moving objects, facilitating scene decoupling and motion editing. Moreover, the learned particle motion model shows superior quantitative results in motion modeling, surpassing existing methods that deform a canonical field.

5.2 Preliminaries on Neural Radiance Fields (NeRFs)

Neural Radiance Fields (NeRFs) [128] represent a static 3D scene as a continuous volumetric field parameterized by a neural network. Specifically, a multilayer perceptron (MLP) is trained to map a 3D spatial location $\boldsymbol{x} \in \mathbb{R}^3$ and a viewing direction $\mathbf{d} \in \mathbb{S}^2$ to a color $\mathbf{c} \in \mathbb{R}^3$ and a volume density $\sigma \in \mathbb{R}^+$:

$$(\sigma, \mathbf{c}) = \text{MLP}(\boldsymbol{x}, \mathbf{d}) \tag{5.1}$$

Given a camera pose, NeRF renders a pixel by casting a ray $\mathbf{r}(s) = \mathbf{o} + s\mathbf{d}$, $s \in \mathbb{R}^+$ from the camera center $\mathbf{o}$ through the pixel, and accumulating color contributions along the ray

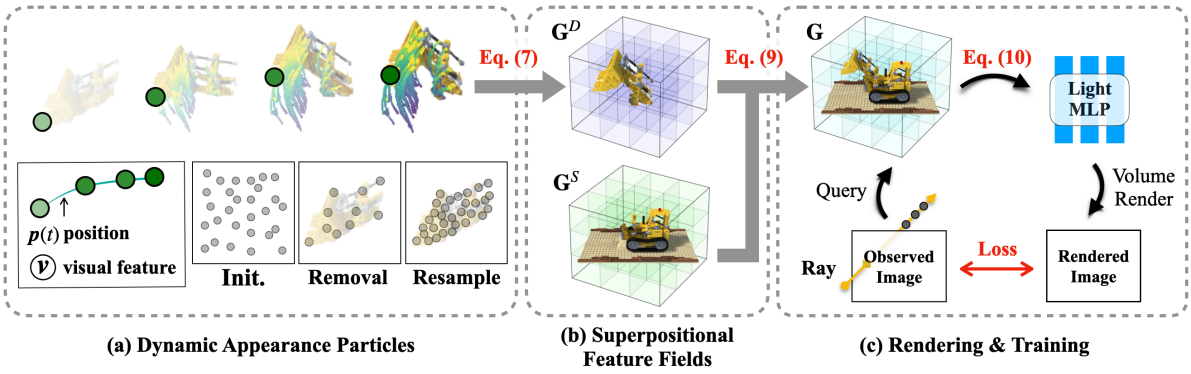

Figure 5.1: Method Overview. **(a)** Particles represent observed movements. Each particle $\boldsymbol{p}$ corresponds to a small volume of material that represents a semantically meaningful part of a moving object. The position $\boldsymbol{p}(t)$ forms an explicit dynamic model of the object part. $\boldsymbol{v}$ denotes the visual feature of the particle (See Sec. 5.3.1). **(b)** Particles are integrated into a grid-based feature field, enabling efficient computation and superposition with the static feature field (See Sec. 5.3.3). **(c)** Colors of rays are computed by querying sampled points on the superpositional radiance field and performing volume rendering. Photometric loss is then calculated to optimize the model (see Sec. 5.3.4).

using volume rendering [119]:

$$\mathbf{C}(\mathbf{r}) = \int_{s_n}^{s_f} T(s)\,\sigma(\mathbf{r}(s))\,\mathbf{c}(\mathbf{r}(s),\mathbf{d})\,ds \tag{5.2}$$

where s_n, s_f denote the near and far bounds of the ray, $T(s) = \exp\left(-\int_{s_n}^{s}\sigma(\mathbf{r}(s'))ds'\right)$ is the accumulated transmittance. The neural field is trained end-to-end by minimizing the photometric error between rendered and ground-truth pixel colors across multiple views.

NeRF achieves high-quality novel view synthesis, but assumes a static scene and requires dense input views, limiting its applicability in dynamic settings.

5.3 Method

Consider the dynamic radiance field model: A pixel of the camera frame at time t is computed by casting a ray $\mathbf{r}(s) = \mathbf{o} + s\mathbf{d}$, $s \in \mathbb{R}^+$ from the camera center $\mathbf{o}$ to the pixel on the image plane:

$$\mathbf{C}(\mathbf{r}, t) = \int_{s_n}^{s_f} T(s)\sigma(\mathbf{r}(s), t)\mathbf{c}(\mathbf{r}(s), \mathbf{d}, t)ds \tag{5.3}$$

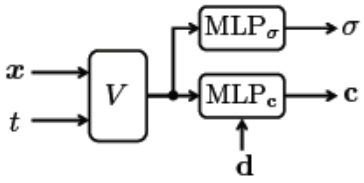

Figure 5.2: Commonly adopted structure of dynamic NeRFs.

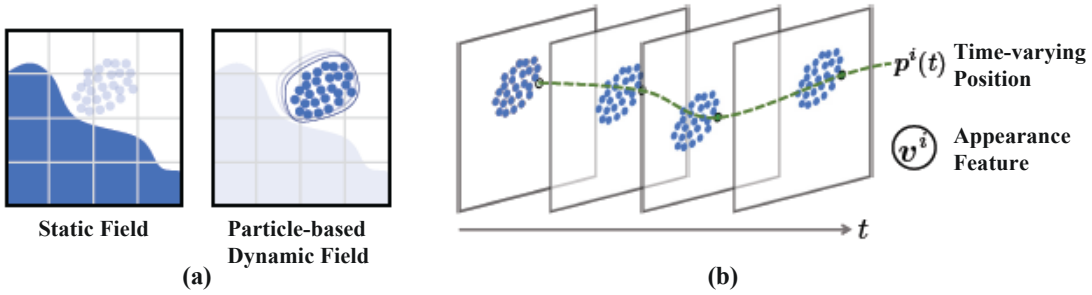

Figure 5.3: Overview of the proposed hybrid representation. **(a)** Superposition of dynamic and static fields modeled by particle-based and Eulerian (voxel-grid-based) representations. We explicitly decouple the reconstruction tasks: the particles represent the moving parts to handle motion modeling, while the static field represents the stationary environment for background reconstruction. **(b)** Two main attributes represented by particles. The particle trajectory $\boldsymbol{p}^i(t)$ is time-varying, while the appearance feature $\boldsymbol{v}^i$ is time invariant.

where $\mathbf{C}(\mathbf{r}, t)$ denotes the pixel color rendered at time $t \in \mathbb{R}^+$, s_n, s_f denote the near and far bounds of the ray. $\mathbf{r}(\cdot)$ returns a 3D location along the ray. $T(s) = \exp\left(-\int_{s_n}^{s} \sigma(\mathbf{r}(s'), t) ds'\right)$ is the accumulated transmittance. Given a location and time t (and direction $\mathbf{d}$), $\sigma(\cdot)$ stands for the volume density, and $\mathbf{c}(\cdot)$ stands for the emitted radiance.

An effective strategy to specify the model of $\sigma(\cdot)$ and $\mathbf{c}(\cdot)$ is to learn small MLPs and localized feature vectors distributed at grid nodes in the 3D region of interest [45]. Fig. 5.2 displays the computational structure. The generic formulations of the components are as follows:

$$
\begin{aligned}
V(\boldsymbol{x}, t) &: \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}^C \\
\mathrm{MLP}_\sigma(\cdot) &: \mathbb{R}^C \mapsto \mathbb{R} \\
\mathrm{MLP}_{\boldsymbol{c}}(\cdot) &: \mathbb{R}^C \times \mathbb{R}^3 \mapsto \mathbb{R}^3
\end{aligned}
\tag{5.4}
$$

where the majority of the field information is modeled by V. We adopt this framework and

embed the Lagrangian dynamic model in V. However, it should be noted that the proposed technique is not tightly coupled to a specific overarching architecture. It is possible to integrate the proposed Lagrangian representation into alternative frameworks, such as Instant-NGP [133].

We model the feature field of the light radiance V by the following superposition:

$$V(\boldsymbol{x}, t) := (1 - w_t(\boldsymbol{x}))\, V^{\mathrm{s}}(\boldsymbol{x}) + w_t(\boldsymbol{x})\, V_t^{\mathrm{d}}(\boldsymbol{x}) \tag{5.5}$$

where $V^{\mathrm{s}}(\boldsymbol{x})$ and $V_t^{\mathrm{d}}(\boldsymbol{x})$ represent the static and dynamic components of the field, respectively. The static field component, $V^{\mathrm{s}}(\boldsymbol{x})$, employs a time-independent grid-based model. This aligns with the canonical feature field in [45]. The particle-based dynamic field component, $V_t^{\mathrm{d}}(\boldsymbol{x})$, is proposed by this work and will be introduced in detail in the following subsections. The superposition is determined by $w_t(\boldsymbol{x})$, which takes value 1 if the dynamic field is effective at $(\boldsymbol{x}, t)$ and 0 otherwise. The support of the dynamic field (where it is effective) is implied by the particle representation of $V_t^{\mathrm{d}}(\boldsymbol{x})$. Fig. 5.3 (a) shows this superposition scheme.

It is helpful to notice a denotation convention due to the hierarchy of concepts that are adopted in this work. We denote time t on the right-hand side of (5.5) in subscripts, as opposed to an independent argument to the functions. This is to clarify the fact that the dynamic component $V_{\cdot}^{\mathrm{d}}(\cdot)$ consists of an ensemble of particles. When considering the composition of particles making up $V_{t_0}^{\mathrm{d}}(\cdot)$, the focus is on the discretization of a field at a specific time t_0. The time argument is fixed and does not affect the construction of the instantaneous status of $V_{t_0}^{\mathrm{d}}(\cdot)$. The dynamics of the system are encoded in the individual particle models.

5.3.1 Particle-based Dynamic Model

The dynamic field V_t^{d} is a continuous function, as a component of a NeRF model. We are concerned with integrating V_t^{d} in finite volumes. Therefore, we formulate V_t^{d} as

$$\begin{aligned} V_t^{\mathrm{d}}(\boldsymbol{x}) &= (V_t^{\mathrm{d}} * \delta)(\boldsymbol{x}) \\ &= \int V_t^{\mathrm{d}}(\boldsymbol{x}')\delta(\boldsymbol{x}-\boldsymbol{x}')d\nu(\boldsymbol{x}') \end{aligned} \tag{5.6}$$

where $\delta(\cdot)$ is the Dirac function $\delta(\boldsymbol{r}) = 0,\ \boldsymbol{r} \neq 0$ and $\int \delta(\boldsymbol{r})d\nu = 1$ and $\nu(\cdot)$ is the volume integration variable.

Replacing the $\delta(\cdot)$ by a kernel with finite support, $\delta \to W$, where W is a kernel locally supported close to 0 ($W(\boldsymbol{x}) = 0$ when $\|\boldsymbol{x}\|$ exceeds a small radius and $\int W(\boldsymbol{x})d\nu(\boldsymbol{x}) = 1$), the finite-width kernel approximation leads to a quantization scheme of a physical field using the notion of *particles* [59]. A particle represents the interesting physical quantities within a small spatial extent. In the NeRF problem, it is the *visual features* that are of the central interest, i.e., the vector quantity produced by the module V in (5.4). The visual feature is fed into subsequent MLPs to output the light radiance properties (density and color). In our model, the visual feature field is represented via smoothed particles [59]. When querying a location $\boldsymbol{x}$ at time t,

$$V_t^{\mathrm{d}}(\boldsymbol{x}) \approx \sum_{i=1}^{N_p} \boldsymbol{v}^i W(\boldsymbol{x} - \boldsymbol{p}^i(t)) \tag{5.7}$$

where i specifies one of the N_p particles. As shown in Fig. 5.3 (b), each particle represents the movements of a small finite volume of certain visual characteristics, consisting of $\boldsymbol{p}^i(t)$ and $\boldsymbol{v}^i$. $\boldsymbol{p}^i(t)$ is a 3D trajectory, mapping time t to a 3D position. The latter, $\boldsymbol{v}^i$, is a time-invariant appearance feature.

The particles are referred to as *appearance particles*, which reflects their contribution to the NeRF model. From the perspective of explicit modeling, these particles function as the "geometric primitives" for dynamic scenes. Unlike implicit deformation fields that

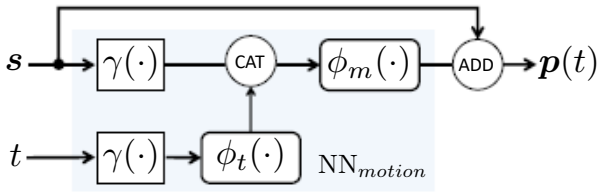

Figure 5.4: Computational structure for time-varying position of a particle. γ is the positional encoding function [128], ϕ_m and ϕ_t are 3 and 2-layer MLP, 'CAT' is concatenate operation and 'ADD' is element-wise addition.

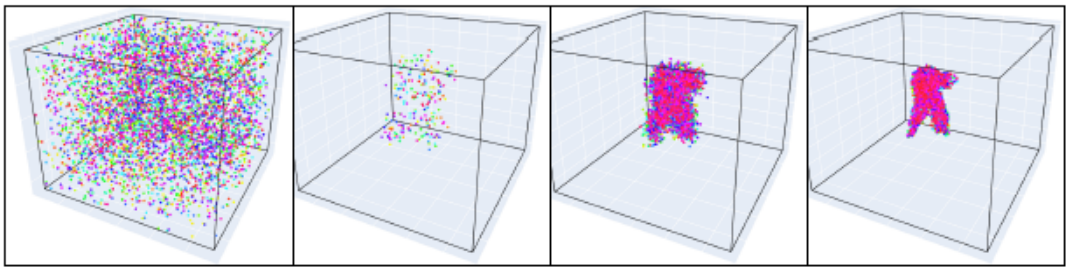

Figure 5.5: Training process of particles at $t = 0$. Panes show the particles at $t = 0$ during different stages of training. (See Sec. 5.3.2)

indirectly warp a continuous coordinate space, appearance particles act as discrete, spatially localized entities. They explicitly define the underlying geometry of moving objects through their positions $\boldsymbol{p}_i(t)$ while carrying the necessary visual attributes $\boldsymbol{v}_i$, effectively bridging the gap between explicit geometric modeling and neural rendering.

While our method and ParticleNeRF [1] share a foundational concept from [184], our approach employs significantly different computational details, which are presented in the next subsection.

5.3.2 Appearance Particle Model

Recall that an individual particle carries two pieces of dynamics information, a visual feature $\boldsymbol{v}$ and time-varying position $\boldsymbol{p}(t)$. In DAP-NeRF, $\boldsymbol{v}^i$ is a per-particle learnable feature vector of appearance characteristics. $\boldsymbol{p}^i(\cdot)$ is implemented using a small neural network,

$$\boldsymbol{p}^i(t) := \mathrm{NN}_{motion}(t, \boldsymbol{s}^i) + \boldsymbol{s}^i \tag{5.8}$$

where NN_{motion} is the neural network that computes the particle motion as the offset from the starting point $\boldsymbol{s}^i$. The computation is illustrated in Fig. 5.4. Therefore, the learning model of particles consists of i) per-particle visual feature $\boldsymbol{v}^i$, ii) per-particle starting point $\boldsymbol{s}^i$ and iii) the network parameters of NN_{motion} that are shared by all particles.

The particles are designed to adaptively model the dynamic components using the following mechanism:

Initialization We initially place a set of particles by randomly distributing them within the pre-defined bounding box encompassing the scene. For each placed particle, we initialize $\boldsymbol{s}^i$ as the spatial coordinate. The setup is illustrated in the first pane of Fig. 5.1 (b).

Particle Removal and Re-sampling To ensure that particles represent only the dynamic components of a scene, we introduce a run-time strategy that automatically removes and re-samples particles during training. Specifically, we

- remove particles that are located in the known free space or rarely move throughout the whole timeline. Free spaces are identified using the occupancy mask technique described in [185], i.e., alpha value less than a threshold ϵ_α. The immobility of a particle i is identified by the length of its trajectory, i.e., $\int_0^1 \| \frac{\partial \boldsymbol{p}^i(r)}{\partial r} \|_2 dr$ not exceeding a threshold ϵ_{traj}. The removal step is performed every few training steps from the beginning. The second pane of Fig. 5.5 shows the result of the removal.

- conduct random re-sampling immediately after each removal step, aiming to preserve the overall particle count. New particles are randomly placed within a small radius around the remaining particles, and each initially inherits the visual feature $\boldsymbol{v}$ of the particle from which it was sampled. The third pane of Fig. 5.5 displays the first re-sampling, and the fourth pane is the final result of alternate removal and re-sampling.

5.3.3 Efficient Computation

Dynamic Feature Field. The computation of (5.7) requires nearest neighbor search, which is expensive when the number of particles N_p is large. We propose a computational scheme referring to the grid structure, inspired by the strategy of [184].

Specifically, we employ a grid $\mathbf{G}^D$ of $N_x \times N_y \times N_z$ nodes and each node is associated with a C-dimension feature. The node feature is of the same format as the particle visual feature $\boldsymbol{v}$. Then the computation of $V_t^{\mathrm{d}}(\boldsymbol{x})$ consists of two steps: i) propagating appearance information from particles to grid nodes, and ii) passing the information to the query location.

At time t, we update features at n-th node of $\mathbf{G}^D$ by:

$$\mathbf{G}^D(n) = \sum_{i \in \mathcal{N}(n)} w_{i \to n} \boldsymbol{v}^i \tag{5.9}$$

where $\mathcal{N}(n)$ represents the particles in one of the cells adjacent to n. The *distribution weight* $w_{i \to n}$ is the tri-linear interpolation weight for particle i in the grid cell containing n. Having appearance features propagated to $\mathbf{G}^D$, $V_t^{\mathrm{d}}(\boldsymbol{x})$ is computed via standard tri-linear interpolation

$$\mathrm{Interp}(\boldsymbol{x}, \mathbf{G}^D) : (\mathbb{R}^3, \mathbb{R}^{C \times N_x \times N_y \times N_z}) \mapsto \mathbb{R}^C \tag{5.10}$$

We query this field by directly accessing neighbor grid nodes instead of searching neighbors over all particles like the previous method [1]. This approach reduces the computational complexity from $\mathrm{O}(N_p)$ to $\mathrm{O}(1)$.

Static Feature Field. For the static field component $V^s(\boldsymbol{x})$, we adopt the canonical feature grid as in [45]. The static feature grid $\mathbf{G}^S \in \mathbb{R}^{C \times N_x \times N_y \times N_z}$ shares the same configuration as $\mathbf{G}^D$, and is initialized with zero values throughout. The querying of this feature field is also done via tri-linear interpolation.

Field Superposition. Since $\mathbf{G}^S$ and $\mathbf{G}^D$ are both grid-based and aligned in shape, the superposition of fields as described in (5.5) (for a convenient simplified reminder, consider $V := (1-\alpha)V^s + \alpha V^d$) is straightforwardly implemented as

$$\begin{aligned} \mathbf{G}(n) &= (1-m(n))\mathbf{G}^S(n) + m(n)\mathbf{G}^D(n) \\ m(n) &= \begin{cases} 1 & \text{if} \quad \sum_i^{N_p} w_{i\to n} > 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.11}$$

where n denotes the node number, and $w_{i\to n}$ is the tri-linear interpolation weight for particle i in the grid cell containing n. (5.11) means that the static field is active only when there is no particle in the neighboring grid cells.

We implement superpositional feature field via

$$\begin{aligned} \boldsymbol{f}_v &= \text{Interp}(\boldsymbol{x}, \mathbf{G}) \\ V(\boldsymbol{x}) &= \phi_v(\gamma(\boldsymbol{f}_v), \gamma(\boldsymbol{x})) \end{aligned} \tag{5.12}$$

where ϕ_v is a small neural network detailed in Fig. 5.13 (a), 'Interp' denotes tri-linear interpolation.

5.3.4 Optimization

Pipeline. The training process of our model follows established practices in dynamic NeRFs [45, 154], involving multiple iterations of model optimization. Each iteration begins by selecting an image from a video sequence. Spatial points are then sampled along rays that extend from the camera center to a batch of random pixel locations. Using the image's timestamp, particle states are updated according to our methods detailed in Sec. 5.3.2. Subsequently, we create the superpositional radiance field, from which appearance information is queried using the sampled spatial points.

Finally, we employ the standard volume rendering [128] on each ray to compute the colors of the pixels and render an image. To optimize the model, losses between the rendered and actual images are computed, as detailed in subsequent paragraphs.

Losses. Following TiNeuVox [45], our training process involves selecting camera rays randomly, querying the radiance field, and applying standard volumetric rendering [128] to compute ray colors and losses. Firstly, we adopt the standard photometric loss, quantified as the squared error between the rendered color $\mathbf{C}(\mathbf{r})$ and the ground-truth color $\hat{\mathbf{C}}(\mathbf{r})$:

$$\mathcal{L}_{\text{photo}} = \|\hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r})\|_2^2 \tag{5.13}$$

Following [45, 185], we apply two additional losses. The per-point RGB loss $\mathcal{L}_{\text{ptrgb}}$ stabilize the optimization process by supervising all sampled points with the target color. The background entropy loss $\mathcal{L}_{\text{bg}}$ encourages the model to better distinguish between foreground and background areas.

Since explicitly modeling dynamic elements is non-trivial, we further employ two regularization terms for the superpositional feature field and the motion model of particles. Specifically, we compute the total variation loss on the superpositional feature grid $\mathbf{G}$ using

$$\begin{aligned}\mathcal{L}_{\text{tvf}} = \text{TV}(\mathbf{G}) := \frac{1}{N}\sum_{i,j,k}\Big(&\left\|\mathbf{G}_{i,j,k} - \mathbf{G}_{i+1,j,k}\right\|_2 + \\ &\left\|\mathbf{G}_{i,j,k} - \mathbf{G}_{i,j+1,k}\right\|_2 + \left\|\mathbf{G}_{i,j,k} - \mathbf{G}_{i,j,k+1}\right\|_2\Big)\end{aligned} \tag{5.14}$$

where N is the total number of grid nodes, and $\text{TV}(\cdot)$ is the operator calculating the total variation loss on a grid. Inspired by the As Rigid As Possible (ARAP) loss, the second regularization term aims to preserve local rigidity within the particles. Due to the costly nearest neighbor search required by the original ARAP loss, we employ an efficient computation method as introduced in Sec. 5.3.3. Specifically, we construct a *motion grid* $\mathbf{G}^m$ where each node summarizes the movement of nearby particles via

$$\mathbf{G}^m(n) = \frac{1}{w_n}\sum_{i\in\mathcal{N}(n)} w_{i\to n}\text{NN}_{motion}(t, \boldsymbol{s}^i) \tag{5.15}$$

where $w_n = \sum_{i\in\mathcal{N}(n)} w_{i\to n}$ is used for normalization. The term $\text{NN}_{motion}(t, \boldsymbol{s}^i)$ denotes particle offset at time t as defined in (5.8). $\mathcal{N}(n)$ and $w_{i\to n}$ are defined as in (5.9). The motion

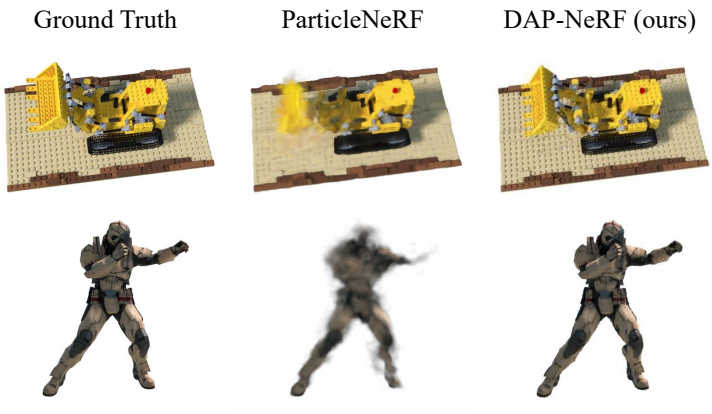

Figure 5.6: Comparative analysis with ParticleNeRF [1] on the D-NeRF dataset [154]. Although both methods utilize particle-based representations, ParticleNeRF struggles with monocular video data.

regularization loss for particles is then implemented by

$$\mathcal{L}_{\text{tvm}} = \text{TV}(\mathbf{G}^m) \tag{5.16}$$

Therefore, the overall loss for the model is defined as

$$\mathcal{L} := \mathcal{L}_{\text{photo}} + w_1\mathcal{L}_{\text{ptrgb}} + w_2\mathcal{L}_{\text{bg}} + w_3\mathcal{L}_{\text{tvf}} + w_4\mathcal{L}_{\text{tvm}} \tag{5.17}$$

Grid-based Coarse to Fine. Following the common practice in [45, 185], all the grids ($\mathbf{G}^D$, $\mathbf{G}^S$ and $\mathbf{G}$) will increase their total voxel number from an initial value (e.g. 80^3) to a final value (e.g. 160^3) during training. Specifically, the bounding box (BBox) that tightly encloses the camera frustums of the training views is initially identified. The lengths of this BBox are denoted as L_x, L_y, and L_z, while the expected number of voxels is represented as M. The size of the grid is then established as $(L_x/s) \times (L_y/s) \times (L_z/s)$, where $s = \sqrt[3]{L_x \cdot L_y \cdot L_z / M}$.

5.4 Experiments

After testing the validity of the proposed DAP-NeRF in novel view synthesis tasks, we demonstrate the capacity and advantage of the particle-based representation in motion modeling,

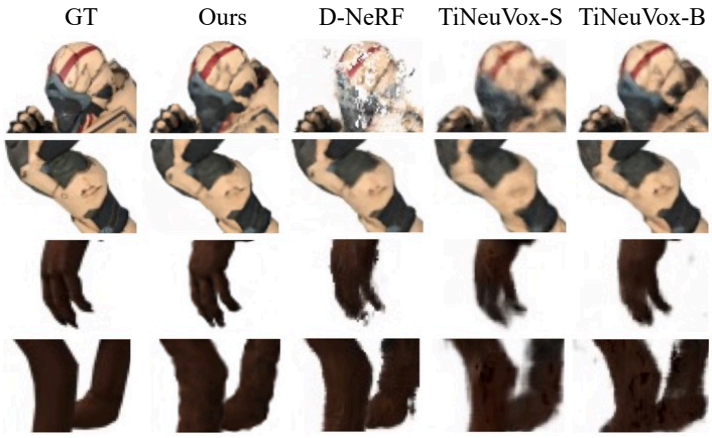

Figure 5.7: Qualitative comparisons on D-NeRF dataset [154].

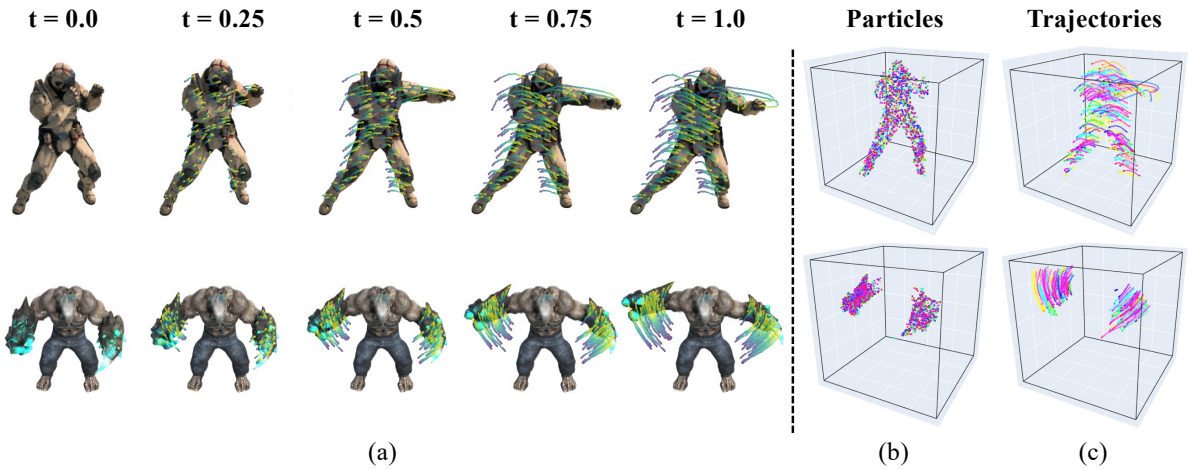

Figure 5.8: Synthesized novel views for D-NeRF dataset [154] and particle-based dynamic models. **(a)** displays the images rendered by the learned DAP-NeRF models, along with the trajectories of 100 sampled particles drawn onto the images. **(b)** shows the particles at time $t = 0$. To enhance visual clarity, we only render 2k randomly sampled particles. **(c)** shows the corresponding learned trajectories of particles, where only 200 randomly sampled particles are rendered.

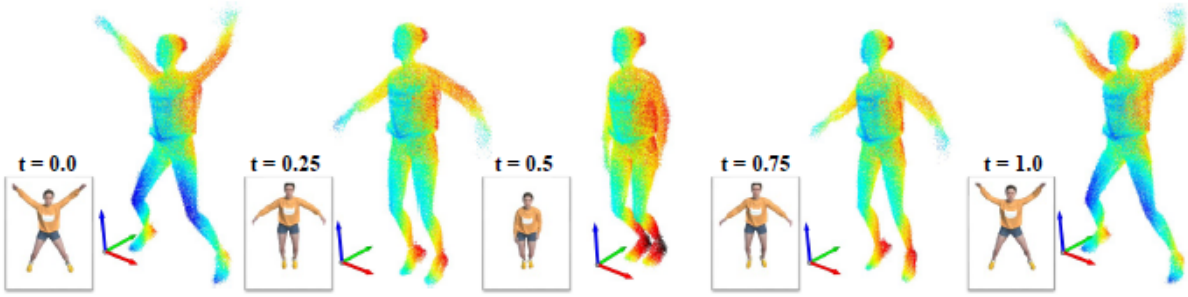

Figure 5.9: Particle positions learned by DAP-NeRF at different frames. The color of each point is determined by its y-axis value.

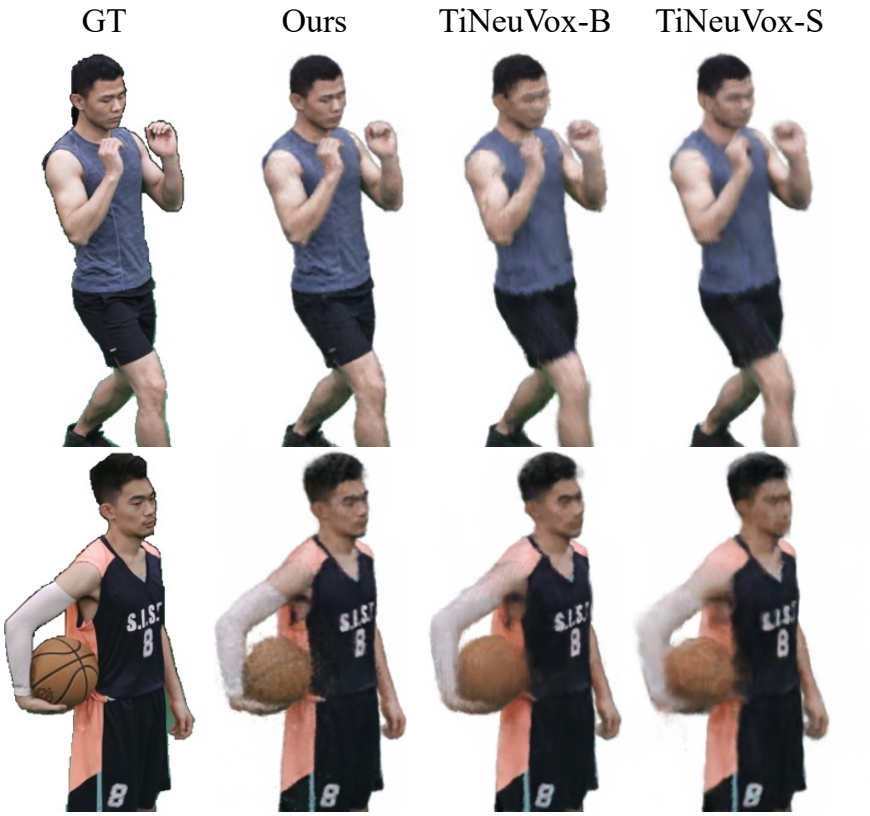

Figure 5.10: Qualitative comparisons on NHR dataset [213].

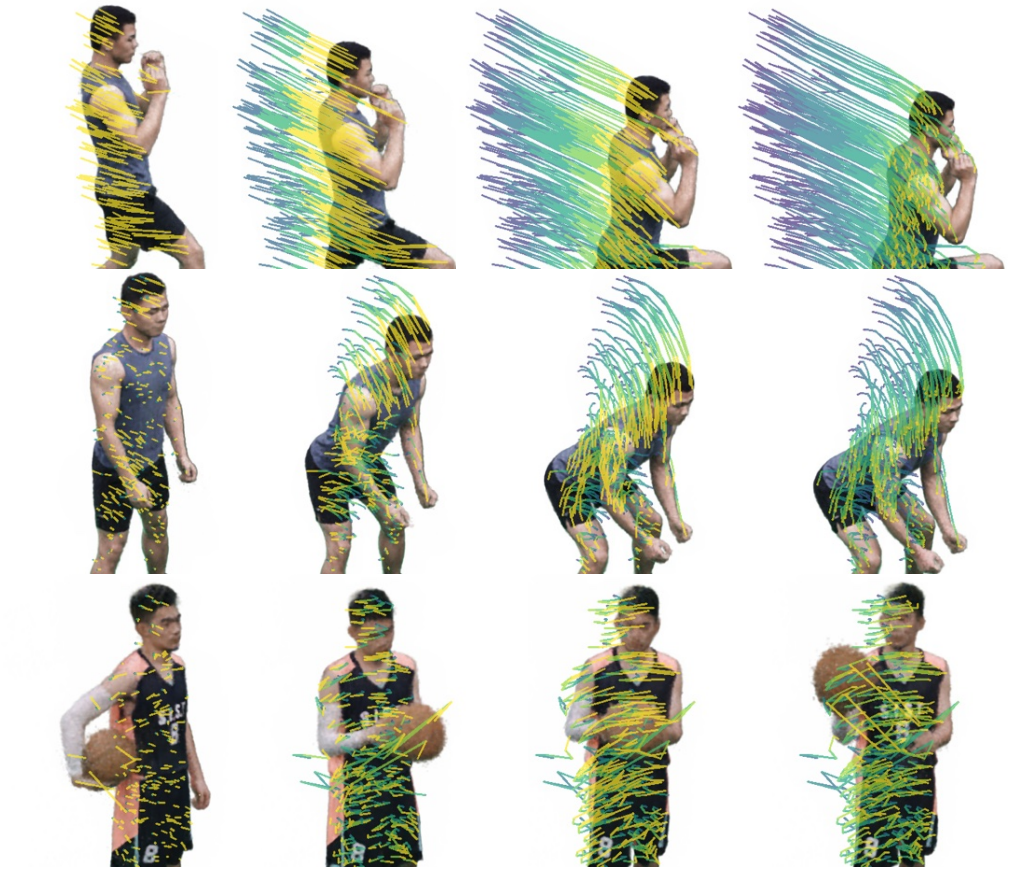

Figure 5.11: This figure displays the novel views rendered by the learned DAP-NeRF models, along with the trajectories of 200 randomly sampled particles.

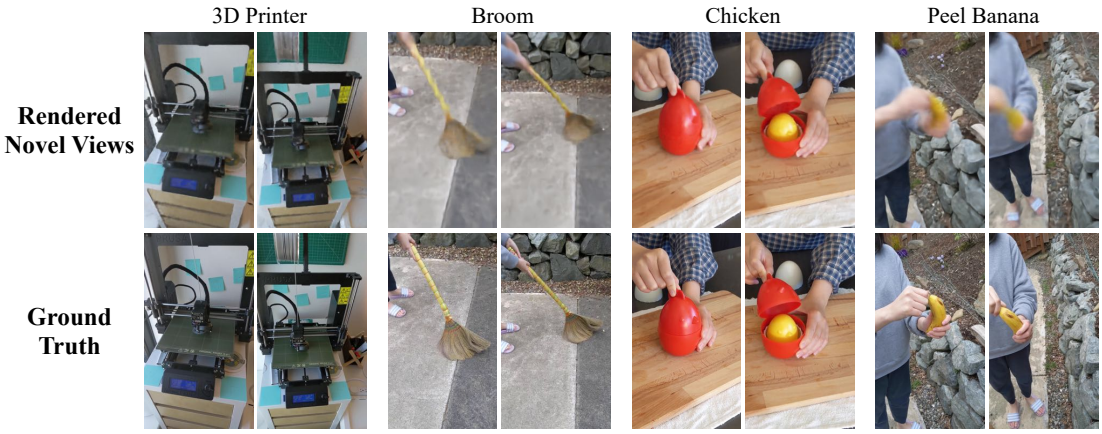

Figure 5.12: Novel views rendered by DAP-NeRF on the HyperNeRF dataset [147]. For each scene, we render images from two perspectives at various times.

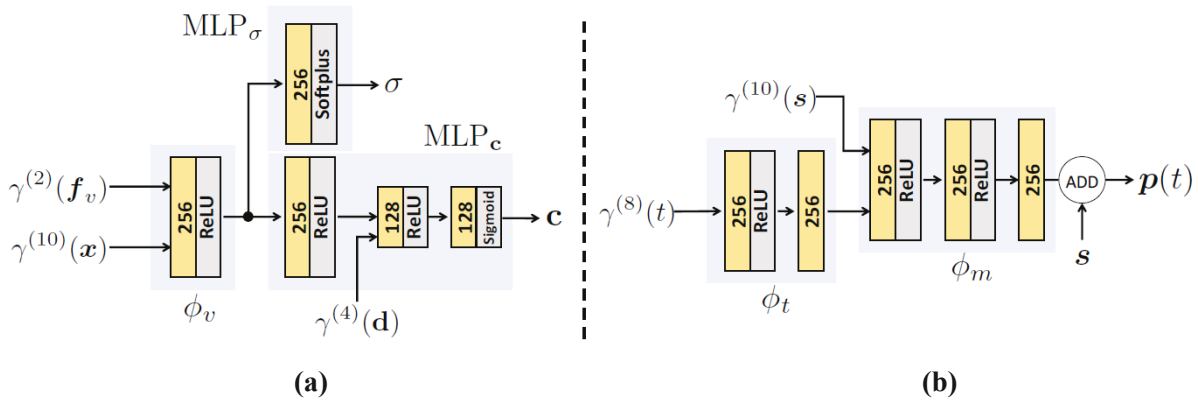

Figure 5.13: Network Architectures. **(a)** shows the implementation of Fig. 5.2, which is the NeRF architecture used in our method. **(b)** details Fig. 5.4, which is the motion model for time-varying position of a particle. In this figure, ϕ_t, ϕ_m, ϕ_v, MLP$_\sigma$ and MLP$_\mathbf{c}$ are neural networks. The superscript of $\gamma^{(\cdot)}$ is the frequency number of positional encoding. Yellow boxes denote linear layers, grey boxes denote activation functions.

as well as related applications. This section also explores ablation studies on particle quantity effects.

5.4.1 Experimental Settings

Datasets. We evaluate our method using four datasets: one synthetic dataset, two real-scene datasets, and our specifically constructed testing scenarios. The *D-NeRF Dataset* [154] includes eight 360° synthetic Scenes. For a fair comparison, we follow the settings of [45] where images are trained and rendered at a resolution of 400 × 400. For real-scene evaluation, the *NHR Dataset* [213] offers four scenes with a single human performing a specific action in each, captured using a multi-camera dome system. We conduct evaluations on 100 frames from each scene, allocating 90% of the camera views for training and the remaining 10% for testing. Another dataset for real scenes is provided by *HyperNeRF* [147], which employs two phones to capture four real unbounded scenes.

Additionally, to comprehensively evaluate models in motion modeling, we've designed three test scenes and built a dataset of 300 images per scene. The dataset also includes data

for ground-truth material motion at specific locations and times. We will provide further details in Sec. 5.4.5.

Model and Optimization Configuration. We implement our framework using PyTorch [149]. Following [45], the resolution of static feature grid $\mathbf{G}^S$ and dynamic feature grid $\mathbf{G}^D$ are set to 160^3. We utilize 200k particles for synthetic datasets, and 500k for real-scene datasets. The dimensions C of both particle and voxel features are set to 12. Details of the network architecture are shown in Fig. 5.13, where the network width is 256, the frequency number of positional encoding is set to 10 for (x, y, z), 4 for view direction $\boldsymbol{d}$, and 8 for time t. Particle removal and re-sampling are performed every 2k training steps. During removal, we set the thresholds for alpha value and trajectory length to $\epsilon_\alpha = 0.0001$ and $\epsilon_{\text{traj}} = 0.1$, respectively. The small radius of the range for random re-sampling is set to 0.1 times the voxel length of the used feature grids.

For the optimization process, we employ the Adam optimizer [93], configured with beta values of $(0.9, 0.99)$ and a minibatch size of $4,096$ rays. The learning rates are set to 0.005 for particle features and 0.001 for motion predictor network ϕ_m. Other learning rates for static feature grid $\mathbf{G}^S$, radiance network $\text{MLP}_\sigma(\cdot)$ and $\text{MLP}_{\boldsymbol{c}}(\cdot)$ are following [45]. The total number of training iterations is 60k. All the learning rates will finally decay by 0.1 with an exponential schedule. The weights for the loss items in (5.17) are set to $w_1 = 0.01$, $w_2 = 0.001$, $w_3 = 0.01$, and $w_4 = 0.01$.

5.4.2 Novel View Synthesis

We begin by comparing ParticleNeRF [1] with our proposed DAP-NeRF on the monocular D-NeRF dataset, since both methods employ a particle-based representation but with fundamentally different implementations. Fig. 5.6 illustrates that our method successfully adapts to challenging monocular video data, while ParticleNeRF struggles with this setup.

Table 5.1: Per-scene quantitative comparisons on D-NeRF dataset [154].

Method	PSNR ↑								
	Hell Warrior	Mutant	Hook	Boucing Balls	Lego	T-Rex	Stand Up	Jumping Jacks	Mean
T-NeRF [154]	23.19	30.56	27.21	37.81	23.82	30.19	31.24	32.01	29.50
D-NeRF [154]	25.02	31.29	29.25	38.93	21.64	31.75	32.79	32.80	30.43
NDVG [67]	25.53	35.53	29.80	34.58	25.23	30.15	34.05	29.45	30.54
TiNeuVox [45]	28.17	33.61	31.45	40.73	25.02	32.70	35.43	34.23	32.67
K-Planes [49]	24.81	32.59	28.13	40.33	25.27	30.75	33.17	31.64	30.84
DAP-NeRF (ours)	**29.51**	**35.75**	**32.69**	**41.29**	**25.43**	**34.07**	**37.86**	**35.90**	**34.06**
Method	**SSIM ↑**								
	Hell Warrior	Mutant	Hook	Boucing Balls	Lego	T-Rex	Stand Up	Jumping Jacks	Mean
T-NeRF [154]	0.93	0.96	0.94	0.98	0.90	0.96	0.97	0.97	0.951
D-NeRF [154]	0.95	0.97	0.96	0.98	0.83	0.97	0.98	0.98	0.953
NDVG [67]	0.95	**0.99**	0.97	0.97	0.93	0.97	0.98	0.96	0.965
TiNeuVox [45]	**0.97**	0.98	0.97	**0.99**	0.92	**0.98**	**0.99**	0.98	0.973
K-Planes [49]	0.95	0.97	0.95	**0.99**	0.94	0.97	0.98	0.97	0.965
DAP-NeRF (ours)	**0.97**	**0.99**	**0.98**	**0.99**	**0.94**	**0.98**	**0.99**	**0.99**	**0.979**

Table 5.2: Per-scene quantitative comparisons on NHR dataset [213].

Method	Sport 1		Sport 2		Sport 3		Basketball		Mean	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
TiNeuVox-S [45]	27.82	0.972	27.84	0.975	27.45	0.968	25.46	0.953	27.14	0.967
TiNeuVox-B [45]	28.07	0.974	28.22	0.978	28.29	0.974	26.31	0.962	27.72	0.972
DAP-NeRF (ours)	**28.58**	**0.978**	**28.78**	**0.981**	**28.50**	**0.976**	**26.41**	**0.964**	**28.07**	**0.975**

Table 5.3: Per-scene quantitative comparisons on HyperNeRF dataset [147]. **Red** text indicates the best result while **blue** text indicate the second best.

Method	3D Printer		Broom		Chicken		Peel Banana		Mean	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
NeRF [128]	20.7	0.780	19.9	0.653	19.9	0.777	20.0	0.769	20.1	0.745
Nerfies [146]	20.6	0.830	19.2	0.567	26.7	0.943	22.4	0.872	22.2	0.803
HyperNeRF [147]	20.0	0.821	19.3	0.591	26.9	**0.948**	**23.3**	**0.896**	22.4	0.814
NDVG [67]	22.4	0.839	**21.5**	**0.703**	27.1	0.939	22.8	0.828	23.3	0.823
TiNeuVox-B [45]	**22.8**	**0.841**	**21.5**	0.686	**28.3**	**0.947**	**24.4**	**0.873**	**24.3**	**0.837**
DAP-NeRF (ours)	**22.9**	**0.845**	**21.9**	**0.713**	**27.6**	0.939	22.5	0.811	**23.7**	**0.827**

This highlights the superiority of our technical design in terms of both performance and application potential.

To further validate the effectiveness of our method, we conduct experiments on the novel view synthesis task and compare DAP-NeRF with state-of-the-art methods. Tab. 5.1 presents the quantitative results on the *D-NeRF Dataset*, evaluated based on peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [207]. Fig. 5.7 provides qualitative comparisons, showcasing image details rendered by different methods.

Results demonstrate that DAP-NeRF matches or surpasses the performance of state-of-the-art models, even though our particle-based representation is primarily designed for explicit motion modeling rather than solely focusing on appearance accuracy. Fig. 5.8 (a) (b) and (c) illustrate the learned particles and their trajectories, which effectively capture and track the dynamic elements within the scenes. Furthermore, Fig. 5.9 offers a visual inspection of the scene geometry discovered by particles at different moments.

For the real scenes in *NHR Dataset*, our method also achieves superior performance in both quantitative comparisons (see Tab. 5.2) and qualitative comparisons (see Fig. 5.10). In Fig. 5.11, we visualize the rendered results as well as the particle trajectories, which visually demonstrate high quality and also reveal the potential applications such as human motion capture. In the *HyperNeRF Dataset*, our method achieves competitive performance (see Tab. 5.3 and Fig. 5.12), although the improvements are not as significant as in the previous two datasets. These results can largely be attributed to the dataset's sparse, forward-facing camera setup, which limits the geometric constraints essential for explicit motion modeling. Note that the core novelty of our method lies in the design of a particle-based representation, primarily aimed at enhancing motion modeling rather than solely improving rendering quality. In the subsequent sections, we will demonstrate our method's unique advantages in faster rendering (Sec. 5.4.4), explicit motion modeling (Sec. 5.4.5), and its potential practical applications (Sec. 5.4.7).

5.4.3 User Study

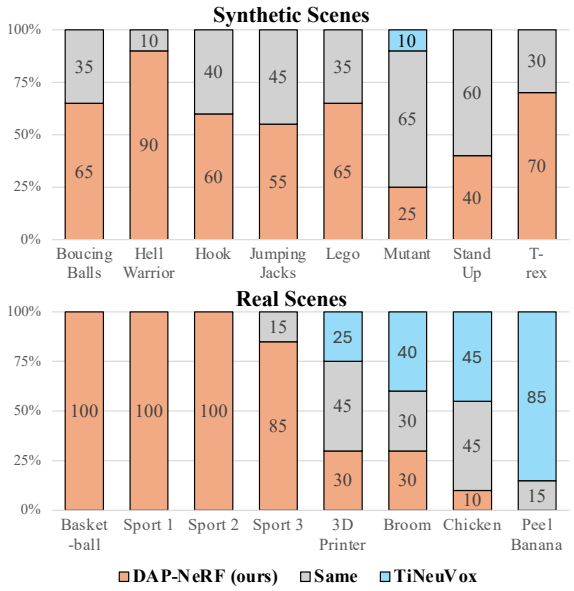

Figure 5.14: User study on rendering quality. This boxplot records the ratio of user preferences between DAP-NeRF (ours) and TiNeuVox [45]. Participants could also select 'Same' to indicate no significant difference between the methods.

To further evaluate reconstruction quality from a human perspective, we conducted a user study comparing our model's rendering performance to TiNeuVox [45], which is closest to ours in terms of PSNR and SSIM quantitative results. For each of the 16 scenes included in the D-NeRF, NHR, and HyperNeRF datasets, participants were presented with 4 random images rendered by both our method and TiNeuVox, alongside the corresponding ground-truth images. They were asked to judge which rendering was closer to the ground truth in terms of detail and overall visual accuracy, or if both renderings were of comparable quality.

We collected questionnaires from 20 participants and compiled the results into a boxplot, as illustrated in Fig. 5.14. This user study yielded results consistent with the qualitative evaluations, further confirming the effectiveness of our method.

5.4.4 Evaluation of Efficiency

Table 5.4: Comparisons in terms of training and rendering times, and parameter counts on the D-NeRF Dataset [154].

Method	Train	Render (one frame)	#.Params
D-NeRF [154]	27.5 hours	8.70s	**1M**
NDVG [67]	23 mins	0.43s	87M
TiNeuVox [45]	28 mins	0.31s	25M
K-Planes [49]	49 mins	0.64s	37M
DAP-NeRF (ours)	**17 mins**	**0.22s**	52M

We compare our model with existing methods in terms of training and rendering times, as well as the total number of parameters. For a fair comparison, we evaluate all methods on the same device (RTX 3090 GPU) and report the average metrics across all scenes of the D-NeRF dataset [154].

As shown in Tab. 5.4, our method achieves improved efficiency in both training and rendering times. Unlike methods that deform all ray-casting points (exceeding 30M points) to the canonical space, our method models the dynamics of only 0.2M particles, resulting in reduced computation. On the other hand, our model requires a reasonable increase in parameters to store particle features.

5.4.5 Evaluation of Motion modeling

Table 5.5: Motion Field Error (MFE) for evaluating motion modeling. The definition of MFE can be found in (5.18).

Method	Scene 1	Scene 2	Scene 3
TiNeuVox [45]	0.0197	0.0363	0.0376
DAP-NeRF (ours)	**0.0029**	**0.0104**	**0.0136**

To further evaluate how well our models capture the physically relevant dynamics of a scene, we developed a dataset with explicit object motion information. As shown in Fig.

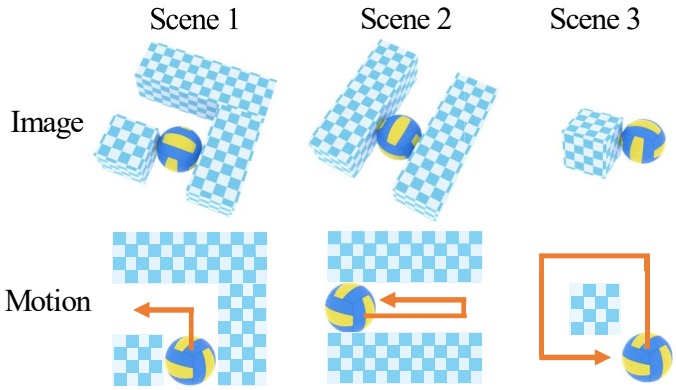

Figure 5.15: The dataset for evaluating motion modeling.

5.15, each scene contains a moving object of interest (a ball), demonstrating varying levels of motion complexity.

Since the scene and object trajectories are constructed in a 3D design software [8], the ground-truth motions are known. Specifically, for any 3D coordinate (x, y, z) at time t, we can determine if it's occupied and, if so, consider the occupying object's velocity as the velocity at (x, y, z, t). With the ground-truth velocity, we design a metric to quantitatively evaluate the motion modeling. Formally, *Motion Field Error* (MFE) is defined as the average Euclidean-norm difference between velocities:

$$\begin{aligned} \mathrm{MFE}(\vec{F}_1, \vec{F}_2) &:= \frac{1}{\mathcal{V}} \int_{\mathcal{V}} \|\vec{F}_1(\boldsymbol{x}) - \vec{F}_2(\boldsymbol{x}))\|_2 \, dv \\ &\approx \frac{1}{N} \sum_{n}^{N} \|\vec{F}_1(\boldsymbol{x}_n) - \vec{F}_2(\boldsymbol{x}_n))\|_2 \end{aligned} \tag{5.18}$$

where $\vec{F}_1, \vec{F}_2$ are the velocity fields to be compared, $\mathcal{V}$ is the volume of the region of interest. The integration (5.18) is approximated by traversing the N voxels, where $\boldsymbol{x}_n$ represents the center of a voxel. i) For existing deformable dynamic NeRF models, e.g. TiNeuVox [45], the velocity at a voxel is approximately implied the deformation field

$$\vec{F}_{\mathrm{df}}(\boldsymbol{x}) \approx \frac{\mathrm{df}(\boldsymbol{x}, t) - \mathrm{df}(\boldsymbol{x}, t + \delta t)}{\delta t} \tag{5.19}$$

where df stands for the deformation field component of a deformable dynamic NeRF. Given a time t, df specifies a 3D offset from a 3D location to a location in the canonical 3D model

df : $\mathbb{R}^3 \times \mathbb{R}^+ \mapsto \mathbb{R}^3$. Note that we need to zero-out the velocities in empty areas where deformable NeRFs are likely to produce incorrect values, as discussed in Sec. 5.4.6. ii) In DAP-NeRF, the velocity field can be computed directly using the particle motions. Specifically, we computing the average velocity of particles that affect the voxel region at time t. And the velocity of a particle is $\boldsymbol{v}_p = \frac{\boldsymbol{p}_{t+\delta t} - \boldsymbol{p}_t}{\delta t}$.

We compare the our dynamic model with TiNeuVox [45], a representative method for deformation fields. Specifically, we calculate the mean MFE metric between the motion fields estimated by each model and the corresponding ground truth for sampled time steps $t = [0.1, 0.3, 0.5, 0.7, 0.9]$. To compute (5.18), we employ $N = 30^3$ voxels evenly distributed within the cubic bounding box of the tested scene. The small time increment δt used for velocity approximation is set to 0.01. As demonstrated in Tab. 5.5, particle-based dynamic model in DAP-NeRF effectively captures motion with lower error in dynamic scenes.

5.4.6 Method Analysis

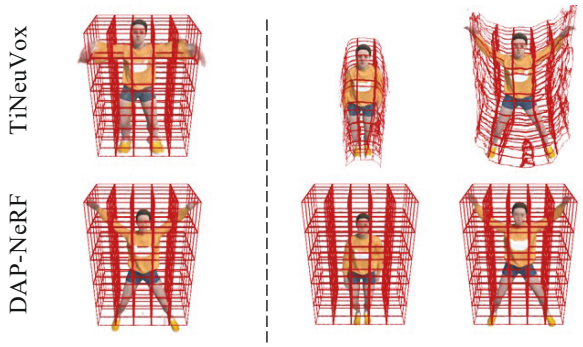

Figure 5.16: Comparison in dealing with dynamics. We add artificial grids (in red) aligned with the coord-axes at the canonical/first frame for TiNeuVox and our method.

Decoupling of Scene Components. To illustrate how different models manage dynamics, we introduce artificial grid lines parallel to the axes in the canonical (first) frame, as shown

Figure 5.17: Decomposition of scene components. The 'full' image is rendered using the final superpositional feature field. The other two images are rendered using only static and dynamic field respectively.

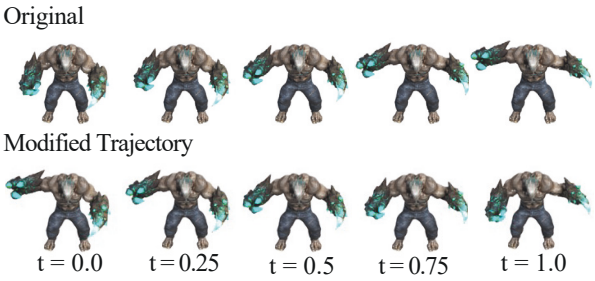

Figure 5.18: Application on editing dynamic scene. We reverse the original trajectories of particles that are responsible for the shape's right hand.

in Fig. 5.16 (a). Note that the grid lines are solely for the visualization of modeled motions; the models are NOT aware of the added lines during training.

The rendering results are shown in Fig. 5.16 (b), which demonstrate that the deforming-based method causes continuous deformation across all space, including empty areas (e.g., air). In comparison, our method decouples the actural moving objects from the static components and only models the motion for objects. Fig. 5.17 shows that a trained DAP-NeRF enables a direct and high-quality decomposition. Fig. 5.18 illustrates how this feature can be helpful in practical applications, such as scene editing.

Table 5.6: Ablation study of particle quantity.

Particle Number	**PSNR↑**	**SSIM↑**
200k	**34.06**	**0.979**
100k	33.77	0.979
50k	33.22	0.974
10k	32.52	0.970

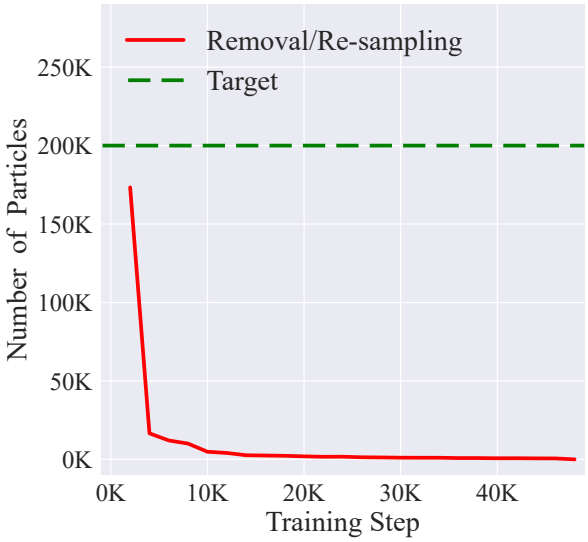

Figure 5.19: This figure demonstrates the effect of particle removal and re-sampling scheme.

Ablation study of Particle Quantity. We examine the impact of particle quantity on performance, employing synthetic scenes from [154] for evaluation. Averaged metrics across all scenes are displayed in Tab. 5.6, which illustrate that the model maintains robust performance even with reduced particle amount.

(a) without $\mathcal{L}_{\text{tvm}}$ (b) with $\mathcal{L}_{\text{tvm}}$

Figure 5.20: Impact of Motion Regularization Term $\mathcal{L}_{\text{tvm}}$. To enhance visual clarity, only 500 particles randomly sampled from around the hand area are rendered.

Ablation study of Loss Components. We examine the impact of four auxiliary loss components used in our model, reporting averaged metrics across all scenes from the D-NeRF

Table 5.7: Ablation studies of Loss Components. We conduct evaluations on the D-NeRF dataset [154] and report the average PSNR across all scenes.

$\mathscr{L}_{\text{ptrgb}}$	$\mathscr{L}_{\text{bg}}$	$\mathscr{L}_{\text{tvf}}$	$\mathscr{L}_{\text{tvm}}$	PSNR↑
x	x	x	x	33.51
x	x	x	✓	33.60
x	x	✓	x	33.57
x	x	✓	✓	33.62
x	✓	x	x	33.72
x	✓	x	✓	33.79
x	✓	✓	x	33.77
x	✓	✓	✓	33.91
✓	x	x	x	33.58
✓	x	x	✓	33.69
✓	x	✓	x	33.67
✓	x	✓	✓	33.76
✓	✓	x	x	33.79
✓	✓	x	✓	33.83
✓	✓	✓	x	33.98
✓	✓	✓	✓	**34.06**

Dataset [154]. Tab. 5.7 shows that each component contributes to improved rendering quality.

Specifically, the two regularization terms $\mathscr{L}_{\text{ptrgb}}$ and $\mathscr{L}_{\text{bg}}$ help the radiance field better distinguish between scene objects and empty space. The total variation loss $\mathscr{L}_{\text{tvf}}$ enhances the continuity of the feature field. The combination of these terms effectively prevents overfitting of the model, especially when dealing with monocular videos that have limited multi-view constraints. Additionally, the motion regularization term $\mathscr{L}_{\text{tvm}}$ encourages particles to preserve local rigidity, leading to more reasonable trajectories and better reconstruction quality. To further validate its effectiveness, we visualized particle trajectories learned with and without $\mathscr{L}_{\text{tvm}}$. Fig. 5.20 reveals that this motion regularization can effectively reduce artifacts, such as spurious particle trajectories that inaccurately model empty areas.

Quantitative Analysis of Removal and Re-sampling. Fig. 5.19 quantitatively demonstrates the impact of alternating removal and re-sampling processes during training. The number of particles re-sampled, which equals the number removed, gradually decreases to zero as the model converges (e.g., after 10K steps for D-NeRF dataset).

Selection of Loss Component Weights. The selection of the weights for the four auxiliary losses in (5.17) is empirical. Specifically, the weights w_1 and w_2, corresponding to $\mathscr{L}_{\text{ptrgb}}$ and $\mathscr{L}_{\text{bg}}$, follow the common settings used in [45, 185]. The other two weights were selected based on hyperparameter tuning, as shown in Tab. 5.8, where $w_3 = w_4 = 0.01$ performed slightly better than smaller values. Additionally, using larger weights would hinder the model's learning.

Table 5.8: Tuning of weights for $\mathscr{L}_{\text{tvf}}$ and $\mathscr{L}_{\text{tvm}}$.

weight for $\mathscr{L}_{\text{tvf}}$	weight for $\mathscr{L}_{\text{tvm}}$	PSNR↑
0.1	0.1	32.35
0.1	0.01	32.77
0.1	0.001	32.73
0.01	0.1	33.04
0.01	0.01	**34.06**
0.01	0.001	34.01
0.001	0.1	32.89
0.001	0.01	33.92
0.001	0.001	33.84

5.4.7 Practical Application

In this section, we provide a simple example to present the potential of our models for practical applications. We first train a dynamic NeRF for the scene where a ball slowly falls. When placing a wall in the middle of the ball's path, we can easily detect collision since our model makes an explicitly interpretable and physically meaningful description (i.e., particles) of object motions. As shown in Fig. 5.21, when some particles touch the wall, we

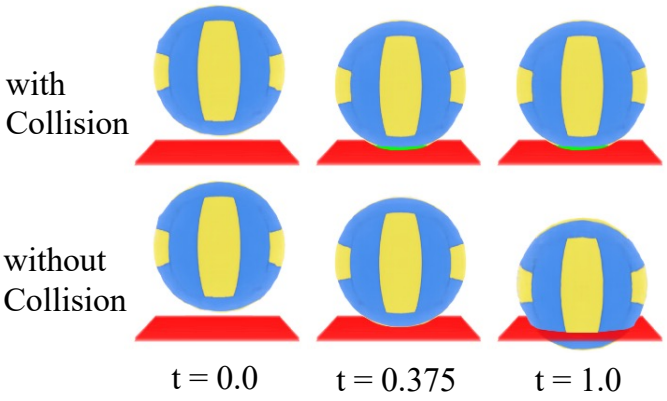

Figure 5.21: Visualization of collision detection. The red plane represents a virtual wall, while the collided surfaces of the ball are marked in green.

can stop the moving object and then mark the collided surfaces of the object by changing the color of the radiance field region affected by those collided particles.

5.5 Summary

We have presented the Dynamic Appearance Particle Neural Radiance Field (DAP-NeRF), a novel framework that introduces Lagrangian particles to construct a superpositional radiance field. The proposed appearance particles can not only carry local light radiance information but also capture object motions in an explicitly interpretable and physically meaningful manner. DAP-NeRF is effective and efficient, requiring only monocular video photometric supervision. We have demonstrated that DAP-NeRF performs well in conventional novel view synthesis and excels in motion modeling tasks. One potential future work is the applications on scenes containing fluids.

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

This thesis aims to advance 3D scene reconstruction by improving explicit geometric modeling, narrowing the gap between advanced reconstruction models and practical applications. Through extensive study of 3D reconstruction techniques, we examined both explicit and implicit approaches, identifying common limitations arising from the lack of explicit geometric modeling in existing methods. To address these challenges, we proposed three contributions that enhance reconstruction accuracy, efficiency, and interpretability.

First, we improve 3D reconstruction from LiDAR scans through surface normal estimation with multi-modal sensor fusion. We introduced the Hybrid Geometric Transformer (HGT), a transformer-based neural network that fuses visual semantic and geometric information while capturing long-range dependencies. Experimental results demonstrate its effectiveness in both synthetic and real-world datasets, with transferable geometric knowledge learned from a simulated traffic environment and successfully applied to the KITTI dataset.

Second, we introduce a hybrid representation that integrates 3D Gaussian Splatting (3DGS) with a mesh, improving the structured organization of Gaussians and enabling high-fidelity surface modeling. The proposed method jointly learns the mesh and appearance in an end-to-end manner, where Gaussians are bound to mesh faces and optimized using differentiable rendering with photometric supervision. This formulation facilitates both high-quality rendering and geometric controllability. Experimental results verify its state-of-the-art performance, demonstrating that our hybrid approach effectively bridges the gap between explicit and implicit representations.

Third, we enable explicit geometric modeling within an implicit reconstruction framework. We propose Dynamic Appearance Particle Neural Radiance Field (DAP-NeRF), which introduces appearance particles as explicit geometric entities within the radiance field. These particles encode local radiance information while capturing object motions in a physically meaningful manner. DAP-NeRF is efficiently learned from monocular video supervision, achieving strong performance in novel view synthesis and excelling in motion modeling tasks.

6.2 Future Directions

Looking ahead, several promising directions can extend the work presented in this thesis:

1. **Scalability and Generalization**: While significant progress has been made in handling dynamic scenes and complex geometries, scaling these solutions to accommodate larger datasets and more diverse environments remains an open challenge. Future work could explore improving the generalization capability of the proposed methods, particularly in real-world, unseen scenarios.

2. **Multi-Sensor Fusion for Robust 3D Reconstruction**: Current approaches rely primarily on LiDAR and RGB cameras, which may struggle in adverse conditions such

as fog, low light, or occlusion. Exploring fusion strategies with event cameras, thermal imaging, and radar could enhance reconstruction robustness and temporal consistency, particularly for dynamic environments.

3. **Neural Rendering with Explicit-Implicit Hybrid Representations**: Hybrid approaches combining 3D Gaussian Splatting, meshes, and implicit neural fields offer advantages in both accuracy and efficiency. Future research could explore their application in real-time rendering, dynamic scene synthesis, and interactive editing, particularly in augmented reality (AR), virtual reality (VR), and digital twin applications.

4. **Efficient Representations and Compression**: Reducing the density of primitives while preserving model fidelity remains crucial for optimizing storage and computational efficiency. Future research could focus on leveraging geometric priors and structural regularities to develop more compact yet expressive scene representations, particularly for deployment in resource-constrained environments.

5. **Real-Time 3D Reconstruction on Edge Devices**: Advancing real-time 3D reconstruction for edge computing applications, such as field robotics and autonomous systems, requires improvements in computational efficiency. Optimizing model architectures and leveraging hardware acceleration could enable faster inference while maintaining high reconstruction quality.

6. **Closed-Loop Simulation for Perception and Planning**: Developing geometrically meaningful closed-loop simulation frameworks can enhance end-to-end perception and planning systems. Future research could focus on integrating learned representations into physically accurate simulations, improving the reliability of models deployed in robotics, autonomous driving, and augmented reality.

BIBLIOGRAPHY

[1] J. ABOU-CHAKRA, F. DAYOUB, AND N. SÜNDERHAUF, *Particlenerf: A particle-based encoding for online neural radiance fields*, in IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2024, Waikoloa, HI, USA, January 3-8, 2024, IEEE, 2024, pp. 5963–5972.

[2] A. BANSAL, B. C. RUSSELL, AND A. GUPTA, *Marr revisited: 2d-3d alignment via surface normal prediction*, in CVPR, IEEE Computer Society, 2016, pp. 5965–5974.

[3] J. T. BARRON, B. MILDENHALL, M. TANCIK, P. HEDMAN, R. MARTIN-BRUALLA, AND P. P. SRINIVASAN, *Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 5835–5844.

[4] J. T. BARRON, B. MILDENHALL, D. VERBIN, P. P. SRINIVASAN, AND P. HEDMAN, *Mip-nerf 360: Unbounded anti-aliased neural radiance fields*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 5460–5469.

[5] Y. BEN-SHABAT AND S. GOULD, *Deepfit: 3d surface fitting via neural network weighted least squares*, in ECCV, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds., vol. 12346 of Lecture Notes in Computer Science, Springer, 2020, pp. 20–34.

[6] Y. BEN-SHABAT, M. LINDENBAUM, AND A. FISCHER, *Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks*, in CVPR, Computer Vision Foundation / IEEE, 2019, pp. 10112–10120.

[7] F. BERNARDINI, J. MITTLEMAN, H. E. RUSHMEIER, C. T. SILVA, AND G. TAUBIN, *The ball-pivoting algorithm for surface reconstruction*, IEEE Trans. Vis. Comput. Graph., 5 (1999), pp. 349–359.

[8] BLENDER-COMMUNITY, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[9] A. CAO AND R. DE CHARETTE, *Monoscene: Monocular 3d semantic scene completion*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 3981–3991.

[10] A. CAO AND J. JOHNSON, *Hexplane: A fast representation for dynamic scenes*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 130–141.

[11] Y. CAO, Z. LIU, Z. KUANG, L. KOBBELT, AND S. HU, *Learning to reconstruct high-quality 3d shapes with cascaded fully convolutional networks*, in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IX, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., vol. 11213 of Lecture Notes in Computer Science, Springer, 2018, pp. 626–643.

[12] A. X. CHANG, T. FUNKHOUSER, L. GUIBAS, P. HANRAHAN, Q. HUANG, Z. LI, S. SAVARESE, M. SAVVA, S. SONG, H. SU, J. XIAO, L. YI, AND F. YU, *ShapeNet: An Information-Rich 3D Model Repository*, Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[13] A. CHEN, Z. XU, A. GEIGER, J. YU, AND H. SU, *Tensorf: Tensorial radiance fields*, in Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXII, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds., vol. 13692 of Lecture Notes in Computer Science, Springer, 2022, pp. 333–350.

[14] A. CHEN, Z. XU, F. ZHAO, X. ZHANG, F. XIANG, J. YU, AND H. SU, *Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 14104–14113.

[15] H. CHEN, C. LI, AND G. H. LEE, *Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance*, CoRR, abs/2312.00846 (2023).

[16] H. CHEN, E. TRETSCHK, T. STUYCK, P. KADLECEK, L. KAVAN, E. VOUGA, AND C. LASSNER, *Virtual elastic objects*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 15806–15816.

[17] X. CHEN, J. SUN, Y. XIE, H. BAO, AND X. ZHOU, *Neuralrecon: Real-time coherent 3d scene reconstruction from monocular video*, IEEE Trans. Pattern Anal. Mach. Intell., 46 (2024), pp. 7542–7555.

[18] Y. CHEN AND X. WANG, *Transformers as meta-learners for implicit neural representations*, in Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds., vol. 13677 of Lecture Notes in Computer Science, Springer, 2022, pp. 170–187.

[19] Z. CHEN, T. A. FUNKHOUSER, P. HEDMAN, AND A. TAGLIASACCHI, *Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on*

mobile architectures, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 16569–16578.

[20] Z. CHEN AND H. ZHANG, *Learning implicit fields for generative shape modeling*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, Computer Vision Foundation / IEEE, 2019, pp. 5939–5948.

[21] A. CHICA, J. WILLIAMS, C. ANDÚJAR, P. BRUNET, I. NAVAZO, J. ROSSIGNAC, AND À. VINACUA, *Pressing: Smooth isosurfaces with flats from binary grids*, Comput. Graph. Forum, 27 (2008), pp. 36–46.

[22] N. CHODOSH, C. WANG, AND S. LUCEY, *Deep convolutional compressed sensing for lidar depth completion*, in Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part I, C. V. Jawahar, H. Li, G. Mori, and K. Schindler, eds., vol. 11361 of Lecture Notes in Computer Science, Springer, 2018, pp. 499–513.

[23] C. B. CHOY, D. XU, J. GWAK, K. CHEN, AND S. SAVARESE, *3d-r2n2: A unified approach for single and multi-view 3d object reconstruction*, in Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., vol. 9912 of Lecture Notes in Computer Science, Springer, 2016, pp. 628–644.

[24] D. COHEN-STEINER AND F. DA, *A greedy delaunay-based surface reconstruction algorithm*, Vis. Comput., 20 (2004), pp. 4–16.

[25] D. J. CRANDALL, A. OWENS, N. SNAVELY, AND D. P. HUTTENLOCHER, *Sfm with mrfs: Discrete-continuous optimization for large-scale structure from motion*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 2841–2853.

[26] H. CUI, X. GAO, S. SHEN, AND Z. HU, *Hsfm: Hybrid structure-from-motion*, in CVPR, IEEE Computer Society, 2017, pp. 2393–2402.

[27] B. CURLESS AND M. LEVOY, *A volumetric method for building complex models from range images*, in Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1996, New Orleans, LA, USA, August 4-9, 1996, J. Fujii, ed., ACM, 1996, pp. 303–312.

[28] P. DAI, J. XU, W. XIE, X. LIU, H. WANG, AND W. XU, *High-quality surface reconstruction using gaussian surfels*, in ACM SIGGRAPH 2024 Conference Papers, SIGGRAPH 2024, Denver, CO, USA, 27 July 2024- 1 August 2024, A. Burbano, D. Zorin, and W. Jarosz, eds., ACM, 2024, p. 22.

[29] K. DENG, A. LIU, J. ZHU, AND D. RAMANAN, *Depth-supervised nerf: Fewer views and faster training for free*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 12872–12881.

[30] D. DETONE, T. MALISIEWICZ, AND A. RABINOVICH, *Superpoint: Self-supervised interest point detection and description*, in CVPR Workshops, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 224–236.

[31] P. DEUTSCH, *DEFLATE compressed data format specification version 1.3*, RFC, 1951 (1996), pp. 1–17.

[32] A. DOSOVITSKIY, L. BEYER, A. KOLESNIKOV, D. WEISSENBORN, X. ZHAI, T. UNTERTHINER, M. DEHGHANI, M. MINDERER, G. HEIGOLD, S. GELLY, J. USZKOREIT, AND N. HOULSBY, *An image is worth 16x16 words: Transformers for image recognition at scale*, in 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, OpenReview.net, 2021.

[33] S. DURVASULA, A. ZHAO, F. CHEN, R. LIANG, P. K. SANJAYA, AND N. VIJAYKUMAR, *DISTWAR: fast differentiable rendering on raster-based rendering pipelines*, CoRR, abs/2401.05345 (2024).

[34] H. EDELSBRUNNER, D. G. KIRKPATRICK, AND R. SEIDEL, *On the shape of a set of points in the plane*, IEEE Trans. Inf. Theory, 29 (1983), pp. 551–558.

[35] H. EDELSBRUNNER AND N. R. SHAH, *Triangulating topological spaces*, Int. J. Comput. Geom. Appl., 7 (1997), pp. 365–378.

[36] D. EIGEN, C. PUHRSCH, AND R. FERGUS, *Depth map prediction from a single image using a multi-scale deep network*, in NeurIPS, 2014, pp. 2366–2374.

[37] A. ELDESOKEY, M. FELSBERG, K. HOLMQUIST, AND M. PERSSON, *Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end*, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, Computer Vision Foundation / IEEE, 2020, pp. 12011–12020.

[38] A. ELFES, *Using occupancy grids for mobile robot perception and navigation*, Computer, 22 (1989), pp. 46–57.

[39] W. EQUITZ, *A new vector quantization clustering algorithm*, IEEE Trans. Acoust. Speech Signal Process., 37 (1989), pp. 1568–1575.

[40] F. EVRARD, F. DENNER, AND B. G. M. VAN WACHEM, *Surface reconstruction from discrete indicator functions*, IEEE Trans. Vis. Comput. Graph., 25 (2019), pp. 1629–1635.

[41] R. FAN, H. WANG, B. XUE, H. HUANG, Y. WANG, M. LIU, AND I. PITAS, *Three-filters-to-normal: An accurate and ultrafast surface normal estimator*, IEEE Robotics Autom. Lett., 6 (2021), pp. 5405–5412.

[42] Z. FAN, K. WANG, K. WEN, Z. ZHU, D. XU, AND Z. WANG, *Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ FPS*, CoRR, abs/2311.17245 (2023).

[43] S. R. FANELLO, J. P. C. VALENTIN, C. RHEMANN, A. KOWDLE, V. TANKOVICH, P. L. DAVIDSON, AND S. IZADI, *Ultrastereo: Efficient learning-based matching for active stereo systems*, in CVPR, IEEE Computer Society, 2017, pp. 6535–6544.

[44] J. FANG, J. WANG, X. ZHANG, L. XIE, AND Q. TIAN, *Gaussianeditor: Editing 3d gaussians delicately with text instructions*, CoRR, abs/2311.16037 (2023).

[45] J. FANG, T. YI, X. WANG, L. XIE, X. ZHANG, W. LIU, M. NIESSNER, AND Q. TIAN, *Fast dynamic radiance fields with time-aware neural voxels*, in SIGGRAPH Asia 2022 Conference Papers, SA 2022, Daegu, Republic of Korea, December 6-9, 2022, S. K. Jung, J. Lee, and A. W. Bargteil, eds., ACM, 2022, pp. 11:1–11:9.

[46] Y. FENG, X. FENG, Y. SHANG, Y. JIANG, C. YU, Z. ZONG, T. SHAO, H. WU, K. ZHOU, C. JIANG, AND Y. YANG, *Gaussian splashing: Dynamic fluid synthesis with gaussian splatting*, CoRR, abs/2401.15318 (2024).

[47] S. FRIDOVICH-KEIL, G. MEANTI, F. R. WARBURG, B. RECHT, AND A. KANAZAWA, *K-planes: Explicit radiance fields in space, time, and appearance*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 12479–12488.

[48] S. FRIDOVICH-KEIL, G. MEANTI, F. R. WARBURG, B. RECHT, AND A. KANAZAWA, *K-planes: Explicit radiance fields in space, time, and appearance*, in CVPR, 2023.

[49] S. FRIDOVICH-KEIL, G. MEANTI, F. R. WARBURG, B. RECHT, AND A. KANAZAWA, *K-planes: Explicit radiance fields in space, time, and appearance*, in IEEE/CVF Con-

ference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 12479–12488.

[50] S. FRIDOVICH-KEIL, A. YU, M. TANCIK, Q. CHEN, B. RECHT, AND A. KANAZAWA, *Plenoxels: Radiance fields without neural networks*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 5491–5500.

[51] Y. FURUKAWA AND J. PONCE, *Accurate, dense, and robust multi-view stereopsis*, in 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA, IEEE Computer Society, 2007.

[52] W. GAN, H. XU, Y. HUANG, S. CHEN, AND N. YOKOYA, *V4D: voxel for 4d novel view synthesis*, IEEE Trans. Vis. Comput. Graph., 30 (2024), pp. 1579–1591.

[53] C. GAO, A. SARAF, J. KOPF, AND J. HUANG, *Dynamic view synthesis from dynamic monocular video*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 5692–5701.

[54] J. GAO, C. GU, Y. LIN, H. ZHU, X. CAO, L. ZHANG, AND Y. YAO, *Relightable 3d gaussian: Real-time point cloud relighting with BRDF decomposition and ray tracing*, CoRR, abs/2311.16043 (2023).

[55] L. GAO, J. YANG, B. ZHANG, J. SUN, Y. YUAN, H. FU, AND Y. LAI, *Mesh-based gaussian splatting for real-time large-scale deformation*, CoRR, abs/2402.04796 (2024).

[56] L. GAO, J. YANG, B.-T. ZHANG, J.-M. SUN, Y.-J. YUAN, H. FU, AND Y.-K. LAI, *Mesh-based gaussian splatting for real-time large-scale deformation*, CoRR, abs/2402.04796 (2024).

[57] Q. GAO, Q. XU, H. SU, U. NEUMANN, AND Z. XU, *Strivec: Sparse tri-vector radiance fields*, CoRR, abs/2307.13226 (2023).

[58] A. GEIGER, P. LENZ, C. STILLER, AND R. URTASUN, *Vision meets robotics: The KITTI dataset*, Int. J. Robotics Res., 32 (2013), pp. 1231–1237.

[59] R. A. GINGOLD AND J. J. MONAGHAN, *Smoothed particle hydrodynamics: theory and application to non-spherical stars*, Monthly notices of the royal astronomical society, 181 (1977), pp. 375–389.

[60] X. GU, Z. FAN, S. ZHU, Z. DAI, F. TAN, AND P. TAN, *Cascade cost volume for high-resolution multi-view stereo and stereo matching*, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, Computer Vision Foundation / IEEE, 2020, pp. 2492–2501.

[61] X. GU, W. YUAN, Z. DAI, S. ZHU, C. TANG, AND P. TAN, *DRO: deep recurrent optimizer for structure-from-motion*, CoRR, abs/2103.13201 (2021).

[62] A. GUÉDON AND V. LEPETIT, *Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering*, CoRR, abs/2311.12775 (2023).

[63] G. GUENNEBAUD AND M. H. GROSS, *Algebraic point set surfaces*, ACM Trans. Graph., 26 (2007), p. 23.

[64] P. GUERRERO, Y. KLEIMAN, M. OVSJANIKOV, AND N. J. MITRA, *Pcpnet learning local shape properties from raw point clouds*, Comput. Graph. Forum, 37 (2018), pp. 75–85.

[65] H. GUO, J. ZHU, AND Y. CHEN, *E-LOAM: lidar odometry and mapping with expanded local structural information*, IEEE Trans. Intell. Veh., 8 (2023), pp. 1911–1921.

[66] M. GUO, J. CAI, Z. LIU, T. MU, R. R. MARTIN, AND S. HU, *PCT: point cloud transformer*, Comput. Vis. Media, 7 (2021), pp. 187–199.

[67] X. GUO, G. CHEN, Y. DAI, X. YE, J. SUN, X. TAN, AND E. DING, *Neural deformable voxel grid for fast optimization of dynamic view synthesis*, in Computer Vision - ACCV 2022 - 16th Asian Conference on Computer Vision, Macao, China, December 4-8, 2022, Proceedings, Part I, L. Wang, J. Gall, T. Chin, I. Sato, and R. Chellappa, eds., vol. 13841 of Lecture Notes in Computer Science, Springer, 2022, pp. 450–468.

[68] X. GUO, J. SUN, Y. DAI, G. CHEN, X. YE, X. TAN, E. DING, Y. ZHANG, AND J. WANG, *Forward flow for novel view synthesis of dynamic scenes*, in IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, IEEE, 2023, pp. 15976–15987.

[69] J. K. HAAS, *A history of the unity game engine*, N/A, (2014).

[70] R. HARTLEY AND A. ZISSERMAN, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004.

[71] X. HE, J. SUN, Y. WANG, S. PENG, Q. HUANG, H. BAO, AND X. ZHOU, *Detector-free structure from motion*, CoRR, abs/2306.15669 (2023).

[72] S. HOCHREITER AND J. SCHMIDHUBER, *Long short-term memory*, Neural Comput., 9 (1997), pp. 1735–1780.

[73] H. HOPPE, T. DEROSE, T. DUCHAMP, J. A. MCDONALD, AND W. STUETZLE, *Surface reconstruction from unorganized points*, in SIGGRAPH, ACM, 1992, pp. 71–78.

[74] A. HORNUNG, K. M. WURM, M. BENNEWITZ, C. STACHNISS, AND W. BURGARD, *Octomap: an efficient probabilistic 3d mapping framework based on octrees*, Auton. Robots, 34 (2013), pp. 189–206.

[75] B. HUANG, Z. YU, A. CHEN, A. GEIGER, AND S. GAO, *2d gaussian splatting for geometrically accurate radiance fields*, in ACM SIGGRAPH 2024 Conference Papers, SIGGRAPH 2024, Denver, CO, USA, 27 July 2024- 1 August 2024, A. Burbano, D. Zorin, and W. Jarosz, eds., ACM, 2024, p. 32.

[76] J. HUANG, Z. GOJCIC, M. ATZMON, O. LITANY, S. FIDLER, AND F. WILLIAMS, *Neural kernel surface reconstruction*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4369–4379.

[77] J. HUANG, Z. GOJCIC, M. ATZMON, O. LITANY, S. FIDLER, AND F. WILLIAMS, *Neural kernel surface reconstruction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 4369–4379.

[78] J. HUANG, H. SU, AND L. J. GUIBAS, *Robust watertight manifold surface generation method for shapenet models*, CoRR, abs/1802.01698 (2018).

[79] P. HUANG, K. MATZEN, J. KOPF, N. AHUJA, AND J. HUANG, *Deepmvs: Learning multi-view stereopsis*, in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 2821–2830.

[80] Y. HUANG, W. ZHENG, Y. ZHANG, J. ZHOU, AND J. LU, *Tri-perspective view for vision-based 3d semantic occupancy prediction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 9223–9232.

[81] J. HWANG AND M. SUNG, *Occupancy-based dual contouring*, in SIGGRAPH Asia 2024 Conference Papers, SA 2024, Tokyo, Japan, December 3-6, 2024, T. Igarashi, A. Shamir, and H. R. Zhang, eds., ACM, 2024, pp. 129:1–129:11.

[82] F. INCAHUANACO AND A. PAIVA, *Surface reconstruction method for particle-based fluids using discrete indicator functions*, Comput. Graph., 114 (2023), pp. 26–35.

[83] Y. JIANG, C. YU, T. XIE, X. LI, Y. FENG, H. WANG, M. LI, H. Y. K. LAU, F. GAO, Y. YANG, AND C. JIANG, *VR-GS: A physical dynamics-aware interactive gaussian splatting system in virtual reality*, CoRR, abs/2401.16663 (2024).

[84] J. JO, H. KIM, AND J. PARK, *Identifying unnecessary 3d gaussians using clustering for fast rendering of 3d gaussian splatting*, CoRR, abs/2402.13827 (2024).

[85] A. JULIANI, V. BERGES, E. VCKAY, Y. GAO, H. HENRY, M. MATTAR, AND D. LANGE, *Unity: A general platform for intelligent agents*, CoRR, abs/1809.02627 (2018).

[86] H. JUN AND A. NICHOL, *Shap-e: Generating conditional 3d implicit functions*, CoRR, abs/2305.02463 (2023).

[87] H. KATO, Y. USHIKU, AND T. HARADA, *Neural 3d mesh renderer*, in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3907–3916.

[88] M. M. KAZHDAN, M. BOLITHO, AND H. HOPPE, *Poisson surface reconstruction*, in Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, June 26-28, 2006, A. Sheffer and K. Polthier, eds., vol. 256 of ACM International Conference Proceeding Series, Eurographics Association, 2006, pp. 61–70.

[89] M. M. KAZHDAN AND H. HOPPE, *Screened poisson surface reconstruction*, ACM Trans. Graph., 32 (2013), pp. 29:1–29:13.

[90] N. V. KEETHA, J. KARHADE, K. M. JATAVALLABHULA, G. YANG, S. A. SCHERER, D. RAMANAN, AND J. LUITEN, *Splatam: Splat, track & map 3d gaussians for dense RGB-D SLAM*, CoRR, abs/2312.02126 (2023).

[91] B. KERBL, G. KOPANAS, T. LEIMKÜHLER, AND G. DRETTAKIS, *3d gaussian splatting for real-time radiance field rendering*, ACM Trans. Graph., 42 (2023), pp. 139:1–139:14.

[92] B. KERBL, A. MEULEMAN, G. KOPANAS, M. WIMMER, A. LANVIN, AND G. DRETTAKIS, *A hierarchical 3d gaussian representation for real-time rendering of very large datasets*, ACM Trans. Graph., 43 (2024), pp. 62:1–62:15.

[93] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2015.

[94] G. KRISPEL, M. OPITZ, G. WALTNER, H. POSSEGGER, AND H. BISCHOF, *Fuseseg: Lidar point cloud segmentation fusing multi-modal data*, in WACV, IEEE, 2020, pp. 1863–1872.

[95] A. KURZ, T. NEFF, Z. LV, M. ZOLLHÖFER, AND M. STEINBERGER, *Adanerf: Adaptive sampling for real-time rendering of neural radiance fields*, in Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds., vol. 13677 of Lecture Notes in Computer Science, Springer, 2022, pp. 254–270.

[96] S. LAINE, J. HELLSTEN, T. KARRAS, Y. SEOL, J. LEHTINEN, AND T. AILA, *Modular primitives for high-performance differentiable rendering*, ACM Trans. Graph., 39 (2020), pp. 194:1–194:14.

[97] J. LEE, S. LEE, J. LEE, J. PARK, AND J. SIM, *Gscore: Efficient radiance field rendering via architectural support for 3d gaussian splatting*, in Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2024, La Jolla, CA, USA, 27 April 2024- 1 May 2024, R. Gupta, N. B. Abu-Ghazaleh, M. Musuvathi, and D. Tsafrir, eds., ACM, 2024, pp. 497–511.

[98] V. S. LEMPITSKY, *Surface extraction from binary volumes with higher-order smoothness*, in The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010, IEEE Computer Society, 2010, pp. 1197–1204.

[99] J. E. LENSSEN, C. OSENDORFER, AND J. MASCI, *Deep iterative surface normal estimation*, in CVPR, IEEE, 2020, pp. 11244–11253.

[100] B. LI, C. SHEN, Y. DAI, A. VAN DEN HENGEL, AND M. HE, *Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs*, in CVPR, IEEE Computer Society, 2015, pp. 1119–1127.

[101] K. LI, M. ZHAO, H. WU, D. YAN, Z. SHEN, F. WANG, AND G. XIONG, *Graphfit: Learning multi-scale graph-convolutional representation for point cloud normal estimation*, in ECCV, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds., vol. 13692, Springer, 2022, pp. 651–667.

[102] Q. LI, Y.-S. LIU, J.-S. CHENG, C. WANG, Y. FANG, AND Z. HAN, *HSurf-Net: Normal estimation for 3D point clouds by learning hyper surfaces*, in Advances in Neural Information Processing Systems (NeurIPS), vol. 35, Curran Associates, Inc., 2022, pp. 4218–4230.

[103] T. LI, M. SLAVCHEVA, M. ZOLLHÖFER, S. GREEN, C. LASSNER, C. KIM, T. SCHMIDT, S. LOVEGROVE, M. GOESELE, R. A. NEWCOMBE, AND Z. LV, *Neural 3d video syn-*

thesis from multi-view video, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 5511–5521.

[104] Z. LI, T. MÜLLER, A. EVANS, R. H. TAYLOR, M. UNBERATH, M. LIU, AND C. LIN, *Neuralangelo: High-fidelity neural surface reconstruction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 8456–8465.

[105] Z. LI, S. NIKLAUS, N. SNAVELY, AND O. WANG, *Neural scene flow fields for space-time view synthesis of dynamic scenes*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 6498–6508.

[106] Z. LI, K. WANG, W. ZUO, D. MENG, AND L. ZHANG, *Detail-preserving and content-aware variational multi-view stereo reconstruction*, IEEE Trans. Image Process., 25 (2016), pp. 864–877.

[107] A. LIN, J. LI, AND Z. MA, *On learning and learned data representation by capsule networks*, IEEE Access, 7 (2019), pp. 50808–50822.

[108] A. LIN, J. LI, L. ZHANG, Z. MA, AND W. LUO, *Multiple-task learning and knowledge transfer using generative adversarial capsule nets*, in AI 2018: Advances in Artificial Intelligence - 31st Australasian Joint Conference, Wellington, New Zealand, December 11-14, 2018, Proceedings, T. Mitrovic, B. Xue, and X. Li, eds., vol. 11320 of Lecture Notes in Computer Science, Springer, 2018, pp. 669–680.

[109] C. LIN, W. MA, A. TORRALBA, AND S. LUCEY, *BARF: bundle-adjusting neural radiance fields*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 5721–5731.

[110] Y. LIN, P. FLORENCE, J. T. BARRON, A. RODRIGUEZ, P. ISOLA, AND T. LIN, *inerf: Inverting neural radiance fields for pose estimation*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021, IEEE, 2021, pp. 1323–1330.

[111] D. B. LINDELL, J. N. P. MARTEL, AND G. WETZSTEIN, *Autoint: Automatic integration for fast neural volume rendering*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 14556–14565.

[112] P. LINDENBERGER, P. SARLIN, AND M. POLLEFEYS, *Lightglue: Local feature matching at light speed*, CoRR, abs/2306.13643 (2023).

[113] J. LIU, Y. CAO, W. MAO, W. ZHANG, D. J. ZHANG, J. KEPPO, Y. SHAN, X. QIE, AND M. Z. SHOU, *Devrf: Fast deformable voxel radiance fields for dynamic scenes*, in Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., 2022.

[114] Y. LIU, C. LUO, L. FAN, N. WANG, J. PENG, AND Z. ZHANG, *Citygaussian: Real-time high-quality large-scale scene rendering with gaussians*, in Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XVI, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, eds., vol. 15074 of Lecture Notes in Computer Science, Springer, 2024, pp. 265–282.

[115] Y. LIU, K. ZHU, G. WU, Y. REN, B. LIU, Y. LIU, AND J. SHAN, *Mv-deepsdf: Implicit modeling with multi-sweep point clouds for 3d vehicle reconstruction in au-*

tonomous driving, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 8306–8316.

[116] Y. LIU, K. ZHU, G. WU, Y. REN, B. LIU, Y. LIU, AND J. SHAN, *Mv-deepsdf: Implicit modeling with multi-sweep point clouds for 3d vehicle reconstruction in autonomous driving*, in IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, IEEE, 2023, pp. 8272–8282.

[117] Z. LIU, Y. CAO, Z. KUANG, L. KOBBELT, AND S. HU, *High-quality textured 3d shape reconstruction with cascaded fully convolutional networks*, IEEE Trans. Vis. Comput. Graph., 27 (2021), pp. 83–97.

[118] A. LOCHER, M. PERDOCH, AND L. V. GOOL, *Progressive prioritized multi-view stereo*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, 2016, pp. 3244–3252.

[119] S. LOMBARDI, T. SIMON, J. M. SARAGIH, G. SCHWARTZ, A. M. LEHRMANN, AND Y. SHEIKH, *Neural volumes: learning dynamic renderable volumes from images*, ACM Trans. Graph., 38 (2019), pp. 65:1–65:14.

[120] W. E. LORENSEN AND H. E. CLINE, *Marching cubes: A high resolution 3d surface construction algorithm*, in Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA, July 27-31, 1987, M. C. Stone, ed., ACM, 1987, pp. 163–169.

[121] T. LU, M. YU, L. XU, Y. XIANGLI, L. WANG, D. LIN, AND B. DAI, *Scaffold-gs: Structured 3d gaussians for view-adaptive rendering*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, IEEE, 2024, pp. 20654–20664.

[122] K. LUO, T. GUAN, L. JU, Y. WANG, Z. CHEN, AND Y. LUO, *Attention-aware multi-view stereo*, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, Computer Vision Foundation / IEEE, 2020, pp. 1587–1596.

[123] Y. LUO, Z. MI, AND W. TAO, *Deepdt: Learning geometry from delaunay triangulation for surface reconstruction*, in Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 2277–2285.

[124] J. MANSON, J. SMITH, AND S. SCHAEFER, *Contouring discrete indicator functions*, Comput. Graph. Forum, 30 (2011), pp. 385–393.

[125] D. MEAGHER, *Geometric modeling using octree encoding*, Comput. Graph. Image Process., 19 (1982), p. 85.

[126] Q. MÉRIGOT, M. OVSJANIKOV, AND L. J. GUIBAS, *Voronoi-based curvature and feature estimation from point clouds*, IEEE Trans. Vis. Comput. Graph., 17 (2011), pp. 743–756.

[127] L. M. MESCHEDER, M. OECHSLE, M. NIEMEYER, S. NOWOZIN, AND A. GEIGER, *Occupancy networks: Learning 3d reconstruction in function space*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, Computer Vision Foundation / IEEE, 2019, pp. 4460–4470.

[128] B. MILDENHALL, P. P. SRINIVASAN, M. TANCIK, J. T. BARRON, R. RAMAMOORTHI, AND R. NG, *Nerf: representing scenes as neural radiance fields for view synthesis*, Commun. ACM, 65 (2022), pp. 99–106.

[129] N. J. MITRA, A. T. NGUYEN, AND L. J. GUIBAS, *Estimating surface normals in noisy point cloud data*, Int. J. Comput. Geom. Appl., 14 (2004), pp. 261–276.

[130] P. MITTAL, Y. CHENG, M. SINGH, AND S. TULSIANI, *Autosdf: Shape priors for 3d completion, reconstruction and generation*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 306–315.

[131] A. MOLAEI, A. AMINIMEHR, A. TAVAKOLI, A. KAZEROUNI, B. AZAD, R. AZAD, AND D. MERHOF, *Implicit neural representation in medical imaging: A comparative survey*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 2381–2391.

[132] J. MU, W. QIU, A. KORTYLEWSKI, A. L. YUILLE, N. VASCONCELOS, AND X. WANG, *A-SDF: learning disentangled signed distance functions for articulated shape representation*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 12981–12991.

[133] T. MÜLLER, A. EVANS, C. SCHIED, AND A. KELLER, *Instant neural graphics primitives with a multiresolution hash encoding*, ACM Trans. Graph., 41 (2022), pp. 102:1–102:15.

[134] J. MUNKBERG, W. CHEN, J. HASSELGREN, A. EVANS, T. SHEN, T. MÜLLER, J. GAO, AND S. FIDLER, *Extracting triangular 3d models, materials, and lighting from images*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 8270–8280.

[135] Z. MUREZ, T. VAN AS, J. BARTOLOZZI, A. SINHA, V. BADRINARAYANAN, AND A. RABINOVICH, *Atlas: End-to-end 3d scene reconstruction from posed images*, in Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-

28, 2020, Proceedings, Part VII, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds., vol. 12352 of Lecture Notes in Computer Science, Springer, 2020, pp. 414–431.

[136] K. L. NAVANEET, K. P. MEIBODI, S. A. KOOHPAYEGANI, AND H. PIRSIAVASH, *Compgs: Smaller and faster gaussian splatting with vector quantization*, in Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXXII, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, eds., vol. 15090 of Lecture Notes in Computer Science, Springer, 2024, pp. 330–349.

[137] R. A. NEWCOMBE, S. IZADI, O. HILLIGES, D. MOLYNEAUX, D. KIM, A. J. DAVISON, P. KOHLI, J. SHOTTON, S. HODGES, AND A. W. FITZGIBBON, *Kinectfusion: Real-time dense surface mapping and tracking*, in 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland, October 26-29, 2011, IEEE Computer Society, 2011, pp. 127–136.

[138] S. NIEDERMAYR, J. STUMPFEGGER, AND R. WESTERMANN, *Compressed 3d gaussian splatting for accelerated novel view synthesis*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, IEEE, 2024, pp. 10349–10358.

[139] G. M. NIELSON, *Dual marching cubes*, in 15th IEEE Visualization Conference, IEEE Vis 2004, Austin, TX, USA, October 10-15, 2004, Proceedings, IEEE Computer Society, 2004, pp. 489–496.

[140] M. NIEMEYER, J. T. BARRON, B. MILDENHALL, M. S. M. SAJJADI, A. GEIGER, AND N. RADWAN, *Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 5470–5480.

[141] J. OH, J. CHUNG, D. LEE, AND K. M. LEE, *Deblurgs: Gaussian splatting for camera motion blur*, CoRR, abs/2404.11358 (2024).

[142] A. OUASFI AND A. BOUKHAYMA, *Unsupervised occupancy learning from sparse point cloud*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, IEEE, 2024, pp. 21729–21739.

[143] J. PAN, X. HAN, W. CHEN, J. TANG, AND K. JIA, *Deep mesh reconstruction from single RGB images via topology modification networks*, in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 9963–9972.

[144] C. M. PARAMESHWARA, G. HARI, C. FERMÜLLER, N. J. SANKET, AND Y. ALOIMONOS, *Diffposenet: Direct differentiable camera pose estimation*, in CVPR, IEEE, 2022, pp. 6835–6844.

[145] J. J. PARK, P. FLORENCE, J. STRAUB, R. NEWCOMBE, AND S. LOVEGROVE, *Deepsdf: Learning continuous signed distance functions for shape representation*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[146] K. PARK, U. SINHA, J. T. BARRON, S. BOUAZIZ, D. B. GOLDMAN, S. M. SEITZ, AND R. MARTIN-BRUALLA, *Nerfies: Deformable neural radiance fields*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 5845–5854.

[147] K. PARK, U. SINHA, P. HEDMAN, J. T. BARRON, S. BOUAZIZ, D. B. GOLDMAN, R. MARTIN-BRUALLA, AND S. M. SEITZ, *Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields*, ACM Trans. Graph., 40 (2021), pp. 238:1–238:12.

[148] D. PASCHALIDOU, A. O. ULUSOY, C. SCHMITT, L. V. GOOL, AND A. GEIGER, *Raynet: Learning volumetric 3d reconstruction with ray potentials*, in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3897–3906.

[149] A. PASZKE, S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEIN, L. ANTIGA, A. DESMAISON, A. KÖPF, E. Z. YANG, Z. DEVITO, M. RAISON, A. TEJANI, S. CHILAMKURTHY, B. STEINER, L. FANG, J. BAI, AND S. CHINTALA, *Pytorch: An imperative style, high-performance deep learning library*, in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds., 2019, pp. 8024–8035.

[150] M. PAULY, R. KEISER, L. KOBBELT, AND M. H. GROSS, *Shape modeling with point-sampled geometry*, ACM Trans. Graph., 22 (2003), pp. 641–650.

[151] S. PENG, C. JIANG, Y. LIAO, M. NIEMEYER, M. POLLEFEYS, AND A. GEIGER, *Shape as points: A differentiable poisson solver*, in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021, pp. 13032–13044.

[152] S. PENG, M. NIEMEYER, L. MESCHEDER, M. POLLEFEYS, AND A. GEIGER, *Convolutional occupancy networks*, in European Conference on Computer Vision (ECCV), 2020.

[153] M. PIALA AND R. CLARK, *Terminerf: Ray termination prediction for efficient neural rendering*, in International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021, IEEE, 2021, pp. 1106–1114.

[154] A. PUMAROLA, E. CORONA, G. PONS-MOLL, AND F. MORENO-NOGUER, *D-nerf: Neural radiance fields for dynamic scenes*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 10318–10327.

[155] C. R. QI, H. SU, K. MO, AND L. J. GUIBAS, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, IEEE Computer Society, 2017, pp. 77–85.

[156] C. R. QI, L. YI, H. SU, AND L. J. GUIBAS, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, in NeurIPS, 2017, pp. 5099–5108.

[157] X. QI, Z. LIU, R. LIAO, P. H. S. TORR, R. URTASUN, AND J. JIA, *Geonet++: Iterative geometric neural network with edge-aware refinement for joint depth and surface normal estimation*, IEEE Trans. Pattern Anal. Mach. Intell., 44 (2022), pp. 969–984.

[158] G. QIAN, A. ABUALSHOUR, G. LI, A. K. THABET, AND B. GHANEM, *PU-GCN: point cloud upsampling using graph convolutional networks*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 11683–11692.

[159] S. QIAN, T. KIRSCHSTEIN, L. SCHONEVELD, D. DAVOLI, S. GIEBENHAIN, AND M. NIESSNER, *Gaussianavatars: Photorealistic head avatars with rigged 3d gaussians*, CoRR, abs/2312.02069 (2023).

[160] Y. QIAO, A. GAO, AND M. C. LIN, *Neuphysics: Editable neural geometry and physics from monocular videos*, in Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., 2022.

[161] J. QIU, Z. CUI, Y. ZHANG, X. ZHANG, S. LIU, B. ZENG, AND M. POLLEFEYS, *Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image*, in CVPR, Computer Vision Foundation / IEEE, 2019, pp. 3313–3322.

[162] M. RAKOTOSAONA, F. MANHARDT, D. M. ARROYO, M. NIEMEYER, A. KUNDU, AND F. TOMBARI, *Nerfmeshing: Distilling neural radiance fields into geometrically-accurate 3d meshes*, CoRR, abs/2303.09431 (2023).

[163] K. REN, L. JIANG, T. LU, M. YU, L. XU, Z. NI, AND B. DAI, *Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians*, CoRR, abs/2403.17898 (2024).

[164] G. RIEGLER, A. O. ULUSOY, AND A. GEIGER, *Octnet: Learning deep 3d representations at high resolutions*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, IEEE Computer Society, 2017, pp. 6620–6629.

[165] O. RONNEBERGER, P. FISCHER, AND T. BROX, *U-net: Convolutional networks for biomedical image segmentation*, in MICCAI, vol. 9351 of Lecture Notes in Computer Science, Springer, 2015, pp. 234–241.

[166] S. SAITO, Z. HUANG, R. NATSUME, S. MORISHIMA, H. LI, AND A. KANAZAWA, *Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization*, in

2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 2304–2314.

[167] K. SAMAL, H. KUMAWAT, P. SAHA, M. WOLF, AND S. MUKHOPADHYAY, *Task-driven rgb-lidar fusion for object tracking in resource-efficient autonomous system*, IEEE Trans. Intell. Veh., 7 (2022), pp. 102–112.

[168] S. SANTURKAR, D. TSIPRAS, A. ILYAS, AND A. MADRY, *How does batch normalization help optimization?*, in NeurIPS, 2018, pp. 2488–2498.

[169] P. SARLIN, D. DETONE, T. MALISIEWICZ, AND A. RABINOVICH, *Superglue: Learning feature matching with graph neural networks*, in CVPR, Computer Vision Foundation / IEEE, 2020, pp. 4937–4946.

[170] S. SCHAEFER, T. JU, AND J. D. WARREN, *Manifold dual contouring*, IEEE Trans. Vis. Comput. Graph., 13 (2007), pp. 610–619.

[171] J. L. SCHÖNBERGER AND J. FRAHM, *Structure-from-motion revisited*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, 2016, pp. 4104–4113.

[172] D. SCULLEY, *Web-scale k-means clustering*, in Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, eds., ACM, 2010, pp. 1177–1178.

[173] T. SHEN, J. GAO, K. YIN, M. LIU, AND S. FIDLER, *Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis*, in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, M. Ranzato,

A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021, pp. 6087–6101.

[174] T. SHEN, J. MUNKBERG, J. HASSELGREN, K. YIN, Z. WANG, W. CHEN, Z. GOJCIC, S. FIDLER, N. SHARP, AND J. GAO, *Flexible isosurface extraction for gradient-based mesh optimization*, ACM Trans. Graph., 42 (2023), pp. 37:1–37:16.

[175] J. SHI, L. XU, L. HENG, AND S. SHEN, *Graph-guided deformation for point cloud completion*, IEEE Robotics Autom. Lett., 6 (2021), pp. 7081–7088.

[176] Y. SHI, B. NI, J. LIU, D. RONG, Y. QIAN, AND W. ZHANG, *Geometric granularity aware pixel-to-mesh*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 13077–13086.

[177] V. SITZMANN, J. N. P. MARTEL, A. W. BERGMAN, D. B. LINDELL, AND G. WETZSTEIN, *Implicit neural representations with periodic activation functions*, in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., 2020.

[178] V. SITZMANN, S. REZCHIKOV, B. FREEMAN, J. TENENBAUM, AND F. DURAND, *Light field networks: Neural scene representations with single-evaluation rendering*, in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021, pp. 19313–19325.

[179] V. SITZMANN, M. ZOLLHÖFER, AND G. WETZSTEIN, *Scene representation networks: Continuous 3d-structure-aware neural scene representations*, in Advances in Neu-

ral Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds., 2019, pp. 1119–1130.

[180] M. SLAVCHEVA, W. KEHL, N. NAVAB, AND S. ILIC, *SDF-2-SDF: highly accurate 3d object reconstruction*, in Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., vol. 9905 of Lecture Notes in Computer Science, Springer, 2016, pp. 680–696.

[181] L. SONG, A. CHEN, Z. LI, Z. CHEN, L. CHEN, J. YUAN, Y. XU, AND A. GEIGER, *Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields*, IEEE Trans. Vis. Comput. Graph., 29 (2023), pp. 2732–2742.

[182] X. SONG, J. ZHENG, S. YUAN, H. GAO, J. ZHAO, X. HE, W. GU, AND H. ZHAO, *SA-GS: scale-adaptive gaussian splatting for training-free anti-aliasing*, CoRR, abs/2403.19615 (2024).

[183] E. SUCAR, S. LIU, J. ORTIZ, AND A. DAVISON, *iMAP: Implicit mapping and positioning in real-time*, in Proceedings of the International Conference on Computer Vision (ICCV), 2021.

[184] D. SULSKY, Z. CHEN, AND H. L. SCHREYER, *A particle method for history-dependent materials*, Computer methods in applied mechanics and engineering, 118 (1994), pp. 179–196.

[185] C. SUN, M. SUN, AND H. CHEN, *Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 5449–5459.

[186] M. SUN, D. YANG, D. KOU, Y. JIANG, W. SHAN, Z. YAN, AND L. ZHANG, *Human 3d avatar modeling with implicit neural representation: A brief survey*, in 2022 14th International Conference on Signal Processing Systems (ICSPS), IEEE, 2022, pp. 818–827.

[187] J. TANG, J. REN, H. ZHOU, Z. LIU, AND G. ZENG, *Dreamgaussian: Generative gaussian splatting for efficient 3d content creation*, CoRR, abs/2309.16653 (2023).

[188] J. TANG, H. ZHOU, X. CHEN, T. HU, E. DING, J. WANG, AND G. ZENG, *Delicate textured mesh recovery from nerf via adaptive surface refinement*, in IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, IEEE, 2023, pp. 17693–17703.

[189] M. TATARCHENKO, A. DOSOVITSKIY, AND T. BROX, *Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs*, in IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, IEEE Computer Society, 2017, pp. 2107–2115.

[190] W. TONG, C. SIMA, T. WANG, L. CHEN, S. WU, H. DENG, Y. GU, L. LU, P. LUO, D. LIN, AND H. LI, *Scene as occupancy*, in IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, IEEE, 2023, pp. 8372–8381.

[191] E. TRETSCHK, A. TEWARI, V. GOLYANIK, M. ZOLLHÖFER, C. LASSNER, AND C. THEOBALT, *Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 12939–12950.

[192] B. TRIGGS, P. F. MCLAUCHLAN, R. I. HARTLEY, AND A. W. FITZGIBBON, *Bundle adjustment - A modern synthesis*, in Workshop on Vision Algorithms, vol. 1883 of Lecture Notes in Computer Science, Springer, 1999, pp. 298–372.

[193] P. TRUONG, M. RAKOTOSAONA, F. MANHARDT, AND F. TOMBARI, *SPARF: neural radiance fields from sparse and noisy poses*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 4190–4200.

[194] A. O. ULUSOY, M. J. BLACK, AND A. GEIGER, *Semantic multi-view stereo: Jointly estimating objects and voxels*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, IEEE Computer Society, 2017, pp. 4531–4540.

[195] L. UZOLAS, E. EISEMANN, AND P. KELLNHOFER, *Template-free articulated neural point clouds for reposable view synthesis*, in Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds., 2023.

[196] I. VIZZO, X. CHEN, N. CHEBROLU, J. BEHLEY, AND C. STACHNISS, *Poisson surface reconstruction for lidar odometry and mapping*, in ICRA, IEEE, 2021, pp. 5624–5630.

[197] J. WACZYNSKA, P. BORYCKI, S. K. TADEJA, J. TABOR, AND P. SPUREK, *Games: Mesh-based adapting and modification of gaussian splatting*, CoRR, abs/2402.01459 (2024).

[198] F. WANG, S. GALLIANI, C. VOGEL, P. SPECIALE, AND M. POLLEFEYS, *Patchmatchnet: Learned multi-view patchmatch stereo*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 14194–14203.

[199] L. WANG, J. ZHANG, X. LIU, F. ZHAO, Y. ZHANG, Y. ZHANG, M. WU, J. YU, AND L. XU, *Fourier plenoctrees for dynamic radiance field rendering in real-time*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 13514–13524.

[200] N. WANG, Y. ZHANG, Z. LI, Y. FU, H. YU, W. LIU, X. XUE, AND Y. JIANG, *Pixel2mesh: 3d mesh model generation via image guided deformation*, IEEE Trans. Pattern Anal. Mach. Intell., 43 (2021), pp. 3600–3613.

[201] P. WANG, L. LIU, Y. LIU, C. THEOBALT, T. KOMURA, AND W. WANG, *Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction*, in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021, pp. 27171–27183.

[202] P. WANG, Y. LIU, Y. GUO, C. SUN, AND X. TONG, *O-CNN: octree-based convolutional neural networks for 3d shape analysis*, ACM Trans. Graph., 36 (2017), pp. 72:1–72:11.

[203] P. WANG, C. SUN, Y. LIU, AND X. TONG, *Adaptive O-CNN: a patch-based deep representation of 3d shapes*, ACM Trans. Graph., 37 (2018), p. 217.

[204] X. WANG, M. H. ANG, AND G. H. LEE, *Voxel-based network for shape completion by leveraging edge generation*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 13169–13178.

[205] Y. WANG, Q. HAN, M. HABERMANN, K. DANIILIDIS, C. THEOBALT, AND L. LIU, *Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction*,

in IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023, IEEE, 2023, pp. 3272–3283.

[206] Y. WANG, Q. HAN, M. HABERMANN, K. DANIILIDIS, C. THEOBALT, AND L. LIU, *Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction*, in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023.

[207] Z. WANG, A. C. BOVIK, H. R. SHEIKH, AND E. P. SIMONCELLI, *Image quality assessment: from error visibility to structural similarity*, IEEE Trans. Image Process., 13 (2004), pp. 600–612.

[208] Z. WANG, S. WU, W. XIE, M. CHEN, AND V. A. PRISACARIU, *Nerf-: Neural radiance fields without known camera parameters*, CoRR, abs/2102.07064 (2021).

[209] X. WEI, F. XIANG, S. BI, A. CHEN, K. SUNKAVALLI, Z. XU, AND H. SU, *Neumanifold: Neural watertight manifold reconstruction with efficient and high-quality rendering support*, CoRR, abs/2305.17134 (2023).

[210] C. WEN, Y. ZHANG, C. CAO, Z. LI, X. XUE, AND Y. FU, *Pixel2mesh++: 3d mesh generation and refinement from multi-view images*, IEEE Trans. Pattern Anal. Mach. Intell., 45 (2023), pp. 2166–2180.

[211] F. WILLIAMS, Z. GOJCIC, S. KHAMIS, D. ZORIN, J. BRUNA, S. FIDLER, AND O. LITANY, *Neural fields as learnable kernels for 3d reconstruction*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, IEEE, 2022, pp. 18479–18489.

[212] K. WILSON AND N. SNAVELY, *Network principles for sfm: Disambiguating repeated structures with local context*, in ICCV, IEEE Computer Society, 2013, pp. 513–520.

[213] M. WU, Y. WANG, Q. HU, AND J. YU, *Multi-view neural human rendering*, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020,

Seattle, WA, USA, June 13-19, 2020, Computer Vision Foundation / IEEE, 2020, pp. 1679–1688.

[214] Z. WU, S. PAN, F. CHEN, G. LONG, C. ZHANG, AND P. S. YU, *A comprehensive survey on graph neural networks*, IEEE Trans. Neural Networks Learn. Syst., 32 (2021), pp. 4–24.

[215] W. XIAN, J. HUANG, J. KOPF, AND C. KIM, *Space-time neural irradiance fields for free-viewpoint video*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 9421–9431.

[216] P. XIANG, X. WEN, Y. LIU, Y. CAO, P. WAN, W. ZHENG, AND Z. HAN, *Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 5479–5489.

[217] H. XIE, H. YAO, X. SUN, S. ZHOU, AND S. ZHANG, *Pix2vox: Context-aware 3d reconstruction from single and multi-view images*, in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, IEEE, 2019, pp. 2690–2698.

[218] H. XIE, H. YAO, S. ZHANG, S. ZHOU, AND W. SUN, *Pix2vox++: Multi-scale context-aware 3d object reconstruction from single and multiple images*, Int. J. Comput. Vis., 128 (2020), pp. 2919–2935.

[219] H. XIE, H. YAO, S. ZHOU, J. MAO, S. ZHANG, AND W. SUN, *Grnet: Gridding residual network for dense point cloud completion*, in Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part IX, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, eds., vol. 12354 of Lecture Notes in Computer Science, Springer, 2020, pp. 365–381.

[220] Q. XU, W. KONG, W. TAO, AND M. POLLEFEYS, *Multi-scale geometric consistency guided and planar prior assisted multi-view stereo*, IEEE Trans. Pattern Anal. Mach. Intell., 45 (2023), pp. 4945–4963.

[221] Q. XU, W. WANG, D. CEYLAN, R. MECH, AND U. NEUMANN, *DISN: deep implicit surface network for high-quality single-view 3d reconstruction*, in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds., 2019, pp. 490–500.

[222] Y. YAN, Y. MAO, AND B. LI, *SECOND: sparsely embedded convolutional detection*, Sensors, 18 (2018), p. 3337.

[223] J. YANG, M. PAVONE, AND Y. WANG, *Freenerf: Improving few-shot neural rendering with free frequency regularization*, in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, 2023, pp. 8254–8263.

[224] Y. YANG, C. FENG, Y. SHEN, AND D. TIAN, *Foldingnet: Point cloud auto-encoder via deep grid deformation*, in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 206–215.

[225] Y. YAO, Z. LUO, S. LI, T. FANG, AND L. QUAN, *Mvsnet: Depth inference for unstructured multi-view stereo*, in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., vol. 11212 of Lecture Notes in Computer Science, Springer, 2018, pp. 785–801.

[226] L. YARIV, J. GU, Y. KASTEN, AND Y. LIPMAN, *Volume rendering of neural implicit surfaces*, in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds., 2021, pp. 4805–4815.

[227] L. YARIV, P. HEDMAN, C. REISER, D. VERBIN, P. P. SRINIVASAN, R. SZELISKI, J. T. BARRON, AND B. MILDENHALL, *Bakedsdf: Meshing neural sdfs for real-time view synthesis*, in ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023, E. Brunvand, A. Sheffer, and M. Wimmer, eds., ACM, 2023, pp. 46:1–46:9.

[228] A. YU, R. LI, M. TANCIK, H. LI, R. NG, AND A. KANAZAWA, *Plenoctrees for real-time rendering of neural radiance fields*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 5732–5741.

[229] A. YU, V. YE, M. TANCIK, AND A. KANAZAWA, *pixelnerf: Neural radiance fields from one or few images*, in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021, Computer Vision Foundation / IEEE, 2021, pp. 4578–4587.

[230] X. YU, Y. RAO, Z. WANG, Z. LIU, J. LU, AND J. ZHOU, *Pointr: Diverse point cloud completion with geometry-aware transformers*, in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021, IEEE, 2021, pp. 12478–12487.

[231] Z. YU, A. CHEN, B. HUANG, T. SATTLER, AND A. GEIGER, *Mip-splatting: Alias-free 3d gaussian splatting*, CoRR, abs/2311.16493 (2023).

[232] Z. YU AND S. GAO, *Fast-mvsnet: Sparse-to-dense multi-view stereo with learned propagation and gauss-newton refinement*, in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, Computer Vision Foundation / IEEE, 2020, pp. 1946–1955.

[233] Z. YU, T. SATTLER, AND A. GEIGER, *Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes*, ACM Trans. Graph., 43 (2024), pp. 271:1–271:13.

[234] W. YUAN, T. KHOT, D. HELD, C. MERTZ, AND M. HEBERT, *Pcn: Point completion network*, in 2018 international conference on 3D vision (3DV), IEEE, 2018, pp. 728–737.

[235] J. ZENG, Y. TONG, Y. HUANG, Q. YAN, W. SUN, J. CHEN, AND Y. WANG, *Deep surface normal estimation with hierarchical RGB-D fusion*, in CVPR, Computer Vision Foundation / IEEE, 2019, pp. 6153–6162.

[236] Y. ZHANG, S. SONG, E. YUMER, M. SAVVA, J. LEE, H. JIN, AND T. A. FUNKHOUSER, *Physically-based rendering for indoor scene understanding using convolutional neural networks*, in CVPR, IEEE Computer Society, 2017, pp. 5057–5065.

[237] X. ZHENG, H. PAN, P. WANG, X. TONG, Y. LIU, AND H. SHUM, *Locally attentional SDF diffusion for controllable 3d shape generation*, ACM Trans. Graph., 42 (2023), pp. 91:1–91:13.

[238] Q.-Y. ZHOU, J. PARK, AND V. KOLTUN, *Open3D: A modern library for 3D data processing*, arXiv:1801.09847, (2018).

[239] R. ZHU, Y. LIU, Z. DONG, Y. WANG, T. JIANG, W. WANG, AND B. YANG, *Adafit: Rethinking learning-based normal estimation on point clouds*, in ICCV, IEEE, 2021, pp. 6098–6107.

[240] Z. ZHU, S. PENG, V. LARSSON, W. XU, H. BAO, Z. CUI, M. R. OSWALD, AND M. POLLEFEYS, *Nice-slam: Neural implicit scalable encoding for slam*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2022.

[241] M. ZWICKER, H. PFISTER, J. VAN BAAR, AND M. H. GROSS, *EWA volume splatting*, in 12th IEEE Visualization Conference, IEEE Vis 2001, San Diego, CA, USA, October 24-26, 2001, Proceedings, T. Ertl, K. I. Joy, and A. Varshney, eds., IEEE Computer Society, 2001, pp. 29–36.