

Imperial College London
Department of Computing

Differentiable Rendering for Dense Visual SLAM

Hidenobu Matsuki

1st October 2025

Supervised by Professor Andrew J. Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the
Diploma of Imperial College London.

Statement of Originality

I hereby declare that this thesis is the result of my own independent work, except where specific acknowledgement is given. All external sources and intellectual contributions have been clearly identified and properly referenced.

Copyright Declaration

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial-No Derivatives 4.0 International Licence (CC BY-NC-ND).

Under this licence, you may copy and redistribute the material in any medium or format on the condition that; you credit the author, do not use it for commercial purposes and do not distribute modified versions of the work.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abstract

This thesis investigates differentiable rendering for dense visual Simultaneous Localization and Mapping (SLAM), specifically examining how novel rendering algorithms and 3D scene representations enhance reconstruction and spatial understanding. The choice of scene representation is fundamental to the efficiency of 3D modeling. Traditional approaches, which rely on direct 3D measurements from sensors or predictions from pre-trained networks, often struggle with limited operational ranges or training data biases. This necessitates 3D representations that can be flexibly optimized using real-time 2D observations.

With advancements in computing hardware, 3D representations optimized through differentiable rendering have gained prominence. These methods achieve high-fidelity inverse rendering and novel view synthesis. While early techniques required lengthy optimization, the introduction of explicit data structures has shifted performance toward real-time, enabling integration into SLAM. Unlike offline reconstruction, online SLAM requires continuous estimation and updating of unknown environments from sequential data, posing unique challenges for optimization stability and memory management.

This thesis examines the use of 3D scene representations in combination with differentiable rendering pipelines to enable real-time SLAM systems. The work begins by developing classical depth map based SLAM methods to establish a foundation in existing standard pipelines. Building on this baseline, the thesis explores advanced scene representations — specifically NeRF and the latest 3D Gaussian Splatting techniques — as novel approaches to SLAM. Through detailed experiments and analysis, the thesis demonstrates how to effectively address the open-world challenges inherent in SLAM tasks using these modern representations. It also highlights the unique advantages that differentiable rendering offers, such as improved accuracy and flexibility, positioning it as a promising direction for the future of visual SLAM.

Acknowledgements

My journey into visual SLAM began during my master's in 2017, when I first saw the ground-breaking works of this lab. I was fortunate enough to pursue my PhD degree here, indeed a journey filled with unexpected challenges. Just as I was preparing to travel to London in March 2020, the global pandemic struck, and my first two years of PhD were conducted remotely from Japan. The lack of face-to-face interaction, time zone differences, and uncertain social circumstances, together with not knowing when I could finally move to the UK, made for a daunting situation until midway through my PhD. Despite these extraordinary circumstances, I was able to complete my PhD thanks to the support and encouragement of many people both in the UK and Japan.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Andrew Davison. His unwavering guidance and passion for research gave me the confidence to explore new and challenging topics. Our numerous discussions on long-term research vision were a constant source of inspiration.

I am also deeply thankful to my friends and colleagues at the Dyson Robotics Lab: Raluca Scona, Tristan Laidlow, Talfan Evans, Jan Czarnowski, Stephen James, Charlie Houseago, Shuaifeng Zhi, Zoe Landgraf, Dorian Henning, Binbin Xu, Kentaro Wada, Joe Ortiz, Edgar Sucar, Shikun Liu, Riku Murai, Seth Nabarro, Aalok Patwardhan, Eric Dexheimer, Kirill Mazur, Ivan Kapelyukh, Xin Kong, Marwan Taher, Anagh Malik, Ignacio Alzugaray, Callum Rhodes, and Gwangbin Bae. From daily conversations to research discussions, every moment we shared together has been invaluable.

I would also like to thank the lab's operations manager, Iosifina Pournara, who provided invaluable help by organizing my attendance at conferences, ensuring I had access to the equipment I needed, and offering general support and advice.

I was fortunate to conduct a research internship during my PhD, and I am grateful to Keisuke Tateno, Federico Tombari, and Michael Niemeyer for hosting me at Google Zurich. Working with such exceptionally talented people at Google inspired me to pursue original and impactful research.

During the pandemic, I had the opportunity to work on applied computer vision projects at ArchTek Corporation in Osaka. This experience was both inspiring and essential in maintaining my social connections during a time of great uncertainty. I extend my sincere thanks to Goki

Kuroda, Yukiya Higashi, Yosuke Furukawa, and Shuichi Takada for their support.

Finally, I would like to thank my parents and sister for their unwavering support throughout my PhD. Completing the first half of my PhD at home, their dedication and encouragement were invaluable in helping me stay focused. Even after I traveled to the UK, their continued support remained a crucial source of strength. My father dreamed of seeing me become the first PhD in our family, and I hope he is smiling down on me from heaven.

Contents

1	Introduction	1
1.1	Spatial Perception for Embodied Agents	1
1.2	Simultaneous Localization and Mapping	2
1.3	3D Scene Representations for SLAM	3
1.4	Differentiable Rendering and Neural Graphic Primitives	4
1.5	Thesis Structure	5
1.6	Publications	6
2	Preliminaries	18
2.1	Classical 3D Scene Representations	18
2.2	Differentiable Rendering	22
2.3	3D Scene Representations with Differentiable Rendering	24
3	iMODE: Neural Fields and Differentiable Rendering for Real-Time Dense Mapping	35
3.1	Introduction	37
3.2	Related Work	38
3.3	Preliminary: Dense Reconstruction via Classical Depth Map Integration	40
3.4	System Overview	41
3.5	3D Scene Representation	42
3.6	Evaluation	45
3.7	Conclusions	52
3.8	Limitations	53
4	NEWTON: Egocentric Neural Fields for Large-scale Mapping	55
4.1	Introduction	56
4.2	Related Work	58
4.3	Method	59
4.4	Evaluation	66

4.5	Conclusion	73
4.6	Limitations	73
5	Gaussian Splatting SLAM: Point Primitive and Differentiable Rasterization for SLAM	75
5.1	Introduction	76
5.2	Related Work	78
5.3	Method	80
5.4	Evaluation	87
5.5	Conclusion	105
5.6	Limitations	105
6	4DTAM: Non-Rigid Gaussian Splatting SLAM for Dynamic Scenes	107
6.1	Introduction	109
6.2	Related Work	110
6.3	Method	113
6.4	Evaluation	122
6.5	Conclusion	129
7	Conclusion and Future Directions	131
7.1	Conclusion	131
	Bibliography	133

List of Tables

2.1	Algorithm Steps for Volumetric TSDF Fusion	22
3.1	Reconstruction results for 8 indoor Replica scenes	46
3.2	Depth map evaluation	47
3.3	Depth RMSE in metres on ScanNet dataset	51
3.4	Quantitative evaluation of semantic label fusion on Replica dataset	51
4.1	Reconstruction Results for On-the-Fly Mapping	67
4.2	Baseline Comparison	71
4.3	Depth Ablation for On-the-Fly Mapping	71
4.4	Ablation of our Feature Propagation	72
4.5	Offline Performance Evaluation on Replica dataset	72
4.6	Mapping Representation Memory Usages	72
4.7	Different Map Sizes for On-the-Fly Mapping	73
5.1	Camera tracking result on TUM for monocular and RGB-D	90
5.2	Camera tracking result on Replica for RGB-D SLAM	90
5.3	Ablation Study on TUM RGB-D dataset	91
5.4	Memory Analysis on TUM RGB-D dataset	91
5.5	Rendering performance comparison of RGB-D SLAM methods on Replica	91
5.6	Pruning Ablation Study on TUM RGB-D dataset (Monocular Input)	92
5.7	Isotropic Loss Ablation Study on TUM RGB-D dataset	93
5.8	Isotropic Loss Ablation Study on Replica dataset (RGB-D input)	93
5.9	ATE RMSE (meter) on EuRoC Machine Hall with Stereo Depth	94
5.10	Mean Rendering metrics for TUM and Replica (RGBD).	94
5.11	Mean Memory and ATE metrics for TUM (RGBD).	94
5.12	Performance Analysis using fr3/office	95
5.13	Performance Analysis using replica/office1	95

5.14	Camera convergence analysis	99
6.1	Non-rigid SLAM Evaluation on Sim4D Dataset	124
6.2	Ablation Study on L_{ARAP_n}	126
6.3	Non-rigid SLAM Evaluation on Sim4D Dataset with Monocular Depth Prior	126
6.4	Static SLAM Ablation on Replica	127
6.5	Static SLAM Ablation on TUM	127
6.6	Memory Analysis on TUM RGB-D dataset	128
6.7	Offline Non-Rigid Reconstruction Ablation	128

List of Figures

1.3	Examples of dense/semi-dense visual SLAM methods	4
1.4	Novel View Synthesis Examples of the MipNeRF360	5
2.1	Point-based Representations for SLAM	20
2.2	Voxel-based Representations for SLAM	21
2.3	Learnable 3D Representations.	23
2.4	Overview of differentiable rendering pipeline.	24
2.5	Overview of NeRF optimization pipeline.	27
2.6	An example of grid-based NeRF.	28
2.7	An example of Point-based NeRF representation	29
2.8	Neural Field for surface reconstruction.	30
2.9	Neural Fields for semantic scene reconstruction.	30
2.10	Overview of 3D Gaussian Splatting Pipeline	30
3.1	Examples of real-time dense monocular reconstructions.	36
3.2	System overview of CodeMapping	40
3.3	TSDF Fusion results in CodeMapping	40
3.4	iMODE system overview	41
3.5	Neural field represented through a 4-layer MLP	43
3.6	The effect of positional encoding frequency separation	43
3.7	Adaptive ray-sampling effect	45
3.8	3D reconstruction results on the Replica dataset	46
3.9	3D Reconstruction results	47
3.10	Large-scale reconstruction	48
3.11	Small-scale reconstruction	49
3.12	View dependent effect	49
3.13	A comparison between different supervision signals	50
3.14	The use of dense depth prior prediction	51

3.15	Semantic label fusion examples on Replica dataset	52
3.16	3D mesh and normal map result by using stereo-inertial data on EuRoC dataset	52
4.1	View-Centric Mapping with NEWTON	57
4.2	Method overview	60
4.3	Spherical Multi-Resolution Grid	61
4.4	NDC vs Spherical Parameterization	62
4.5	Scene Contraction of Mip-NeRF360	63
4.6	Feature Propagation	65
4.7	Loop Closure	68
4.8	Timeseries For On-the-Fly Mapping	69
4.9	View Synthesis on ScanNet (scene0000)	70
5.1	Qualitative reconstruction of Gaussian Splatting SLAM	76
5.2	SLAM System Overview	80
5.3	Effect of isotropic regularization	86
5.4	Rendering examples on Replica	92
5.5	Convergence basin analysis	96
5.6	Monocular SLAM result on fr1/desk sequence	96
5.7	2D Visualization of the camera pose distributions used for convergence basin analysis	97
5.8	Self-captured Scenes	100
5.9	Novel view rendering and Gaussian visualizations on TUM fr1/desk	101
5.10	Rendering comparison on TUM fr1/desk	101
5.11	Novel view rendering and Gaussian visualizations on TUM fr2/xyz	102
5.12	Rendering comparison on TUM fr2/xyz	102
5.13	Novel view rendering and Gaussian visualizations on TUM fr3/office	103
5.14	Rendering comparison on TUM fr3/office	103
5.15	Novel view rendering and Gaussian visualizations on Replica	104
5.16	Rendering comparison on Replica	104
6.1	4DTAM Overview	108
6.2	Method overview diagram	113
6.3	2D Gaussian's Surface Normal Rendering based on Different Initialization	119
6.4	Sim4D dataset	121
6.5	Training and Test Views on the Sim4D Dataset	122
6.6	Qualitative comparison to SurfelWarp	125

6.7 Qualitative Results on Real-World Dataset.	126
6.8 3D Reconstruction Result on Replica Office4	127
6.9 Non-rigid Reconstruction Results	128

Introduction

Contents

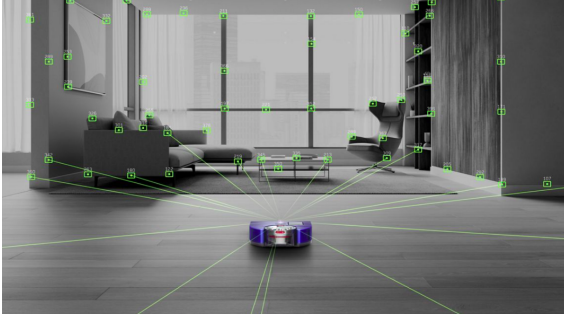
1.1	Spatial Perception for Embodied Agents	1
1.2	Simultaneous Localization and Mapping	2
1.3	3D Scene Representations for SLAM	3
1.4	Differentiable Rendering and Neural Graphic Primitives	4
1.5	Thesis Structure	5
1.6	Publications	6

In this chapter, we establish the foundations and motivations for this thesis. We begin by defining the task and providing a historical background of visual SLAM (section 1.1 and 1.2). Next, we explore the classical 3D scene representations and their applications, which have traditionally been used in visual SLAM systems (section 1.3). We then address the limitations of these classical representations and introduce an overview of modern 3D representation and differentiable rendering (section 1.4). Finally, we outline our research contributions and present the roadmap for this thesis (section 1.5 and 1.6).

1.1 Spatial Perception for Embodied Agents

Achieving human-level intelligence in embodied agents is one of the ultimate goals of robotics research. Agents that can autonomously navigate their environment, manipulate objects, interact with the world, and collaborate with others have the potential to significantly improve human life. A precise spatial understanding of their surroundings — knowing their own location, the locations of other objects or agents, and identifying what those objects or agents are — dramatically enhances the ability of these autonomous agents to perform intelligent tasks. Geometric perception

is particularly essential for enabling physical interactions with the environment, including tasks such as robot navigation, manipulation, and AR/VR device tracking and content rendering. This perception must be processed in real-time based on sensor data to ensure the agent can respond quickly and effectively to changing environments. Achieving this requires a compact yet detailed scene representation and an efficient method to construct it from real-time sensor observations.



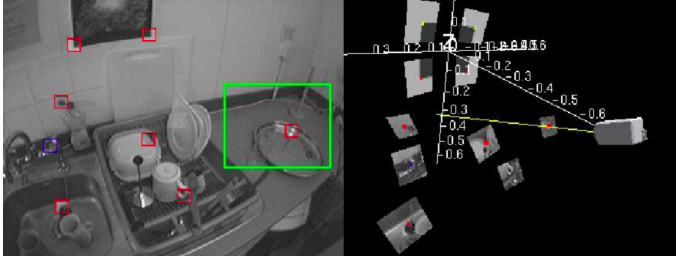
(a) Dyson Robot Vacuum Cleaner [Dyson.,]



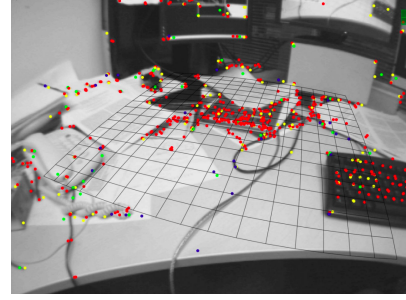
(b) Apple Vision Pro [Apple.,]

1.2 Simultaneous Localization and Mapping

Vision-based Simultaneous Localization and Mapping (Visual SLAM) has been a fundamental tool for real-time spatial perception in autonomous agents using low-cost vision sensors. From a real-time camera image stream, visual SLAM simultaneously estimates the device’s ego-motion and maps the surrounding geometry, playing a critical role in commercial applications such as robot vacuum cleaner and AR glass. Visual SLAM has a long history of research and has continually advanced with the development of computing hardware and sensors. Early SLAM methods in the 2000s are based on probabilistic filtering framework such as Kalman Filters, to estimate both camera pose and the surrounding map represented by sparse set of landmarks in real-time [Davison et al., 2007]. However, these filtering-based approaches, which updated all landmark positions and camera poses at every time step, became computationally prohibitive for large-scale environments. A major breakthrough came in 2007 with the introduction of PTAM [Klein and Murray, 2007], a keyframe-based SLAM method that separated tracking and mapping into parallel processes. This separation allowed for real-time, locally consistent tracking while global mapping could be processed more slowly but accurately. Most of the modern SLAM algorithms have since adopted this approach [Engel et al., 2014, Engel et al., 2017, Mur-Artal and Tardós, 2017a, Bloesch et al., 2018, Sucar et al., 2021], leveraging keyframes and multi-threading to maintain a balance between real-time tracking and globally consistent mapping.



(a) MonoSLAM [Davison et al., 2007]



(b) PTAM [Klein and Murray, 2007]

1.3 3D Scene Representations for SLAM

Another fundamental aspect of SLAM is the choice of scene representation. A 3D representation refers to the parameterization or data structure used to store the spatial contents of a scene. The choice of this representation directly affects the performance of SLAM systems, as it determines how effectively the environment is modeled and understood.

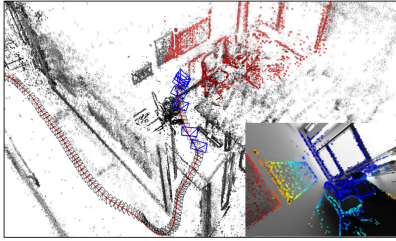
Early Visual SLAM systems primarily relied on sparse point clouds as their core representation [Davison et al., 2007, Engel et al., 2017, Mur-Artal and Tardós, 2017a]. While these were efficient for real-time localization, they discarded much of the scene detail, limiting their usefulness for more advanced robotic vision tasks. With improvements in computing hardware, SLAM systems have started incorporating richer map representations capable of capturing denser and more detailed information, which led to the emergence of the Dense SLAM research field.

For example, volumetric Signed Distance Functions (SDF) provide dense and watertight scene geometry, making them highly valuable for downstream robotics tasks such as manipulation and navigation [Newcombe et al., 2011b, Vespa et al., 2018]. Surfels, which represent the scene using point clouds with surface disks, offer a flexible approach to representing dense scene geometry and are especially effective in warping or deformation [Whelan et al., 2015b, Schöps et al., 2019]. The ongoing research in SLAM focuses on optimizing 3D scene representations to strike a balance between accuracy, efficiency, and practical applicability in diverse environments (Fig. 1.3).

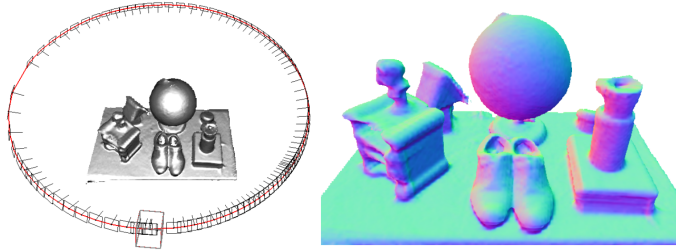
The uniqueness of the Visual SLAM problem lies in its need for real-time, online state estimation based on ego-centric observations. This introduces several key requirements for effective map representations. First, the representation must efficiently encode dense scene information, as the richness of these data directly impacts downstream tasks. Second, the map should be constructed incrementally in real-time from sensor data, as instant inference and interaction are critical for embodied agents. Third, the representation must remain consistent with the history of 2D obser-

vations. Finally, the map must be flexible enough to dynamically update based on real-time sensor input, as SLAM involves online exploration where the agent frequently updates the estimation.

Traditionally, dense SLAM methods struggled to fully meet these requirements. Most existing approaches relied on casting 2D observation/estimation results into 3D through a separate hand-designed fusion process [Newcombe et al., 2011b, Koestler et al., 2021], which hindered consistent map construction. First, dense 2D depth maps or point clouds were directly observed by sensors or estimated from multiple views, then fused into a 3D representation via weighted averaging, rather than directly optimizing the 3D map. This approach resulted in representations that were less consistent in real-time applications, as they did not fully leverage the 3D map’s properties, such as occlusion handling. The inability to propagate gradients from pixels to the 3D map model further limited the map’s adaptability, preventing it from meeting the full range of conditions.



(a) Direct Sparse Odometry [Engel et al., 2017]



(b) KinectFusion [Newcombe et al., 2011a]

Figure 1.3: Examples of dense/semi-dense visual SLAM methods

1.4 Differentiable Rendering and Neural Graphic Primitives

Recent advances in graphics hardware and automatic differentiation frameworks for machine learning have led to significant progress in differentiable rendering. Initially, differentiable rendering was developed to optimize classical explicit 3D data structures, such as meshes or occupancy grids, using only 2D image supervision [Kato et al., 2018a, Liu et al., 2019]. However, new representations better suited to the differentiable rendering framework, known as Neural Graphic Primitives, have since emerged. Two of the most notable examples are NeRF [Mildenhall et al., 2020] and 3D Gaussian Splatting [Kerbl et al., 2023], which represent a scene with a radiance field that maps location to color and density, and are rendered through a differentiable volumetric rendering pipeline. These representations capture dense and highly detailed scene information through their volumetric parameterization and can be directly optimized via differentiable rendering, eliminating the need for potentially inconsistent hand-designed fusion modules. While these

primitives and pipelines were originally designed for offline novel view synthesis tasks with posed images, their ability to produce highly detailed, real-time, consistent, and flexible outputs shows significant potential for integration into visual SLAM systems.



Figure 1.4: Novel View Synthesis Examples of MipNeRF360 [Barron et al., 2022]. (left:) and 3DGS [Kerbl et al., 2023] (right:) on the MipNeRF 360 dataset. Both methods can capture high fidelity radiance field from posed RGB images. The images are from [Barron et al., 2022, Kerbl et al., 2023]

In this thesis, we explore dense real-time visual SLAM methods that achieve high-detail scene capture and long-term consistency through differentiable rendering and 3D scene representations. We introduce methods that leverage high-fidelity rendering and end-to-end consistent optimization to enhance the visual SLAM pipeline. We address key challenges involved in deploying these representations for online SLAM tasks, ensuring both flexibility and real-time performance. While the definition of "real-time" varies depending on available computing resources — ranging from immediate deployment on mobile devices to research-oriented exploration on more powerful server computers — this thesis primarily targets real-time performance on a desktop machine equipped with a consumer-grade GPU available at the time of this study.

1.5 Thesis Structure

This thesis is structured as follows to provide a clear progression through the concepts:

Chapter 2 – Preliminaries. In this chapter, we formally introduce differentiable rendering and 3D representations. We cover various rendering techniques, including ray marching and rasterization, and then highlight NeRF and 3D Gaussian Splatting as widely adopted frameworks based on differentiable rendering, laying the groundwork for the subsequent chapters.

Chapter 3 – iMODE: Neural Fields and Differentiable Rendering for Real-Time Dense Mapping. This chapter presents [Matsuki et al., 2023], which explores the potential of differentiable

rendering for SLAM through the design and evaluation of a NeRF-based dense mapping system combined with an off-the-shelf sparse tracker. The system achieves real-time, dense reconstruction from monocular images, offering flexible and domain-agnostic performance with the option to seamlessly incorporate additional 2D priors for more advanced mapping.

Chapter 4 – NEWTON: Egocentric Neural Fields and rendering for Large-scale Mapping.

This chapter details [Matsuki et al., 2024b], which addresses the limitations of existing Neural Field SLAM/Mapping methods in large-scale scenes, particularly during loop closures. We identify the world-centric nature of Neural Fields as a fundamental drawback and propose a “view-centric” approach, consisting of multiple local Neural Fields tied to the device’s coordinate system.

Chapter 5 – Gaussian Splatting SLAM: Point-Based SLAM via Differentiable Rasterization

In this chapter, we introduce our Gaussian Splatting SLAM [Matsuki et al., 2024a], arguing that 3D Gaussian Splatting offers distinct advantages for online SLAM tasks — advantages not found in Neural Field approaches — due to its discrete, flexible spatial allocation and differentiable rasterization. We show how to harness these properties and present the first Gaussian Splatting-based SLAM system that operates solely on RGB images. As a joint first-author, my role involved independent initial development and subsequent collaborative leadership. I established an initial working prototype for both RGB-D and Monocular offline reconstruction/SLAM pipelines and developed the evaluation protocol within the first two months. I then introduced and guided the collaboration to continuously refine and improve key performance indicators throughout the remainder of the project.

Chapter 6 – 4DTAM: Gaussian Splatting SLAM for Dynamic Scenes Finally, we introduce 4DTAM [Matsuki et al., 2025], a Gaussian Splatting-based SLAM method capable of handling dynamically changing scenes, which is a more natural and realistic scenario than existing static SLAM settings. We demonstrate that complex dynamic scenes can be modeled by a simple extension of 3D Gaussians and optimized from 2.5D sensor data, enabled by end-to-end optimization through differentiable rendering.

Chapter 7 – Conclusion and Future Directions. We conclude the thesis by summarizing the key contributions and discussing potential future research directions.

1.6 Publications

The work described in this thesis resulted in the following publications:

- Hiddenobu Matsuki, Edgar Sucar, Tristan Laidlow, Kentaro Wada, Raluca Scona, Andrew J. Davison, **iMODE:Real-Time Incremental Monocular Dense Mapping Using Neural Field**, IEEE International Conference of Robotics and Automation (ICRA), 2023 (**Oral and Best Navigation Award Finalist**) [[Matsuki et al., 2023](#)].
- Hiddenobu Matsuki, Keisuke Tateno, Michael Niemeyer, Federico Tombari **NEWTON: Neural View-Centric Mapping for On-the-Fly Large-Scale SLAM**, IEEE Robotics and Automation Letters (RA-L), 2024 [[Matsuki et al., 2024b](#)].
- Hiddenobu Matsuki*, Riku Murai*, Paul H.J. Kelly, Andrew J. Davison, **Gaussian Splatting SLAM**, IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR), 2024 (**Highlight and Best Demo Award**) (*Equal contribution) [[Matsuki et al., 2024a](#)]
- Hiddenobu Matsuki, Gwangbin Bae, Andrew J. Davison, **4DTAM: Non-Rigid Tracking and Mapping via Dynamic Surface Gaussians**, IEEE / CVF Computer Vision and Pattern Recognition Conference (CVPR), 2025 [[Matsuki et al., 2025](#)]

While not explicitly included in the thesis, the following work was done in conjunction:

- Hiddenobu Matsuki, Raluca Scona, Jan Czarnowski, Andrew J. Davison **CodeMapping: Real-Time Dense Mapping for Sparse SLAM using Compact Scene Representations**, IEEE Robotics and Automation Letters, 2021 [[Matsuki et al., 2021](#)].
- Raluca Scona, Hiddenobu Matsuki, Andrew J. Davison, **From Scene Flow to Visual Odometry through Local and Global Regularisation in Markov Random Fields**, IEEE Robotics and Automation Letters, 2023 [[Scona et al., 2022](#)]

Preliminaries

Contents

2.1	Classical 3D Scene Representations	18
2.1.1	Point-based Representation	19
2.1.2	Polygon-based Representation	20
2.1.3	Voxel-based Representation	20
2.2	Differentiable Rendering	22
2.2.1	Optimization-Friendly 3D Representations	23
2.2.2	Rendering Process	23
2.3	3D Scene Representations with Differentiable Rendering	24
2.3.1	NeRF	25
2.3.2	3D Gaussian Splatting	30

This chapter explores the technical details of 3D representations and differentiable rendering. We begin by reviewing classical 3D representations (Section 2.1), then introduce the differentiable rendering framework (Section 2.3). Next, we highlight NeRF and 3D Gaussian Splatting as two widely adopted differentiable rendering-based representations (also in Section 2.3). Applications to SLAM are subsequently detailed in chapters 3 through 6.

2.1 Classical 3D Scene Representations

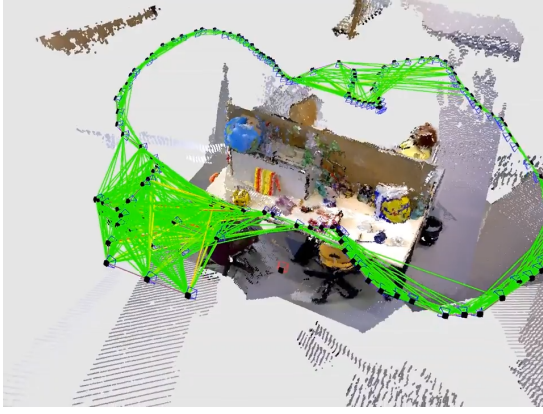
There are various 3D map representations used in dense visual SLAM. Broadly, these representations can be categorized into *point-based* (e.g., point clouds and surfels) *polygon-based* (e.g., triangle meshes) and *volumetric-based* (e.g., occupancy grids and truncated signed distance functions (TSDF)). While point-based methods provide flexibility and simplicity, volumetric approaches

maintain more structured spatial information that is beneficial for tasks such as surface extraction and collision checking. Among these representations, volumetric TSDFs have emerged as one of the most widely adopted choices due to their balance of accuracy, robustness, and ease of integration. In this section, we provide an overview of commonly used representations, followed by a detailed explanation of TSDF fusion, which is frequently employed algorithm to estimate dense 3D model.

2.1.1 Point-based Representation

Point Clouds provide an unstructured collection of 3D points that represent discrete samples of the surface of objects within the environment. They primarily describe a scene as a set of 3D coordinates (x, y, z) , but can also include additional attributes such as color and semantic labels. Due to their intuitive and straightforward nature, point clouds are a popular choice and are employed by various visual SLAM methods [Kerl et al., 2013a, Engel et al., 2014, Bloesch et al., 2018, Czarnowski et al., 2020a]. Since each point is independently estimated, point clouds can facilitate real-time non-linear least squares optimization via second-order optimizer, which is beneficial for real-time applications. However, the lack of inherent structure and connectivity between points presents challenges in downstream tasks, such as collision avoidance, where understanding the continuity of surfaces is crucial. Moreover, point clouds are often inefficient in terms of data compactness, as they require storing large amounts of data to represent surfaces accurately, which can be a drawback in resource-constrained environments.

Surfels (Surface Elements) offer a compact, efficient alternative to point clouds by representing surfaces as oriented disks. Surfel is parameterized by its location (x, y, z) , radius r and surface normal vector (n_x, n_y, n_z) , providing a balance between the dense information of point clouds and the structured simplicity of surface models. This representation captures more continuous surface information than point clouds. Several methods employ Surfels as a core representation for dense SLAM [Whelan et al., 2015b, Schöps et al., 2019] and demonstrates accurate dense scene modeling performance [Whelan et al., 2016, Schöps et al., 2019]. However, the estimation of the radius and normal vector typically requires an accurate initial dense point cloud, which can limit the application of surfels to systems like RGB-D SLAM, where depth information is readily available.



(a) DVO-SLAM (Point Clouds) [Kerl et al., 2013b]



(b) ElasticFusion (Surfels) [Whelan et al., 2016]

Figure 2.1: Dense SLAM methods based on Point-based Representations

2.1.2 Polygon-based Representation

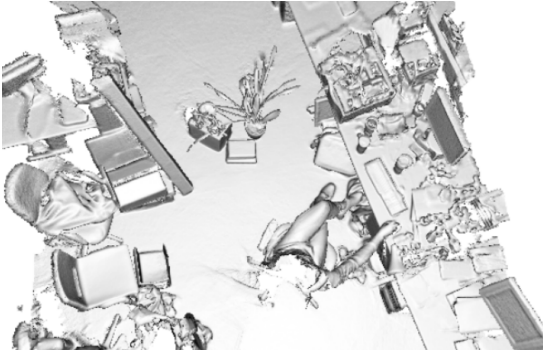
Triangle Meshes are among the most widely employed representations in 3D modeling, especially within video game engines. They are lightweight and provide a structured way to represent complex surfaces using vertices and faces. This structured nature makes them ideal for visual rendering, detailed surface analysis, and even physics simulations supported by modern game engines. However, optimizing a mesh directly from sensor measurements poses significant challenges. In particular, changes in topology, such as merging or splitting surfaces, introduce non-smooth transitions in vertex connectivity, making gradient-based optimization difficult. As a result, meshes are seldom used as a core representation in SLAM systems. Instead, they are typically derived from other SLAM-based representations using dedicated meshing algorithms, such as Marching Cubes or Poisson Meshing.

2.1.3 Voxel-based Representation

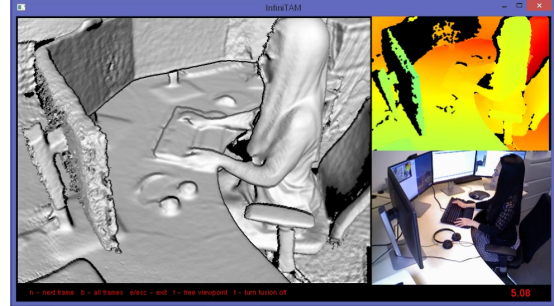
Occupancy Grids discretize the environment into a grid of cells, each cell representing the probability of being occupied by an obstacle. Several SLAM approaches use this representation [Labbe and Michaud, 2019, Ho et al., 2018] because it is particularly beneficial for robotic navigation, where distinguishing between free and occupied space is crucial for safe and efficient movement. However, the resolution of occupancy grids has a direct impact on memory usage and computational demands — higher resolutions provide more detail but at the cost of increased resource consumption. Moreover, occupancy grids struggle with representing complex geometries at a fine level of detail due to their discrete nature. Additionally, the grid must be allocated before the system begins operating, which requires prior knowledge of the scene’s boundaries. This static

allocation can limit the system’s flexibility, as updating the grid dynamically during runtime is not straightforward, posing challenges in environments where the boundaries are unknown or constantly changing. These issues can be partially addressed by employing adaptive grid structures, such as Octrees [Zeng et al., 2013, Hornung et al., 2013], but it involves additional engineering effort and complicates the representation.

Truncated Signed Distance Functions (TSDF) represent the environment as a volumetric field where each voxel stores the signed distance to the nearest surface, and the most popular choice for dense SLAM map representation [Newcombe et al., 2011a, Vespa et al., 2018, Prisacariu et al., 2017]. This distance information allows for more efficient rendering and collision checking through techniques like sphere tracing, providing a significant advantage over naive occupancy grids. TSDF is typically estimated via depth map fusion algorithm: By integrating depth maps captured from different viewpoints, it fuse them into a coherent volumetric representation that approximates the true geometry of the scene. The key idea is to update the TSDF values of voxels based on the differences between observed depth values and the distances from the camera to the voxels.



(a) KinectFusion [Newcombe et al., 2011a]



(b) InfiniTAM [Prisacariu et al., 2017]

Figure 2.2: Dense SLAM methods based on TSDF

For each voxel $\mathbf{v}$, the TSDF value $F(\mathbf{v})$ is defined as:

$$F(\mathbf{v}) = \begin{cases} \text{clamp}\left(\frac{d_s - d}{\mu}, -1, 1\right), & \text{if } |d_s - d| \leq \mu, \\ \text{null}, & \text{otherwise,} \end{cases} \quad (2.1)$$

where $d \in \mathbb{R}$ is the distance from the camera center to $\mathbf{v}$, $d_s \in \mathbb{R}$ is the measured depth along the ray through $\mathbf{v}$, $\mu \in \mathbb{R}$ is the truncation distance, and $\text{clamp}(x, -1, 1)$ limits x to $[-1, 1]$. Each voxel also maintains a weight $W(\mathbf{v})$ representing confidence in $F(\mathbf{v})$, updated to account for multiple observations.

Initialize a 3D voxel grid where each voxel $\mathbf{v}$ has $F(\mathbf{v}) = 0$ and $W(\mathbf{v}) = 0$. For each depth map D_i and camera pose $(\mathbf{R}_i, \mathbf{t}_i)$ where $\mathbf{R}_i \in \mathbb{R}^{3 \times 3}$ is the rotation matrix and $\mathbf{t}_i \in \mathbb{R}^3$ is the translation vector:

Step	Description
For each voxel $\mathbf{v}$:	
1.	Transform $\mathbf{v}$ to camera frame: $\mathbf{v}_{\text{cam}} = \mathbf{R}_i \mathbf{v} + \mathbf{t}_i$
2.	Project onto image plane: $(u, v, w)^\top = \mathbf{K} \mathbf{v}_{\text{cam}}$ where $\mathbf{K}$ is the intrinsic camera matrix
3.	If within image bounds , proceed; else, skip voxel
4.	Compute distance: $d = \ \mathbf{v}_{\text{cam}}\ $
5.	Retrieve depth value from depth map: $d_s = D_i\left(\frac{u}{w}, \frac{v}{w}\right)$
6.	Compute signed distance: $\phi = d_s - d$
7.	Update TSDF and weight: If $-\mu < \phi \leq \mu$: Set $F_{\text{new}} = \frac{\phi}{\mu}$ Update $F(\mathbf{v}) = \frac{W(\mathbf{v})F(\mathbf{v}) + wF_{\text{new}}}{W(\mathbf{v}) + w}$ Update $W(\mathbf{v}) = W(\mathbf{v}) + w$ Else if $\phi > \mu$: Set $F_{\text{new}} = 1$ Update $F(\mathbf{v}), W(\mathbf{v})$ accordingly Else if $\phi \leq -\mu$: Skip voxel

Table 2.1: Algorithm Steps for Volumetric TSDF Fusion

While TSDF fusion is widely used due to its simplicity and computational efficiency, it remains a heuristic approach based on weighted averaging of observed depths. As a result, it cannot directly leverage gradient-based optimization to enforce end-to-end alignment with raw observation data.

2.2 Differentiable Rendering

To ensure that a 3D representation aligns with observed 2D data, one needs a formulation that enables gradient-based optimization directly from the image domain. *differentiable rendering* (DR) provides this capability by allowing gradients to flow from 2D images back into the 3D model parameters. Early DR works [Loper and Black, 2014, Zienkiewicz et al., 2016] focused mainly on triangle meshes with limited topological changes. However, DR has since gained popularity as a method to train neural networks that model complex 3D scenes directly from 2D observations, without relying on 3D ground truth data. The advancement of DR relies on two key elements: (1) **Optimization-Friendly 3D Representations** and (2) **Rendering Pipelines**.

2.2.1 Optimization-Friendly 3D Representations

With the success of deep learning in the 2D image domain [Deng et al., 2009, He et al., 2016, He et al., 2017], the research community has progressively explored learning-based approaches for 3D tasks. Early works leveraged traditional 3D data structures such as points [Qi et al., 2017] and occupancy grids [Riegler et al., 2017], processing them to perform various downstream tasks. However, these classical representations, being inherently discrete, present challenges for deep learning. They often struggle to represent watertight and continuous surfaces with flexible topologies, which is crucial for gradient-based optimization in neural network training. To address these limitations, researchers have introduced more learning-friendly 3D representations by utilizing neural networks as scalar function approximators. These networks define occupancy or signed distance functions [Mescheder et al., 2019, Park et al., 2019, Chen and Zhang, 2019], enabling direct 3D supervision. For instance, DeepSDF [Park et al., 2019] uses an 8-layer Multi-Layer Perceptron (MLP) to regress a signed distance function (SDF) from a 3D coordinate, optionally conditioned on a latent code, as summarized in Figure 2.3. These methods have demonstrated the ability to reconstruct continuous and compact 3D shapes without being constrained by the limitations of topology and discretization errors.

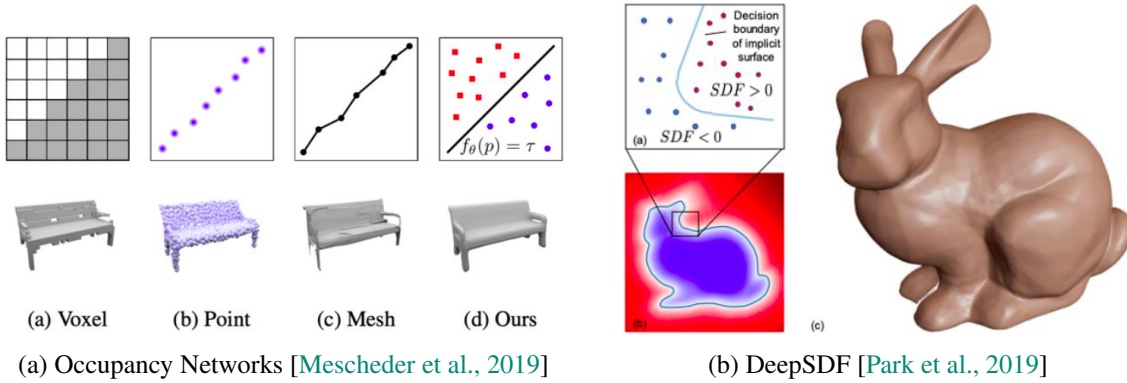


Figure 2.3: Learnable 3D Representations. Both ONet and DeepSDF regresses continuous implicit surface functions modelled by neural network.

2.2.2 Rendering Process

Supervised training on 3D data poses challenges due to the difficulty and expense of obtaining large-scale ground truth annotations. To overcome these issues, recent works integrate graphics rendering into neural network training, leading to the advancement of DR. In a graphics pipeline, a 3D model can be represented as surface data, explicitly encoding solid object boundaries (e.g., polygons or implicit surfaces), or as volume data, describing continuous fields like density or color

and representing the object’s internal properties and material distribution. Correspondingly, different rendering methods are employed: rasterization projects surface geometry onto a 2D plane and benefits from hardware acceleration, making it efficient for explicit surfaces, while ray marching, which casts rays and samples values along their paths, is suited for handling volume data and continuous fields at the cost of increased computational effort.

Although early DR approaches used softened rasterization for meshes [Loper and Black, 2014, Kato et al., 2018a, Liu et al., 2019], volume rendering has emerged as a more powerful approach for gradient-based optimization. By providing gradients over 3D space — even before identifying exact surface positions — volume rendering makes neural network training easier and thus is widely adopted as a DR backbone. Following NeuralVolume [Lombardi et al., 2019], works like differentiable volumetric rendering [Niemeyer et al., 2020a] and Scene Representation Networks [Sitzmann et al., 2019a] use differentiable ray marching to represent implicit surfaces. Although initially limited to simple shapes, these methods laid the groundwork for more complex and realistic 3D reconstructions.

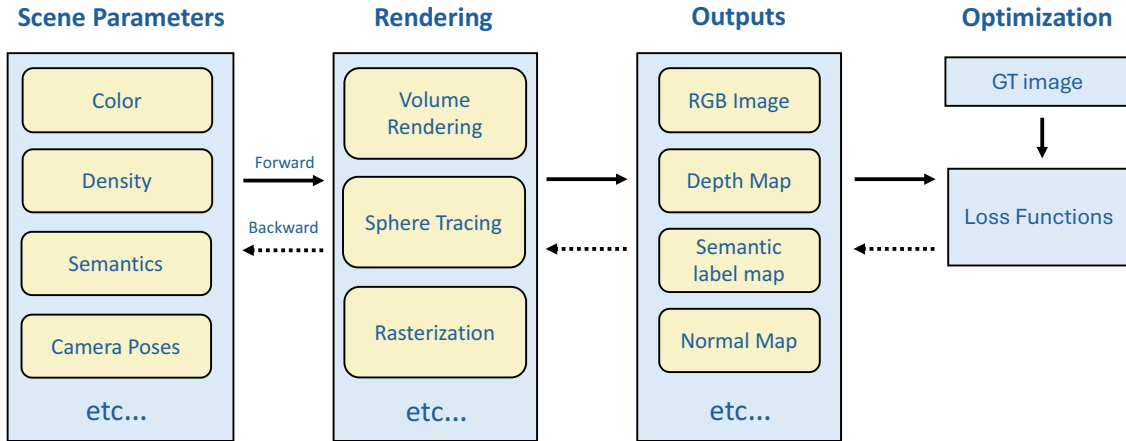


Figure 2.4: Overview of differentiable rendering pipeline. DR allows to optimize 3D scene parameters from multi-view 2d images via end-to-end manner thanks to its consistent gradient propagation.

2.3 3D Scene Representations with Differentiable Rendering

A major breakthrough occurred in 2020, when Mildenhall et al. introduced NeRF [Mildenhall et al., 2020], an end-to-end radiance field reconstruction method from 2D images. NeRF inspired a significant amount of subsequent research. Later, 3D Gaussian Splatting [Kerbl et al., 2023] further advanced the field by providing an ultra-fast differentiable rendering framework for volu-

metric Gaussian primitives. This section offers an introduction and the mathematical details of these methods.

2.3.1 NeRF

NeRF employs a single MLP as the scene representation, performing volume rendering by marching along pixel rays and querying the continuous radiance field from the MLP for opacity and color. Because volume rendering is inherently differentiable, the MLP parameters can be optimized to minimize a rendering loss function using multiview information, resulting in high-quality novel view synthesis. To capture high-frequency scene details, NeRF’s MLP takes both spatial coordinates and viewing directions as inputs, supplemented by positional encoding.

The NeRF model is expressed as a function $f_{\Theta}(\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, with MLP weights Θ , in-scene 3D coordinates $\mathbf{x} = (x, y, z)$, viewing direction $\mathbf{d} = (\theta, \phi)$, color $\mathbf{c} = (r, g, b)$ and volume density σ .

To synthesize novel views, the NeRF workflow involves casting camera rays through the scene, generating sampling points for each pixel. The local color and density at each sampling point are computed using the NeRF MLP(s), and volume rendering is used to synthesize the 2D image. The expected color $C(\mathbf{r})$ of camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is calculated using the following integral formulation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), \mathbf{d}) dt \quad (2.2)$$

Here, dt denotes the differential distance traveled by the ray at each integration step and t_n and t_f denote near and far bounds. The terms $\sigma(\mathbf{r}(t))$ and $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ represent the volume density and color at point $\mathbf{r}(t)$ along the camera ray with viewing direction $\mathbf{d}$, respectively. The accumulated transmittance from t_n to t is given by $T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right)$.

The continuous integral is estimated numerically with quadrature, which requires sampling a finite set of points along the ray. Naively sampling a large number of points is computationally expensive. To address this, NeRF employs a hierarchical volume sampling scheme using two networks: a coarse model ($\hat{C}_c$) and a fine model ($\hat{C}_f$).

1. Coarse Sampling (Stratified Sampling): The coarse model first samples N_c points along the ray using a stratified sampling approach. The ray is divided into N_c evenly-spaced bins, and

one sample is drawn randomly within each bin. This ensures that samples are distributed throughout the entire ray segment.

The i -th sample point t_i for the coarse model is drawn from the uniform distribution:

$$t_i \sim \mathcal{U} \left[t_{\text{near}} + \frac{i-1}{N_c}(t_{\text{far}} - t_{\text{near}}), t_{\text{near}} + \frac{i}{N_c}(t_{\text{far}} - t_{\text{near}}) \right]. \quad (2.3)$$

2. Fine Sampling (Importance Sampling): The resulting density (σ_i) and color ($\mathbf{c}_i$) from the coarse samples are used to estimate the ray’s color via quadrature:

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \quad \text{where } \delta_i = t_{i+1} - t_i.$$

This coarse rendering result is then used to construct a piecewise constant Probability Density Function (PDF) that models where most of the relevant scene content (i.e., surfaces) lies. The fine model then performs importance sampling by drawing N_f additional samples from this PDF. This concentrates the sampling budget on regions with higher estimated opacity and volume density, where light is most likely to be blocked or emitted.

The final color is then estimated by evaluating the volume rendering equation on the combined set of $N_c + N_f$ samples using the fine model’s MLP weights Θ_f :

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N_c+N_f} \alpha_i T_i \mathbf{c}_i, \quad \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \quad (2.4)$$

where, $\delta_i = t_{i+1} - t_i$ denotes the interval between consecutive samples t_i and t_{i+1} , while σ_i and $\mathbf{c}_i$ indicate the density and color evaluated at the sample point i along the ray, respectively. Additionally, $\alpha_i = (1 - \exp(-\sigma_i \delta_i))$ denotes the opacity resulting from the alpha compositing at the sample point i .

The expected depth along a ray D , representing a ray-termination point, can be estimated using the accumulated transmittance:

$$D(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \cdot \sigma(\mathbf{r}(t)) \cdot t \, dt, \quad (2.5)$$

This continuous integration can also be approximated via quadrature as:

$$\hat{D}(\mathbf{r}) = \sum_{i=1}^N \alpha_i t_i T_i. \quad (2.6)$$

To optimize the model, a quadratic photometric error between the rendered image and the ground truth color image is used:

$$L = \sum_{r \in R} \|\hat{C}(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2. \quad (2.7)$$

Here, $C_{gt}(r)$ denotes the ground truth color for the pixel associated with $\mathbf{r}$ in the training image, and R denotes the batch of rays for synthesizing the target image.

In the context of SLAM, several methods [Sucar et al., 2021, Zhu et al., 2022, Wang et al., 2023] also employ depth rendering loss to minimize the discrepancy between expected depth and sensor/predicted depth.

$$L_{depth} = \sum_{r \in R} \|\hat{D}(\mathbf{r}) - D_{gt}(\mathbf{r})\|_2^2. \quad (2.8)$$

Other geometric regularization techniques such as depth smoothing or normal consistency can be applied to further improve the geometric consistency.

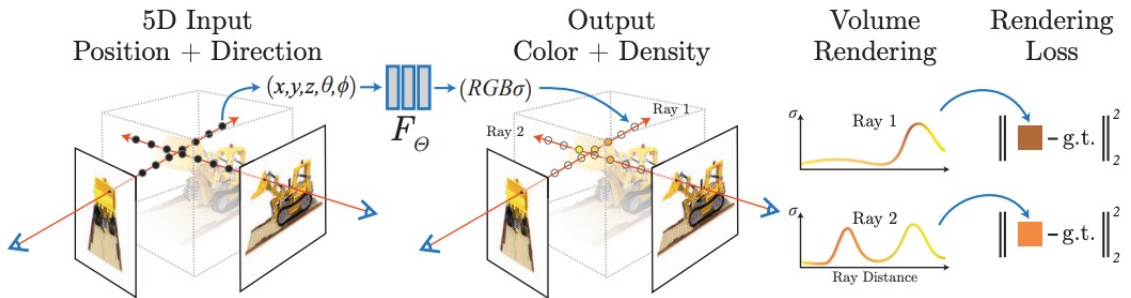


Figure 2.5: Overview of NeRF optimization pipeline. Radiance Field is represented by MLP and 2D images are rendered via differentiable volume rendering. MLP parameters are optimized to minimize the photometric error between rendered color image and ground truth color image. The image is from [Mildenhall et al., 2020]

While originally proposed as an offline radiance field reconstruction method for photorealistic scene capture, NeRF’s straightforward and general formulation has inspired numerous follow-up works aimed at enhancing performance. In the following paragraphs, we provide an overview of key NeRF extensions, highlighting alternative data structures and the diverse field representations that extend beyond the radiance field itself.

Data Structure

The original NeRF [Mildenhall et al., 2020] employs a single MLP to represent the entire radiance field, requiring a forward pass to evaluate color and density at specific 3D locations. This approach involves all MLP parameters during each evaluation, resulting in significantly slow inference and training times. To achieve faster operation, several approach introduced explicit data structure either partially or completely, and achieved fast data-access.

Voxel/Hash Grid Early works introduced voxel grids to speed up forward inference [Reiser et al., 2021, Yu et al., 2021a]. Later, Plenoxel [Fridovich-Keil et al., 2022] and DirectVoxGo [Sun et al., 2022b] introduced voxel feature grids as trainable properties and encoded the radiance field into them. Compared to MLP-based NeRF, color and density at specific locations are queried through simple index lookups and trilinear interpolation, which significantly accelerates the entire pipeline, reducing training time from days to approximately 10 minutes. Instant-NGP [Müller et al., 2022] further introduced multi-resolution hash encoding, efficiently encoding volumetric features in hash tables. Combined with a highly optimized CUDA implementation, Instant-NGP achieves NeRF training in a few seconds.

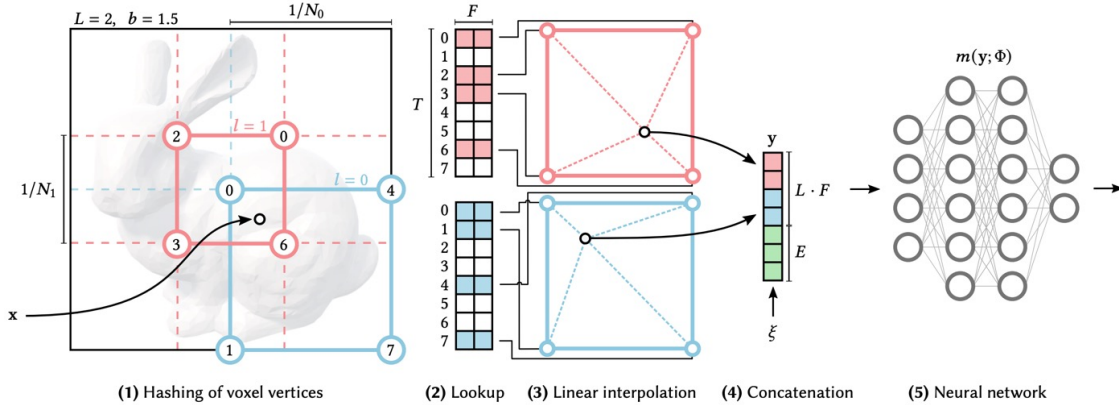


Figure 2.6: An example of grid-based NeRF (Instant-NGP [Müller et al., 2022]). The radiance field is modelled by a combination of multi-resolution hash grid and tiny MLP decoder. The image is from [Müller et al., 2022]

Point Clouds/Surfels integrated into NeRF frameworks. For instance, Point-NeRF [Xu et al., 2022] optimizes point clouds with associated features and employs a lightweight MLP decoder to render images using a volume rendering framework. This approach demonstrates that point-based representations offer greater spatial allocation flexibility compared to regular voxel grids, allowing the model to skip empty spaces and achieving a threefold speed-up over vanilla NeRF. Similarly, surfels can also be utilized to represent radiance fields [Gao et al., 2023].

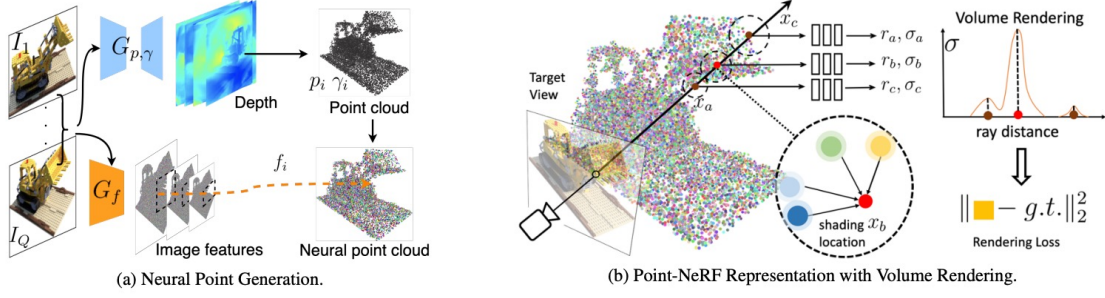


Figure 2.7: An example of Point-based NeRF representation (PointNeRF [Xu et al., 2022]). Radiance field is modeled by a combination of pointcloud with trainable features and tiny MLP decoder. The image is from [Xu et al., 2022]

Neural Fields: Generalized Field Representations using Neural Network

While the original NeRF primarily focuses on representing a radiance field — mapping 3D positions to color and density for photorealistic scene capture — the underlying concept can be extended to various other outputs, depending on the downstream task. Such generalized field representations are termed **Neural Fields**. Here, we introduce a few major examples.

Surface (SDF/Occupancy) To achieve more geometrically accurate 3D reconstructions, Neural Fields can utilize representations like Signed Distance Functions (SDF) and Occupancy Fields. Techniques such as NeuS [Wang et al., 2021a] and VolSDF [Yariv et al., 2021a] leverage SDF to represent surfaces. SDF encodes the distance of a point from the nearest surface, with the sign indicating whether the point is inside or outside the object. By integrating SDF into NeRF, it is possible to achieve precise surface modeling. Techniques like UNISURF [Oechsle et al., 2021] employ occupancy fields to represent the presence or absence of surfaces in the 3D space. Occupancy fields divide the space into a grid, where each cell indicates whether it is occupied by a surface. This method can be combined with NeRF to handle complex scenes effectively. To extract meshes from these representations, techniques such as marching cubes can be applied to convert the continuous surface representation into a discrete mesh format.

Semantics Incorporating semantic information into Neural Fields enables the decomposition of scenes into meaningful components, enhancing scene understanding and interaction. Early works such as [Zhi et al., 2021, Zhi et al., 2022] extend NeRF by outputting semantic labels in addition to color and density. This is achieved by training the model with multi-view semantic images, enabling it to predict the semantic category of each part of the scene. These fields fuse high-dimensional features from multiple views to produce a 3D consistent semantic feature field.

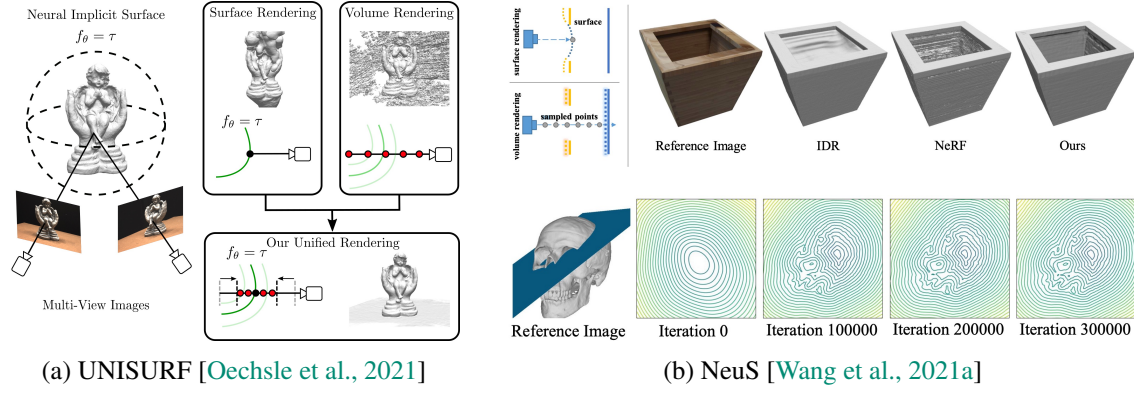


Figure 2.8: Neural Field for surface reconstruction. UNISURF represents occupancy field and NeuS represents continuous Signed Distance Functions via MLP which can be rendered via volume rendering. The images are from [Oechsle et al., 2021, Wang et al., 2021a]

More generally, NeRF can fuse 2D features from pre-trained foundation networks and perform more advanced semantic tasks such as scene decomposition and interactive editing via language prompting or clicking [Kobayashi et al., 2022, Tschernezki et al., 2022, Mazur et al., 2023].

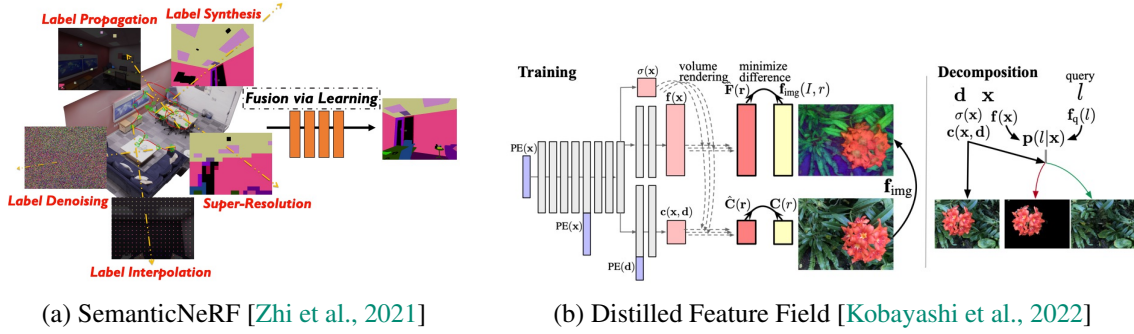
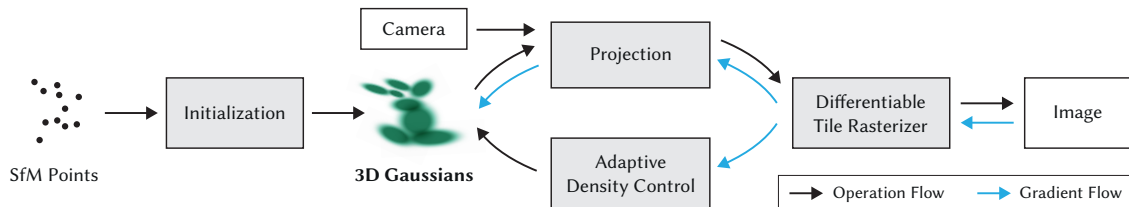


Figure 2.9: Neural Field for semantic scene reconstruction. By using additional semantic head on an MLP, NeRF can output multi-view consistent semantic label/features which can be trained by per-view semantic prediction. The images are from [Zhi et al., 2021, Kobayashi et al., 2022]

2.3.2 3D Gaussian Splatting



In parallel with the rapid growth of NeRF and volume rendering, differentiable rasterization techniques has also been researched [Kopanas et al., 2021, Rückert et al., 2022, Keselman and Hebert, 2022]. These methods use point-based representation such as point cloud [Kopanas et al., 2021, Rückert et al., 2022] or metaball [Keselman and Hebert, 2022], and define differentiable

rendering pipeline using CNN based rasterizer or approximated renderer. In 2023, Kerbl et al. proposed a radically different rendering approach for radiance field capture with 3D Gaussian Splatting [Kerbl et al., 2023] which are rendered via volume-rendering [Zwicker et al., 2002]. Similar to traditional graphics rasterization, 3DGS iterates over the primitives to be rasterized rather than marching along rays.

Although 3DGS employs a rasterization-like strategy, its primitives represent a volume rather than a solid surface. This volumetric representation provides a smoothly varying gradient distribution throughout the entire space, similar to NeRF, making the optimization process easier. Crucially, 3DGS achieves real-time rendering speed by deviating from the rigorous volumetric rendering integral used in NeRF. The key distinction lies in its parameterization: while NeRF relies on the density of the medium to model the physical attenuation of light along a ray, 3DGS parameterizes each primitive with an explicit opacity value. This opacity is a learned parameter used directly in the α -blending process during the tile-based rasterization. This opacity-based approach is a highly effective approximation of the full volumetric rendering integral, enabling high-quality synthesis while avoiding the computational overhead associated with ray marching. This method leverages the natural sparsity of a 3D scene, resulting in a representation that is expressive enough to capture high-fidelity 3D scenes while offering significantly faster rendering with similar training speed to the most efficient volumetric NeRF.

The input to 3DGS pipeline consists of a set of color images with corresponding poses estimated by SFM. In addition, 3DGS takes the sparse point cloud generated during the SFM process, which is used for constructing a set of initial 3D Gaussians as graphical primitives.

Each 3D Gaussian is parameterized by its 3D mean position $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3 \times 3}$, opacity o , and direction-dependent color $\mathbf{c}$ represented by spherical harmonics (SH). Given a scaling matrix $\mathbf{S} \in \mathbb{R}^{3 \times 3}$ and rotation matrix $\mathbf{R} \in \mathbb{R}^{3 \times 3}$, covariance matrix $\boldsymbol{\Sigma}$ is represented as :

$$\boldsymbol{\Sigma} = \mathbf{R} \mathbf{S} \mathbf{S}^T \mathbf{R}^T \quad (2.9)$$

These 3D Gaussians are projected onto the image plane ($z = 1$ plane) via an approximated projection function for differentiable rasterization. Here we describe the projection process.

In the world space, we define a 3D Gaussian function by G_{3D}^w , characterized by its mean $\boldsymbol{\mu}_{3D}^w$ and covariance matrix $\boldsymbol{\Sigma}_{3D}^w$.

$$G_{3D}^w(\mathbf{x}_{3D}^w) = \exp \left\{ -1/2 (\mathbf{x}_{3D}^w - \boldsymbol{\mu}_{3D}^w)^\top (\boldsymbol{\Sigma}_{3D}^w)^{-1} (\mathbf{x}_{3D}^w - \boldsymbol{\mu}_{3D}^w) \right\}. \quad (2.10)$$

We firstly transform 3D Gaussian from the world space to the camera space via a viewing transformation matrix $\mathbf{W}$. The transformation is formulated as:

$$G_{3D}^w(\mathbf{x}_{3D}^w) = \exp \left\{ -1/2 (\mathbf{W}\mathbf{x}_{3D}^w - \mathbf{W}\boldsymbol{\mu}_{3D}^w)^\top (\mathbf{W}\boldsymbol{\Sigma}_{3D}^w \mathbf{W}^\top)^{-1} (\mathbf{W}\mathbf{x}_{3D}^w - \mathbf{W}\boldsymbol{\mu}_{3D}^w) \right\}. \quad (2.11)$$

Let $\boldsymbol{\Sigma}_{3D}^c$ be a camera space 3D Gaussian function, with $\mathbf{x}_{3D}^c = \mathbf{W}\mathbf{x}$, $\boldsymbol{\mu}_{3D}^c = \mathbf{W}\boldsymbol{\mu}_{3D}^w$ and $\boldsymbol{\Sigma}_{3D}^c = \mathbf{W}\boldsymbol{\Sigma}_{3D}^w \mathbf{W}^\top$ as the point, mean and covariance matrix in the camera space, the 3D Gaussian function in camera space G_{3D}^c is formulated as:

$$G_{3D}^c(\mathbf{x}_{3D}^c) = \exp \left\{ -1/2 (\mathbf{x}_{3D}^c - \boldsymbol{\mu}_{3D}^c)^\top (\boldsymbol{\Sigma}_{3D}^c)^{-1} (\mathbf{x}_{3D}^c - \boldsymbol{\mu}_{3D}^c) \right\}. \quad (2.12)$$

Subsequently, the camera-space 3D Gaussian is projected onto the $z = 1$ plane, with the projection function φ . This projection function is approximated by the first order Taylor series expansion:

$$\varphi(\mathbf{x}_{3D}^c) \approx \varphi(\boldsymbol{\mu}_{3D}^c) + \mathbf{J}(\mathbf{x}_{3D}^c - \boldsymbol{\mu}_{3D}^c) \quad (2.13)$$

where $\mathbf{J} = \frac{\partial \varphi}{\partial \mathbf{x}_{3D}^c}(\boldsymbol{\mu}_{3D}^c)$ denotes the Jacobian of the affine approximation of the projective transformation. By applying this local affine approximation, we can transform camera space 3D Gaussian $\boldsymbol{\Sigma}_{3D}^c$ to image space projected onto the $z = 1$ plane. Eq. 2.12 is reformulated as:

$$\begin{aligned} G_{3D}^c(\mathbf{x}_{3D}^c) &= \exp \left\{ -1/2 (\mathbf{J}\mathbf{x}_{3D}^c - \mathbf{J}\boldsymbol{\mu}_{3D}^c)^\top (\mathbf{J}\boldsymbol{\Sigma}_{3D}^c \mathbf{J}^\top)^{-1} (\mathbf{J}\mathbf{x}_{3D}^c - \mathbf{J}\boldsymbol{\mu}_{3D}^c) \right\} \\ &= \exp \left\{ -1/2 (\varphi(\mathbf{x}_{3D}^c) - \varphi(\boldsymbol{\mu}_{3D}^c))^\top (\mathbf{J}\boldsymbol{\Sigma}_{3D}^c \mathbf{J}^\top)^{-1} (\varphi(\mathbf{x}_{3D}^c) - \varphi(\boldsymbol{\mu}_{3D}^c)) \right\}. \end{aligned} \quad (2.14)$$

We can obtain the covariance matrix of the 2D image space Gaussian by skipping the third row and column of $\mathbf{J}\boldsymbol{\Sigma}_{3D}^c \mathbf{J}^\top$. We denote $\mathbf{x}_{2D} = \varphi(\mathbf{x}_{3D}^c)$, $\boldsymbol{\mu}_{2D} = \varphi(\boldsymbol{\mu}_{3D}^c)$ and $\boldsymbol{\Sigma}_{2D} = \mathbf{J}\boldsymbol{\Sigma}_{3D}^c \mathbf{J}^\top$ as the point, mean and 2×2 covariance matrix of G_{2D} in the image space, respectively. Subsequently, we can formulate 2D Gaussian function G_{2D} as:

$$G_{2D}(\mathbf{x}_{2D}) \approx \exp \left\{ -1/2 (\mathbf{x}_{2D} - \boldsymbol{\mu}_{2D})^\top (\boldsymbol{\Sigma}_{2D})^{-1} (\mathbf{x}_{2D} - \boldsymbol{\mu}_{2D}) \right\}. \quad (2.15)$$

We obtain the value of the 2D Gaussian function by querying a pixel $\mathbf{x}_{2D} = \begin{bmatrix} u, v \end{bmatrix}^\top$.

As a result, the final color C of the pixel $\mathbf{x}_{2D}$ is determined through alpha blending of the 3D Gaussians, which are sorted by depth using GPU-accelerated Radix sort:

$$C = \sum_{i \in \mathcal{N}} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (2.16)$$

where $\mathcal{N}$ is the set of Gaussians along the pixel's ray, and α_i is the product of the opacity o_i the pixel-space Gaussian function:

$$\alpha_i = o_i G_{2D}(\mathbf{x}_{2D}) \quad (2.17)$$

The Gaussian parameters are optimized by minimizing the L1 loss and the SSIM loss between the rendered image and the ground truth image. Furthermore, during the optimization process, adaptive densification and pruning are performed to flexibly adjust the number of Gaussians based on their gradients and opacities.

iMODE: Neural Fields and Differentiable Rendering for Real-Time Dense Mapping

Contents

3.1	Introduction	37
3.2	Related Work	38
3.3	Preliminary: Dense Reconstruction via Classical Depth Map Integration . .	40
3.4	System Overview	41
3.4.1	Localization Process	41
3.4.2	Semi-Dense Mapping Process	41
3.4.3	Dense Reconstruction Process	42
3.5	3D Scene Representation	42
3.5.1	Network Architecture	42
3.5.2	Neural Field Optimization	43
3.5.3	View-based Adaptive Ray Sampling	44
3.6	Evaluation	45
3.6.1	Dense Scene Reconstruction: Quantitative Results	46
3.6.2	Dense Scene Reconstruction: Qualitative Results	48
3.6.3	Multi-modal Data Fusion	50
3.7	Conclusions	52
3.8	Limitations	53

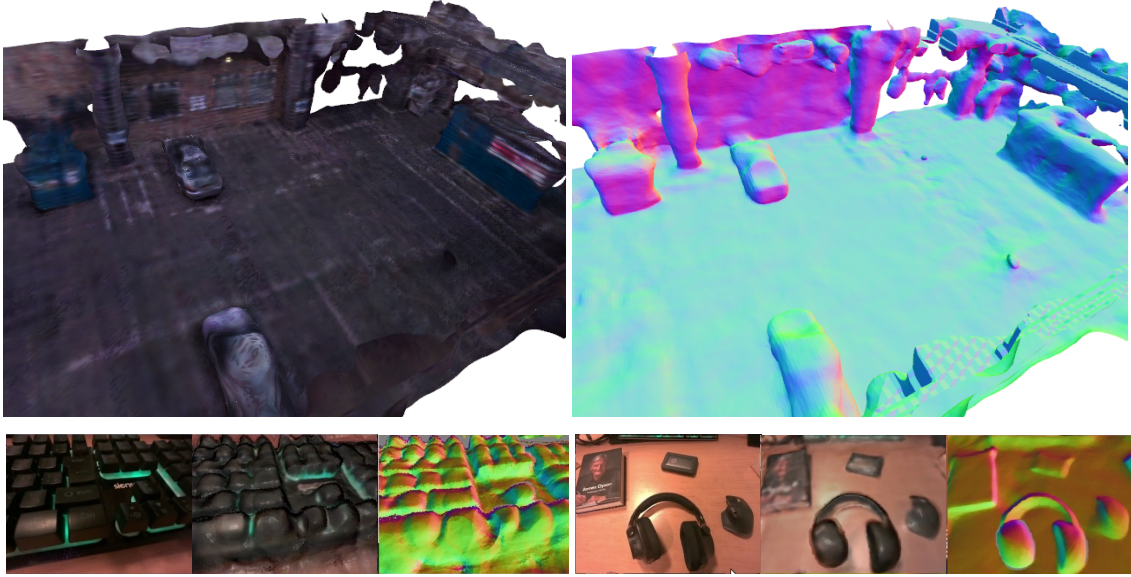


Figure 3.1: Examples of real-time dense monocular reconstructions. (top) 3D mesh and its normal map of a large-scale parking lot. (bottom) Small object reconstruction examples. We show keyframe image and the rendering of shaded color and normal from the network.

This chapter explores a potential of differentiable rendering for dense mapping/SLAM task by building and evaluating a real-time dense monocular mapping framework based on NeRF [Mildenhall et al., 2020]. First, we briefly review the classical multi-stage dense mapping approach by proposed in CodeMapping [Matsuki et al., 2021] paper. It achieves accurate reconstruction but requires multiple separate hand-designed algorithms that lack guaranteed consistency across modules. Motivated by these limitations, we then introduce differentiable rendering into the mapping pipeline and present iMODE [Matsuki et al., 2023], a NeRF-based dense mapping system which only requires monocular rgb images. iMODE employs a dense continuous radiance field represented by a Multi-Layer Perceptron (MLP) together with sparse visual SLAM, demonstrating accurate real-time 3D reconstruction and continuous scene completion. Because it does not rely on prior training, iMODE can flexibly adapt to arbitrary scales and structures at runtime, even handling strong specular reflections. We showcase its performance on various scenes, from small indoor environments to large outdoor spaces. Furthermore, iMODE can straightforwardly incorporate additional inputs such as learned depth priors or semantic labels, demonstrating the flexibility and extensibility of the end-to-end optimization via differentiable rendering.

3.1 Introduction

To be effective for intelligent embodied devices, a visual SLAM system must construct an accurate, robust, real-time 3D map that supports localization, safe navigation, and tasks such as object recognition and physical interaction. Consequently, the 3D model should be efficient, scalable, and capable of inferring properties in unobserved areas. To address these requirements, we propose a novel method, **iMODE**, which combines the accuracy and robustness of sparse landmark-based visual SLAM systems with a neural field to incrementally obtain a dense, continuous 3D reconstruction from a monocular camera in real time.

Inspired by iMAP [Sucar et al., 2021], we represent the geometry and appearance of the 3D scene with a Neural Field. Recent advances in Neural Field representations have shown that a small multi-layer perceptron can be optimized, from scratch, to accurately represent the geometry and appearance of a 3D scene [Mildenhall et al., 2020, Yariv et al., 2021a]. Additionally, the smoothness priors inherent in the structure of the MLP allow these networks to make watertight reconstructions from partial input data without any pre-training. iMAP showed how these neural fields could be used in an accurate, real-time, dense SLAM system, but relied on a depth camera, meaning it was limited to certain depth ranges and operating conditions.

In our system, the incoming stream of RGB images is fed into an off-the-shelf sparse SLAM system that creates and tracks against keyframe images. A set of sparse but accurate depth estimates are generated for these keyframes using a semi-dense mapping system. In a parallel thread during live operation, the weights of the MLP are optimized to minimize the photometric error between the observed and rendered keyframe images, as well as the geometric distance between the rendered depth and the semi-dense depth estimates. To further improve the depth reconstruction, we also include a prior on surface normals. Our system can run at more than 30fps on a single desktop CPU/GPU system.

As our system only uses monocular images and no pre-training, we are capable of reconstructing scenes from vastly different domains and scales. We perform exhaustive quantitative evaluations of our system on a number of standard benchmark datasets and show that iMODE achieves comparable performance to state-of-the-art learning-based methods; however, while these learning-based methods only perform well when tested on domains close to their training data, iMODE’s performance remains consistent across all datasets. We also show a number of qualitative results of our system on a wide variety of scenes including small complex objects, room-scale indoor environments, and large outdoor scenes. Furthermore, we demonstrate that our approach is very

general and modular, and can easily be extended to include semantic information, multi-modal sensor fusion, and dense depth priors. Example reconstructions generated by iMODE are provided in Fig. 3.1.

In summary, the key contributions are:

- The first real-time monocular dense mapping system using a neural field representation of the scene.
- Accurate and complete dense reconstructions of scenes from arbitrary domains and with arbitrary scales.
- A demonstration of how our system can be flexibly extended to include semantic information, dense depth priors and multi-modal sensor fusion.

3.2 Related Work

SLAM with a single camera is attractive for applications due to minimal hardware requirements and flexibility, and there has been steady progress since early real-time methods [Davison et al., 2007, Klein and Murray, 2007] which built sparse feature maps to enable accurate camera tracking. Similar methods are now widespread in industry, and are also well represented by open source systems such as ORB-SLAM [Mur-Artal and Tardós, 2017b].

A SLAM system becomes even more useful when its output is not just localization but also detailed scene reconstruction, and rising processing power in particular made it feasible to build real-time systems which could build dense scene models. In DTAM [Newcombe et al., 2011c], an efficient regularized cost volume estimation technique running on a GPU enabled the RGB information from many frames to be fused into detailed dense depth maps, which could be put together into a scene mesh good enough for camera tracking in small scenes. LSD-SLAM [Engel et al., 2014] showed that by focusing similar dense reconstruction methods only on high gradient pixels, a highly scalable semi-dense reconstruction could be achieved. However, it remains the case that real-time fully dense scene reconstruction from only monocular input is both computationally demanding and somewhat fragile, since the regularization assumptions needed to regularize low-texture areas well are often specific to certain scenes. Therefore, most real-time visual SLAM systems that are close to being useful for applications have required additional sensing, particular from RGB-D cameras [Whelan et al., 2015b, Schöps et al., 2019, Dai et al., 2017b, Kerl

et al., 2013b]. Note though that depth cameras have their own limitations, especially limited depth measurement range with both minimum and maximum bounds.

Deep learning offers new tools and promise for monocular SLAM, with the possibility to move away from hand-designed regularizers and densely reconstruct scenes in a data-driven manner. Several ways to use neural network representations in SLAM have been proposed, and the main approach has been to pre-learn a representation from a dataset with some ability to use code optimization or similar at runtime to fit this to a live scenario [Bloesch et al., 2018, Sucar et al., 2020, Matsuki et al., 2021, Tateno et al., 2017, Yang et al., 2020]. It has often been found, however, that it is difficult to get reliable and general high performance from such methods apart from in quite specific domains such as on-road driving, because it is difficult to train on data which has the generality to span more varied scenes and camera motions.

A different way to use neural networks in scene reconstruction is to train a network from scratch to represent one specific scene, and to take advantage of priors which are inherent in the structure of the network itself rather than datasets. This approach has become well known in very high performance offline methods for scene and radiance reconstruction [Niemeyer et al., 2020b, Yariv et al., 2021a, Mildenhall et al., 2020], where it was shown that a Multi-Layer Perceptron (MLP) can represent scenes accurately using neural coordinate fields. While these methods were originally designed for offline use, and took hours or days to optimize for a single scene, there has been much recent interest in speeding them up both for training and rendering. iMAP [Sucar et al., 2021] was the first system which showed that a neural field representation could be trained from scratch in real-time as the only scene model in a real-time SLAM system. The keys to real-time use in iMAP were the use of RGB-D camera input, a reduced MLP design and uncertainty-guided keyframe and ray sampling within a PTAM-inspired system architecture of parallel tracking and mapping. iMAP is the primary inspiration for our system. We show that similar performance can be obtained without depth input by using a sparse monocular system to offer sparse depth measurements and camera tracking. Besides the reduction in hardware complexity required, our monocular system has the benefit of being able to work across a large range of scales inaccessible to depth cameras.

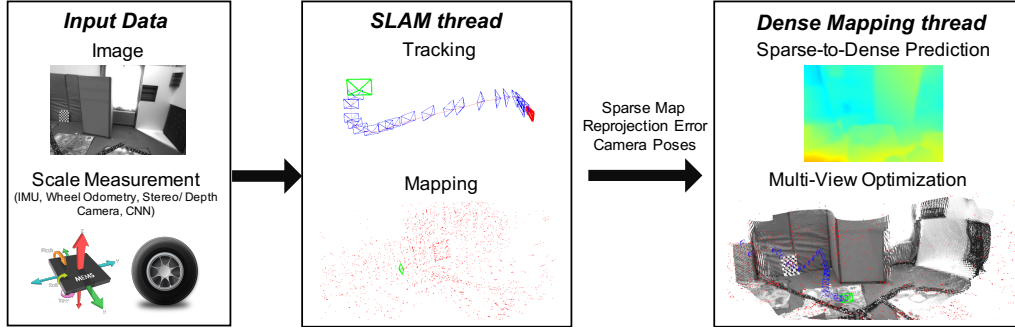


Figure 3.2: System overview of CodeMapping [Matsuki et al., 2021]: the dense mapper receives sparse point sets and camera poses to run dense prediction and multi-view optimization.

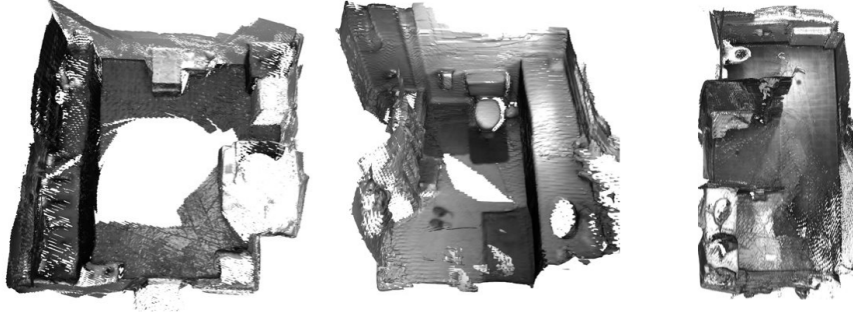


Figure 3.3: TSDf Fusion results of CodeMapping

3.3 Preliminary: Dense Reconstruction via Classical Depth Map Integration

This section provides an overview of the classical dense reconstruction pipeline by reviewing our paper CodeMapping [Matsuki et al., 2021], which estimates an accurate and globally consistent dense map based on the commonly used multi-stage fusion approach as shown in Fig 3.2. Using the camera pose and sparse points from the front-end camera tracking system, CodeMapping first performs per-view depth completion using Neural Network, then local multi-view optimization, and finally global depth map fusion via TSDf integration. This results in an accurate reconstruction as shown in Fig 3.3; however, it relies on different hand-designed algorithms for each stage, which does not guarantee consistency between modules. This can complicate the overall algorithm flow and may lead to suboptimal solutions.

3.4 System Overview

Our system runs three loosely-coupled processes concurrently. The first is localization, which uses an off-the-shelf sparse landmark-based SLAM system to provide real-time camera pose estimates. The second process uses these poses to run semi-dense mapping for selected keyframes. Both localization and semi-dense mapping systems run on the CPU. The camera poses and semi-dense maps are passed to the dense reconstruction system that runs on a GPU and optimizes the weights of an MLP representing a 3D neural field. Both color and depth rendering losses are minimized to create a watertight reconstruction of the scene geometry and appearance. Each of these sub-systems is discussed in detail below; see the overview in Fig. 3.4.

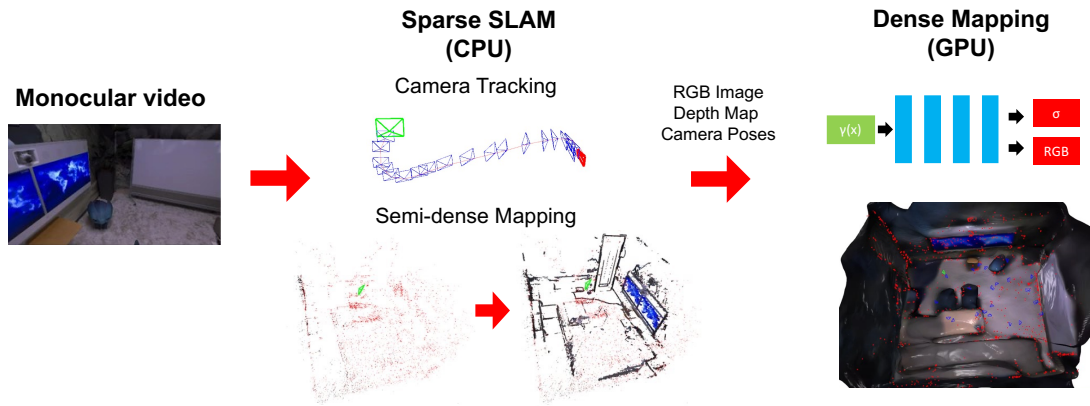


Figure 3.4: iMODE system overview: from a monocular input stream sparse SLAM is used for camera localization, a semi dense point cloud is then obtained in high gradient regions, which is used along with photometric errors for obtaining a full dense reconstruction through training an MLP neural field representation.

3.4.1 Localization Process

We use the sparse landmark-based SLAM system ORB-SLAM [Mur-Artal and Tardós, 2017b] for localization, which is highly accurate and robust, and comfortably runs at frame rate on a CPU. We use a subset of the keyframes selected by ORB-SLAM in our semi-dense mapping and dense reconstruction processes.

3.4.2 Semi-Dense Mapping Process

As shown in iMAP, the accuracy and convergence speed of a neural-field-based reconstruction system can be dramatically improved by directly supervising the geometry via depth rendering

during the online training process. Unlike iMAP, however, iMODE does not use a depth camera, so any depth measurements used must come from monocular multi-view stereo (MVS) methods. DS-NeRF [Deng et al., 2021] showed that sparse depth supervision speeds up the convergence with fewer frames, thus we further increase supervision signals using the CPU-based real-time monocular semi-dense mapping system proposed in [Mur-Artal and Tardós, 2015]. This system exploits high-gradient regions to produce semi-dense depth maps for a set of keyframes and camera poses obtained from the localization procedure. This process runs in a separate thread and it does not delay the main localization system.

3.4.3 Dense Reconstruction Process

In the dense reconstruction process, the weights of the MLP representing the neural field are optimized against a subset of the keyframe images and corresponding semi-dense depth maps produced by the first two processes. To keep the system running at real time, we limit the optimization window to a fixed size of 7 representative keyframes. To balance adding new information to the map without losing information from previously viewed regions, the representative keyframes consist of the two most recently added keyframes and 5 older keyframes selected at random. This windowed optimization is run for 20 iterations each time a new keyframe is received. Details of the network architecture, loss formulation and ray sampling procedure are provided in the next section.

3.5 3D Scene Representation

3.5.1 Network Architecture

The neural field representing the scene is modeled by an MLP, F_θ , with weights θ . F_θ consists of 4 hidden layers of size 256 and a color and a density head. The 3D coordinates of a query point, $\mathbf{p}$, are mapped to the frequency domain using the positional encoding functions γ_{low} and γ_{high} (mapping to higher and lower sets of frequencies, respectively), and then passed as input to F_θ along with the viewing direction, $\mathbf{v}$. F_θ then predicts the density, σ , and color, $\mathbf{c}$, corresponding to the query point and viewing direction:

$$F_\theta(\gamma(\mathbf{p}), \mathbf{v}) = (\mathbf{c}, \sigma) \quad (3.1)$$

Our network architecture has two notable distinctions to the one in iMAP: (i) *View Dependency*: Because our system relies more heavily on photometric consistency we model view dependant effects such as specularities by concatenating the viewing direction to the color head of the network (following the design of NeRF). (ii) *Frequency Separation*: As natural scenes tend to present higher frequencies in texture than in geometry, we introduce this bias into our network by feeding a lower frequency embedding as the initial input (avoiding jaggy artifacts in geometry), and a higher frequency embedding only to the color head. A diagram of our proposed network architecture is provided in Fig. 3.5. Fig. 3.6 shows the influence of frequency separation by comparing the rendering quality after running multi-view optimization.

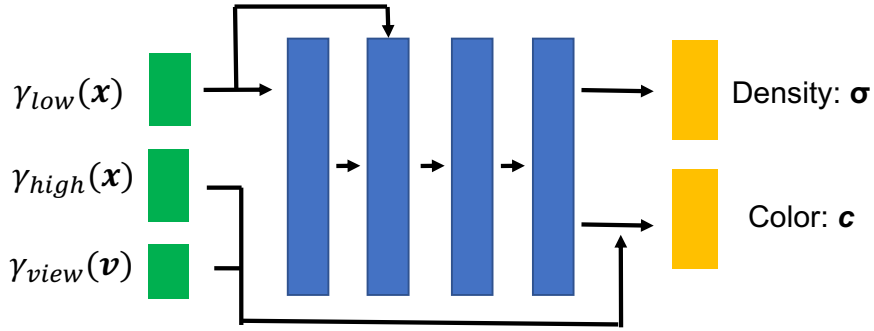


Figure 3.5: Neural field represented through a 4-layer MLP, low frequency embedding is used for geometry while a high frequency one along with the viewing direction are used to reconstruct high detailed texture with view-dependant effects.

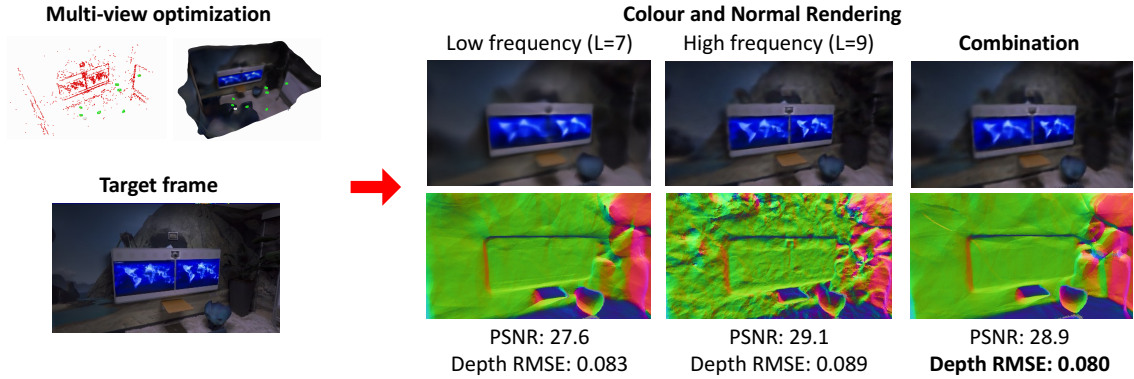


Figure 3.6: The effect of positional encoding frequency separation. By using low frequency for geometry input and high frequency for color head separately, the MLP can output both noiseless surface reconstruction and detailed color. L denotes the frequency of axis-aligned positional encoding [Mildenhall et al., 2020].

3.5.2 Neural Field Optimization

The MLP is optimized with respect to the semi-dense depth estimates and color of the keyframes by volume renderings computed by combining the queried network values along the 3D ray of

pixel $[u, v]$:

$$\hat{D}[u, v] = \sum_{i=1}^N w_i d_i, \quad \hat{D}_{var}[u, v] = \sum_{i=1}^N w_i (\hat{D}[u, v] - d_i), \quad \hat{I}[u, v] = \sum_{i=1}^N w_i c_i \quad (3.2)$$

where $\hat{D}$ and $\hat{D}_{var}$ are the rendered depth and depth variance respectively, and $\hat{I}$ is the rendered color image. The weighting $w_i = o_i \prod_{j=1}^{i-1} (1 - o_j)$ is the ray-termination probability of the i th sample along the ray at depth d_i , where $o_i = 1 - \exp(-\rho_i \delta_i)$ is the occupancy activation function with the inter-sample distance $\delta_i = d_{i+1} - d_i$. We minimize the color and depth consistency loss for a small number of pixels from selected keyframes.

For M pixel samples and W keyframes, the photometric loss, L_p , is the sum of the difference between rendered and measured color values, and the geometric loss, L_g , is the difference between the depth rendering and the semi-dense depth estimates, normalized by $\hat{D}_{var}$:

$$L_p = \frac{1}{M} \sum_{i=1}^W \sum_{s_i} |I_i[u, v] - \hat{I}_i[u, v]|, \quad L_g = \frac{1}{M} \sum_{i=1}^W \sum_{s_i} \frac{|D_i[u, v] - \hat{D}_i[u, v]|}{\sqrt{\hat{D}_{var}[u, v]}}. \quad (3.3)$$

We also apply a surface smoothness loss to the the predicted normals as proposed in UNISURF [Oechsle et al., 2021]. Given $\mathbf{n}(\mathbf{x}_t)$, the predicted unit normal vector at the ray-termination point, $\mathbf{x}_t$, we enforce the nearby normals to be similar:

$$L_n = \frac{1}{M} \sum_{i=1}^W \sum_{(u,v) \in s_i} \|\mathbf{n}(\mathbf{x}_t) - \mathbf{n}(\mathbf{x}_t + \epsilon)\|, \quad (3.4)$$

where ϵ is a small random perturbation in 3D.

We minimize the weighted loss sum with constants λ_p and λ_n (5.0 for both):

$$\arg \min_{\theta} \frac{1}{M} \sum_{i=1}^W \sum_{(u,v) \in s_i} L_g + \lambda_p L_p + \lambda_n L_n \quad (3.5)$$

We sample 200 points from pixels where semi-dense depth values are available (mainly corners or edges) for full supervision and another 200 points from randomly selected pixels for color supervision.

3.5.3 View-based Adaptive Ray Sampling

Methods that use neural fields for novel view synthesis tasks typically sample along a fixed length of the ray as the distance to the surface is approximately constant in these applications. In a robot navigation context, however, the distance to the surface changes often and using a fixed length

for ray sampling can be inefficient. Therefore, we use an adaptive interval based on the scale of estimated pixel depth. Let $d_{\min}$ and $d_{\max}$ be the per-frame minimum and maximum sampling distances along the ray, $\mathbf{D}_0$ be the depth map of initial frame in ORB-SLAM, and $\mathbf{D}$ be the depth map of the target frame. We change the ray sampling distance from the initial interval value $(d_{\max,0}, d_{\min,0})$ based on the scale of the view, calculated by the median value of sparse depth map:

$$(d_{\max}, d_{\min}) = \frac{\text{Median}(\mathbf{D}_0)}{\text{Median}(\mathbf{D})} (d_{\max,0}, d_{\min,0}). \quad (3.6)$$

Fig. 3.7 shows the effect of adaptive ray sampling, which helps to recover fine detail by providing denser training signals around the surface. Without it we observe spotted artifacts due to wide sampling steps.

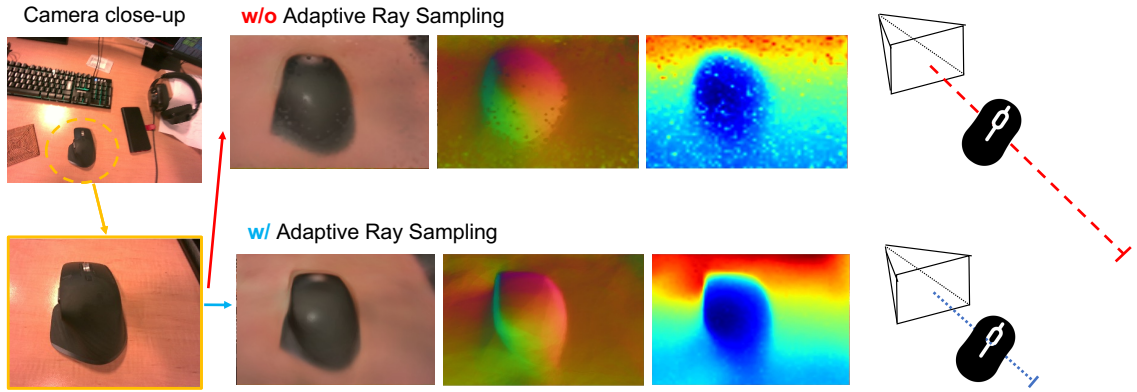


Figure 3.7: By adaptively changing the ray-sampling interval based on the depth, we can sample the points around the object surface more efficiently, resulting in better reconstructions. From left to right, the middle images show the rendered color, normals and depth with and without adaptive sampling.

3.6 Evaluation

We demonstrate the flexibility of our system by evaluating it in a variety of scenes from very small scale indoor scenes to large-scale outdoors. Our monocular system can reconstruct dense geometry in these domains without any prior training. We also show that it can flexibly make use of prior information (such as domain-specific network depth prediction or multi-modal sensor inputs) if available to improve performance. We also demonstrate the dense semantic label fusion. All experiments use a desktop system with an Intel Core i9 12900K 3.50GHz CPU and an NVIDIA Geforce RTX 3090 GPU. As in iMAP, the meshes and view renderings we show are for visualization and evaluation purposes and do not form part of our system. Sparse SLAM and

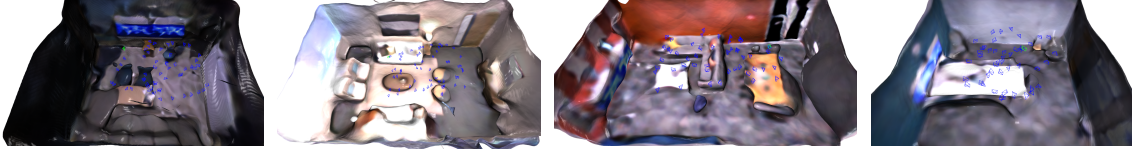


Figure 3.8: 3D reconstruction results on the Replica dataset: our system generates dense and watertight mesh models in real-time from monocular video.

the semi-dense mapper are implemented with C++ and dense mapping in Python (PyTorch). The communication is done via ROS. The input image stream is 15hz for benchmark datasets and 30hz for our demos. For low frequency encoding, we use axis-aligned encoding with $L_{\text{low}} = 7$ or 8 for our demos and gaussian embedding [Tancik et al., 2020] with $\sigma = 35$ and size 123 for dataset evaluation. We use axis-aligned encoding with $L_{\text{high}} = 9$ and $L_{\text{view}} = 6$ for higher frequency and view direction encoding respectively. We do not use a view direction on Replica dataset because the dataset does not contain a specular reflection. 3D coordinates are normalized by $\frac{1}{10}$.

3.6.1 Dense Scene Reconstruction: Quantitative Results

3D Scene Reconstruction We evaluate 3D reconstruction quality against the ground truth meshes of the Replica dataset [Straub et al., 2019]. As in [Sucar et al., 2021], we sample 200,000 points from both ground truth and reconstructed meshes and evaluate the accuracy, completion and completion ratio. We first run Sim(3) alignment between the monocular SLAM trajectory and groundtruth trajectory and then apply the Sim(3) matrix to the SLAM mesh for alignment. As Table 3.1 shows, our monocular system achieves a completion ratio similar to RGB-D methods with less than 20cm error. Figure 3.8 shows 3D meshes from our system.

		r0	r1	r2	o0	o1	o2	o3	o4	avg
iMAP (RGBD)	Acc. [cm] ↓	3.58	3.69	4.68	5.87	3.71	4.81	4.27	4.83	4.43
	Comp. [cm] ↓	5.06	4.87	5.51	6.11	5.26	5.65	5.45	6.59	5.56
	Comp. Ratio [<5cm %] ↑	83.9	83.4	75.5	77.7	79.6	77.2	77.3	77.6	79.0
Ours (Monocular)	Acc. [cm] ↓	7.4	6.4	9.3	6.6	11.8	11.4	9.4	8.0	8.78
	Comp. [cm] ↓	13.5	10.1	19.2	9.7	17	14.5	11.8	15.4	13.9
	Comp. Ratio [<5cm %] ↑	38.7	46.1	36.1	49.3	30.1	29.8	36.0	31.0	37.1
	Comp. Ratio [<20cm %] ↑	79.3	85.8	73.1	88.7	75	74.7	81.0	74.8	79.1

Table 3.1: Reconstruction results for 8 indoor Replica scenes.

Depth Evaluation We compare our system to other dense depth estimation methods by evaluating the 2D depth map quality on public benchmark datasets [Sturm et al., 2012, Burri et al., 2016, Straub et al., 2019] (Table 3.2). Following [Tateno et al., 2017], we evaluate the accuracy and density of the depth map by computing the percentage of depth values with an error less than 10% of the ground truth depth. We compare against two real-time multi-view stereo algorithms

with pure geometric formulation (Plane Sweep Stereo [Gallup et al., 2007] and REMODE [Pizzoli et al., 2014]), a learning-based multi-view dense reconstruction method (NeuralRecon [Sun et al., 2021]) and a single-view depth completion method (SparseToDense [Ma and Karaman, 2018]). To generate pseudo-ground-truth depth maps for the EuRoC dataset, we first generate a global mesh model using the provided stereo camera data and ground truth camera poses and render it to each view. We believe that this is sufficient to demonstrate the domain generalization capability of iMODE. For Plane Sweep Stereo and REMODE, we use 5 neighboring ORB-SLAM keyframes per image for dense stereo. For NeuralRecon and SparseToDense, we use the public models pre-trained on ScanNet [Dai et al., 2017a] and NYU dataset [Silberman et al., 2012] respectively. We provide ground truth camera poses to NeuralRecon to run its own keyframe selection. As Fig 3.9 shows our method can smoothly reconstruct scenes thanks to the inherent prior in MLP and multi-view consistent single global scene model. Learning-based methods, in particular NeuralRecon can vary a lot in performance depending on the test dataset. This is because NeuralRecon is trained on small room-scale data (ScanNet) and learns a strong prior regarding the object domain or scene scale. This cannot easily be applied to different scene domains such as the large scale hall in the EuRoC dataset.

	TUM RGB-D			EuRoC MH		Replica				Avg.
	fr1/desk	fr2/xyz	fr3/office	MH01	MH02	office0	office1	room0	room1	
iMODE	80.7	85.7	61.9	63.6	66.6	87.4	80.6	88.1	91.1	78.4
Plane Sweep Stereo [Gallup et al., 2007]	39.3	41.5	38.7	36.4	30.5	60.4	45.8	52	47.8	43.6
REMODE [Pizzoli et al., 2014]	25.62	24.28	7.83	13.3	11.1	13.5	14.1	22.8	20.1	16.9
NeuralRecon [Sun et al., 2021]	84.7	79.3	76.2	1.24	2.09	18.4	26.4	0.99	2.1	32.4
SparseToDense [Ma and Karaman, 2018]	24.4	32.3	32.0	35.9	35.8	52.6	42.3	56.7	52.9	40.5

Table 3.2: Depth map evaluation

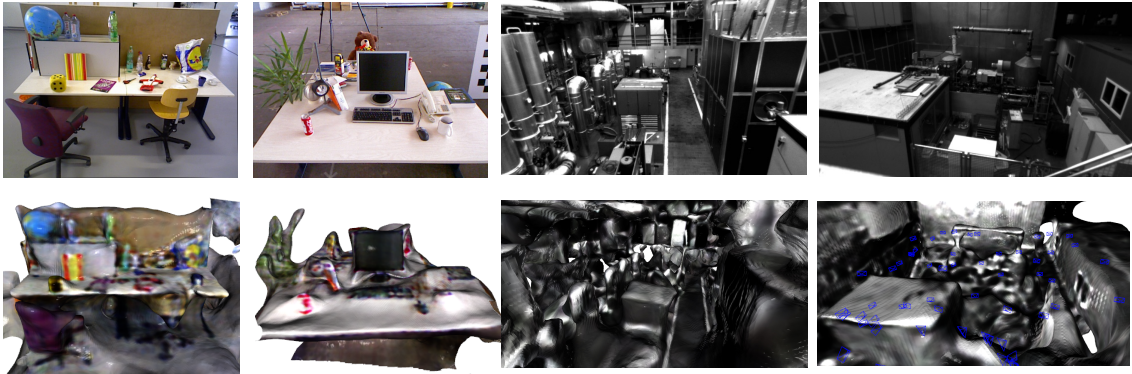


Figure 3.9: Example images (top) and reconstructed 3D meshes (bottom) on TUM and EuRoC dataset. Meshes are phong-shaded to highlight the geometry.

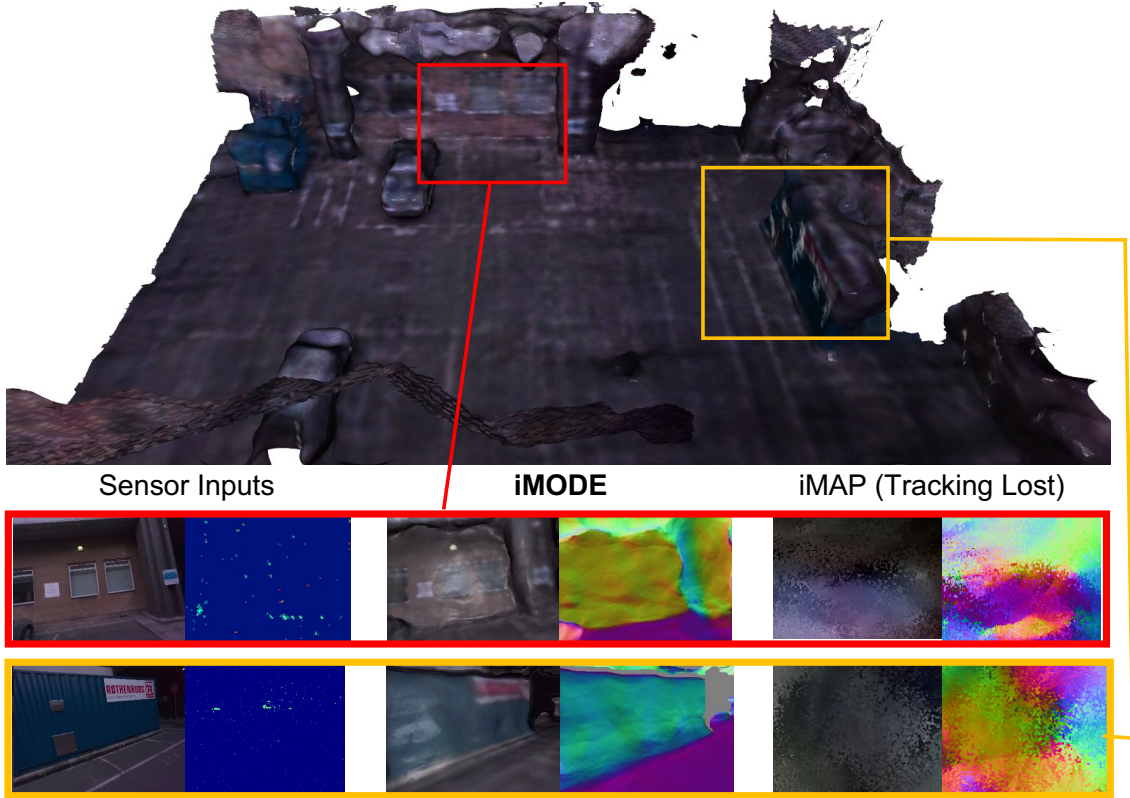


Figure 3.10: Large-scale reconstruction: iMODE can reconstruct outdoor scenes measuring tens of metres where iMAP which relies on depth camera input fails.

3.6.2 Dense Scene Reconstruction: Qualitative Results

We further demonstrate the uniqueness of our method by using three different types of recorded scenes. Please also check the video submission to see the real-time incremental reconstruction.

Large-scale Outdoor Scene: As Fig. 3.10 shows, we can reconstruct a dense 3D model in real-time even in a large-scale outdoor parking lot. A commodity ToF sensor fails to measure the depth in this scene.

Small-Scale Scene: Another uniqueness of our method is we can also reconstruct small, detailed objects, which is also out of the measurement range of depth sensors. Fig. 3.11 is an example of mechanical keyboard reconstruction. While the depth from the RGB-D camera has huge holes and an RGB-D based SLAM system (iMAP) completely fails, our purely monocular system is still able to track and reconstruct the detail of the each individual key.



Figure 3.11: Small-scale reconstruction: While commodity depth sensors cannot measure surfaces closer than 10cm and iMAP completely fails at this scale, our purely monocular system can reconstruct each key of the mechanical keyboard.

Specular Reflection: Including the viewing-direction vector allows iMODE to model specularities which break the inter-frame photometric consistency. Fig. 3.12 demonstrates how our method can handle a strong specular reflection on a desk and accurately reconstruct the dense geometry, while a system that does not include the viewing direction corrupts the geometry in an attempt to account for the lack of color consistency.

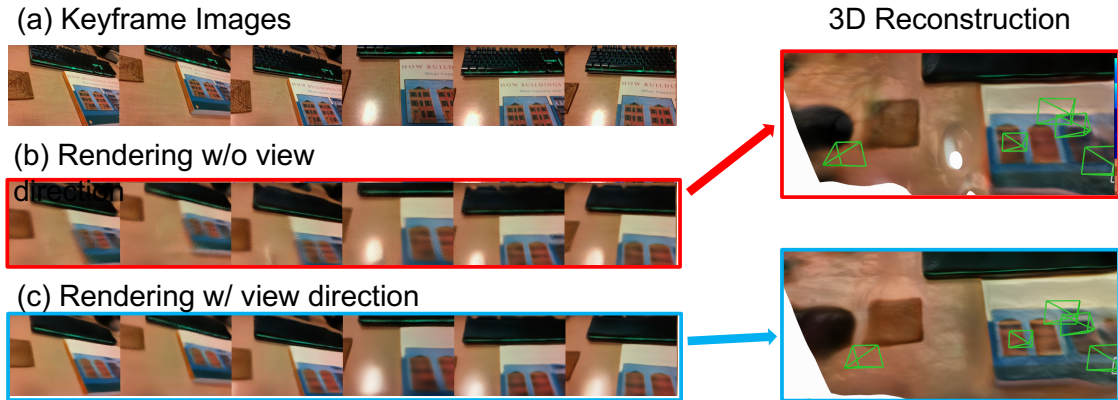


Figure 3.12: View dependent effect: The use of view-direction vector allows our system to deal with strong specular reflection.

Analysis of Supervision Signals We investigate the effect of semi-dense mapping and different loss terms by running our system on the Replica dataset. As Fig. 3.13 shows, semi-dense depth maps provide more geometric constraints, speeding up convergence and increasing reconstruction accuracy. We also analyzed the effect of depth and color supervision independently and found that depth supervision enables fast coarse geometry convergence and color supervision helps recover

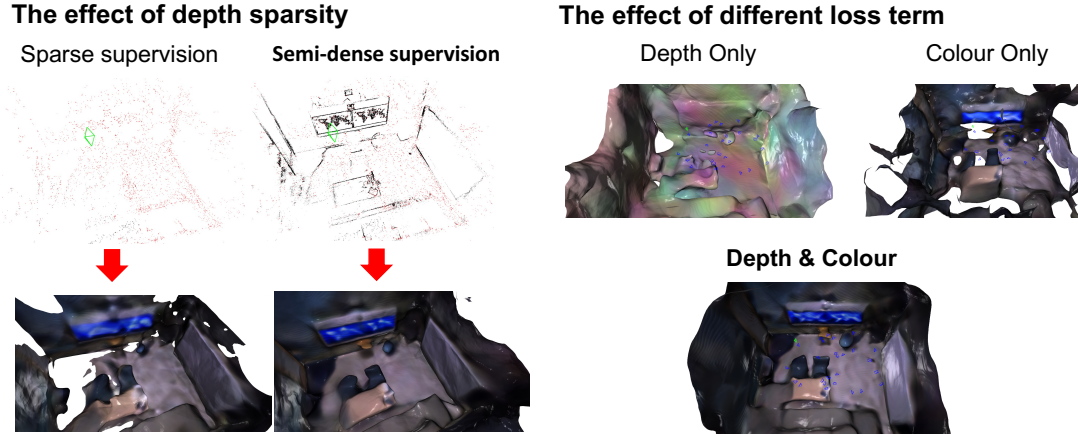


Figure 3.13: A comparison between different supervision signals. (left) Depth sparsity analysis. (right) Loss function analysis. The combination of semi-dense depth and color supervision generates the best reconstruction.

the detail where a depth signal from MVS is not available.

3.6.3 Multi-modal Data Fusion

While we discussed the domain-agnostic performance of our system in the previous sections, our method can easily make full use of an available domain-specific depth prior. All the data fusion examples shown in Fig. 3.14 are easily done without any major modification to our system. If a dense depth prior is available, we do not need to change the MLP architecture, but simply add more supervisory signals to geometric loss function. For semantic label fusion, we only need to add a small semantic prediction head to the MLP and a semantic loss function during training. We will explain each in detail below.

Predicted Dense Depth Fusion

We show that our system can use and improve upon a dense depth prior provided by a CNN. For this experiment, we trained an off-the-shelf depth-completion network [Ma and Karaman, 2018] on ScanNet and evaluate it on sequences from the test split. We run RGB-D ORB-SLAM to get a metric sparse depth map and camera poses in real-time and then apply depth completion for each keyframe to obtain a dense depth prior. Note that we do not use dense depth information for dense mapping. We sample depth from pixels with sparse SLAM measurements and the dense CNN-based predictions equally. As Fig 3.14 shows, the learned priors have inaccurate depth values but our method can fuse them to coherent 3D models. The reconstruction can be further improved using the uncertainty of a dense depth prior as proposed in [Roessle et al., 2021, Wei et al., 2021].



Figure 3.14: The use of dense depth prior prediction. (A, C) Back projection of depth map predicted by depth-completion network. (B, D) 3D mesh reconstruction by our system using the dense depth prior.

	0084_00	0100_00	0356_02	0406_02	0535_00
iMODE + SparseToDense [Ma and Karaman, 2018]	0.279	0.061	0.207	0.068	0.21
iMODE (Sparse input only)	1.08	0.23	0.689	0.22	0.457
SparseToDense [Ma and Karaman, 2018]	1.19	0.065	1.17	0.151	0.829

Table 3.3: Depth RMSE in metres on ScanNet dataset.

(a) Semantic View Synthesis			(b) Semantic Label Denoising		
mIoU	Avg Acc	Total Acc	Noize Ratio	Label type	mIoU Avg Acc Total Acc
0.776	0.863	0.938	50%	Input	0.182 0.286 0.538
				Denoized	0.676 0.77 0.918
			90%	Input	0.048 0.132 0.169
				Denoized	0.556 0.640 0.896

Table 3.4: Quantitative evaluation of semantic label fusion on Replica dataset.

Semantic Label Fusion

Our monocular mapping system allows for the fusion and propagation of partially observed semantic labels. Following [Zhi et al., 2021], we add a semantic prediction head to the MLP to predict semantic labels at a queried 3D point and supervise it by comparing the rendered semantic labels to the semantic labels of the keyframes. For the semantic view synthesis task, we provide ground truth semantic labels to keyframes and evaluate novel view label synthesis performance after the SLAM process finishes. For the semantic label denoising task, we provide labels with pixel-wise random noise and evaluate the denoising effect by multi-view label fusion. Table 3.4a and 3.4b shows quantitative results and Fig 3.15 shows qualitative results on the 3 sequences from Replica (office0, room0, room1). consistent semantic mesh model and is able to denoise partial input label data, achieving 10x more accuracy as measured by mIoU.

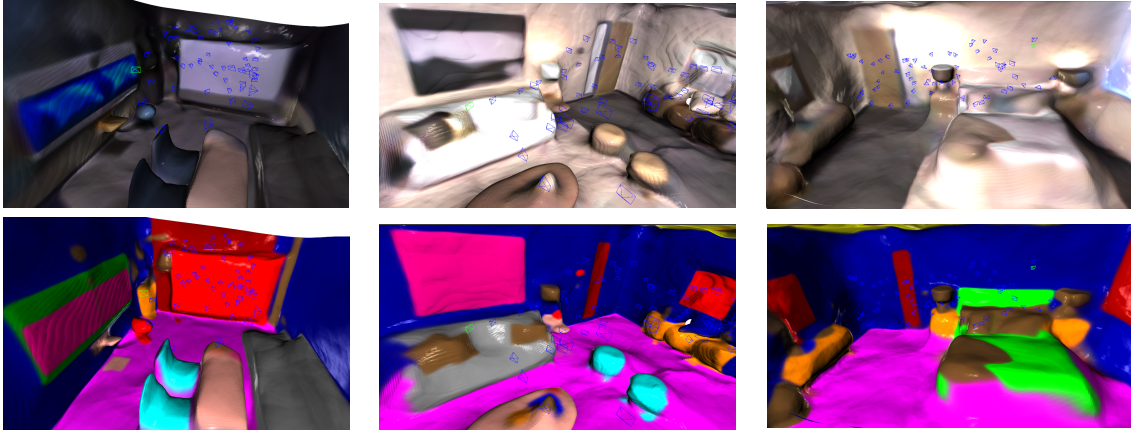


Figure 3.15: Semantic label fusion examples on Replica dataset.

Sensor Fusion

We also show iMODE can use multi-modal sensor inputs commonly used in robotics applications such as stereo-inertial data. The sparse SLAM module [Campos et al., 2021b] works as an interface to the multi-modal sensor inputs and sends more accurate camera poses and dense depth maps to dense mapping process. Fig. 3.16 is a qualitative result on EuRoC dataset.

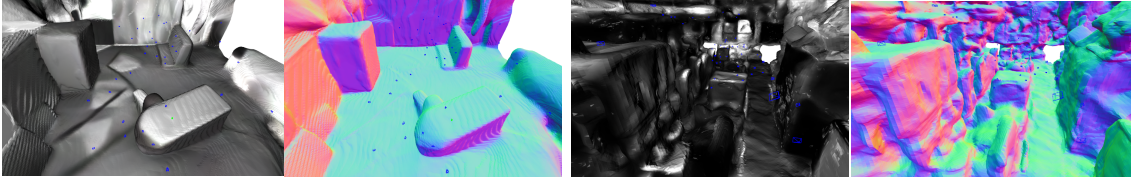


Figure 3.16: 3D mesh and normal map result by using stereo-inertial data on EuRoC dataset. Our method can effectively use multi-modal inputs thanks to the use of a sparse SLAM system as an interface to the sensor data.

3.7 Conclusions

We have presented iMODE, the first real-time dense monocular SLAM system using a neural field. Our system achieves key advantages by requiring no prior training and operating exclusively with a passive monocular camera. The absence of prior training eliminates domain dependency often associated with learned methods and allows iMODE to adapt instantly to any novel scene at runtime without fine-tuning. Furthermore, operating without active sensors (e.g., LiDAR or structured light cameras) ensures cost-effectiveness and simplicity of deployment, while also providing domain flexibility by avoiding operational constraints like limited range and susceptibility to reflective surfaces. Our evaluation shows the flexibility of our method to achieve complete and watertight reconstructions over multiple domains, including small objects and large outdoor

scenes. Our system is also capable of harnessing extra domain-specific prior information that may be available. In the future, we hope to further improve the accuracy of the method over multiple scales, using adaptive positional encoding.

3.8 Limitations

- The system relies heavily on an external sparse tracking module, preventing a fully end-to-end SLAM pipeline. A promising direction for future work is to tightly integrate pose tracking with surface reconstruction.
- Due to the world-centric nature of the current field representation, the system cannot handle large pose updates (such as those arising during loop closure), limiting its applicability to controlled environments.

NEWTON: Egocentric Neural Fields for Large-scale Mapping

Contents

4.1	Introduction	56
4.2	Related Work	58
4.3	Method	59
4.3.1	Camera Tracking	60
4.3.2	Scene Representation	60
4.3.3	Dynamic Neural Field Allocation and Training	64
4.4	Evaluation	66
4.4.1	Experimental Setup	66
4.4.2	On-the-fly Mapping	67
4.4.3	Baseline Comparison	68
4.4.4	Ablation	71
4.5	Conclusion	73
4.6	Limitations	73

This chapter details our work [Matsuki et al., 2024b]¹, which proposes an ego-centric Neural Field approach to address the limitations of existing Neural Field-based SLAM systems for large-scale scenes. As discussed in the previous chapter on iMODE (3), Neural Field-based 3D representations have recently been showing a promising research direction for SLAM. Current Neural Field SLAM or online mapping systems lead to impressive results in the presence of simple captures, but they rely on a world-centric map representation as only a single neural field model is used. To

¹This work was conducted during an internship at Google.

define such a world-centric representation, accurate and static prior information about the scene, such as its boundaries and initial camera poses, are required. However, in real-time and on-the-fly scene capture applications, this prior knowledge cannot be assumed as fixed or static, since it dynamically changes and it is subject to significant updates based on run-time observations. Particularly in the context of large-scale mapping, significant camera pose drift is inevitable, necessitating the correction via loop closure. To overcome this limitation, we propose **NEWTON**, a view-centric mapping method that dynamically constructs neural fields based on run-time observation. In contrast to prior works, our method enables camera pose updates using loop closures and scene boundary updates by representing the scene with multiple neural fields, where each is defined in a local coordinate system of a selected keyframe. The experimental results demonstrate the superior performance of our method over existing world-centric neural field-based SLAM systems, in particular for large-scale scenes subject to camera pose updates.

4.1 Introduction

Real-time scene capture is an essential technology for embodied device scene understanding and interaction, and the choice of the underlying scene representation greatly influences the performance and efficiency of the mapping. In recent years, neural field-based representations have enabled remarkable progress in many 3D vision applications. Recent work [Sucar et al., 2021, Zhu et al., 2022, Rosinol et al., 2023] has adapted neural field-based scene representation for the task of SLAM. Those existing approaches rely on world-centric representations since they employ one ‘global’ neural field to represent the underlying scene. As a result, they have fundamental limitations for open world on-the-fly mapping, which requires loop closure and dynamic definition of scene boundaries. In particular, related work uses scene representations through a MLP [Sucar et al., 2021] or a Feature Grid [Rosinol et al., 2023, Zhu et al., 2022] defined in a global world coordinate system and allocate features within predetermined boundaries. These representations, however, do not allow for loop closure modules that can perform large and instant pose updates, as it either causes the tracking to fail due to the resulting inconsistent map or it requires re-training the entire field for online-mapping, since camera pose optimization tends to distort the geometry around the corrected camera positions causing inconsistencies. Especially under large-scale settings this results in misalignments and performance degradation, hence limiting the use of these methods to more controlled and small room-scale scenarios only.

In this work, we propose *NEWTON*, a view-centric neural field-based representation for mapping

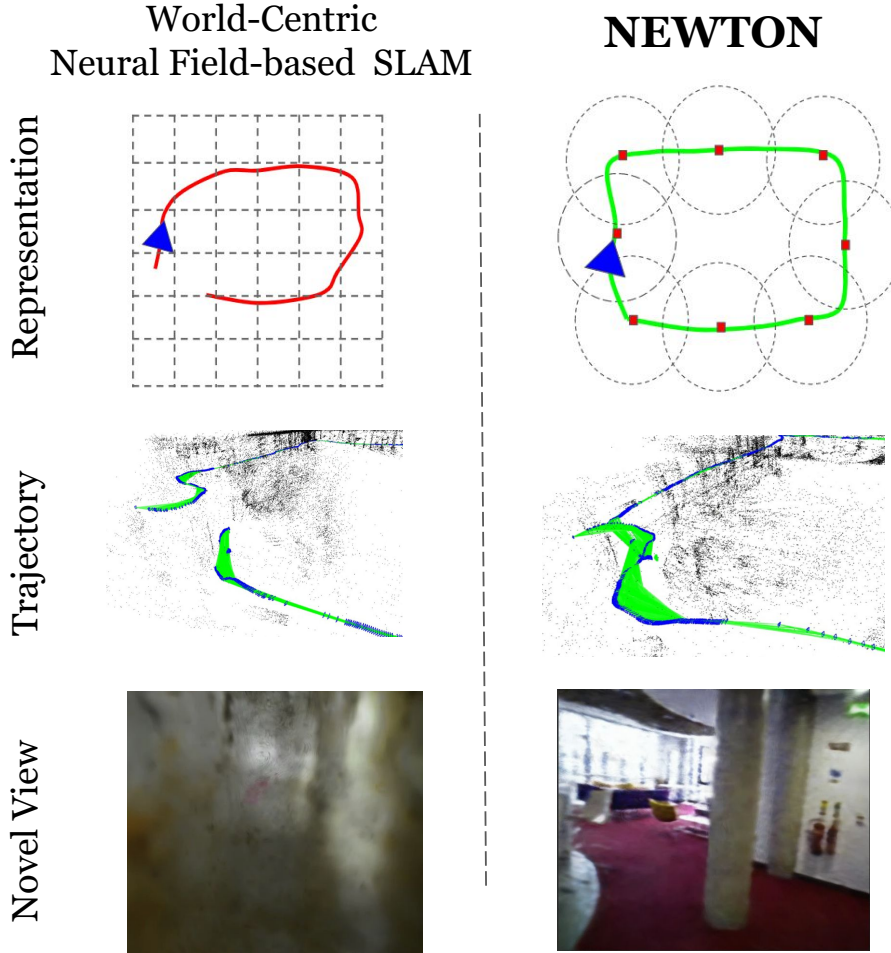


Figure 4.1: **View-Centric Mapping with NEWTON.** Prior neural field-based SLAM systems [Sucar et al., 2021, Zhu et al., 2022, Rosinol et al., 2023] use a world-centric representation and thus cannot handle large-scale real-world scenarios where online pose updates by loop closure are required. In contrast, we propose a *view-centric* mapping approach with multiple, local neural fields which are dynamically allocated. This improves performance and enables on-the-fly mapping for large-scale scenes.

which can handle dynamically changing input data streams in real-time. In contrast to prior works, we achieve this by representing the scene, as multiple neural fields defined in the local coordinate system of the respective keyframes. Further, our system is capable of dynamically allocating the individual representations based on run-time observations, which enables flexible field updates without re-training the entire scene representations. To enable full camera pose tracking and mapping from input video stream, we combine our mapping system with the camera tracker of a loop-closable state-of-the-art SLAM system [Mur-Artal and Tardós, 2017b]. We exploit Neural Radiance Field (NeRF) as an underlying Neural-Field, and our method demonstrates superior performance and robustness on large-scale scenes. To summarize, our contributions are as follows:

- A neural field-based map representation which can handle dynamic pose updates by using multiple local models.

- A spherical parameterization for neural fields to efficiently represent unbounded scenes not limited by object-centric motion.
- Model creation, initialization, training, and view synthesis policies for the proposed representation.

4.2 Related Work

Neural Fields While first works on neural fields for 3D reconstruction [Mescheder et al., 2019, Park et al., 2019, Chen and Zhang, 2019] or view synthesis [Sitzmann et al., 2019b, Niemeyer et al., 2020a, Yariv et al., 2021b, Mildenhall et al., 2020] propose to use a single multi-layer perceptron (MLP) as scene representation, recent works propose to use spatially-distributed grid features with MLPs [Peng et al., 2020, Müller et al., 2022], separate fore- and background models [Zhang et al., 2020], space contractions [Barron et al., 2022], or multiple individual models [Tancik et al., 2022]. For example, BlockNeRF [Tancik et al., 2022] partitions large scenes into individual blocks and reconstructs them independently. While leading to improvements for the respective scenarios, all of these methods are world-centric, *i.e.* they need to know a priori where to reconstruct and how to structure the scene. While works that combine image-based rendering techniques with neural fields [Wizadwongsa et al., 2021, Wang et al., 2021b, Yu et al., 2021b] do not reconstruct in a canonical world coordinate system, they still assume a single coordinate system with aligned poses. This, however, is not a realistic assumption for on-the-fly mapping where optimal model allocation such as uniform block partitions in the scene is impossible to define prior to the optimization. In contrast, we propose a view-centric representation that can handle dynamically-changing input by representing scenes using multiple fields in local coordinate systems. While [Meuleman et al., 2023] also introduces multiple local radiance field for offline NeRF training, our method is intended for on-the-fly SLAM reconstruction and proposes novel techniques for the integration of the SLAM system.

Visual SLAM Visual SLAM systems can be broadly classified by their underlying representation into world- and view-centric approaches.

World-centric representations assume that rich scene information such as accurate scene boundaries, *etc.*, are provided. While pioneering works such as KinectFusion [Newcombe et al., 2011a] and SuperEight [Vespa et al., 2018] tracks against underlying voxel grid-based Truncated Signed Distance Functions (TSDF) with classical depth map fusion, later works such as NeuralRecon [Sun

et al., 2021] and Atlas [Murez et al., 2020] incorporate learned priors into the volumetric TSDF mapping. More recently, neural field-based scene representations have been incorporated due to their simplicity and state-of-the-art performance in tasks such as 3D reconstruction. iMAP [Sucar et al., 2021] optimizes a single MLP-based scene representation from RGB-D input. NICE-SLAM [Zhu et al., 2022] improves the scene representation by combining a small MLP with spatially-distributed latent grid features, improving the mapping performance. NeRF-SLAM [Rosinol et al., 2023] further utilizes uncertainty information given by the depth maps. While these methods perform well for small scenes and simple camera trajectories, their performance drops significantly for real-world scenarios where dynamic pose updates and loop closure is required.

View-centric representations in contrast construct the local scene representation on a per-view/keyframe basis and hence no prior global scene information such as scene boundaries or grid resolution are required for pose and map optimization. As a result, most of the state-of-the-art camera tracking systems are view-centric methods with representations such as depth maps [Mur-Artal and Tardós, 2017b, Engel et al., 2017, Bloesch et al., 2018, Teed and Deng, 2021] or surfels [Schöps et al., 2019]. While view-centric representations cannot directly optimize a scene globally, having multiple local maps simplifies the camera pose drift correction. This idea is powerful and some volumetric SLAM methods implement loop closure by representing a scene by multiple local chunks or submaps [Dai et al., 2017b, Kähler et al., 2016]. As a result, in this work we propose a view-centric scene representation and show how neural fields can be used as local models, improving mapping performance over prior neural-field based approaches in particular for large-scale scenes where dynamic pose updates are required.

4.3 Method

This section introduces the proposed neural view-centric mapping approach for real-time large-scale SLAM. More specifically, we first obtain camera pose estimates and a co-visibility graph from an input RGB stream by means of a state-of-the-art camera tracking component [Mur-Artal and Tardós, 2017b] (Sec. 4.3.1). Next, for mapping we represent scenes using multiple neural radiance fields, each defined in local coordinate systems (Sec. 4.3.2). We dynamically allocate new fields based on co-visibility and perform feature propagation to facilitate training (Sec. 4.3.3). Fig. 4.2 shows an overview over our method.

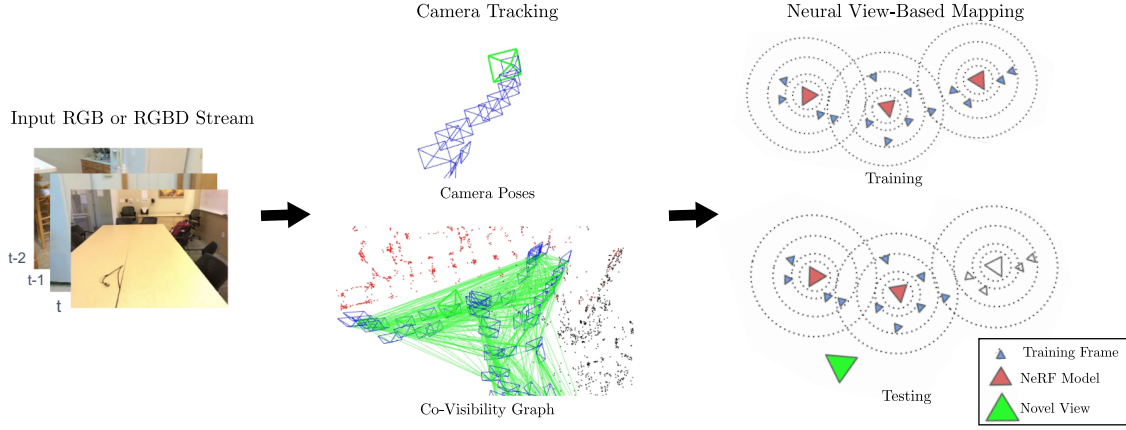


Figure 4.2: **Method overview.** The input RGB or RGB-D stream is fed to a state-of-the-art camera tracking system [Mur-Artal and Tardós, 2017b] to obtain camera poses and a covisibility graph. Next, for mapping we train multiple neural radiance fields each defined in a local coordinate system. At test time, neighboring models are queried and images can be rendered from arbitrary viewpoints, where blending is performed in 2D image space. These multiple local maps effectively handle loop closure (Fig. 4.7).

4.3.1 Camera Tracking

Our proposed neural field-based mapping approach runs in combination with a separate camera tracking component. More specifically, the input RGB or RGB-D stream is first fed to the tracking component which returns camera poses and a co-visibility graph. In this work, we use ORB-SLAM [Mur-Artal and Tardós, 2017b] as it provides reliable pose estimation regardless of the scale and domain of the scene, and it performs loop-closure during runtime. Our mapping system takes in the current keyframe pose and image as training frames and assigns them to the training view pool for the NeRF models using the frame covisibility. Since each model is defined in a local coordinate system of the selected keyframe, it can be flexibly adjusted to the pose update and the map is much more robust than world-centric representations.

4.3.2 Scene Representation

Spherical Multi-Resolution Hash Encoding

We represent scenes with multiple NeRF models defined in the local coordinate system of selected keyframes. We represent each NeRF model as a multi-resolution feature grid [Müller et al., 2022] in a spherical coordinate system (see Fig. 4.3). We convert the input location represented in

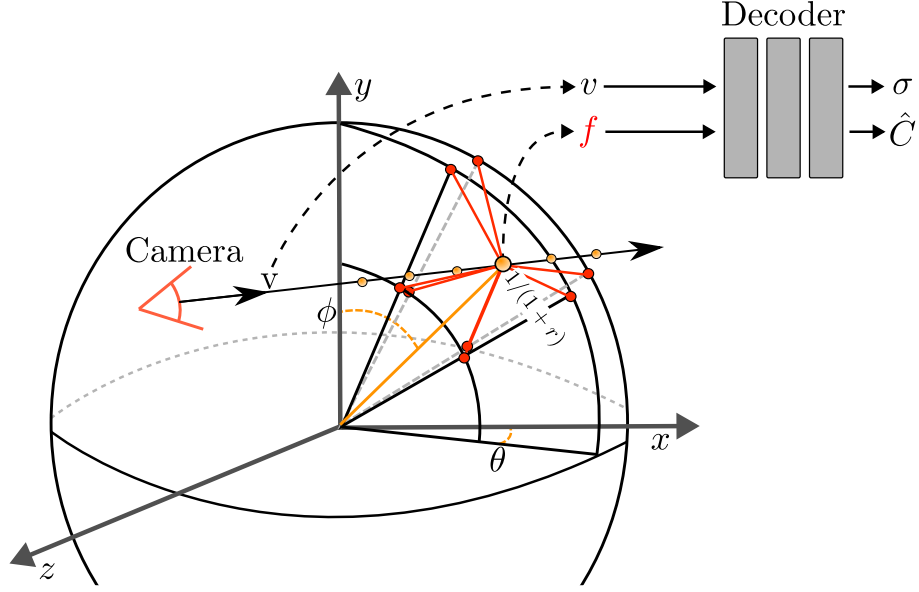


Figure 4.3: **Spherical Multi-Resolution Grid.** This representation allocates features evenly along inverse distance and naturally supports unbounded scene without a limitation of the field of view.

Cartesian coordinates $(x, y, z) \in \mathbb{R}^3$ to spherical coordinates $(\phi, \theta, \frac{1}{1+r}) \in [0, 1]^3$ as follows:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \frac{2}{\pi} \arctan\left(\frac{z}{x}\right) \\ \phi &= \frac{1}{\pi} \arcsin \frac{y}{r}. \end{aligned} \tag{4.1}$$

This enables at the same time an efficient feature allocation strategy related to the pixel’s measurement accuracy and a representation that is not limited by the frustum boundary.

Feature Allocation An increasing distance to the camera center leads to a larger measurement uncertainty as disparity is inversely-proportional to its depth [Civera et al., 2008]. To allocate more capacity in areas of high certainty, it is therefore efficient to allocate the features evenly in inverse depth space.

This representation can be interpreted as an omnidirectional extension of a Normalized Device Coordinates (NDC) used for novel view synthesis in forward facing scenes in [Mildenhall et al., 2020]. While the NDC parameterization contracts the scene along inverse depth, the region is strictly bounded by the model view frustum, which discards the training view information outside of the frustum and also causes artifacts around the frustum boundary as shown in Fig. 4.4. In contrast, our spherical representation can make full use of the training view information without being limited to the field of view.

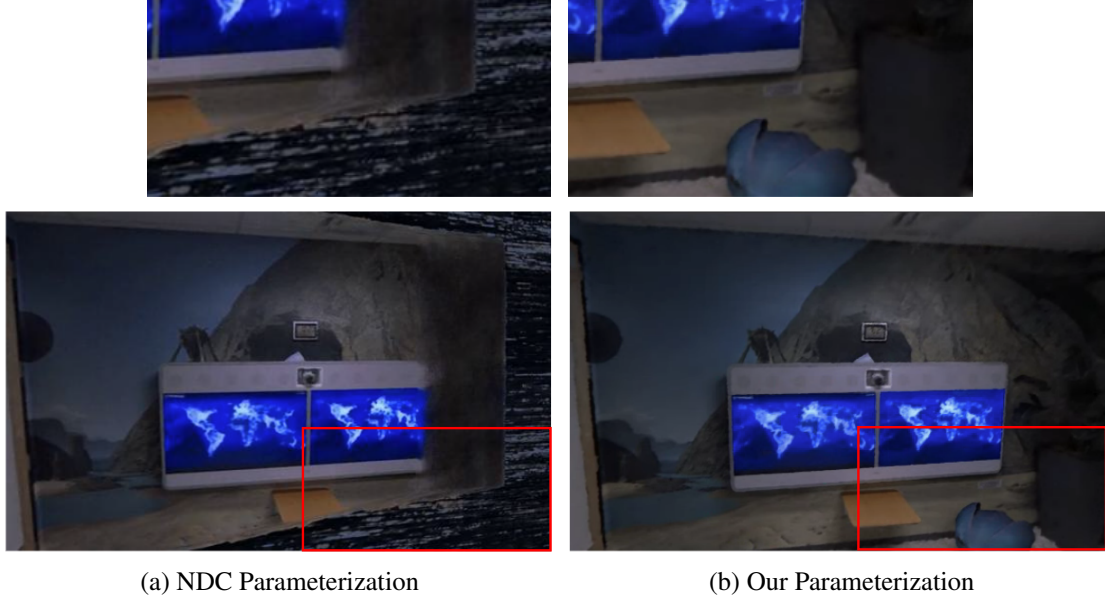


Figure 4.4: **NDC vs Spherical Parameterization.** While both parameterizations allocate features along inverse depth, NDC lead suffers from artifacts around the boundaries. In contrast, our proposed spherical parameterization leads to high-quality view synthesis results for the entire image.

Unbounded Scenes Our representation can naturally capture unbounded scenes. While other parameterizations such as the recently proposed space contraction from [Barron et al., 2022] can also represent 360° unbounded scenes, it explicitly separates foreground and background. This is effective only when the camera trajectory is object-centric and inward-looking, and the region of interest needs to be known a priori in world coordinates. However, in case of general odometry-type video sequences, the camera trajectory is not limited to being object-centric and it is not possible to define the region of interest beforehand. While Mip-NeRF 360 [Barron et al., 2022] also introduces a scene-contraction background model, our spherical parameterization offers the advantage of fully leveraging grid-based features. In spherical coordinates, a position is parameterized as $(\phi, \theta, \frac{1}{1+r})$ and these three variables can take arbitrary values in $[0, 1]^3$ independently. On the other hand, Mip-NeRF360 contracts the position $\mathbf{x} = (x, y, z)$ to $\mathbf{x}_c = (x_c, y_c, z_c) \in [-2, 2]^3$ by the following contract function,

$$\text{contract}(\mathbf{x}) = \begin{cases} \mathbf{x} & \|\mathbf{x}\| \leq 1 \\ (2 - \frac{1}{\|\mathbf{x}\|})(\frac{\mathbf{x}}{\|\mathbf{x}\|}) & \|\mathbf{x}\| > 1 \end{cases}$$

However, the contracted parameters (x_c, y_c, z_c) of the background model are not independent but need to be within a sphere of radius 2. As a result, a uniform grid $[-2, 2]^3$ in (x_c, y_c, z_c) space is not fully filled with the sphere area of variables and some features are not used at all as shown

in Fig. 4.5. To address this, nerfstudio [Tancik et al., 2023] uses a L_∞ norm for normalization to cover the grid with the parameters. However, this is not an ideal parameterization because points need to be allocated proportionally to inverse Euclidean distance (L_2 norm). In contrast, the spherical coordinate parameterization can naturally allocate features evenly in the parameter space without facing these issues.

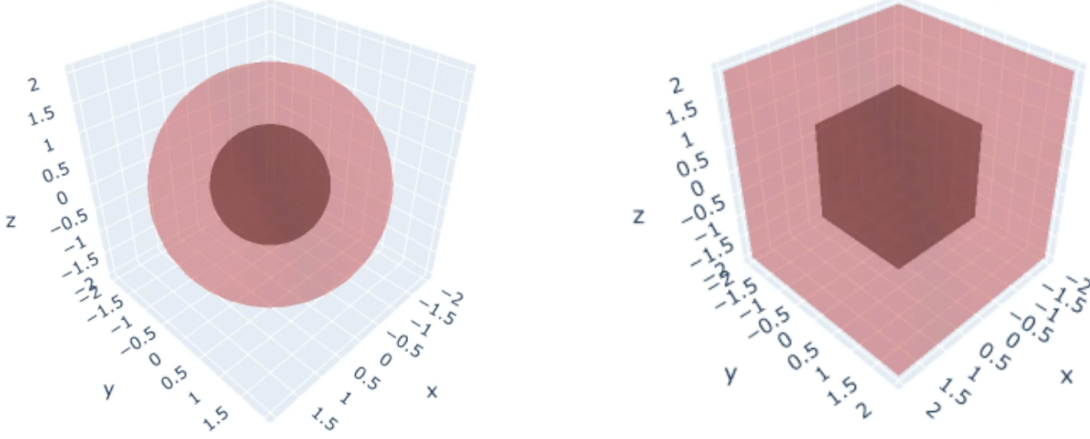


Figure 4.5: **Scene Contraction of Mip-NeRF360.** (Left) Mip-NeRF360’s original scene contraction (Right) the same contraction using L_∞ norm (used in nerfstudio). The inner black region represents the foreground model and the outer red region represents the background model. Mip-NeRF360’s background model cannot fill the entire grid $[-2, 2]^3$ and cannot make full use of the allocated grid features. While nerfstudio’s contraction fills the parameter space, L_∞ norm is different from the original definition of inverse distance defined in Euclidean space (L_2 norm). Figures are generated by using nerfstudio’s interactive documentation [team, 2022].

Loss Functions

We use classic volume rendering techniques [Mildenhall et al., 2020, Müller et al., 2022] to render pixel color $\hat{C}$. We train each NeRF model by minimizing the MSE loss against ground truth color C :

$$L_{\text{rgb}} = \sum_{r \in \mathcal{R}} \|\hat{C}_r - C_r\|_2^2 \quad (4.2)$$

where $r \in \mathcal{R}$ are rays subsampled from all pixels/rays.

We have further incorporated a distortion regularization as detailed in [Barron et al., 2022]. Utilizing the set of normalized ray distances $\mathbf{s}$ and weights $\mathbf{w}$ that parameterize each ray, the regularization is defined as follows:

$$L_{\text{dist}}(\mathbf{s}, \mathbf{w}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{w}_{\mathbf{s}}(u) \mathbf{w}_{\mathbf{s}}(v) |u - v| d_u d_v. \quad (4.3)$$

In this equation, $\mathbf{w}_s(u)$ represents an interpolation into the step function, defined by the pair $(s, \mathbf{w})$ at u . This is expressed as $\mathbf{w}_s(u) = \sum_i w_i \mathbb{1}_{[s_i, s_{i+1})}(u)$, where i denotes the index of bins for ray sampling.

Given that $\mathbf{w}_s(\cdot)$ maintains a constant value within each interval, we have reformulated the distortion regularization as follows:

$$L_{\text{dist}} = \sum_{i,j} w_i w_j \left| \frac{s_i + s_{i+1}}{2} - \frac{s_j + s_{j+1}}{2} \right| + \frac{1}{3} \sum_i w_i^2 (s_{i+1} - s_i), \quad (4.4)$$

where j also denotes the index of bins for ray sampling.

We also follow [Barron et al., 2022] and add a small proposal network for each NeRF model. Here we define ray interval and predicted weight vector from the main NeRF network as $\mathbf{t}, \mathbf{w}$ and those from the proposal network as $\hat{\mathbf{t}}, \hat{\mathbf{w}}$. Following [Barron et al., 2022], the bound function with interval T is defined as:

$$\text{bound}(\hat{\mathbf{t}}, \hat{\mathbf{w}}, T) = \sum_{j: T \cap \hat{T}_j \neq \emptyset} \hat{w}_j. \quad (4.5)$$

We train the proposal network with the proposal loss:

$$L_{\text{prop}}(t, w, \hat{t}, \hat{w}) = \sum_i \max(0, w_i - \text{bound}(\hat{t}, \hat{w}, T_i))^2 \quad (4.6)$$

We can optionally incorporate depth measurement if available. Similar to a pixel’s color value, we use volume rendering to obtain a pixel’s rendered $\hat{D}_r$ and minimize

$$L_{\text{depth}} = \sum_{r \in \mathcal{R}} |\hat{D}_r - D_r|_1, \quad (4.7)$$

where D is the GT depth for ray r . The full loss we optimize is given by:

$$L_{\text{total}} = L_{\text{rgb}} + \lambda_{\text{dist}} L_{\text{dist}} + \lambda_{\text{prop}} L_{\text{prop}} + \lambda_{\text{depth}} L_{\text{depth}}. \quad (4.8)$$

4.3.3 Dynamic Neural Field Allocation and Training

Training Policy

When a new keyframe is created in our camera tracking module, we check for covisibility between the keyframe and the existing training frames of NEWTON by using the covisibility graph. Each

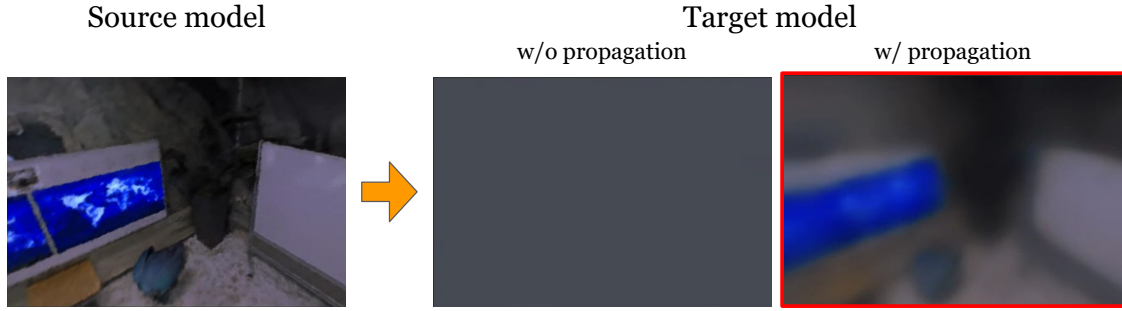


Figure 4.6: **Feature Propagation.** We initialize new models by propagating parameters of previous models which improves both photometric and geometric accuracy.

training frame is assigned to a single model ID which represents the primary model it belongs to. We check the model IDs of the covisible training frames, assign the new keyframe to the newest covisible model, and further assign this newest model ID to the frame. We also add the new keyframe to the training batch of the other older covisible models to keep appearance consistency of spatially-overlapping regions for different models. For every iteration, we train up to three models simultaneously; two of the newest covisible models from the latest training frame and one randomly chosen model.

Model Creation

Creation Policy If the input keyframe is further away than a distance threshold d_{th} from any of the covisible models, we create a new model. To better initialize the new model we apply our proposed feature propagation.

Feature Propagation When creating a new model, it is better to initialize the model parameters using the information from previously trained models instead of training from scratch. Here we describe how to propagate the trained feature information from the previous model to the new one. For the coarse level of multi-resolution hash grids which does not have any hash collision, we query the copy source model and trilinearly interpolate the feature on each corner of the new feature grid. We do not propagate the fine level features from the hash table because the hash function is uniquely defined in one local coordinate and the features are learned to handle collisions in the specific coordinate system. This cannot be easily transferred to another coordinate system. We use the weights of the MLPs as a coarse initialization of the newly created field. We investigate the effect of the feature propagation in an ablation study.

Novel View Rendering

For novel view synthesis, we first perform model selection based on the camera position and query multiple NeRF models. We then perform image blending in 2D image space.

Model Selection For a novel view, we first check the nearest training view position, then take the top three closest models for this training view. We render an image for each model and blend them. For each model, we also store the coarse voxel grid (of resolution 16^3) that contains information whether a block in the grid is used during training. For rendering at test time, we check the visibility for queried sample points and perform empty space skipping.

Blending We blend rendered images by weighting them based on the distance between the novel view and the nearest training view of each queries model. Let the camera origin of the test view be o and t_i the position of the nearest training view for the i -th model. Similar to [Tancik et al., 2022], we use the inverse distance weighting function:

$$w_i \propto \|o - t_i\|_2^{-p}. \quad (4.9)$$

4.4 Evaluation

4.4.1 Experimental Setup

We first evaluate our method for **on-the-fly mapping** and investigate its performance when dynamic pose updates are required. Next, we run a **baseline comparison** against the following state-of-the-art neural field-based SLAM systems: NeRF-SLAM [Rosinol et al., 2023], NICE-SLAM [Zhu et al., 2022], and iMAP [Sucar et al., 2021]. Finally, we perform an **ablation study** of various components of our method.

Implementation Details We train our method on a single NVIDIA RTX3090Ti GPU which can process our system with a 20 fps video stream input. We build our method on an unbounded instant-NGP implementation in Nerfstudio [Tancik et al., 2023] and use the same hash size and ray parameters. We set $\lambda_{dist}=0.002$, $\lambda_{prop}=1.0$, $\lambda_{depth}=0.5$, $p=4$, $d_{th}=0.3$.

		Kintinuous		ScanNet													
		loop		0000		0059		0169		0181		0207		0106		Avg.	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
World-Centric Single Model	PSNR $\uparrow$	20.01	2.021	23.36	1.243	19.40	1.673	24.13	2.551	21.49	2.002	22.21	2.278	20.35	0.659	21.57	1.770
	SSIM $\uparrow$	0.654	0.079	0.765	0.059	0.646	0.104	0.714	0.074	0.707	0.062	0.676	0.087	0.686	0.033	0.692	0.071
	LPIPS $\downarrow$	0.204	0.038	0.098	0.014	0.159	0.016	0.167	0.024	0.139	0.029	0.176	0.024	0.164	0.018	0.176	0.023
	L1Depth $\downarrow$	0.364	0.211	0.187	0.010	0.267	0.035	0.217	0.035	0.199	0.026	0.200	0.025	0.247	0.016	0.240	0.051
Ours	PSNR $\uparrow$	20.45	0.96	23.58	0.772	19.36	0.806	24.95	0.643	22.09	0.627	23.15	0.967	20.16	0.659	21.96	0.777
	SSIM $\uparrow$	0.684	0.037	0.773	0.036	0.631	0.036	0.720	0.032	0.717	0.019	0.702	0.036	0.683	0.029	0.701	0.032
	LPIPS $\downarrow$	0.174	0.014	0.100	0.012	0.213	0.008	0.186	0.019	0.138	0.015	0.183	0.028	0.178	0.018	0.167	0.016
	L1Depth $\downarrow$	0.311	0.066	0.188	0.007	0.264	0.010	0.215	0.021	0.193	0.016	0.184	0.005	0.274	0.059	0.233	0.026

Table 4.1: **Reconstruction Results for On-the-Fly Mapping.** For each scene, we report the average over the full time series.

Datasets We use in total seven real-world scenes where ORB-SLAM [Mur-Artal and Tardós, 2017b] detects loop closure, ranging from small room-scale to large building-scale sequences. We take one sequence from Kintinuous [Whelan et al., 2015a] recorded by a hand-held Kinect v1 camera in a large building which has a large loop closure at the end of the sequence. We further follow previous works [Zhu et al., 2022] and take six sequences from the ScanNet dataset [Dai et al., 2017a] which contains small to middle-sized room sequences recorded by Kinect v1. While we run the input video stream at 20Hz for our method, the baseline methods run offline and process frames sequentially.

Metrics We report reconstruction metrics of unseen views *w.r.t.* both, the predicted appearance as well as the geometry. For the first, we report PSNR, SSIM, and LPIPS, while for measuring the predicted geometry, we report the L1 metric on predicted and GT depth maps in meter scale.

4.4.2 On-the-fly Mapping

Experiment Design We evaluate our on-the-fly mapping performance by comparing NEWTON to a world-centric variant of our approach with a single NeRF model which we call World-Centric Single Model. We use the ORB-SLAM’s trajectory from RGBD as input for both representations.

We use 10% of ORB-SLAM’s localized keyframes as test frames and exclude them from training. To measure temporal run-time performance, we report reconstruction metrics of all test views for a certain interval (every 50 keyframe creation by ORB-SLAM). For each dataset, we run NeRFStudio [Tancik et al., 2023]’s preprocessing pipeline and rescale the scene to fit the default model and ray parameters.

Result The quantitative results in Table 4.1 and qualitative results in Fig. 4.7 show that our method is minimally affected by loop closure while for the World-Centric Single Model, the per-

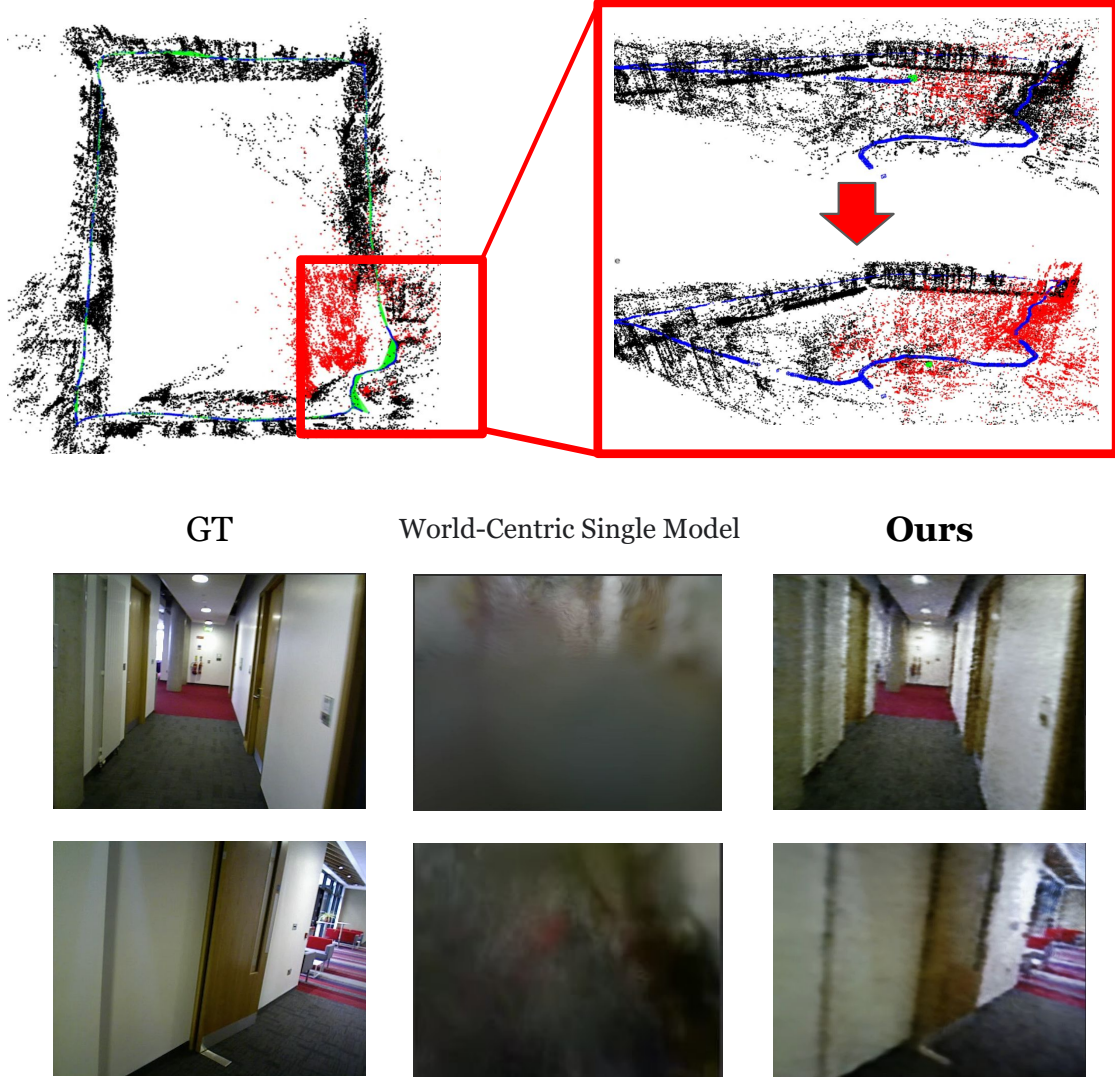
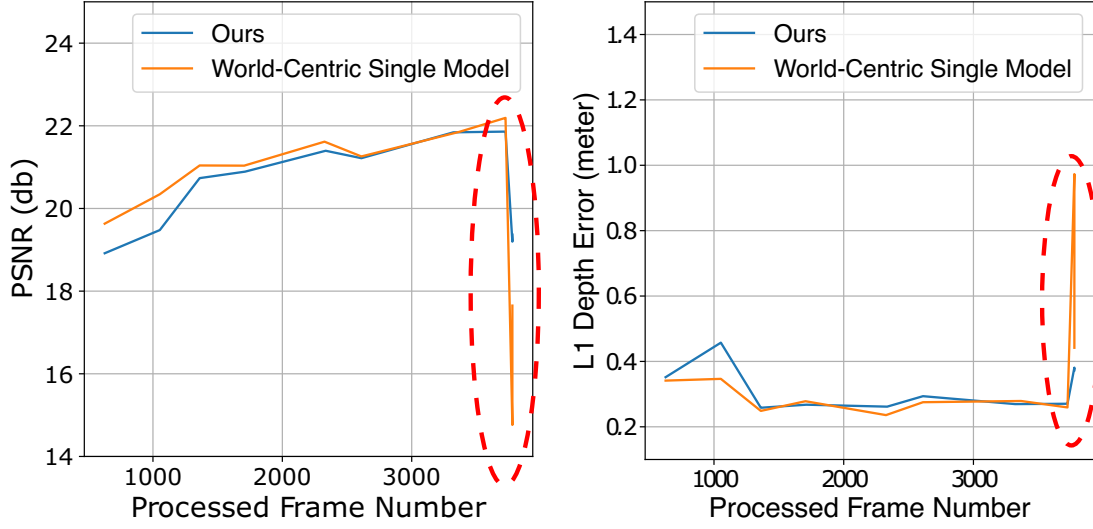


Figure 4.7: **Loop Closure.** While for a world-centric single model loop closure leads to degraded view synthesis, our view-centric approach with multiple local models allows for high-quality view synthesis even when performing loop closure (here shown for the Kintinuous dataset).

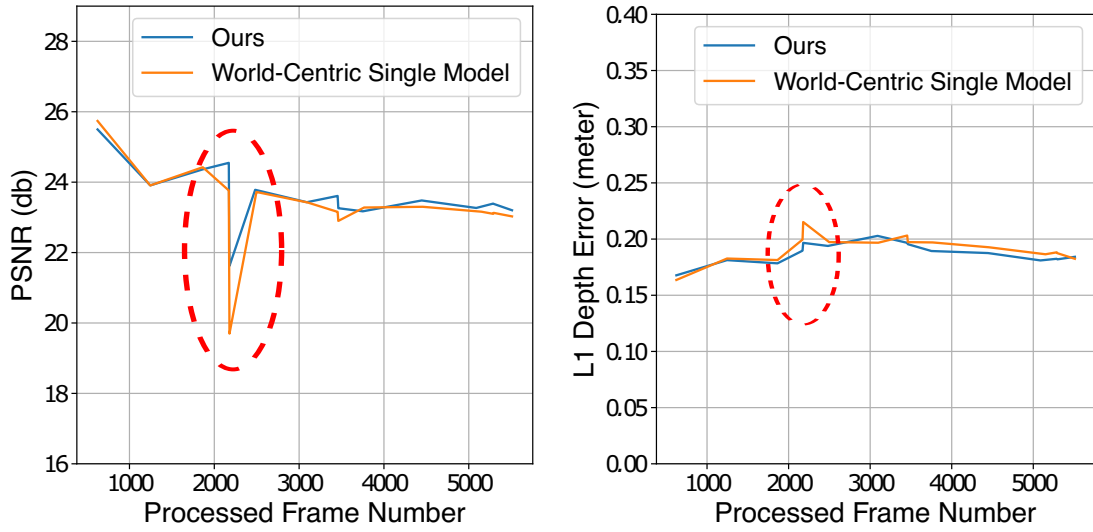
formance drops significantly. Further, the time series plot in Fig. 4.8 shows stronger robustness of our approach to the dynamic pose update and lower deviation values compared to the baseline.

4.4.3 Baseline Comparison

Experiment Design We compare our method to the state-of-the-art neural field based SLAM methods NeRF-SLAM [Rosinol et al., 2023], NICE-SLAM [Zhu et al., 2022], iMAP* [Sucar et al., 2021] [Zhu et al., 2022]. iMAP* denotes an open-source re-implementation of iMAP introduced in [Zhu et al., 2022]. As these are world-centric approaches, the drift correction by loop closure is not performed and shows a clear trajectory drift. NeRF-SLAM uses DROID-SLAM [Teed and Deng, 2021] with its own optimization backend and it does not perform drift



(a) Kintinuus Dataset.



(b) ScanNet Dataset (scene0000).

Figure 4.8: **Timeseries For On-the-Fly Mapping.** Our method is more robust to loop closure while the baseline World-Centric Single Model shows a significant performance drop (see red circles for where loop closure appears).

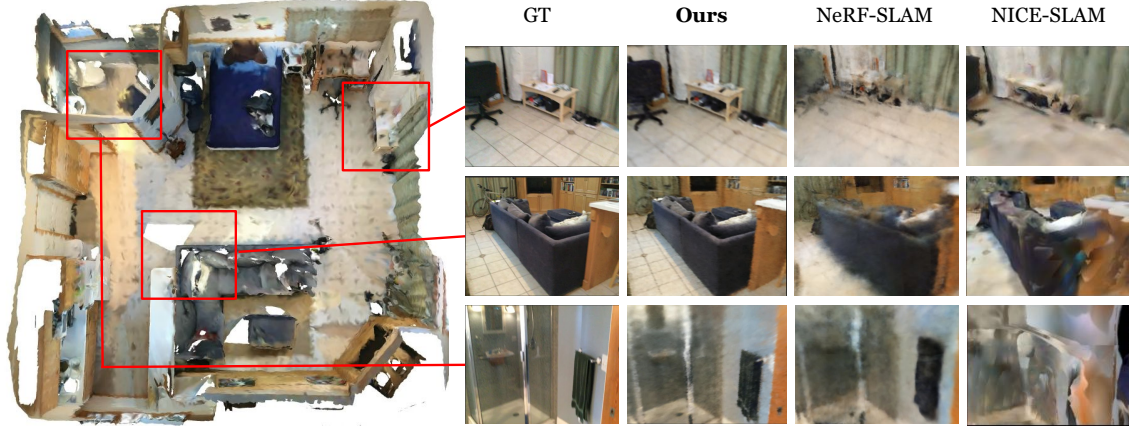


Figure 4.9: **View Synthesis on ScanNet (scene0000).** Left: GT mesh. Right: Novel view synthesis results. Existing world-centric SLAM methods such as NeRF-SLAM [Rosinol et al., 2023] or NICE-SLAM [Zhu et al., 2022] cannot handle loop closure and show misalignment in rendered views. In contrast, our representation flexibly adjusts to the pose update and leads to high-quality view synthesis.

correction by loop closure on these test sequences. We provide sensor depth measurements with NeRF-SLAM’s frontend tracking system for fair comparison. To evaluate the effect of the drift, we first align each trajectory to ground truth, and then render the images on the associated ground truth frame position. For large-scale scenes, it is difficult to get a perfect ground truth trajectory by using a motion capture system. In our settings, Kintinuous dataset does not provide ground truth trajectory data and the ScanNet dataset only provides trajectories estimated from BundleFusion [Dai et al., 2017b] which are less accurate than ORB-SLAM’s trajectories because the sensor (Kinect v1) is not perfectly synchronized and has the rolling shutter effect [Schöps et al., 2019]. Therefore, we adopt the frame trajectory from RGBD ORB-SLAM as ground truth for evaluation. We compare the final reconstruction result of these methods.

Result Table 4.2 shows the quantitative evaluation of our baseline comparison. We observe that our method achieves better performance for the majority of sequences. This indicates the importance of our view-centric mapping approach allowing for online drift correction and online camera pose updates when optimizing the neural field-based mapping representation. We further show qualitative results in Fig. 4.9 highlighting the effect of the trajectory drift. Notably, although our method and NeRF-SLAM both use volumetric NeRF as an underlying neural field representation, our view synthesis quality is significantly better due to the training camera pose accuracy. These results demonstrates the limitations of existing world-centric neural field-based mapping approaches for large scenes and long trajectories as well as the necessity of scene representations to be able to handle online state updates.

		Kintinuous	ScanNet						
		loop	0000	0059	0169	0181	0207	0106	Avg.
Ours	PSNR $\uparrow$	19.85	23.46	19.62	24.34	21.04	22.94	20.35	21.66
	SSIM $\uparrow$	0.634	0.769	0.644	0.698	0.686	0.693	0.697	0.689
	LPIPS $\downarrow$	0.193	0.108	0.211	0.198	0.162	0.178	0.173	0.175
	L1Depth $\downarrow$	0.376	0.184	0.264	0.223	0.214	0.183	0.294	0.248
NeRF-SLAM	PSNR $\uparrow$	13.94	16.98	15.14	17.06	19.76	16.96	13.33	15.85
	SSIM $\uparrow$	0.068	0.132	0.094	0.137	0.215	0.157	0.067	0.114
	LPIPS $\downarrow$	0.506	0.410	0.52	0.394	0.350	0.408	0.5646	0.465
	L1Depth $\downarrow$	2.06	0.297	0.588	0.373	0.272	0.341	0.773	0.690
NICE-SLAM	PSNR $\uparrow$	Fail	15.45	13.86	14.25	13.74	16.49	14.72	14.75
	SSIM $\uparrow$		0.573	0.502	0.585	0.610	0.634	0.579	0.580
	LPIPS $\downarrow$		0.502	0.511	0.491	0.497	0.425	0.459	0.480
	L1Depth $\downarrow$		0.205	0.397	0.292	0.384	0.191	0.226	0.282
iMAP*	PSNR $\uparrow$	Fail	13.10	13.45	13.44	11.84	16.35	11.53	13.28
	SSIM $\uparrow$		0.510	0.465	0.559	0.535	0.609	0.484	0.527
	LPIPS $\downarrow$		0.540	0.529	0.516	0.563	0.448	0.554	0.524
	L1Depth $\downarrow$		0.269	0.391	0.294	0.470	0.175	0.586	0.295

Table 4.2: **Baseline Comparison.** For each scene, each metric is averaged over the frames.

4.4.4 Ablation

	PSNR $\uparrow$		SSIM $\uparrow$		LPIPS $\downarrow$		L1Depth $\downarrow$	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Ours	21.96	0.777	0.701	0.032	0.167	0.016	0.233	0.026
Ours w/o depth	22.55	0.854	0.725	0.033	0.148	0.013	0.726	0.169

Table 4.3: **Depth Ablation for On-the-Fly Mapping.**

Color Only Supervision Table 4.3 shows ablation analysis of the on-the-fly performance result without depth supervision averaged over all sequences from Table 4.1. While we observe a drop in geometry prediction quality, view synthesis results are very similar. This demonstrates that our method can also be used for photorealistic mapping by using large-scale Monocular SLAM system with RGB streams as input.

Feature Propagation In Table 4.4 we ablate our feature propagation strategy (see Sec. 4.3.3). It improves all metrics for both, RGB and RGBD inputs, and indicates the importance of propagating the trained information between models for better initialization.

Memory Consumption We report memory size used for the map representation in Table 4.6 and the performance with different map sizes in Table 4.7. We use up to 12 models for the map representation and therefore the maximum map size is 12 times the NeRF model. The map size ablation in Table 4.7 shows our method still shows competitive performance even when we limit

	PSNR $\uparrow$		SSIM $\uparrow$		LPIPS $\downarrow$		L1Depth $\downarrow$	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
RGBD w/o propagation	21.84	0.799	0.692	0.034	0.175	0.017	0.239	0.033
Ours	21.96	0.777	0.701	0.032	0.167	0.016	0.233	0.026
RGB w/o propagation	22.40	0.885	0.721	0.034	0.151	0.013	0.790	0.217
Ours	22.55	0.854	0.725	0.033	0.148	0.013	0.726	0.169

Table 4.4: Ablation of our Feature Propagation.

	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	L1Depth $\downarrow$
World-Centric Single Model	40.33	0.984	0.008	0.0019
4 models	39.87	0.980	0.011	0.0020
Ours 8 models	40.29	0.981	0.010	0.0021
12 models	40.55	0.983	0.009	0.0016

Table 4.5: Offline Performance Evaluation on Replica dataset

Method	Memory Size (MB)
iMAP [Sucar et al., 2021]	1.02
NICE-SLAM [Zhu et al., 2022]	12.0
Unbounded Instant-NGP [Tancik et al., 2023]	67.1
Ours (up to 12 models)	805.2

Table 4.6: Mapping Representation Memory Usages.

the map size thanks to the flexible loop closure handling and effective feature allocation by our proposed spherical encoding.

Offline Performance Evaluation While our method is aiming for on-the-fly mapping where dynamic state updates happen frequently, we evaluate the performance in the case of offline training. For this evaluation, we compare the baseline World-Centric Single Model to our method with multiple spherical NeRF models by training them on the synthetic Replica dataset [Straub et al., 2019] which provides perfect GT data in the form of RGB images, depth maps, and poses. We use the office0 sequence which is also used for iMAP [Sucar et al., 2021] evaluation and use 1800 images for training and 200 images for testing. We give static ground truth camera poses and RGBD images as input to the model, run 30000 training iterations and report the final numbers. For our multiple model training, we first divide the training images equally among the number of models in the order of the frame index. We set the first image in each training image group as the keyframe and allocate model centers on it. In Table 4.5 we report the quantitative results. The final metrics after offline training with ground truth input data are almost on par between model numbers and this demonstrates our representation does not sacrifice the offline performance.

		PSNR $\uparrow$		SSIM $\uparrow$		LPIPS $\downarrow$		L1Depth $\downarrow$	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
World-Centric	67MB	21.46	1.67	0.690	0.069	0.162	0.023	0.238	0.049
Single Model	805MB	21.57	1.77	0.692	0.071	0.158	0.023	0.240	0.051
Ours	67MB	21.75	0.841	0.687	0.031	0.180	0.016	0.235	0.026
	805MB	21.96	0.777	0.701	0.032	0.167	0.016	0.233	0.026

Table 4.7: **Different Map Sizes for On-the-Fly Mapping.**

4.5 Conclusion

We have proposed NEWTON, a novel neural-field based mapping method which can handle online and dynamic state updates from the camera tracking module. We efficiently allocate and train multiple local neural fields with the proposed parametrization and model training strategy. The strong advantage of our method is its robustness and flexibility to large pose updates by loop closure and its superior performance on real-world large-scale scenes compared to existing world-centric approaches. As future work, we will investigate a loop-closable end-to-end neural field SLAM system that includes camera localization by using the proposed scene representation.

4.6 Limitations

- The proposed method currently employs an external camera tracking system, which increases the overall system complexity. In order to develop a SLAM system that performs tracking and mapping within the same framework, we should use more geometry-oriented Neural Field representation such as SDFs or occupancy field for accurate camera tracking instead of radiance field.
- The time series plot in Fig. 4.8 indicates that, while our approach is significantly less affected by loop closure than the baseline model, we still observe a small drop in performance because ORB-SLAM’s pose graph optimization distributes the drift correction to all keyframes and relative pose updates in the models still happen. However, this can be mitigated by exchanging the loop closure module with *e.g.* hierarchical pose graph optimization where large drift correction by loop closure is done only for the selected model-center keyframe and the camera poses of local training frames are separately optimized similar to [Dai et al., 2017b].

Gaussian Splatting SLAM: Point Primitive and Differentiable Rasterization for SLAM

Contents

5.1	Introduction	76
5.2	Related Work	78
5.3	Method	80
5.3.1	Gaussian Splatting	80
5.3.2	Camera Pose Optimization	81
5.3.3	SLAM	83
5.4	Evaluation	87
5.4.1	Experimental Setup	87
5.4.2	Quantitative Evaluation	89
5.4.3	Qualitative Results	99
5.5	Conclusion	105
5.6	Limitations	105

As demonstrated in previous sections, adaptive and flexible scene representation is essential for online SLAM, where estimations must be continuously updated based on live observations. While *NEWTON* demonstrated the effectiveness of a submapping framework using grid-based NeRF, it relies on additional heuristics for map creation and blended rendering, resulting in a more complex system. In contrast, point-based representations are inherently better suited for dynamic resource allocation and resampling due to their discrete nature.



Figure 5.1: From a single monocular camera, we reconstruct a high fidelity *3D scene* live at 3fps. For every incoming RGB frame, 3D Gaussians are incrementally formed and optimized together with the camera poses. We show both the rasterized Gaussians (left) and Gaussians shaded to highlight the geometry (right). Notice the details and the complex material properties (e.g. transparency) captured. Thin structures such as wires are accurately represented by numerous small, elongated Gaussians, and transparent objects are effectively represented by placing the Gaussians along the rim. Our system significantly advances the fidelity a live monocular SLAM system can capture.

Recently, 3D Gaussian Splatting [Kerbl et al., 2023] has emerged as a new scene representation for ultra-fast radiance field rendering. In this chapter, we present the first application of 3D Gaussian Splatting in monocular SLAM, the most fundamental but the hardest setup for Visual SLAM. Our method, which runs live at 3fps, utilizes Gaussians as the only 3D representation, unifying the required representation for accurate, efficient tracking, mapping, and high-quality rendering. Designed for challenging monocular settings, our approach is seamlessly extendable to RGB-D SLAM when an external depth sensor is available. Several innovations are required to continuously reconstruct 3D scenes with high fidelity from a live camera. First, to move beyond the original 3DGS algorithm, which requires accurate poses from an offline Structure from Motion (SfM) system, we formulate camera tracking for 3DGS using direct optimization against the 3D Gaussians, and show that this enables fast and robust tracking with a wide basin of convergence. Second, by utilizing the explicit nature of the Gaussians, we introduce geometric verification and regularization to handle the ambiguities occurring in incremental 3D dense reconstruction. Finally, we introduce a full SLAM system which not only achieves state-of-the-art results in novel view synthesis and trajectory estimation but also reconstruction of tiny and even transparent objects. Unlike the methods introduced in previous chapters works, our method jointly optimizes the 3D Gaussian map and camera pose in an end-to-end manner.

5.1 Introduction

A long-term goal of online reconstruction with a single moving camera is near-photorealistic fidelity, which will surely allow new levels of performance in many areas of Spatial AI and robotics

as well as opening up a whole range of new applications. While we increasingly see the benefit of applying powerful pre-trained priors to 3D reconstruction, a key avenue for progress is still the invention and development of core 3D representations with advantageous properties. Many “layered” SLAM methods exist which tackle the SLAM problem by integrating multiple different 3D representations or existing SLAM components; however, the most interesting advances are when a new unified dense representation can be used for all aspects of a system’s operation: local representation of detail, large-scale geometric mapping and also camera tracking by direct alignment.

In this chapter, we present the first online visual SLAM system based solely on the 3D Gaussian Splatting (3DGS) representation [Kerbl et al., 2023] recently making a big impact in offline scene reconstruction. As a SLAM representation, 3DGS is most similar to point and surfel clouds, and inherits their efficiency, locality and ability to be easily warped or modified. However, it also represents geometry in a smooth, continuously differentiable way: a dense cloud of Gaussians merge together and jointly define a continuous volumetric function. And crucially, the design of modern graphics cards means that a large number of Gaussians can be efficiently rendered via “splatting” rasterization, up to 200fps at 1080p. This rapid, differentiable rendering is integral to the tracking and map optimization loops in our system.

The 3DGS representation has up until now only been used in offline systems for 3D reconstruction with known camera poses, and we present several innovations to enable online SLAM. We first derive the analytic Jacobian on Lie group of camera pose with respect to a 3D Gaussians map, and show that this can be seamlessly integrated into the existing differentiable rasterization pipeline to enable camera poses to be optimized alongside scene geometry. Second, we introduce a novel Gaussian isotropic shape regularization to ensure geometric consistency, which we have found is important for incremental reconstruction. Third, we propose a novel Gaussian resource allocation and pruning method to keep the geometry clean and enable accurate camera tracking. Our experimental results demonstrate photorealistic online local scene reconstruction, as well as state-of-the-art camera trajectory estimation and mapping for larger scenes compared to other rendering-based SLAM methods. We further show the uniqueness of the Gaussian-based SLAM method such as an extremely large camera pose convergence basin, which can also be useful for map-based camera localization. Our method works with only monocular input, one of the most challenging scenarios in SLAM. To highlight the intrinsic capability of 3D Gaussian for camera localization, our method does not use any pre-trained monocular depth predictor or other existing tracking modules, but relies solely on RGB image inputs in line with the original 3DGS. Since this is one of the most challenging SLAM scenario, we also show our method can easily be extended

to RGB-D SLAM when depth measurements are available.

In summary, our contributions are as follows:

- The first near real-time SLAM system which works with a 3DGS as the only underlying scene representation, which can handle monocular only inputs.
- Novel techniques within the SLAM framework, including the analytic Jacobian on Lie group for direct camera pose estimation, isotropic regularization of the Gaussian shape, and geometric verification.
- Extensive evaluations on a variety of datasets both for monocular and RGB-D settings, demonstrating competitive performance, particularly in real-world scenarios.

5.2 Related Work

Dense SLAM: Dense visual SLAM focuses on reconstructing detailed 3D maps, unlike sparse SLAM methods which excel in pose estimation [Mur-Artal and Tardós, 2017a, Engel et al., 2017, Forster et al., 2014] but typically yield maps useful mainly for localization. In contrast, dense SLAM creates interactive maps beneficial for broader applications, including AR and robotics. Dense SLAM methods are generally divided into two primary categories: Frame-centric and Map-centric. **Frame-centric SLAM** minimizes photometric error across consecutive frames, jointly estimating per-frame depth and frame-to-frame camera motion. Frame-centric approaches [Teed and Deng, 2021, Czarnowski et al., 2020b] are efficient, as individual frames host local rather than global geometry (e.g. depth maps), and are attractive for long-session SLAM, but if a dense global map is needed, it must be constructed on demand by assembling all of these parts which are not necessarily fully consistent. In contrast, **Map-centric SLAM** uses a unified 3D representation across the SLAM pipeline, enabling a compact and streamlined system. Compared to purely local frame-to-frame tracking, a map-centric approach leverages global information by tracking against the reconstructed 3D consistent map. Classical map-centric approaches often use voxel grids [Newcombe et al., 2011b, Dai et al., 2017b, Whelan et al., 2015a, Prisacariu et al., 2014] or points [Keller et al., 2013, Whelan et al., 2015b, Schöps et al., 2019] as the underlying 3D representation. While voxels enable a fast look-up of features in 3D, the representation is expensive, and the fixed voxel resolution and distribution are problematic when the spatial characteristics of the environment are not known in advance. On the other hand, a point-based map representation, such as surfel clouds, enables adaptive changes in resolution and spatial distribution by dynamic

allocation of point primitives in the 3D space. Such flexibility benefits online applications such as SLAM with deformation-based loop closure [Whelan et al., 2015b, Schöps et al., 2019]. However, optimising the representation to capture high fidelity is challenging due to the lack of correlation among the primitives. Recently, in addition to classical graphic primitives, neural network-based map representations are a promising alternative. iMAP [Sucar et al., 2021] demonstrated the interesting properties of neural representation, such as sensible hole filling of unobserved geometry. Many recent approaches combine the classical and neural representations to capture finer details [Zhu et al., 2022, Sandström et al., 2023, Johari et al., 2023, Zhu et al., 2024]; however, the large amount of computation required for neural rendering makes the live operation of such systems challenging.

Differentiable Rendering: The classical method for creating a 3D representation was to unproject 2D observations into 3D space and to fuse them via weighted averaging [Newcombe et al., 2011b, McCormac et al., 2017]. Such an averaging scheme suffers from over-smooth representation and lacks the expressiveness to capture high-quality details. To capture a scene with photo-realistic quality, differentiable volumetric rendering [Niemeyer et al., 2020c] has recently been popularized with Neural Radiance Fields (NeRF) [Mildenhall et al., 2020]. Using a single Multi-Layer Perceptron (MLP) as a scene representation, NeRF performs volume rendering by marching along pixel rays, querying the MLP for opacity and color. Since volume rendering is naturally differentiable, the MLP representation is optimized to minimize the rendering loss using multiview information to achieve high-quality novel view synthesis. The main weakness of NeRF is its training speed. Recent developments have introduced explicit volume structures such as multi-resolution voxel grids [Fridovich-Keil et al., 2022, Sun et al., 2022a, Liu et al., 2020] or hash functions [Müller et al., 2022] to improve performance. Interestingly, these projects demonstrate that the main contributor to high-quality novel view synthesis is not the neural network but rather differentiable volumetric rendering, and that it is possible to avoid the use of an MLP and yet achieve comparable rendering quality to NeRF [Fridovich-Keil et al., 2022]. However, even in these systems, per-pixel ray marching remains a significant bottleneck for rendering speed. This issue is particularly critical in SLAM, where immediate interaction with the map is essential for tracking. In contrast to NeRF, 3DGS performs differentiable rasterization. Similar to regular graphics rasterizations, by iterating over the primitives to be rasterized rather than marching along rays, 3DGS leverages the natural sparsity of a 3D scene and achieves a representation which is expressive to capture high-fidelity 3D scenes while offering significantly faster rendering. Several works have applied 3D Gaussians and differentiable rendering to static scene capture [Keselman and Hebert, 2022, Wang et al., 2022], and in particular more recent works util-

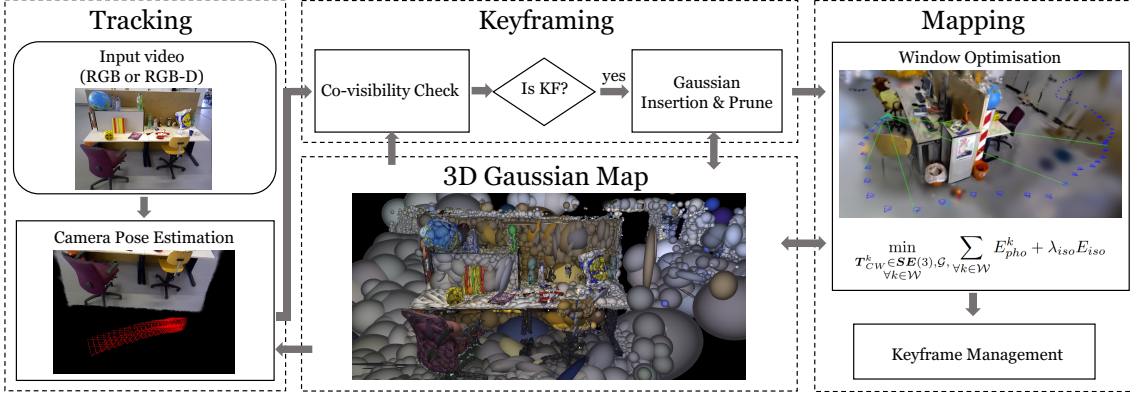


Figure 5.2: **SLAM System Overview:** Our SLAM system uses 3D Gaussians as the only representation, unifying all components of SLAM, including tracking, mapping, keyframe management, and novel view synthesis.

ize 3DGS and demonstrate superior results in vision tasks such as dynamic scene capture [Luiten et al., 2024, Yang et al., 2024b, Wu et al., 2024] and 3D generation [Tang et al., 2024, Yi et al., 2024]. Our method adopts a Map-centric approach, utilizing 3D Gaussians as the only SLAM representation. Similar to surfel-based SLAM, we dynamically allocate the 3D Gaussians, enabling us to model an arbitrary spatial distribution in the scene. Unlike other methods such as ElasticFusion [Whelan et al., 2015b] and PointFusion [Keller et al., 2013], however, by using differentiable rasterization, our SLAM system can capture high-fidelity scene details and represent challenging object properties by direct optimization against information from every pixel.

5.3 Method

5.3.1 Gaussian Splatting

Our SLAM representation is 3DGS, mapping the scene with a set of anisotropic Gaussians $\mathcal{G}$. Each Gaussian $\mathcal{G}^i$ contains optical properties: color c^i and opacity α^i . For continuous 3D representation, the mean μ_W^i and covariance Σ_W^i , defined in the world coordinate, represent the Gaussian’s position and its ellipsoidal shape. We omit the spherical harmonics (SHs) representing view-dependent radiance for simplicity. Since 3DGS uses volume rendering, explicit extraction of the surface is not required. Instead, by splatting and blending $\mathcal{N}$ Gaussians, a pixel color $\mathcal{C}_p$ is synthesized:

$$\mathcal{C}_p = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) . \quad (5.1)$$

3DGS performs rasterization, iterating over the Gaussians rather than marching along the camera rays, and hence, free spaces are ignored during rendering. During rasterization, the contributions

of α are decayed via a Gaussian function, based on the 2D Gaussian formed by splatting a 3D Gaussian. The 3D Gaussians $\mathcal{N}(\boldsymbol{\mu}_W, \boldsymbol{\Sigma}_W)$ in world coordinates are related to the 2D Gaussians $\mathcal{N}(\boldsymbol{\mu}_I, \boldsymbol{\Sigma}_I)$ on the image plane through a projective transformation:

$$\boldsymbol{\mu}_I = \pi(\mathbf{T}_{CW} \cdot \boldsymbol{\mu}_W), \boldsymbol{\Sigma}_I = \mathbf{J}\mathbf{W}\boldsymbol{\Sigma}_W\mathbf{W}^T\mathbf{J}^T, \quad (5.2)$$

where π is the projection operation and $\mathbf{T}_{CW} \in \mathbf{SE}(3)$ is the camera pose of the viewpoint. $\mathbf{J}$ is the Jacobian of the linear approximation of the projective transformation and $\mathbf{W} \in \mathbf{SO}(3)$ is the rotational component of $\mathbf{T}_{CW}$. This formulation enables the 3D Gaussians to be differentiable and the blending operation provides gradient flow to the Gaussians. Using first-order gradient descent [Kingma and Ba, 2015], Gaussians gradually refine both their optic and geometric parameters to represent the captured scene with high fidelity.

5.3.2 Camera Pose Optimization

To achieve accurate tracking, we typically require at least 50 iterations of gradient descent per frame. This requirement emphasizes the necessity of a representation with computationally efficient view synthesis and gradient computation, making the choice of 3D representation a crucial part of designing a SLAM system. In order to avoid the overhead of automatic differentiation, 3DGS implements rasterization with CUDA with derivatives for all parameters calculated explicitly. Since rasterization is performance critical, we similarly derive the camera Jacobians explicitly. To the best of our knowledge, we provide the first analytical Jacobian of $\mathbf{SE}(3)$ camera pose with respect to the 3D Gaussians used in EWA splatting [Zwicker et al., 2002] and 3DGS. This opens up new applications of 3DGS beyond SLAM.

We use Lie algebra to derive the minimal Jacobians, ensuring that the dimensionality of the Jacobians matches the degrees of freedom, eliminating any redundant computations. The terms of Eq. (5.2) are differentiable with respect to the camera pose $\mathbf{T}_{CW}$; using the chain rule:

$$\frac{\mathcal{D}\boldsymbol{\mu}_I}{\mathcal{D}\mathbf{T}_{CW}} = \frac{\partial\boldsymbol{\mu}_I}{\partial\boldsymbol{\mu}_C} \frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\mathbf{T}_{CW}}, \quad (5.3)$$

$$\frac{\mathcal{D}\boldsymbol{\Sigma}_I}{\mathcal{D}\mathbf{T}_{CW}} = \frac{\partial\boldsymbol{\Sigma}_I}{\partial\mathbf{J}} \frac{\partial\mathbf{J}}{\partial\boldsymbol{\mu}_C} \frac{\mathcal{D}\boldsymbol{\mu}_C}{\mathcal{D}\mathbf{T}_{CW}} + \frac{\partial\boldsymbol{\Sigma}_I}{\partial\mathbf{W}} \frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{T}_{CW}}. \quad (5.4)$$

where $\mathbf{T}_{CW}$ represents the 3D position of Gaussian in the camera coordinate. We take the derivatives on the manifold to derive minimal parameterization. Borrowing the notation from [Solà et al., 2018], let $\mathbf{T} \in \mathbf{SE}(3)$ and $\boldsymbol{\tau} = (\boldsymbol{\rho}, \boldsymbol{\theta}) \in \mathfrak{se}(3)$. We define the partial derivative on the manifold as:

$$\frac{\mathcal{D}f(\mathbf{T})}{\mathcal{D}\mathbf{T}} \triangleq \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{\text{Log}(f(\text{Exp}(\boldsymbol{\tau})) \circ \mathbf{T}) \circ f(\mathbf{T})^{-1}}{\boldsymbol{\tau}}, \quad (5.5)$$

where $\circ$ is a group composition, and Exp, Log are the exponential and logarithmic mappings between Lie algebra and Lie Group.

With this, we provide the derivation of the Jacobian:

$$\frac{\mathcal{D}\mu_C}{\mathcal{D}\mathbf{T}_{CW}} = \lim_{\tau \rightarrow 0} \frac{\text{Exp}(\tau) \cdot \mu_C - \mu_C}{\tau} \quad (5.6)$$

$$= \lim_{\tau \rightarrow 0} \frac{(\mathbf{I} + \tau^\wedge) \cdot \mu_C - \mu_C}{\tau} \quad (5.7)$$

$$= \lim_{\tau \rightarrow 0} \frac{\tau^\wedge \cdot \mu_C}{\tau} \quad (5.8)$$

$$= \lim_{\tau \rightarrow 0} \frac{\theta^\times \mu_C + \rho}{\tau} \quad (5.9)$$

$$= \lim_{\tau \rightarrow 0} \frac{-\mu_C^\times \theta + \rho}{\tau} \quad (5.10)$$

$$= \begin{bmatrix} \mathbf{I} & -\mu_C^\times \end{bmatrix} \quad (5.11)$$

where $\mathbf{T} \cdot \mathbf{x}$ is the group action of $\mathbf{T} \in \text{SE}(3)$ on $\mathbf{x} \in \mathbb{R}^3$.

Similarly, we compute the Jacobian with respect to $\mathbf{W}$. Since the translational component is not involved, we only consider the rotational part $\mathbf{R}_{CW}$ of $\mathbf{T}_{CW}$.

$$\begin{aligned} \mathbf{R}_{CW} &= \text{Exp}(\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}^\wedge) \\ &= \exp \left(\sum_{j=1}^3 \mathbf{G}_j \tau_j \right), \quad \boldsymbol{\theta} \in \mathbb{R}^3. \end{aligned} \quad (5.12)$$

where the matrices $\mathbf{G}_j \in \mathbb{R}^{3 \times 3}$, $j = 1, \dots, 3$ are the $\text{SO}(3)$ group generators and form a basis for $\mathfrak{so}(3)$:

$$\mathbf{G}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathbf{G}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad \mathbf{G}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (5.13)$$

Therefore, the full derivative is given as:

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{R}_{CW}} = \frac{\partial \mathbf{W}}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=0} = \mathbf{W} \frac{\partial \left(\sum_{j=1}^3 \mathbf{G}_j \theta_j \right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=0} \quad (5.14)$$

After column-wise vectorization, and stacking horizontally we get:

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{R}_{CW}} = \begin{bmatrix} -\mathbf{W}_{:,1}^\times \\ -\mathbf{W}_{:,2}^\times \\ -\mathbf{W}_{:,3}^\times \end{bmatrix}, \quad (5.15)$$

where $\mathbf{W}_{:,i}$ refers to the i th column of the matrix. Since the translational part is all zeros, we get

$$\frac{\mathcal{D}\mathbf{W}}{\mathcal{D}\mathbf{T}_{CW}} = \begin{bmatrix} \mathbf{0} & -\mathbf{W}_{:,1}^\times \\ \mathbf{0} & -\mathbf{W}_{:,2}^\times \\ \mathbf{0} & -\mathbf{W}_{:,3}^\times \end{bmatrix}, \quad (5.16)$$

5.3.3 SLAM

In this section, we present details of full SLAM framework. The overview of the system is summarized in Fig. 5.2.

In tracking only the current camera pose is optimized, without updates to the map representation. In the monocular case, we minimize the following photometric residual:

$$E_{pho} = \|I(\mathcal{G}, \mathbf{T}_{CW}) - \bar{I}\|_1, \quad (5.17)$$

where $I(\mathcal{G}, \mathbf{T}_{CW})$ renders the Gaussians $\mathcal{G}$ from $\mathbf{T}_{CW}$, and $\bar{I}$ is an observed image.

We further optimize affine brightness parameters for varying exposure and penalize non-edge or low-opacity pixels. When depth observations are available, we define the geometric residual as:

$$E_{geo} = \|D(\mathcal{G}, \mathbf{T}_{CW}) - \bar{D}\|_1, \quad (5.18)$$

where $D(\mathcal{G}, \mathbf{T}_{CW})$ is depth rasterization and $\bar{D}$ is the observed depth. Rather than simply using the depth measurements to initialize the Gaussians, we minimize both photometric and geometric residuals: $\lambda_{pho}E_{pho} + (1 - \lambda_{pho})E_{geo}$, where λ_{pho} is a hyperparameter.

As in Eq. (5.1), per-pixel depth is rasterized by alpha-blending:

$$\mathcal{D}_p = \sum_{i \in \mathcal{N}} z_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (5.19)$$

where z_i is the distance to the mean $\boldsymbol{\mu}_W$ of Gaussian i along the camera ray. We derive analytical Jacobians for the camera pose optimization in a similar manner to Eq. (5.3), (5.4).

Keyframing

Since using all the images from a video stream to jointly optimize the Gaussians and camera poses online is infeasible, we maintain a small window $\mathcal{W}_k$ consisting of carefully selected keyframes based on inter-frame covisibility. Ideal keyframe management will select non-redundant keyframes observing the same area, spanning a wide baseline to provide better multiview constraints.

Selection and Management Every tracked frame is checked for keyframe registration based on our simple yet effective criteria. We measure the covisibility by measuring the intersection over the union of the observed Gaussians between the current frame i and the last keyframe j . If the covisibility drops below a threshold, or if the relative translation t_{ij} is large with respect to the median depth, frame i is registered as a keyframe. For efficiency, we maintain only a small number of keyframes in the current window $\mathcal{W}_k$ following the keyframe management heuristics of DSO [Engel et al., 2017]. The main difference is that a keyframe is removed from the current window if the overlap coefficient with the latest keyframe drops below a threshold.

Gaussian Covisibility An accurate estimate of covisibility simplifies keyframe selection and management. 3DGS respects visibility ordering since the 3D Gaussians are sorted along the camera ray. This property is desirable for covisibility estimation as occlusions are handled by design. A Gaussian is marked to be visible from a view if used in the rasterization and if the ray’s accumulated α has not yet reached 0.5. This enables our estimated covisibility to handle occlusions without requiring additional heuristics.

Between two keyframes i, j , we define the covisibility using the Intersection of Union (IOU) and Overlap Coefficient (OC):

$$IOU_{cov}(i, j) = \frac{|\mathcal{G}_i^v \cap \mathcal{G}_j^v|}{|\mathcal{G}_i^v \cup \mathcal{G}_j^v|}, \quad (5.20)$$

$$OC_{cov}(i, j) = \frac{|\mathcal{G}_i^v \cap \mathcal{G}_j^v|}{\min(|\mathcal{G}_i^v|, |\mathcal{G}_j^v|)}, \quad (5.21)$$

where $\mathcal{G}_i^v$ is the Gaussians visible in keyframe i , based on visibility check described in Section 5.3.3, Gaussian Covisibility. A keyframe i is added to the keyframe window $\mathcal{W}_k$ if given last keyframe j , $IOU_{cov}(i, j) < kf_{cov}$ or if the relative translation $t_{ij} > kf_m \hat{D}_i$, where $\hat{D}_i$ is the median depth of frame i . For Replica $kf_{cov} = 0.95, kf_m = 0.04$ and for TUM $kf_{cov} = 0.90, kf_m = 0.08$. We remove the registered keyframe j in $\mathcal{W}_k$ if the $OC_{cov}(i, j) < kf_c$, where keyframe i is

the latest added keyframe. For both Replica and TUM, we set the cutoff to $k f_c = 0.3$. We set the size of the keyframe window to be for Replica, $|\mathcal{W}_k| = 10$, and for TUM, $|\mathcal{W}_k| = 8$.

Gaussian Insertion and Pruning At every keyframe, new Gaussians are inserted into the scene to capture newly visible scene elements and to refine the fine details. When depth measurements are available, Gaussian means μ_W are initialized by back-projecting the depth. In the monocular case, we render the depth at the current frame. For pixels with depth estimates, μ_W are initialized around those depths with low variance; for pixels without the depth estimates, we initialize μ_W around the median depth of the rendered image with high variance.

In the monocular case, the positions of many newly inserted Gaussians are incorrect. While the majority will quickly vanish during optimization as they violate multiview consistency, we further prune the excess Gaussians by checking the visibility amongst the current window $\mathcal{W}_k$. If the Gaussians inserted within the last 3 keyframes are unobserved by at least 3 other frames, we prune them out as they are geometrically unstable.

When **inserting** new Gaussians in a monocular setting, we randomly sample the Gaussians position μ_W using rendered depth D . Since the estimated depth may sometimes be incorrect, we account for this by initialising the Gaussians with some variance. For a pixel p where the rendered depth $\mathcal{D}_p$ exists, we sample the depth from $\mathcal{N}(\mathcal{D}_p, 0.2\sigma_D)$. Otherwise, for unobserved regions, we initialize the Gaussians by sampling from $\mathcal{N}(\hat{D}, 0.5\sigma_D)$, where $\hat{D}$ is the median of D . For **pruning**, as described in Section 5.3.3, we perform visibility-based pruning, where if new Gaussians inserted within the last 3 keyframes are not observed by at least 3 other frames, they are pruned. We only perform visibility-based pruning once the keyframe window $\mathcal{W}_k$ is full. Additionally, we prune all Gaussians with opacity of less than 0.7.

Mapping

The purpose of mapping is to maintain a coherent 3D structure and to optimize the newly inserted Gaussians. During mapping, the keyframes in $\mathcal{W}_k$ are used to reconstruct currently visible regions. Additionally, two random past keyframes $\mathcal{W}_r$ are selected per iteration to avoid forgetting the global map. Rasterization of 3DGS imposes no constraint on the Gaussians along the viewing ray direction, even with a depth observation. This is not a problem when sufficient carefully selected viewpoints are provided (e.g. in the novel view synthesis case); however, in continuous SLAM this causes many artefacts, making tracking challenging. We therefore introduce an isotropic

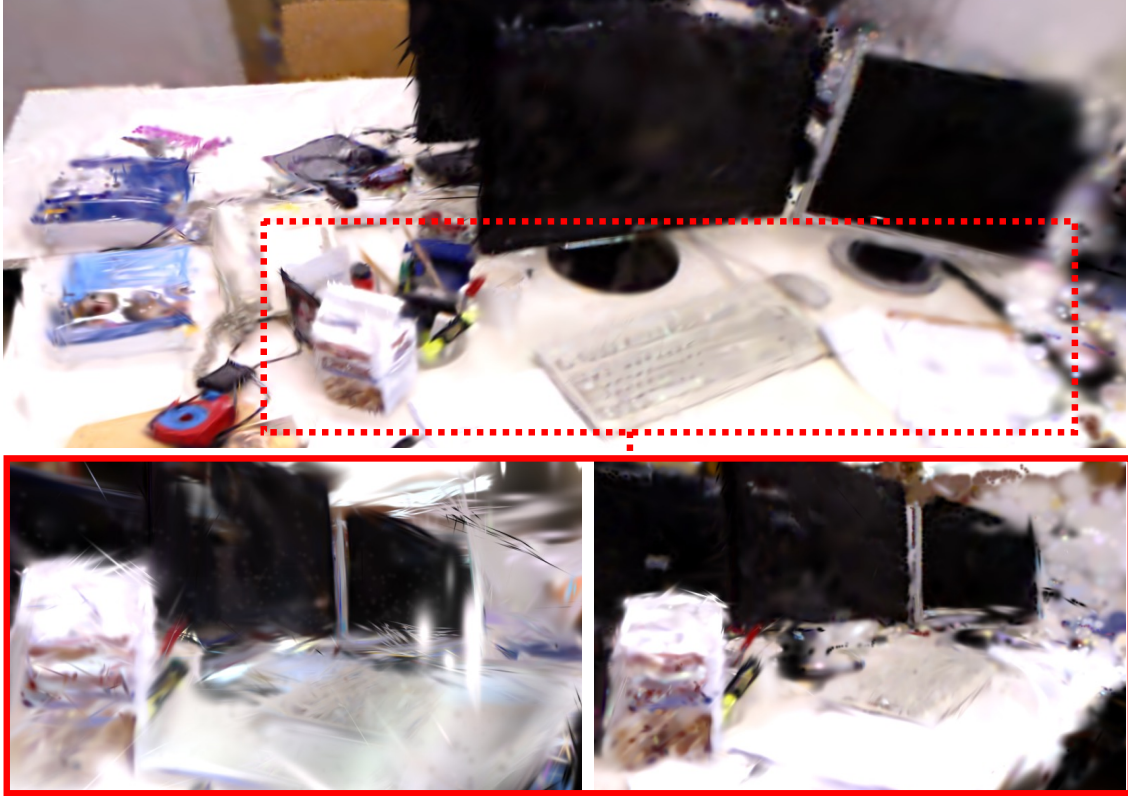


Figure 5.3: **Effect of isotropic regularization:** **Top:** Rendering close to a training view (looking at the keyboard). **Bottom:** Rendering 3D Gaussians far from the training views (view from a side of the keyboard) without (left) and with (right) the isotropic loss. When the photometric constraints are insufficient, the Gaussians tend to elongate along the viewing direction, creating artefacts in the novel views, and affecting the camera tracking.

regularization:

$$E_{iso} = \sum_{i=1}^{|\mathcal{G}|} \|\mathbf{s}_i - \tilde{\mathbf{s}}_i \cdot \mathbf{1}\|_1 \quad (5.22)$$

to penalize the scaling parameters $\mathbf{s}_i$ (i.e. stretch of the ellipsoid) by its difference to the mean $\tilde{\mathbf{s}}_i$. As shown in Fig 5.3, this encourages sphericity, and avoids the problem of Gaussians which are highly elongated along the viewing direction creating artefacts. Let the union of the keyframes in the current window and the randomly selected one be $\mathcal{W} = \mathcal{W}_k \cup \mathcal{W}_r$. For mapping, we solve the following problem:

$$\min_{\substack{T_{CW}^k \in \mathbf{SE}(3), \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{\forall k \in \mathcal{W}} E_{pho}^k + \lambda_{iso} E_{iso} . \quad (5.23)$$

If depth observations are available, as in tracking, geometric residuals Eq. (5.18) are added to the optimization problem.

5.4 Evaluation

We conduct a comprehensive evaluation of our system across a range of both real and synthetic datasets. Additionally, we perform an ablation study to justify our design choices. Finally, we present qualitative results of our system operating live using a monocular camera, illustrating its practicality and high fidelity reconstruction.

5.4.1 Experimental Setup

Datasets For our quantitative analysis, we evaluate our method on the TUM RGB-D dataset [Sturm et al., 2012] (3 sequences) and the Replica dataset [Straub et al., 2019] (8 sequences), following the evaluation in [Sucar et al., 2021]. For qualitative results, we use self-captured real-world sequences recorded by Intel Realsense d455. Since the Replica dataset is designed for RGB-D SLAM evaluation, it contains challenging purely rotational camera motions. We hence use the Replica dataset for RGB-D evaluation only. The TUM RGB-D dataset is used for both monocular and RGB-D evaluation.

Implementation Details We run our SLAM on a desktop with Intel Core i9 12900K 3.50GHz and a single NVIDIA GeForce RTX 4090. We present results from our multi-process implementation aimed at real-time applications. For a fair comparison with other methods on Replica, we additionally report result for single-process implementation which performs more mapping iterations. As with 3DGS, time-critical rasterization and gradient computation are implemented using CUDA. The rest of the SLAM pipeline is developed with PyTorch.

Learning Rates We use the Adam optimizer for both camera poses and Gaussian parameters optimization. For camera poses, we used 0.003 for rotation and 0.001 for translation. For 3D Gaussians, we used the default learning parameters of the original Gaussian Splatting implementation [Kerbl et al., 2023], apart from in monocular setting where we increase the learning rate of the positions of the Gaussians μ_W by a factor of 10.

Iteration numbers 100 tracking iterations are performed per frame for across all experiments. However, we terminate the iterations early if the magnitude of the pose update becomes less than 10^{-4} . For mapping, 150 iterations are used for the single-process implementation.

Loss Weights Given a depth observation, for tracking we minimize both photometric Eq. (5.17) and geometric residual Eq. (5.18) as:

$$\min_{\mathbf{T}_{CW} \in \mathbf{SE}(3)} \lambda_{pho} E_{pho} + (1 - \lambda_{pho}) E_{geo} , \quad (5.24)$$

and similarly, for mapping we modify Eq. (5.23) to:

$$\min_{\substack{\mathbf{T}_{CW}^k \in \mathbf{SE}(3), \mathcal{G}, \\ \forall k \in \mathcal{W}}} \sum_{\forall k \in \mathcal{W}} (\lambda_{pho} E_{pho}^k + (1 - \lambda_{pho}) E_{geo}^k) + \lambda_{iso} E_{iso} . \quad (5.25)$$

We set $\lambda_{pho} = 0.9$ for all RGB-D experiments, and $\lambda_{iso} = 10$ for both monocular and RGB-D experiments.

Metrics For camera tracking accuracy, we report the Root Mean Square Error (RMSE) of the Absolute Trajectory Error (ATE) of the keyframes. To evaluate map quality, we report standard photometric rendering quality metrics (PSNR, SSIM and LPIPS) following the evaluation protocol used in [Sandström et al., 2023]. To evaluate the map quality, on every fifth frame, rendering metrics are computed. We exclude the keyframes (training views). We report the average across three runs for all our evaluations. In the tables, the best result is in bold, and the second best is underlined.

Baseline Methods We primarily benchmark our SLAM method against other approaches that, like ours, do not have explicit loop closure. In monocular settings, we compare with state-of-the-art classical and learning-based direct visual odometry (VO) methods. Specifically, we compare DSO [Engel et al., 2017], DepthCov [Dexheimer and Davison, 2023], and DROID-SLAM [Teed and Deng, 2021] in VO configurations. These methods are selected based on their public reporting of results on the benchmark (TUM dataset) or the availability of their source code for getting the benchmark result. Since one of our focuses is the online scale estimation under monocular scale ambiguity, the method which uses ground truth poses for the system initialization such as [Li et al., 2023] is not considered for the comparison. In the RGB-D case, we compare against neural-implicit SLAM methods [Sucar et al., 2021, Zhu et al., 2022, Huang et al., 2021, Yang et al., 2022, Johari et al., 2023, Wang et al., 2023, Sandström et al., 2023] which are also map-centric, rendering-based and do not perform loop closure.

5.4.2 Quantitative Evaluation

Camera Tracking Accuracy

Table 5.1 shows the tracking results on the TUM RGB-D dataset. In the monocular setting, our method surpasses other baselines without requiring any deep priors. Furthermore, our performance is comparable to systems which perform explicit loop closure. This clearly highlights that there still remains potential for enhancing the tracking of monocular SLAM by exploring fundamental SLAM representations.

Our RGB-D method shows better performance than any other baseline method. Notably, our system surpasses ORB-SLAM in the fr1 sequences, narrowing the gap between Map-centric SLAM and the state-of-the-art sparse frame-centric methods. Table 5.2 reports results on the synthetic Replica dataset. Our single-process implementation shows competitive performance and achieves the best result in 6 out of 8 sequences. Our multi-process implementation which performs fewer mapping iterations still performs comparably. In contrast to other methods, our system demonstrates higher performance on real-world data (TUM RGB-D), by optimising the Gaussian positions to compensate for the sensor noise.

Novel View Rendering

Table 5.5 summarizes the novel view rendering performance of our method with RGB-D input. We consistently show the best performance across most sequences and is least second best. Our rendering FPS is hundreds of times faster than other methods, offering a significant advantage for applications which require real-time map interaction. While Point-SLAM is competitive, that method focuses on view synthesis rather than novel-view synthesis. Their view synthesis is conditional on the availability of depth due to the depth-guided ray-sampling, making novel-view synthesis challenging. On the other hand, our rasterization-based approach does not require depth guidance and achieves efficient, high-quality, novel view synthesis. Fig. 5.4 provides a qualitative comparison of the rendering of ours and Point-SLAM (with depth guidance). The rendering time refers to the duration necessary for full-resolution rendering (1200×680 for the Replica sequence). For each method, we perform 100 renderings and report the average time taken per rendering. The reported rendering fps is found by taking 1 and dividing it by the average rendering time. Note that the rendering FPS means the fps just for the forward rendering, which differs from the end-to-end system FPS reported in Table 5.12 and 5.13.

Input	Loop-closure	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Monocular	w/o	DSO [Engel et al., 2017]	22.4	1.10	9.50	11.0
		DROID-VO [Teed and Deng, 2021]	<u>5.20</u>	10.7	<u>7.30</u>	<u>7.73</u>
		DepthCov-VO [Dexheimer and Davison, 2023]	5.60	<u>1.20</u>	68.8	25.2
		Ours	3.78	4.60	3.50	3.96
	w/	DROID-SLAM [Teed and Deng, 2021]	1.80	0.50	2.80	1.70
		ORB-SLAM2 [Mur-Artal and Tardós, 2017a]	1.90	0.60	2.40	1.60
RGB-D	w/o	iMAP [Sucar et al., 2021]	4.90	2.00	5.80	4.23
		NICE-SLAM [Zhu et al., 2022]	4.26	6.19	3.87	4.77
		DI-Fusion [Huang et al., 2021]	4.40	2.00	5.80	4.07
		Vox-Fusion [Yang et al., 2022]	3.52	1.49	26.01	10.34
		ESLAM [Johari et al., 2023]	2.47	1.11	2.42	<u>2.00</u>
		Co-SLAM [Wang et al., 2023]	<u>2.40</u>	1.70	<u>2.40</u>	2.17
		Point-SLAM [Sandström et al., 2023]	4.34	<u>1.31</u>	3.48	3.04
		Ours	1.50	1.44	1.49	1.47
	w/	BAD-SLAM [Schöps et al., 2019]	1.70	1.10	1.70	1.50
		Kintinous [Whelan et al., 2015a]	3.70	2.90	3.00	3.20
		ORB-SLAM2 [Mur-Artal and Tardós, 2017a]	1.60	0.40	1.00	1.00

Table 5.1: **Camera tracking result on TUM for monocular and RGB-D.** ATE RMSE in cm is reported. In both monocular and RGB-D cases, we achieve state-of-the-art performance. In particular, in the monocular case, not only do we outperform systems which use deep prior, but we achieve comparable performance with many of the RGB-D systems.

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
iMAP [Sucar et al., 2021]	3.12	2.54	2.31	1.69	1.03	3.99	4.05	1.93	2.58
NICE-SLAM	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.07
Vox-Fusion [Yang et al., 2022]	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
ESLAM [Johari et al., 2023]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	0.63	0.63
Point-SLAM [Sandström et al., 2023]	0.61	0.41	0.37	<u>0.38</u>	<u>0.48</u>	0.54	0.69	<u>0.72</u>	<u>0.53</u>
Ours	<u>0.44</u>	<u>0.32</u>	<u>0.31</u>	0.44	0.52	0.23	<u>0.17</u>	2.25	0.58
Ours (sp)	0.33	0.22	0.29	0.36	0.19	<u>0.25</u>	0.12	0.81	0.32

Table 5.2: **Camera tracking result on Replica for RGB-D SLAM.** ATE RMSE in cm is reported. We achieve best performance across most sequences. Here, Ours is our multi-process implementation and Ours (sp) is the single-process implementation which ensures a certain amount of mapping iteration similar to other works.

Ablative Analysis

In Table 5.3, we perform ablation to confirm our design choices. Isotropic regularization and geometric residual improve the tracking of monocular and RGB-D SLAM respectively, as they aid in constraining the geometry when photometric signals are weak. For both cases, keyframe selection

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	w/o E_{iso}	4.16	4.66	5.73	4.83
	w/o kf selection	13.2	4.36	8.65	8.73
	Ours	3.78	4.60	3.50	3.96
RGB-D	w/o E_{geo}	2.39	0.62	4.98	2.66
	w/o kf selection	1.64	1.49	2.60	1.90
	Ours	1.50	1.44	1.49	1.47

Table 5.3: **Ablation Study on TUM RGB-D dataset.** We analyse the usefulness of isotropic regularization, geometric residual, and keyframe selection to our SLAM system.

Memory Usage [MB]				
iMAP [Sucar et al., 2021]	NICE-SLAM [Zhu et al., 2022]	Co-SLAM [Wang et al., 2023]	Ours (Mono)	Ours (RGB-D)
0.8MB	40.34MB	6.4MB	<u>2.6MB</u>	3.97MB

Table 5.4: **Memory Analysis on TUM RGB-D dataset.** The baseline numbers are computed from the parameter numbers in [Wang et al., 2023]

Method	Metric	r0	r1	r2	o0	o1	o2	o3	o4	Avg.	FPS
NICE-SLAM	PSNR↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42	0.54
	SSIM↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809	
	LPIPS↓	0.33	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233	
Vox-Fusion	PSNR↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41	<u>2.17</u>
	SSIM↑	0.683	0.751	0.798	0.857	0.876	0.794	0.803	0.847	0.801	
	LPIPS↓	0.303	0.269	0.234	0.241	0.184	0.243	0.213	0.199	0.236	
Point-SLAM	PSNR↑	<u>32.40</u>	<u>34.08</u>	<u>35.5</u>	<u>38.26</u>	<u>39.16</u>	<u>33.99</u>	<u>33.48</u>	<u>33.49</u>	<u>35.17</u>	1.33
	SSIM↑	0.974	0.977	0.982	0.983	0.986	<u>0.96</u>	<u>0.960</u>	0.979	0.975	
	LPIPS↓	<u>0.113</u>	<u>0.116</u>	<u>0.111</u>	<u>0.1</u>	<u>0.118</u>	<u>0.156</u>	<u>0.132</u>	<u>0.142</u>	<u>0.124</u>	
Ours	PSNR↑	36.04	37.99	38.55	42.95	43.38	37.30	36.85	38.45	38.94	769
	SSIM↑	<u>0.959</u>	<u>0.965</u>	<u>0.968</u>	<u>0.983</u>	<u>0.981</u>	0.964	0.962	<u>0.965</u>	<u>0.968</u>	
	LPIPS↓	0.074	0.078	0.077	0.043	0.048	0.093	0.066	0.079	0.070	

Table 5.5: **Rendering performance comparison of RGB-D SLAM methods on Replica.** Our method outperforms most of the rendering metrics compared to existing methods. Note that Point-SLAM uses ground-truth depth to guide sampling along rays.

significantly improves system performance, as it automatically chooses suitable keyframes based on our occlusion-aware keyframe selection and management. We further compare the memory usage of different 3D representations in Table 5.4. MLP-based iMAP is clearly more memory efficient, but it struggles to express high-fidelity 3D scenes due to the limited capacity of small MLP. Compared with a voxel grid of features used in NICE-SLAM, our method uses significantly less memory.

Pruning Ablation (Monocular input) In Table 5.4.2, we report the ablation study of our proposed Gaussian pruning, which prunes randomly initialized 3D Gaussians effectively in a monocular SLAM setting. The results demonstrate that Gaussian pruning plays a significant role in

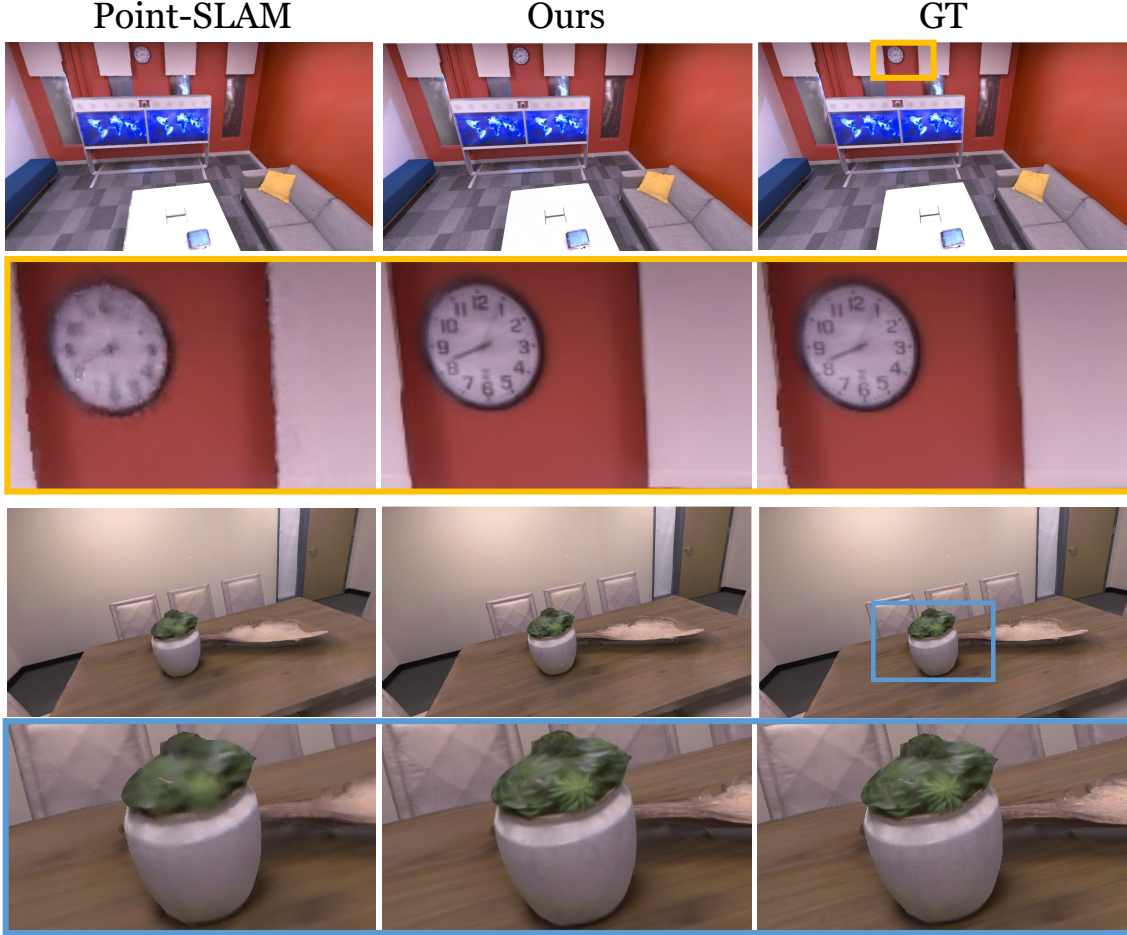


Figure 5.4: **Rendering examples on Replica.** Point-SLAM struggle with rendering fine details due to the stochastic ray sampling.

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
Mono	w/o pruning	78.2	4.5	57.0	46.6
	Ours	3.78	4.60	3.50	3.96

Table 5.6: **Pruning Ablation Study on TUM RGB-D dataset (Monocular Input).** Numbers are camera tracking error (ATE RMSE) in cm.

enhancing camera tracking performance. This improvement is primarily because, without pruning, randomly initialized Gaussians persist in the 3D space, potentially leading to incorrect initial geometry for other views.

Isotropic Loss Ablation (RGB-D input) Table 5.7 and 5.8 report the ablation study of the effect of isotropic loss E_{iso} for RGB-D input. In TUM, as Table 5.7 shows, isotropic regularization does not improve the performance but only shows a marginal difference. However, for Replica, as summarized in Table 5.8, isotropic loss significantly improves camera tracking performance. Even with the depth measurement, since rasterization does not consider the elongation along the viewing axis. Isotropic regularization is required to prevent the Gaussians from over-stretching,

Input	Method	fr1/desk	fr2/xyz	fr3/office	Avg.
RGB-D	w/o E_{iso}	1.60	1.42	1.32	1.43
	Ours	1.50	1.44	1.49	1.47

Table 5.7: **Isotropic Loss Ablation Study on TUM RGB-D dataset (RGB-D input)**. Numbers are camera tracking error (ATE RMSE) in cm.

Method	r0	r1	r2	o0	o1	o2	o3	o4	Avg.
w/o E_{iso}	0.44	0.86	0.28	0.75	0.99	0.36	0.28	2.6	0.82
Ours	0.44	0.32	0.31	0.44	0.52	0.23	0.17	2.25	0.58

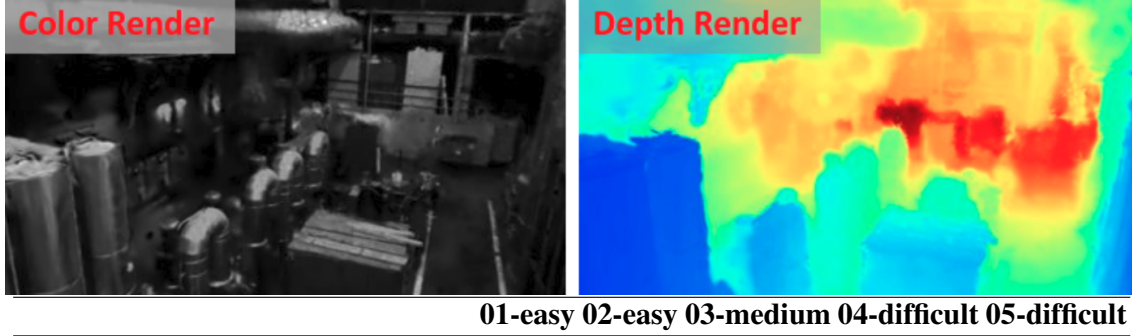
Table 5.8: **Isotropic Loss Ablation Study on Replica dataset (RGB-D input)**. Numbers are camera tracking error (ATE RMSE) in cm.

especially for textureless regions, which are common in Replica.

Effect of Spherical Harmonics (SH) While we disabled SHs in the main paper for simplicity, here we report the ablation study of the effect of SHs. The 3DGS paper [Kerbl et al., 2023] shows that addition of SH leads to small improvements in rendering metrics, and we have found similar improvement with SH enabled in our system (Tab. 5.10a). We did not observe a significant change in runtime with SH enabled, but it notably increases Gaussian map size and hence GPU memory usage. Though an analytical Jacobian propagates the gradients from SH to camera poses, ATE marginally gets worse when SH is enabled (Tab. 5.11), as SH may incorrectly explain non-view directional effects caused by the camera motion, degrading the trajectory estimate.

Mapping Performance with ORB-SLAM One of the most straightforward approaches for real-time operation is to combine an existing tracking system and 3DGS. In particular, frame-based SLAM methods have been well-studied for years and is capable of providing reliable tracking. In this section, we compare our unified 3DGS-based method to the combined approach. We have run RGB-D ORB-SLAM to recover the poses and train 3DGS with the poses and sensor depth of the keyframes, equivalent to performing offline splatting. Though ORB-SLAM is best in terms of ATE (Tab. 1 main), we find no significant difference across the rendering metrics (Tab. 5.10b). SH is omitted in the synthetic Replica dataset as it contains no view-directional effects. While using an off-the-shelf tracker with a 3DGS mapper is possible, this work has focused on the value of the 3DGS throughout the entire algorithms, which is unexplored and therefore provides new insights. Further performance improvement of the unified approach will be an interesting future work.

Large-scale Scenes with Stereo Inputs: This work focuses on pioneering 3DGS-based SLAM for live operation in small-scale scenes. However, we tested our method on the large-scale EuRoC



Point-SLAM [Sandström et al., 2023]	-	-	-	-	-
Ours	0.121	0.141	2.197	4.515	3.190
Vins-Fusion [Qin et al., 2019]	0.540	0.460	0.330	0.780	0.500
SVO [Forster et al., 2014]	0.040	0.070	0.270	0.170	0.120
ORB-SLAM3 [Campos et al., 2021a]	0.029	0.019	0.024	0.085	0.052

Table 5.9: ATE RMSE (meter) on EuRoC Machine Hall with Stereo Depth. Baseline numbers are from [Campos et al., 2021a]. The third best result is highlighted with a dash line.

Method	TUM			Replica		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
Ours (w/o SH)	21.89	0.733	0.327	38.94	0.968	0.0703
(a) Ours (w. SH)	24.37	0.804	0.225	-	-	-
Point-SLAM	21.39	0.727	0.463	24.37	0.840	0.185
(b) ORB+GS (w/o SH)	25.12	0.837	0.161	37.11	0.964	0.040
ORB+GS (w.SH)	25.44	0.842	0.146	-	-	-

Table 5.10: Mean Rendering metrics for TUM and Replica (RGBD).

Memory Usage for RGB-D SLAM					ATE RMSE	
Ours (w/o SH)	Ours (w. SH)	Point-SLAM	ORB+GS (w/o SH)	ORB+GS (w. SH)	Ours (w/o SH)	Ours (w. SH)
3.97MB	11.47MB	38.0MB	45.97MB	186.5MB	1.47cm	1.56cm

Table 5.11: Mean Memory and ATE metrics for TUM (RGBD).

Machine Hall dataset with depth from stereo (Tab.5.9). Fig.1 is a qualitative reconstruction result from our system. Our method is competitive in “easy” sequences, although performance drops for more difficult, longer sequences. Note that Point-SLAM [Sandström et al., 2023] fails on all sequences in this dataset (over 3m ATE within the first 500 frames). In future work, we expect to improve our method by incorporating loop closure. In principle, loop closure will be easier to incorporate compared to other representations such as voxel grids (where feature allocations are fixed), via a method similar to surfel-based approaches like ElasticFusion [Whelan et al., 2015b].

Timing Analysis To analyse the processing time of our monocular/RGB-D SLAM system, we measure the total time required to process all frames in the TUM-RGBD fr3/office dataset. This approach assesses the performance of our system as a whole, rather than isolating individual com-

Method	Total Time [s]	FPS
Monocular	798.9	3.2
RGB-D	986.7	2.5

Table 5.12: **Performance Analysis using fr3/office.** Both monocular and RGB-D implementations use multiprocessing. We report **the total execution time of our system**, FPS computed by dividing the total number of processed frames by the total time.

Method	Total Time [s]	FPS
RGB-D	1111.1	1.8
RGB-D (sp)	1904.7	1.1

Table 5.13: **Performance Analysis using replica/office1.** RGB-D uses a multi-process implementation and RGB-D-sp is the single-process implementation. We report **the total execution time of our system**, FPS computed by dividing the total number of processed frames by the total time.

ponents. By adopting this approach, we gain a more realistic understanding of the system’s true performance which better reflects the real-world operating conditions, as it avoids the assumption of an idealized, sequential interleaving of the tracking and mapping processes. As shown in Table 5.13, our system operates at 3.2 FPS with monocular and 2.5 FPS with depth. The FPS is found by dividing the number of processed frames by the total time. We conducted a similar analysis with the Replica dataset office2. Here, we compare the RGB-D method with and without multiprocessing. As expected, single-process implementation takes longer as it performs more mapping iterations.

Convergence Basin Analysis

In our SLAM experiments, we discovered that 3D Gaussian maps have a notably large convergence basin for camera localization. To investigate further, we conducted a convergence funnel analysis, an evaluation methodology proposed in [Mitra et al., 2004] and used in [Newcombe, 2012]. Here, we train a 3D representation (e.g. 3DGS) using 9 fixed views arranged in a square. We set the viewpoint in the middle of the square to be the target view. As shown in Fig 5.5, we uniformly sample a position, creating a funnel. From the sampled position, given the RGB image of the target view, we perform camera pose optimization for 1000 iterations. The optimization is successful if it converges to within 1cm of the target view within the fixed iterations.

The detail of the benchmark Dataset: For convergence basin analysis, we create three datasets by rendering the synthetic Replica dataset. In addition to the qualitative visualization in Figure 5.5, we report more detailed camera pose distributions in Figure 5.7. Figure 5.7 shows the camera view

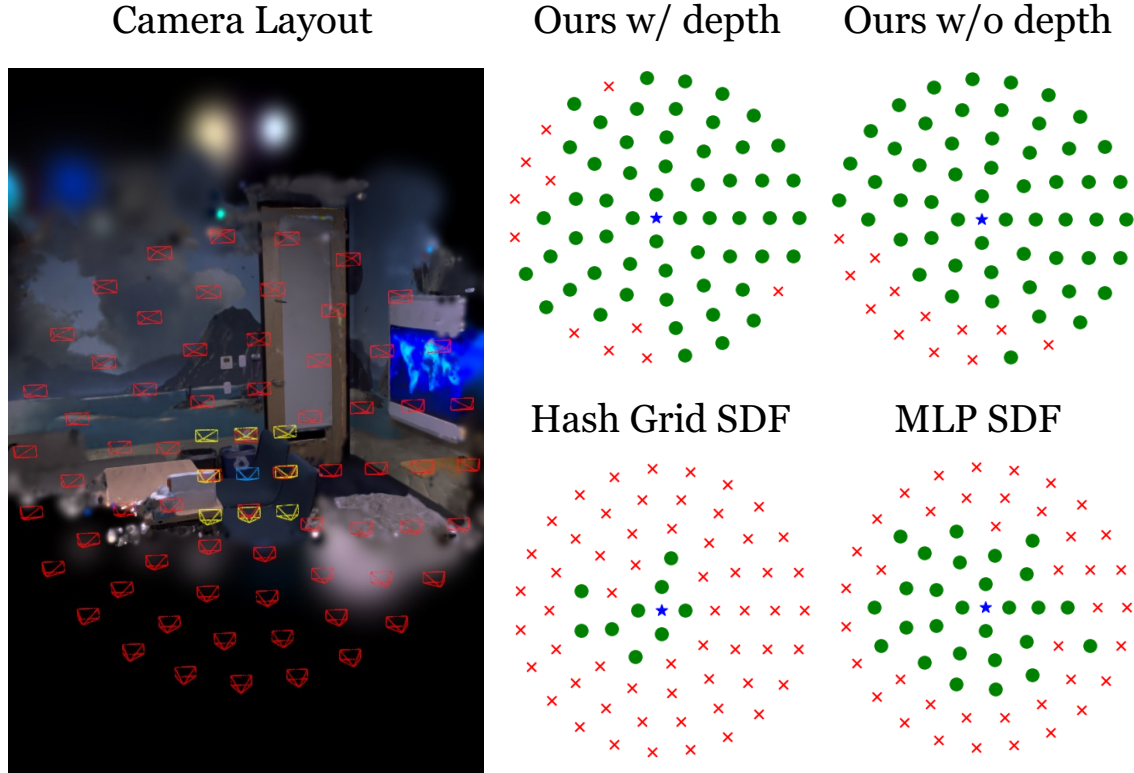


Figure 5.5: **Convergence basin analysis:** **Left:** 3D Gaussian map from training views (Yellow) and visualization of the test poses (Red) and target pose (Blue). **Right:** Convergence basin of our method. The green marks success, and the red marks failure.

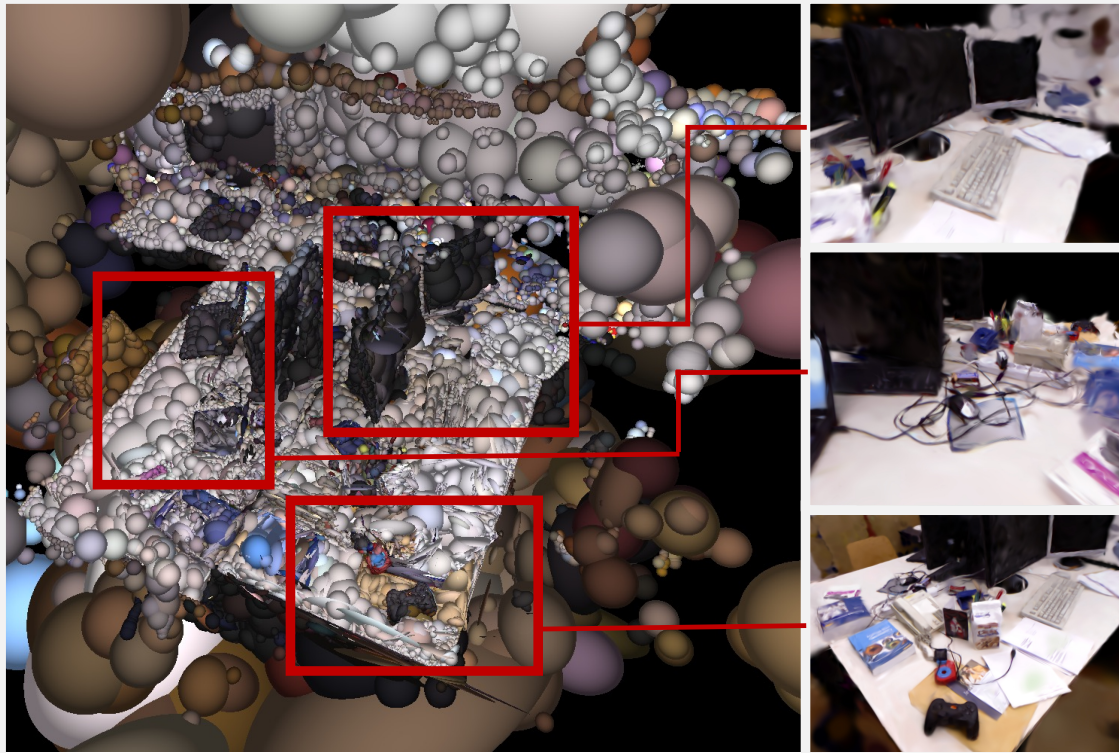


Figure 5.6: **Monocular SLAM result on fr1/desk sequence:** We show the reconstructed 3D Gaussian maps (Left) and novel view synthesis result (Right).

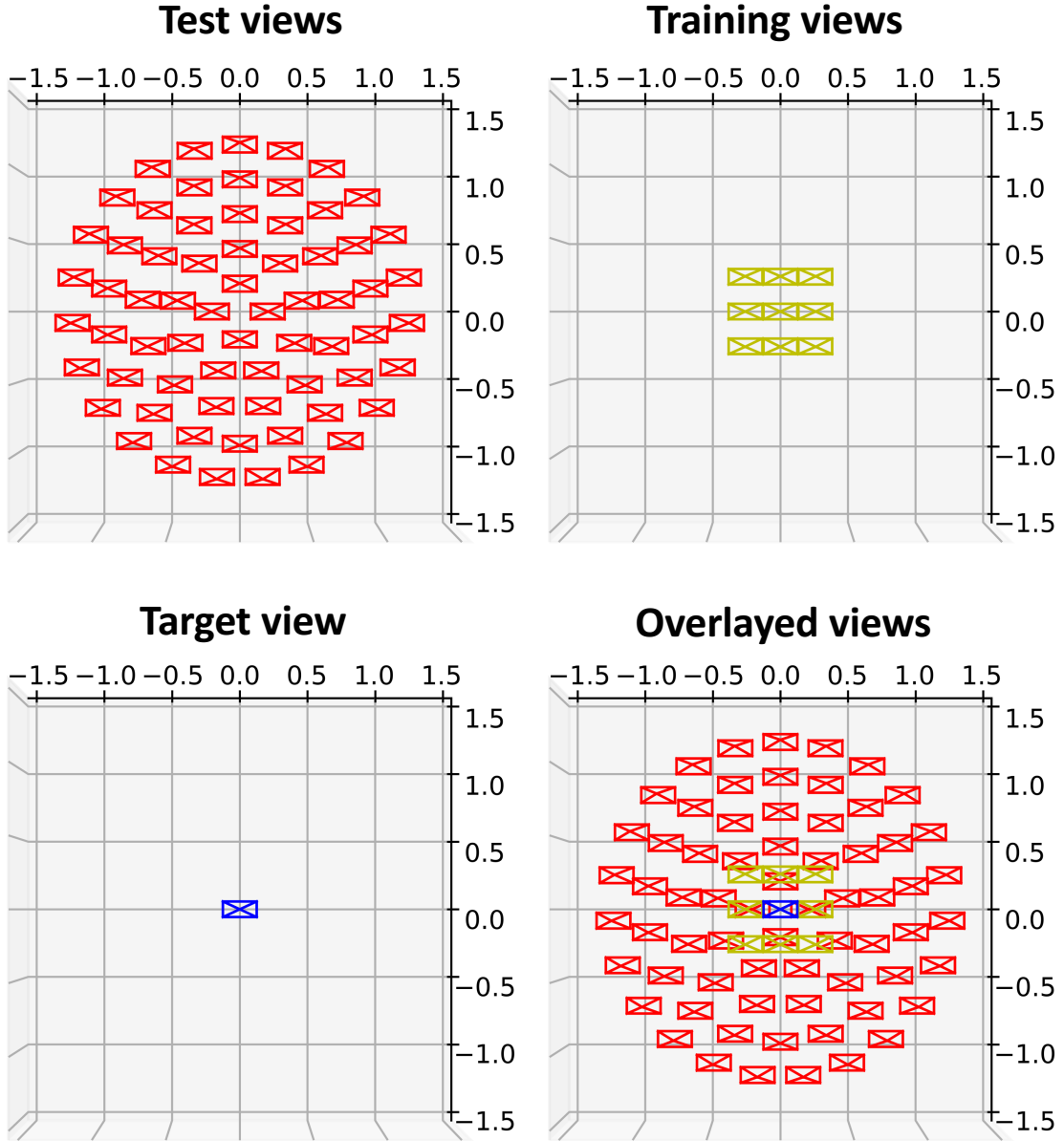


Figure 5.7: **2D Visualization of the camera pose distributions used for convergence basin analysis in Figure 5.5.**

frustums of the test (red), training (yellow) and target (blue) views. As we mentioned in the main paper, we set the training view in the shape of a square with a width of 0.5m and test views are distributed with radii ranging from 0.2m to 1.2m, covering a larger area than the training views. We only apply displacements to the camera translation but not to the rotation. For each sequence, we use a total of 67 test views.

Training setup: For each method, the 3D representation is trained for 30000 iterations using the training views. Here, we detail the training setup of each of the methods:

Ours We evaluated our method under two settings: “w/ depth” and “w/o depth”, where we train the initial 3D Gaussian map $\mathcal{G}_{init}$ with and without depth supervision. In the “w/o depth” setting, the 3D Gaussians’ positions are randomly initialized, and we minimize the monocular mapping cost Eq. (5.23) for the 3D Gaussian training, but keeping the camera poses fixed. Specifically, let $k \in \mathbb{N}$ be a number of training views and 3D Gaussians $\mathcal{G}$, we find $\mathcal{G}_{init}$ by:

$$\mathcal{G}_{init} = \arg \min_{\mathcal{G}} \sum_{\forall k \in \mathcal{W}} E_{pho}^k + \lambda_{iso} E_{iso} . \quad (5.26)$$

Note that training views’ camera poses $\mathbf{T}_{CW}^k$ are fixed during the optimization.

In the “w/ depth” setting, we train the Gaussian map by minimising

$$\begin{aligned} \mathcal{G}_{init} = \arg \min_{\mathcal{G}} \sum_{\forall k \in \mathcal{W}} (\lambda_{pho} E_{pho}^k + (1 - \lambda_{pho}) E_{geo}^k) \\ + \lambda_{iso} E_{iso} , \end{aligned} \quad (5.27)$$

where we use $\lambda_{pho} = 0.9$ and $\lambda_{iso} = 10$ for all the experiments

Baseline Methods For Hash Grid SDF, we trained the same network architecture as Co-SLAM [Wang et al., 2023]. For MLP SDF, we trained the network of iMAP [Sucar et al., 2021]. For both baselines, we supervised networks with the same loss functions as Co-SLAM, which are color rendering loss L_{rgb} , depth rendering loss L_{depth} , SDF loss L_{fs} , free-space loss L_{fs} , and smoothness loss L_{smooth} . Please refer to the original Co-SLAM paper for the exact formulation (equation (6) - (9)). All the training hyperparameters (e.g. learning rate of the network, number of sampling points, loss weight) are the same as Co-SLAM’s default configuration of the Replica dataset. While Co-SLAM stores training view information by downsampling the color and depth images, we store the full pixel information because the number of training views is small.

Testing Setup: For testing, we localize the camera pose by minimising only the photometric error against the ground-truth color image of the target view.

Ours Let the camera pose $\mathbf{T}_{CW} \in \mathbf{SE}(3)$ and initial 3D Gaussians $\mathcal{G}_{init}$, the localized camera pose $\mathbf{T}_{CW}^{est}$ is found by:

$$\mathbf{T}_{CW}^{est} = \arg \min_{\mathbf{T}_{CW}} \|I(\mathcal{G}_{init}, \mathbf{T}_{CW}) - \bar{I}_{target}\|_1 . \quad (5.28)$$

Note that $\mathcal{G}_{init}$ is fixed during the optimization. We initialize T_{CW} at one of the test view’s positions, and optimization is performed for 1000 iterations. We perform this localization process for all the test views and measure the success rate. Camera localization is successful if the estimated pose converges to within 1cm of the target view within the 1000 iterations.

Baseline Methods For the baseline methods, the camera localization is performed by minimising color volume rendering loss L_{rgb} , while all the other trainable network parameters are fixed. The learning rates of the pose optimizer are also the same as Co-SLAM’s default configuration of Replica dataset.

Results: We compare our Gaussian approach with Co-SLAM [Wang et al., 2023]’s network (Hash Grid SDF) and iMAP’s [Sucar et al., 2021] network with Co-SLAM’s SDF loss for further geometric accuracy (MLP Neural SDF). We render the training views using a synthetic Replica dataset and create three sequences for testing (seq1, seq2 and seq3). The width of the square formed by the training view is 0.5m, and the test cameras are distributed with radii ranging from 0.2m to 1.2m, covering a larger area than the training view. When training the map, the three methods—Ours w/depth, Hash Grid SDF, and MLP SDF—use RGB-D images, whereas Ours w/o depth utilizes only color images. Fig. 5.5 shows the qualitative results and Table 5.14 reports the success rate. For both with and without depth for training, our method shows better convergence. Unlike hashing and positional encoding which can lead to signal conflict, anisotropic Gaussians form a smooth gradient in 3D space, increasing the convergence basin.

Method	seq1	seq2	seq3	Avg.
Neural SDF (Hash Grid)	0.13	0.15	0.16	0.14
Neural SDF (MLP)	0.40	0.38	0.22	0.33
Ours w/o depth	<u>0.82</u>	<u>0.91</u>	0.65	<u>0.79</u>
Ours w/ depth	0.83	1.0	0.65	0.82

Table 5.14: **Camera convergence analysis.** We report the ratio of successful camera convergence for the different sequences, across different differentiable 3D representations.

5.4.3 Qualitative Results

We report both the 3D reconstruction of the SLAM dataset and self-captured sequences. In Fig. 5.6, we visualize the monocular SLAM reconstruction of fr1/desk. The placements of the Gaussians are geometrically sensible and are 3D coherent, and our rendering from the different viewpoints highlights the quality of our systems’ novel view synthesis. In Fig. 5.8, we self-capture



Figure 5.8: **Self-captured Scenes:** Challenging scenes and objects, for example, transparent glasses and crinkled texture of salad are captured by our monocular SLAM running live.

challenging scenes for monocular SLAM. By not explicitly modelling a surface, our system naturally handles transparent objects which is challenging for many other SLAM systems.

In Fig. 5.9 - Fig. 5.16, we further show additional qualitative results. We visually compare other state-of-the-art SLAM methods using differentiable rendering (Point-SLAM [Sandström et al., 2023] and ESLAM [Johari et al., 2023]).

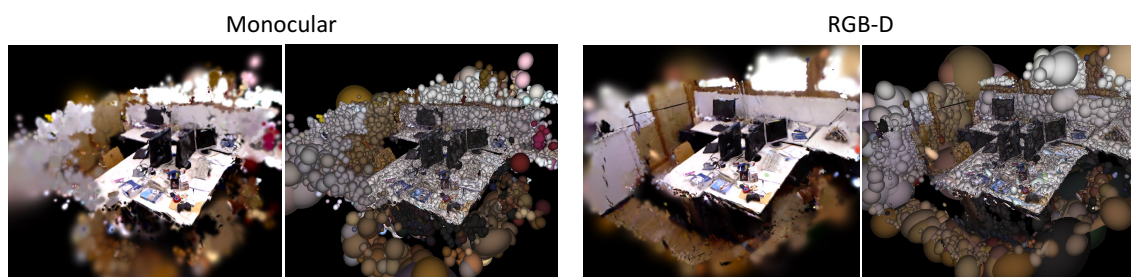


Figure 5.9: Novel view rendering and Gaussian visualizations on TUM fr1/desk

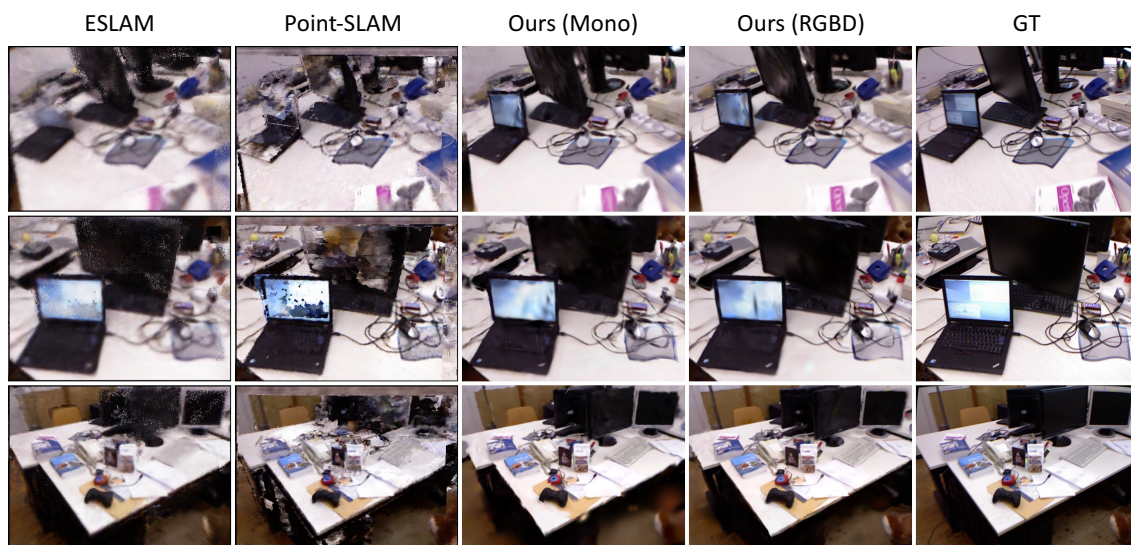


Figure 5.10: Rendering comparison on TUM fr1/desk

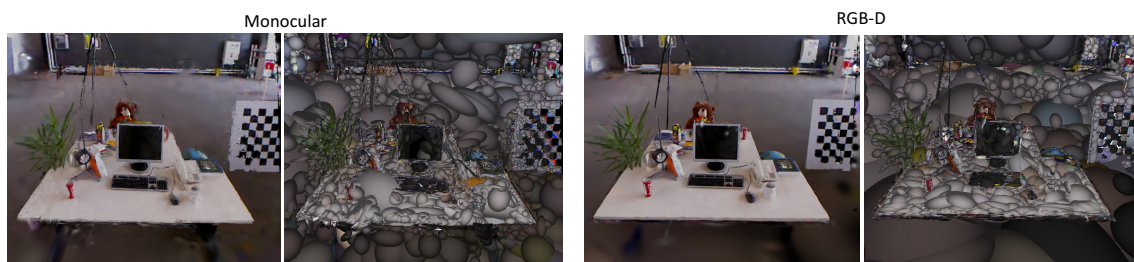


Figure 5.11: Novel view rendering and Gaussian visualizations on TUM fr2/xyz

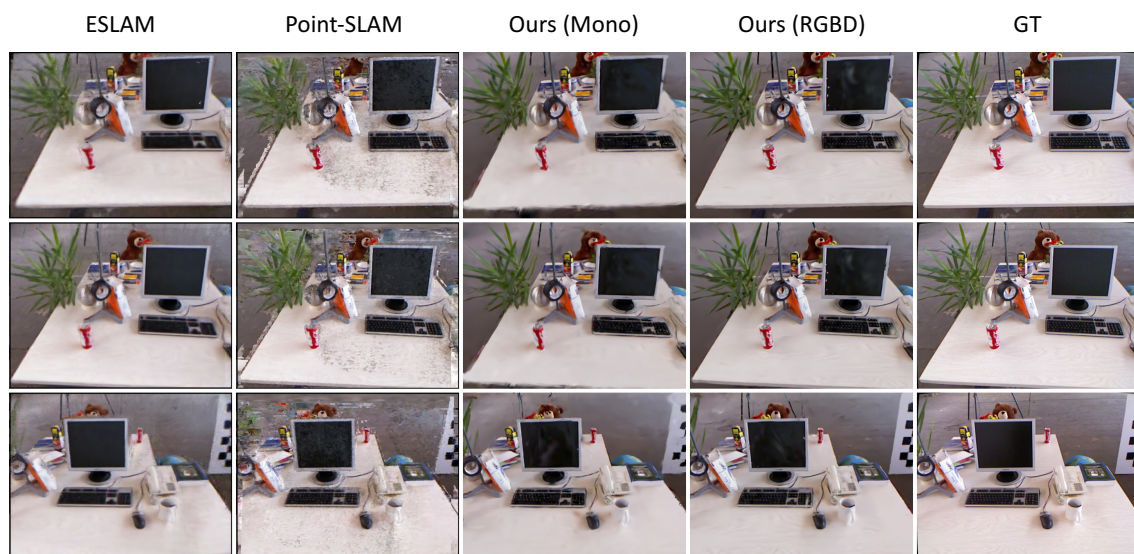


Figure 5.12: Rendering comparison on TUM fr2/xyz

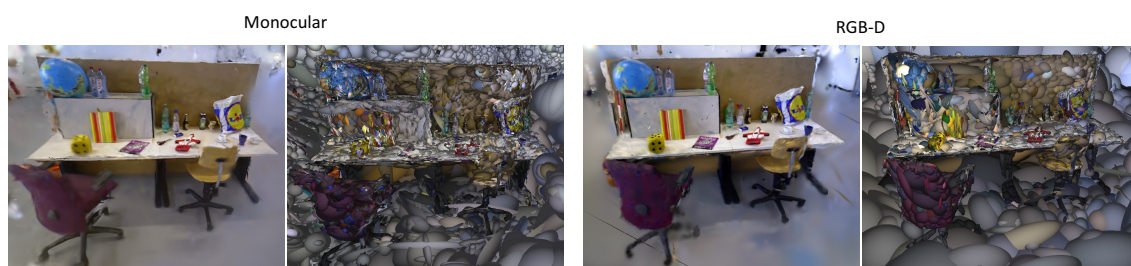


Figure 5.13: Novel view rendering and Gaussian visualizations on TUM fr3/office

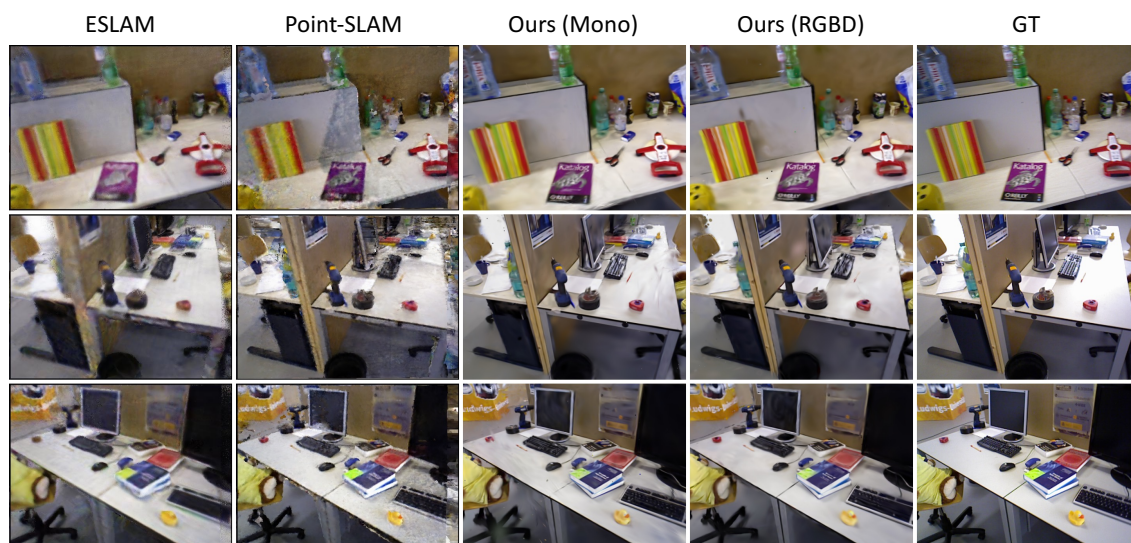


Figure 5.14: Rendering comparison on TUM fr3/office



Figure 5.15: Novel view rendering and Gaussian visualizations on Replica

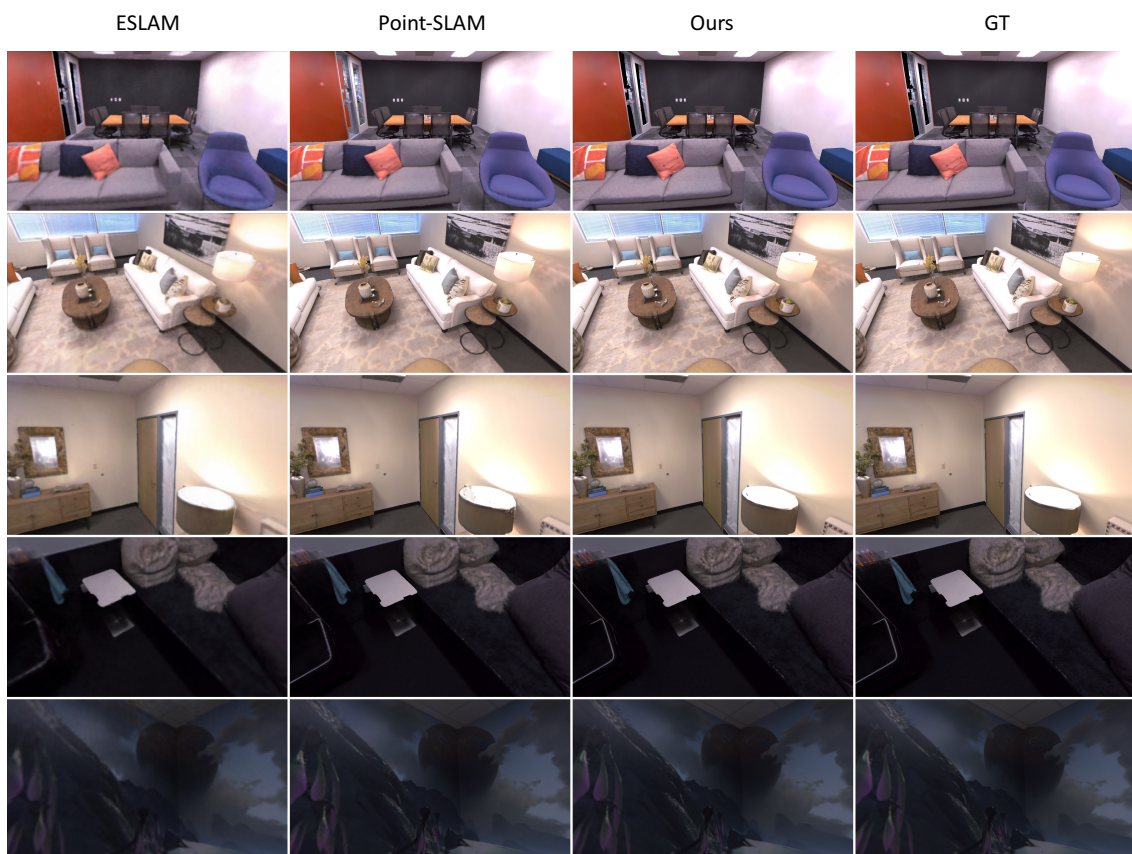


Figure 5.16: Rendering comparison on Replica

5.5 Conclusion

We have proposed the first SLAM method using 3D Gaussians as a SLAM representation. Via efficient volume rendering, our system significantly advances the fidelity and diversity of object materials a live SLAM system can capture. Our system achieves state-of-the-art performance across benchmarks for both monocular and RGB-D cases. Interesting directions for future research are the integration of loop closure for handling large-scale scenes and extraction of geometry such as surface normal as Gaussians do not explicitly represent the surface.

5.6 Limitations

- Currently, the proposed method is tested only on small room-scale scenes. For larger real-world scenes, the trajectory drift is inevitable. This could be addressed by integrating a loop closure module into our existing pipeline.
- Although we achieve interactive live operation, hard real-time operation on the benchmark dataset (30 fps on TUM sequences) is not achieved in this work. To improve speed, exploring a second-order optimizer would be an interesting direction.
- The current system assumes a static environment and can fail in the presence of dynamic objects.

4DTAM: Non-Rigid Gaussian Splatting SLAM for Dynamic Scenes

Contents

6.1	Introduction	109
6.2	Related Work	110
6.2.1	Visual SLAM	110
6.2.2	SLAM for Non-Rigid Reconstruction	111
6.2.3	Datasets for 4D Reconstruction	112
6.3	Method	113
6.3.1	2D Gaussian Splatting	113
6.3.2	Analytic Camera Pose Jacobian	115
6.3.3	Warp Field	117
6.3.4	Tracking and Mapping Framework	118
6.3.5	Dataset Generation	120
6.4	Evaluation	122
6.4.1	Experimental Setup	122
6.4.2	Quantitative Evaluation	125
6.4.3	Qualitative Evaluation	125
6.4.4	Ablation Study	126
6.5	Conclusion	129

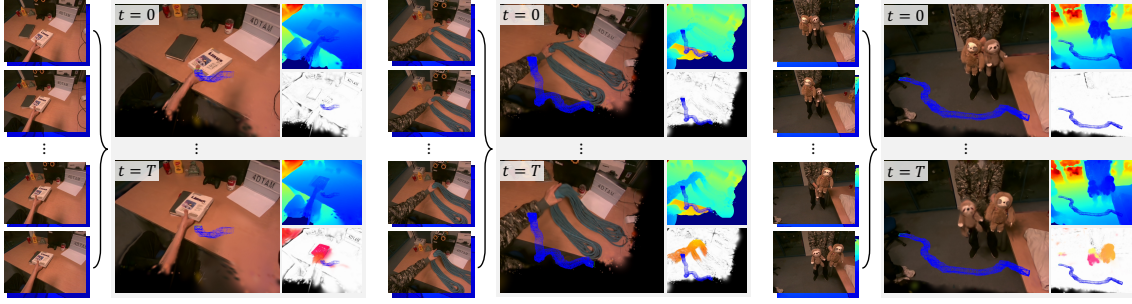


Figure 6.1: **4DTAM** jointly estimates camera-egomotion, appearance, geometry and scene dynamics without any template.

Real-world environments are often dynamic, yet most geometric computer vision and SLAM techniques focus primarily on static scenes. This static-scene assumption, which also underpins the methods proposed in previous chapters of this thesis, significantly limits their applicability in real-world scenarios. Recent advances in differentiable rendering and scene representations have enabled major progress in modeling dynamic scenes from 2D observations in an offline setting. Building on these developments, in this final chapter, we propose the first tracking and mapping approach for a single RGB-D camera capable of non-rigid surface reconstruction via differentiable rendering. We perform 4D scene capture from an online stream by jointly optimizing geometry, appearance, dynamics, and camera ego-motion. Although the natural environment contains complex non-rigid motions, non-rigid SLAM has remained difficult and relatively unexplored problem; even with 2.5D sensor measurements, it is still ill-posed because the primary challenge of single-camera non-rigid SLAM is that the observed geometry can deform significantly while the camera is focused elsewhere. Unlike static SLAM, where the scene geometry is temporally constant, dynamic scene deformation introduces a severe ambiguity: many different combinations of 3D deformation and camera motion can explain the same sequence of 2D observations. This high dimensionality of the joint optimization problem exacerbates the difficulty. Our novel SLAM method based on Gaussian surface primitives allows accurate 3D reconstruction and real-time rendering without any template, using a warp-field represented by a multi-layer perceptron (MLP) and regularization terms to enable spatio-temporal reconstruction. A challenge in non-rigid SLAM research is the lack of publicly available datasets with reliable ground truth and standardized evaluation protocols. To address this, we introduce a novel synthetic dataset of everyday objects featuring diverse motions, leveraging availability of large-scale objects and advancements in animation modeling.

6.1 Introduction

The world we live in has many moving elements. Rivers flow, trees sway, cookies crumble, and humans walk. Although SLAM methods which assume that most of the world is static are highly useful, embodied agents which aim to navigate and interact with their environments in the most general way should be able to operate in dynamic scenes. There are various possible ways to segment out and ignore moving scene elements such that SLAM systems can still reconstruct the static parts of scenes and estimate camera ego-motion, but here we focus on general spatio-temporal (4D) reconstruction of scenes with significant dynamic motion within a single unified framework, and without making strong prior assumptions about the moving elements present.

4D SLAM with general scene motion is difficult primarily because of the complex and high-dimensional nature of modeling non-rigid motions (and potential topological changes) while simultaneously optimizing the pose of a moving camera. There is much more redundancy than in rigid SLAM, and some prior assumptions are needed to combat this. Another challenge lies in the lack of datasets to train and/or evaluate techniques. Recent advances in computer vision and graphics make it a good time to revisit this problem. New 3D representations (e.g. neural fields and Gaussian splats) allow differentiable rendering of complex 3D scenes and optimization via 2D observations, and to model deformation fields smoothly without more specific assumptions. Also, the availability of high-quality 3D meshes and rendering software (e.g. Blender) gives the ability to render non-rigidly moving objects with ground truth.

We present **4DTAM**, a novel approach for **4D Tracking And Mapping** in dynamic scenes. We use Gaussian surface primitives to represent the scene and introduce a neural warp-field represented by a multi-layer perception (MLP) to model continuous temporal changes. We then utilize differentiable rendering to jointly optimize the scene geometry, appearance, dynamics, and camera ego-motion from an online stream of a single RGB-D camera. This enables accurate 3D reconstruction and real-time rendering, even in the presence of complex non-rigid deformations.

To facilitate future research, we also introduce a new synthetic dataset of dynamic objects. Our focus in this dataset is realistic, complex motion of scenes that are not well represented by existing deformable object models. Animated 3D meshes are rendered and the ground truth depth, surface normals, and foreground masks are extracted together with the camera poses/intrinsics. This dataset provides challenging scenarios for 4D reconstruction methods. We also release the full rendering script to allow the generation of custom 4D datasets.

Our experimental results demonstrate that **4DTAM** achieves good performance in both camera tracking and scene reconstruction in the presence of dynamic objects. It can handle the complex motion of articulated objects (e.g., drawers) and non-rigid objects (e.g., curtains, flags, and animals), showcasing its potential for applications in robotics, augmented reality, and other fields requiring real-time dynamic scene understanding.

In summary, the contributions of this method are:

- We present **4DTAM**, the first 4D tracking and mapping approach utilizing differentiable rendering and Gaussian surface primitives to handle dynamic environments.
- We propose the first 2DGS [Huang et al., 2024a] based SLAM method. We formulate analytic camera pose gradients, initialization and regularization techniques to fully utilize an RGB-D signal. We further introduce a warp-field represented by an MLP to model scene dynamics including non-rigid deformations.
- We present a novel synthetic dataset of objects with complex motions to facilitate research in dense non-rigid SLAM and provide a new benchmark for evaluation.
- We demonstrate through extensive experiments that our method achieves state-of-the-art performance in tracking and scene reconstruction in dynamic environments.

6.2 Related Work

6.2.1 Visual SLAM

Visual SLAM (Simultaneous Localization and Mapping) has been an extensively researched field, with Dense SLAM specifically focusing on capturing detailed scene geometry and semantics. A central aspect of these methods lies in the choice of scene representation and the corresponding optimization framework.

Dense SLAM methods based on traditional scene representations, such as volumetric Truncated Signed Distance Functions (TSDF) [Newcombe et al., 2011b, Whelan et al., 2015a, Kähler et al., 2016] or Surfels [Whelan et al., 2015b, Schöps et al., 2019], project 2D observations into 3D space and employ specific data fusion algorithms. While effective, these methods often fail to keep consistency between the model and sensor observations across multiple viewpoints, posing challenges for long-term operation.

However, recent advances in graphics hardware have facilitated the adoption of differentiable rendering frameworks, which have revolutionized inverse rendering and scene reconstruction [Kato et al., 2018b, Niemeyer et al., 2020d, Mildenhall et al., 2020, Müller et al., 2022]. Differentiable rendering ensures multi-view consistency through streamlined backpropagation, enhancing scene reconstruction accuracy. Notably, 3D Gaussian Splatting (3DGS) [Kerbl et al., 2023] has gained attention due to its flexible resource allocation and rapid forward rendering capabilities. Initially developed for photorealistic view synthesis, recent research has extended its application to surface reconstruction [Guédon and Lepetit, 2024, Yu et al., 2024]. Enhanced methods, such as 2D Gaussian Splatting (2DGS) [Huang et al., 2024a], achieve superior geometry reconstruction by reducing the Gaussian dimension and explicitly defining surface normals. These differentiable rendering representations have been applied to visual SLAM, from coordinate-based MLPs [Sucar et al., 2021] to explicit voxel grids [Zhu et al., 2022, Yang et al., 2022, Johari et al., 2023, Wang et al., 2023], points [Sandström et al., 2023], and 3D Gaussians [Matsuki et al., 2024a, Keetha et al., 2024, Yan et al., 2024].

6.2.2 SLAM for Non-Rigid Reconstruction

3D reconstruction of dynamic scenes has a long history in computer vision, and impressive results have been achieved via optimization methods even in the most challenging case of unknown non-rigid objects observed by a single moving RGB camera, [Torresani et al., 2008, Garg et al., 2013]. These methods and their successors require batch optimization of whole sequences, are usually limited to small scene regions.

Dynamic SLAM aims to reconstruct and track scenes with continuous motion via incremental, ideally real-time processing with the ability to work in large scenes and for long periods of time. Most methods to date have relied on RGB-D data from moving depth cameras. If the focus is camera motion estimation and reconstruction of the static parts of a scene, it can be sufficient to detect and ignore dynamic objects, either by learned semantic priors or by detecting outliers to geometric consistency (e.g. [Scona et al., 2018]). A more advanced method is to separately track and build 3D reconstructions of the moving scene elements, on the assumption that each is a solid, rigid object (e.g. [Rünz and Agapito, 2017]).

If the semantic class of the moving objects is known or can be detected, per-object parametric models which capture the ways that their shape changes as they move can be used. This approach is well motivated for instance when scenes contain humans or animals, for which high

quality models exist (e.g. [Loper et al., 2015]). Similarly, for endoscopic images, non-rigid 4D SLAM has been addressed by adding scene-specific priors [Ma et al., 2021] or deformation models [Rodriguez et al., 2023].

Incremental SLAM for general dynamic scenes has remained more challenging but has been addressed based on various regularizing assumptions and representations. DynamicFusion [Newcombe et al., 2015] pioneered a line of work [Slavcheva et al., 2017, Slavcheva et al., 2018, Innmann et al., 2016, Gao and Tedrake, 2018] which captures temporal evolution in the scene geometry by jointly optimizing a canonical volumetric representation (e.g., TSDF volume [Newcombe et al., 2015]) and a deformation field. As the solution space is extremely high-dimensional, additional constraints are often introduced to regularize the motion field [Slavcheva et al., 2017, Slavcheva et al., 2018] or to align visual features [Innmann et al., 2016, Božič et al., 2020a].

Recent advances in 3D representations, such as neural fields and Gaussian primitives, have opened new possibilities for dynamic scene reconstruction. Canonical radiance and motion fields can be jointly optimized via differentiable rendering, as demonstrated with NeRF [Park et al., 2021, Pumarola et al., 2020, Tretschk et al., 2021] and SDF [Cai et al., 2022, Wang et al., 2024a]. For 3D Gaussians, which can explicitly represent points, motion can be estimated either through per-primitive trajectories [Luiten et al., 2024] or learnable motion bases [Wang et al., 2024b]. However, warp-field-based motion representation offers inherent smoothness regularization, leveraging the properties of neural fields [Yang et al., 2024b, Yang et al., 2024a, Duisterhof et al., 2024, Huang et al., 2024b].

Most existing methods, however, rely on known camera poses or multi-camera setups to capture dense spatiotemporal observations. While DyNoMo [Seidenschwarz et al., 2024] supports camera pose optimization, its 3D Gaussian representation is not suited for geometrically accurate reconstruction. In contrast, our 4DTAM framework enables 4D reconstruction using a single RGB-D camera, jointly optimizing camera poses, appearance, geometry, and dynamics, making it practical for most embodied agents.

6.2.3 Datasets for 4D Reconstruction

4D reconstruction has been studied extensively for the case of the human body. Datasets like Human3.6M [Ionescu et al., 2013], DeepCap [Habermann et al., 2020], and ZJU-MoCap [Peng et al., 2021] capture diverse human motions under a multi-camera setup. The cameras are fixed, synchronized, and calibrated to reduce the difficulty in establishing dense multi-view correspond-

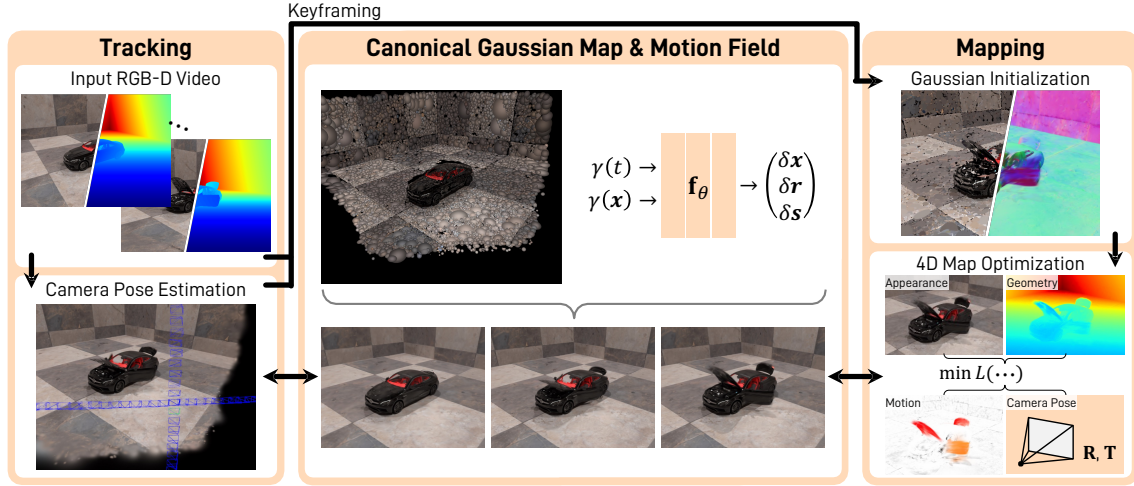


Figure 6.2: Method overview.

ences.

Only a small number of datasets provide single-stream RGB-D sequences captured from a moving camera [Slavcheva et al., 2017, Božič et al., 2020b, Gao et al., 2022]. Recovering the camera poses is not trivial for such real-world captures, and additional post-processing (e.g. robust depth map alignment [Wang et al., 2024a]) is required. Another challenge lies in ground truth acquisition. Besides the depth measurements, other ground truths (e.g., scene flow, object mask) often require manual labeling. In contrast, synthetic data sets [Wang et al., 2020, Butler et al., 2012] provide perfect ground truths. Recent advances in open-source datasets [Deitke et al., 2023] and rendering software [Community, 2018] also close the synthetic-to-real domain gap significantly. To this end, we introduce a new high-quality synthetic dataset tailored for 4D reconstruction.

6.3 Method

6.3.1 2D Gaussian Splatting

Our geometric scene representation is based on 2D Gaussian Splatting (2DGS) [Huang et al., 2024a]. Unlike 3D Gaussian Splatting (3DGS), which uses blob-like splats, 2DGS functions as a stretchable surfel with explicitly defined surface normal directions, enabling precise surface representation. This property makes 2DGS particularly well-suited for non-rigid scene reconstruction with a single camera, where effectively handling 2.5D sensor signals is critical.

Each 2D Gaussian $\mathcal{G}$ is represented by its 3D mean position $\mathbf{P}_\mu$, rotation $\mathbf{R} \in \text{SO}(3)$, color $\mathbf{c}$, opacity o , and a scaling vector $\mathbf{S} \in \mathbb{R}^2$. The rotation matrix $\mathbf{R}$ is decomposed as $\mathbf{R} = [\mathbf{t}_u, \mathbf{t}_v, \mathbf{t}_w]$,

where $\mathbf{t}_u$ and $\mathbf{t}_v$ represent two principal tangential vectors, and $\mathbf{t}_w$ is the normal vector, defined as $\mathbf{t}_w = \mathbf{t}_u \times \mathbf{t}_v$. For simplicity, spherical harmonics are omitted in this work.

The 2D Gaussian function is parameterized on the local tangent plane in world space as:

$$P(u, v) = \mathbf{P}_\mu + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v = \mathbf{H}(u, v, 1, 1)^T \quad (6.1)$$

$$\text{where } \mathbf{H} = \begin{bmatrix} s_u \mathbf{t}_u & s_v \mathbf{t}_v & \mathbf{0} & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{p}_k \\ \mathbf{0} & 1 \end{bmatrix} \quad (6.2)$$

For a point $\mathbf{u} = (u, v)$ in the tangential plane of 2D Gaussian (uv space), its projection onto the image plane is given by

$$\mathbf{x} = (xz, yz, z, 1)^T = \mathbf{W}\mathbf{P}(u, v) = \mathbf{W}\mathbf{H}(u, v, 1, 1)^T \quad (6.3)$$

where $\mathbf{W} \in \mathbb{R}^{4 \times 4}$ is the transformation matrix from world space to screen space.

To avoid numerically unstable matrix inversion of $\mathbf{M} = (\mathbf{W}\mathbf{H})^{-1}$, 2DGS applies ray-splat intersection by finding the intersection of non-parallel planes (x-plane and y-plane). The ray $\mathbf{x} = (x, y)$ is determined by the intersection of the x -plane $\mathbf{h}_x$ and the y -plane $\mathbf{h}_y$, represented as $\mathbf{h}_x = (-1, 0, 0, x)^T$ and $\mathbf{h}_y = (0, -1, 0, y)^T$, respectively.

In the uv coordinates of the 2D Gaussian, this is expressed as:

$$\mathbf{h}_u = (\mathbf{W}\mathbf{H})^T \mathbf{h}_x \quad \text{and} \quad \mathbf{h}_v = (\mathbf{W}\mathbf{H})^T \mathbf{h}_y \quad (6.4)$$

The intersection point meets the following condition,

$$\mathbf{h}_u \cdot (u, v, 1, 1)^T = \mathbf{h}_v \cdot (u, v, 1, 1)^T = 0 \quad (6.5)$$

This leads to an solution for the intersection point $\mathbf{u}(\mathbf{x})$:

$$u(\mathbf{x}) = \frac{\mathbf{h}_u^2 \mathbf{h}_v^4 - \mathbf{h}_u^4 \mathbf{h}_v^2}{\mathbf{h}_u^1 \mathbf{h}_v^2 - \mathbf{h}_u^2 \mathbf{h}_v^1} \quad v(\mathbf{x}) = \frac{\mathbf{h}_u^4 \mathbf{h}_v^1 - \mathbf{h}_u^1 \mathbf{h}_v^4}{\mathbf{h}_u^1 \mathbf{h}_v^2 - \mathbf{h}_u^2 \mathbf{h}_v^1} \quad (6.6)$$

where $\mathbf{h}_u^i, \mathbf{h}_v^i$ are the i -th parameter of the 4D homogeneous plane parameters.

The 2D Gaussian at (u, v) is evaluated as:

$$\mathcal{G}(\mathbf{u}) = \exp\left(-\frac{u^2 + v^2}{2}\right) \quad (6.7)$$

The 2D Gaussians are sorted along the camera ray by their center depth and organized into image tiles. Per-pixel color is rendered via volumetric alpha blending:

$$c(\mathbf{x}) = \sum_{i=1} \mathbf{c}_i \alpha_i \mathcal{G}_i(\mathbf{u}(x)) \prod_{j=1}^{i-1} (1 - \alpha_j \mathcal{G}_j(\mathbf{u}(x))) \quad (6.8)$$

where depth and normal can be rendered similarly.

6.3.2 Analytic Camera Pose Jacobian

One major advantage of Gaussian Splatting is its analytical formulation of gradient flow for model parameters, enabling real-time full-resolution rendering. However, it assumes posed images as input and does not provide gradients for camera poses. To accelerate optimization, we derive the analytic Jacobian of the camera pose for 2D Gaussian Splatting and implement it using a CUDA kernel. This formulation has potential applications for a wide range of tasks involving pose estimation in surface-based Gaussian Splatting.

We use Lie algebra to derive the minimal Jacobians for the camera pose matrix from the world coordinate system to the camera's local coordinate system, defining $\mathbf{T}_{CW} \in \mathbf{SE}(3)$ and $\boldsymbol{\tau} \in \mathfrak{se}(3)$. Since 2DGS backpropagates gradients to $\mathbf{M}^T = \mathbf{W}\mathbf{H}$ during the optimization of the 3D mean, we require the partial derivative $\frac{\partial \mathbf{M}^T}{\partial \boldsymbol{\tau}}$. Let $\mathbf{K} \in \mathbb{R}^{4 \times 4}$ represent the camera projection matrix. Then, equation 6.3 is rewritten as:

$$\mathbf{x} = \mathbf{M}^T(u, v, 1, 1)^T = \mathbf{K} \mathbf{T}_{CW} \mathbf{H}(u, v, 1, 1)^T \quad (6.9)$$

Using the chain rule, the partial derivatives are computed as:

$$\frac{\partial \mathbf{M}^T}{\partial \boldsymbol{\tau}} = \frac{\partial \mathbf{M}^T}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \mathbf{T}_{CW}} \frac{\partial \mathbf{T}_{CW}}{\partial \boldsymbol{\tau}}, \quad (6.10)$$

$$\frac{\partial \mathbf{T}_{CW}}{\partial \boldsymbol{\tau}} = \begin{bmatrix} \mathbf{0} & -\mathbf{R}_{CW}^{\times}_{:,1} \\ \mathbf{0} & -\mathbf{R}_{CW}^{\times}_{:,2} \\ \mathbf{0} & -\mathbf{R}_{CW}^{\times}_{:,3} \\ \mathbf{I} & -\mathbf{t}_{CW}^{\times} \end{bmatrix} \quad (6.11)$$

where $\mathbf{R}_{CW} \in \mathbf{SO}(3)$ and $\mathbf{t}_{CW} \in \mathbb{R}^3$ denote the rotation and translation parts of $\mathbf{T}_{CW}$, respectively. The notation $^{\times}$ represents the skew-symmetric matrix of a 3D vector, and $\mathbf{R}_{CW}^{\times}_{:,i}$ denotes the i th column of $\mathbf{R}_{CW}^{\times}$. Here we provide the detail of the derivation;

Let $\mathbf{T} \in \mathbf{SE}(3)$ and $\boldsymbol{\tau} = (\boldsymbol{\rho}, \boldsymbol{\theta}) \in \mathfrak{se}(3)$, $\mathbf{T}$ is represented as follows;

$$\begin{aligned} \mathbf{T} &= \text{Exp}(\boldsymbol{\tau}) = \exp(\boldsymbol{\tau}^\wedge) \\ &= \exp\left(\sum_{j=1}^6 \mathbf{E}_j \tau_j\right), \quad \boldsymbol{\tau} \in \mathbb{R}^6. \end{aligned} \quad (6.12)$$

where the matrices $\mathbf{E}_j \in \mathbb{R}^{4 \times 4}$ are the $\mathbf{SE}(3)$ group generators and form a basis for $\mathfrak{se}(3)$:

$$\begin{aligned} \mathbf{E}_1 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} & \mathbf{E}_2 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{E}_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} & \mathbf{E}_4 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{E}_5 &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} & \mathbf{E}_6 &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \quad (6.13)$$

We get the partial derivative as follows:

$$\left. \frac{\partial}{\partial \tau_j} \exp(\boldsymbol{\tau}^\wedge) \right|_{\boldsymbol{\tau}=0} = \mathbf{E}_j, \quad j = 1, \dots, 6. \quad (6.14)$$

Therefore, the full derivative is given as:

$$\left. \frac{\partial \mathbf{T}}{\partial \boldsymbol{\tau}} \right|_{\boldsymbol{\tau}=0} = \mathbf{T} \left. \frac{\partial \left(\sum_{j=1}^6 \mathbf{E}_j \tau_j \right)}{\partial \boldsymbol{\tau}} \right|_{\boldsymbol{\tau}=0} \quad (6.15)$$

Since the meaningful elements of the camera $\mathbf{T}$ is 12 number variables, we stack the elements for 12×6 matrix and we obtain

$$\left. \frac{\partial \mathbf{T}}{\partial \boldsymbol{\tau}} \right|_{\boldsymbol{\tau}=0} = \begin{bmatrix} \mathbf{0} & -\mathbf{R}_{:,1}^\times \\ \mathbf{0} & -\mathbf{R}_{:,2}^\times \\ \mathbf{0} & -\mathbf{R}_{:,3}^\times \\ \mathbf{I} & -\mathbf{t}^\times \end{bmatrix}. \quad (6.16)$$

where $\mathbf{R} \in \mathbf{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$ denote the rotation and translation parts of $\mathbf{T}$.

2DGS also renders a normal map, which can be supervised using the loss computed from the rendered normals. Let $\mathbf{n}_c$ denote the camera-space normal. The normal of a 2D Gaussian in the camera’s local coordinate system is defined as:

$$\mathbf{n}_c = \mathbf{T}_{CW} \mathbf{t}_w \quad (6.17)$$

where $\mathbf{t}_w$ is the surface normal in the world coordinate system.

Borrowing the notation of the left Jacobian for Lie groups from [Solà et al., 2018], the left-side partial derivative on the manifold is defined as:

$$\frac{\mathcal{D}f(\mathbf{T})}{\mathcal{D}\mathbf{T}} \triangleq \lim_{\tau \rightarrow 0} \frac{\text{Log}(f(\text{Exp}(\tau) \circ \mathbf{T}) \circ f(\mathbf{T})^{-1})}{\tau} \quad (6.18)$$

By using this, the partial derivative is given by:

$$\left. \frac{\partial \mathbf{n}_c}{\partial \tau} \right|_{\tau=0} = \frac{\mathcal{D}\mathbf{n}_c}{\mathcal{D}\mathbf{T}_{CW}} = \lim_{\tau \rightarrow 0} \frac{\text{Exp}(\tau) \mathbf{n}_c - \mathbf{n}_c}{\tau} \quad (6.19)$$

$$= \lim_{\tau \rightarrow 0} \frac{(\mathbf{I} + \tau^\wedge) \cdot \mathbf{n}_c - \mathbf{n}_c}{\tau} \quad (6.20)$$

$$= \lim_{\tau \rightarrow 0} \frac{\tau^\wedge \cdot \mathbf{n}_c}{\tau} \quad (6.21)$$

$$= \lim_{\tau \rightarrow 0} \frac{\boldsymbol{\theta}^\times \mathbf{n}_c + \boldsymbol{\rho}}{\tau} \quad (6.22)$$

$$= \lim_{\tau \rightarrow 0} \frac{-\mathbf{n}_c^\times \boldsymbol{\theta} + \boldsymbol{\rho}}{\tau} \quad (6.23)$$

$$= \begin{bmatrix} \mathbf{I} & -\mathbf{n}_c^\times \end{bmatrix} \quad (6.24)$$

6.3.3 Warp Field

To model time-varying deformations, we use a warp-field represented by a coordinate-based network [Yang et al., 2024a, Yang et al., 2024b]. In our hand-held single-camera setup, the limited view coverage of dynamic objects necessitates structural priors in the motion representation. For this, we employ a compact MLP as the warp-field to estimate transitions from the canonical Gaussians following [Yang et al., 2024a].

Given time t and center position $\mathbf{x}$ of 2D Gaussians in canonical space as inputs, the deformation MLP $\mathbf{f}_\theta$ produces offsets, which subsequently transform the canonical 2D Gaussians to the deformed space:

$$(\delta \mathbf{x}, \delta \mathbf{r}, \delta \mathbf{s}) = \mathbf{f}_\theta(\gamma_1(\mathbf{x}), \gamma_2(t)) \quad (6.25)$$

where $\delta \mathbf{x} \in \mathbb{R}^3$, $\delta \mathbf{r} \in \mathbf{SO}(3)$, $\delta \mathbf{s} \in \mathbb{R}^2$ denotes the offsets of 2D Gaussian’s mean position, rotation and scale respectively, γ denotes the frequency-based positional encoding [Mildenhall et al., 2020]. For deformable SLAM applications, we leverage a CUDA-optimized MLP implementation [Müller et al., 2022] to enable fast, interactive reconstruction.

6.3.4 Tracking and Mapping Framework

Our SLAM method follows the standard tracking and mapping architecture, where the tracking module is in charge of fast online camera pose estimation while the mapping performs a relatively more involved joint optimization of the camera poses, geometry and motion of selected keyframes.

Tracking

The tracking module estimates the coarse camera pose for the latest incoming frame. This is achieved by minimizing the photometric and depth rendering errors between the sensor observation and the rendering from the deformable Gaussian model. Unlike static 3DGS SLAM methods, we estimate the camera pose relative to the warped Gaussians at the latest keyframes timestamp t_{kf} , assuming the deformed scene structure at t_{kf} is closest to the current state. We define photometric rendering loss as:

$$L_p = \|I(\mathcal{G}_{cano}, \mathbf{T}_{CW}, t_{kf}) - \bar{I}\|_1 \quad (6.26)$$

Here $I(\mathcal{G}, \mathbf{T}_{CW})$ denotes a rendered color image from the canonical Gaussians $\mathcal{G}_{cano}$, timestamp of the latest keyframe t_{kf} and camera pose $\mathbf{T}_{CW}$, and $\bar{I}$ is an observed image. Similarly, we also minimize geometric depth error:

$$L_g = \|D(\mathcal{G}_{cano}, \mathbf{T}_{CW}, t_{kf}) - \bar{D}\|_1 \quad (6.27)$$

Following MonoGS [Matsuki et al., 2024a], we further optimize affine brightness parameters. Keyframes are selected every N-th frame and sent to the mapping process for further refinement.

Mapping

The mapping module performs joint optimization of the camera pose, canonical Gaussians, and the warp field within a sliding window.

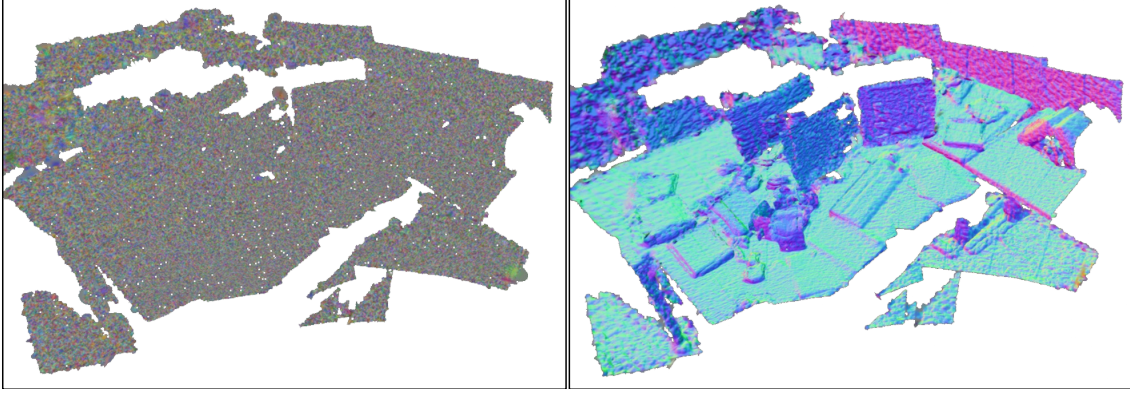


Figure 6.3: **2D Gaussian’s Surface Normal Rendering based on Different Initialization.** Left: Random initialization. Right: Our initialization aligned with sensor measurement.

Gaussian Management When a new keyframe is registered, we add new Gaussians to the canonical Gaussians $\mathcal{G}_{cano}$, based on the back-projected point cloud from the RGB-D observations. Unlike 3DGS, 2DGS explicitly encodes surface normal information in its rotation vector, making it beneficial to initialize using surface normals estimated from sensor depth measurements. To achieve this, we compute the surface normals of the current depth observation by taking the finite difference of neighboring back-projected depth points and assign them as the normal vectors of the 2D Gaussians $\mathbf{t}_w$. This is formulated as:

$$\mathbf{t}_w = \frac{\nabla_x \mathbf{p}_d \times \nabla_y \mathbf{p}_d}{|\nabla_x \mathbf{p}_d \times \nabla_y \mathbf{p}_d|} \quad (6.28)$$

where $\mathbf{p}_d$ denotes points back-projected by the current sensor depth observation. We store the computed normal information as a 2D image $\mathbf{N}_{sensor}$ for normal supervision. Gaussian pruning and densification parameters follow MonoGS, which effectively prunes the wrongly inserted Gaussians in the canonical space due to the object movement.

4D Map optimization We perform joint optimization of the camera ego-motion, appearance, geometry and scene dynamics. In a single-camera setup, the lack of spatiotemporally dense observations makes fully capturing dynamic scenes challenging, as complete spatial (xyz) coverage over time (t) is only feasible with multi-camera systems. To address this, we introduce regularization terms for both shape and motion.

In addition to photometric and depth losses, we apply a normal regularization based on sensor measurements to better align 2D Gaussians. Unlike the original 2DGS methods, which compute normals by finite differences of rendered depth during every optimization step—leading to high computational costs—we instead propose to use normals precomputed from sensor depth as supervision. This reduces computational overhead, as normals are calculated only when a new

keyframe is inserted:

$$L_n = \sum_{i \in h \times w} (1 - \mathbf{n}_i^T \mathbf{N}_{sensor,i}) \quad (6.29)$$

To constrain motion in unobserved regions, we apply an as-rigid-as possible regularization loss L_{ARAP} from [Luiten et al., 2024] to the Gaussian means. Additionally, we introduce a novel surface normal rigidity loss, constraining the 2D Gaussians’ surface normals to stay similar between timesteps t_1 and t_2 , preserving local surface rigidity:

$$L_{ARAP_n} = w_{i,j} \left\| (\mathbf{t}_w)_{i,t_1}^T (\mathbf{t}_w)_{j,t_1} - (\mathbf{t}_w)_{i,t_2}^T (\mathbf{t}_w)_{j,t_2} \right\|_1 \quad (6.30)$$

where $w_{i,j}$ is a distance-based weighting factor like L_{ARAP} . We apply ARAP regularizers between the oldest and latest keyframe in the current window.

Together with the isotropic loss L_{iso} proposed in [Matsuki et al., 2024a], we minimize the following total cost function:

$$\begin{aligned} L_{total} = & \lambda_p L_p + \lambda_g L_g + \lambda_n L_n \\ & + \lambda_{iso} L_{iso} + L_{ARAP} + L_{ARAP_n} \end{aligned} \quad (6.31)$$

The optimization is based on the sliding window heuristics in [Engel et al., 2017], with two additional keyframes randomly selected from the history.

Global Optimization Sliding window-based optimization prioritizes the latest frame, causing past keyframe information to degrade over time. After tracking, if required we can perform global optimization to finalize the map, which takes less than 1 minute on an RTX 4090. During this step, the poses and number of Gaussians are fixed, and one keyframe is randomly selected per iteration. The process uses the normal consistency loss of 2DGS, ensuring global consistency despite being relatively slow.

6.3.5 Dataset Generation

We introduce **Sim4D**, a new synthetic dataset for 4D reconstruction. Recently, a large number of photo-realistic, animated 3D meshes have become available [Deitke et al., 2023, Ske,]. Combined with open-source graphics software (e.g., Blender [Community, 2018]), such meshes provide a scalable way of generating datasets for non-rigid 4D reconstruction. The data generation pipeline is illustrated in Fig. 6.4.



Figure 6.4: **Sim4D dataset.** We create a new dataset for 4D reconstruction by rendering animated 3D meshes.

Meshes and background. We collected 50 high-quality, animated 3D meshes from Objaverse [Deitke et al., 2023] and Sketchfab [Ske,], all of which are under CC-BY license. The collected meshes exhibit a wide variety of motions, including non-rigid deformation and topological changes. We then place the object inside a cube and randomize the background texture. Texture maps are collected from Poly Haven [Pol,] and are all under CC0 license.

Rendering. We render 240 to 540 frames for each object. The camera trajectories are defined along arcs of 20 degrees, and test viewpoints are defined outside of these arcs to evaluate the performance of novel-view synthesis and to quantify the accuracy of the reconstructed geometry. At each timestamp, the RGB image, ground truth depth, surface normals, and foreground mask are rendered and the camera intrinsics/extrinsics saved.

Training/Test Views We define the training and test views on a sphere, with its center representing the target object. In spherical coordinates (r, θ, ϕ) , we set $r = 2.0$. The training view is sampled from two arcs on the sphere’s surface, defined by $\theta \in [-10^\circ, 10^\circ]$ and $\phi \in [-10^\circ, 10^\circ]$. The test views are sampled from a circle on the sphere’s surface that pass through four key points: $(\theta, \phi) = (5^\circ, 0^\circ)$, $(0^\circ, 5^\circ)$, $(-5^\circ, 0^\circ)$, and $(0^\circ, -5^\circ)$. These points are chosen to ensure uniform sampling around the target object while maintaining a clear separation between the training and test views.

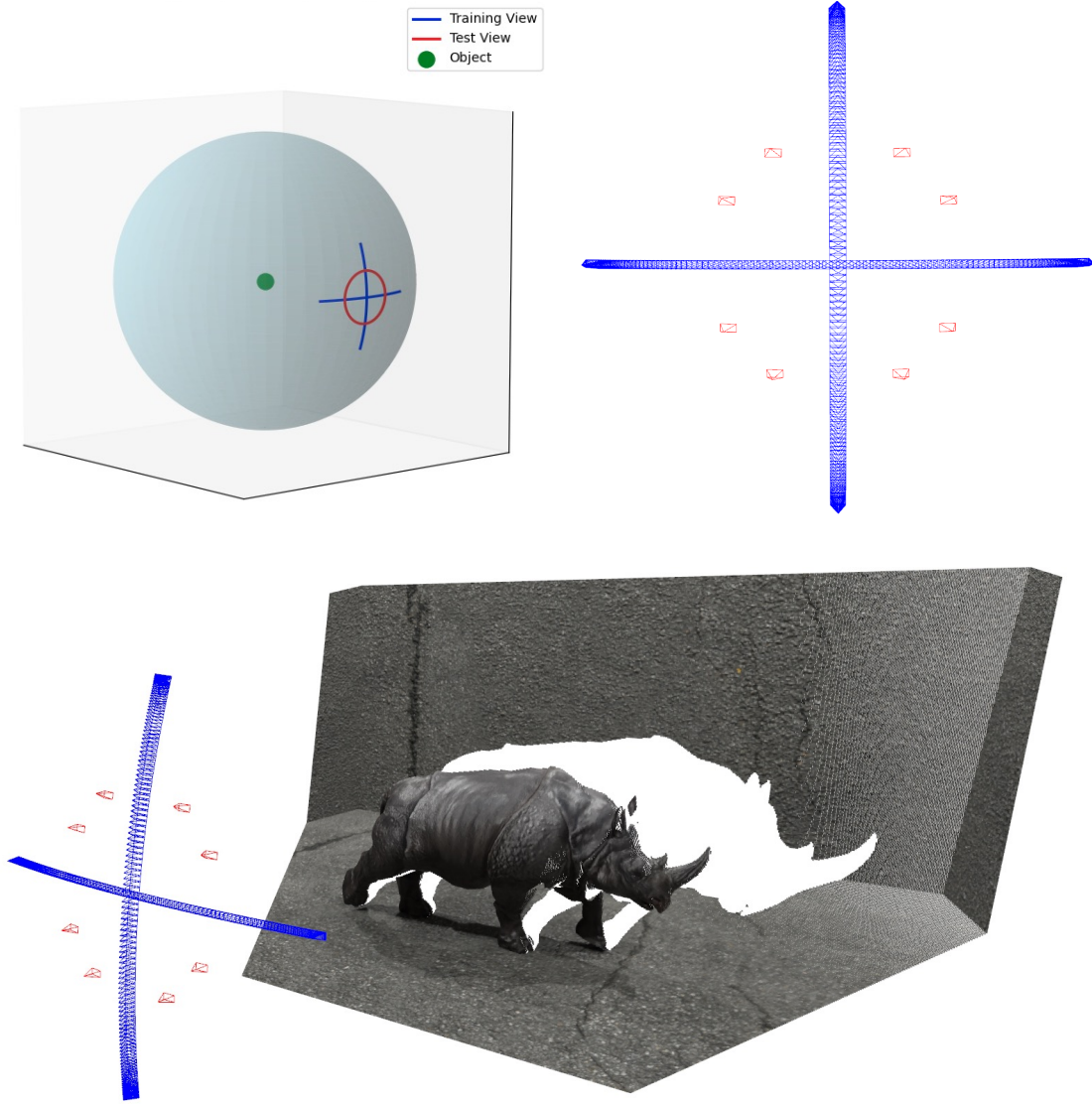


Figure 6.5: **Training and Test Views on the Sim4D Dataset:** **Blue** indicates training views, and **Red** indicates test views. Views are sampled (top right) from an arc on an object-centered sphere (top left) for dynamic scene reconstruction (bottom).

6.4 Evaluation

6.4.1 Experimental Setup

We extensively evaluate our non-rigid SLAM method on both synthetic and real-world datasets. Previous non-rigid RGB-D SLAM work has primarily focused on qualitative demonstrations using limited datasets, showcasing the early-stage potential of the field. To advance research, we introduce a quantitative evaluation protocol with the new Sim4D dataset. Our evaluation covers camera pose accuracy, as well as the appearance and geometric quality of the reconstructed models. Additionally, we demonstrate real-world performance using a self-captured dataset.

While designed primarily for dynamic scenes, our method is the first to leverage surface Gaussian splatting for both static SLAM and non-rigid RGB-D reconstruction. To further validate our approach, we perform a detailed quantitative component-wise ablation analysis.

Metrics and Datasets For our main **Non-Rigid SLAM evaluation**, we evaluate our method on 8 sequences from the Sim4D dataset. We first report ATE RMSE for trajectory evaluation. To assess SLAM map quality, we report depth rendering error (L1 error) for geometry and PSNR, SSIM, and LPIPS for appearance evaluation. For Sim4D, metrics are calculated from test views (extrapolated positions across different timestamps). The estimated and ground truth trajectories are aligned on the first frame, and test view positions are queried in the ground truth trajectory’s coordinate system. Since SurfelWarp [Gao and Tedrake, 2018] requires explicit foreground segmentation, we collect its results only on pixels with valid reconstruction. For **Static SLAM ablation**, we report ATE RMSE, rendering performance, and TSDF-fused mesh metrics, following the protocol in [Sandström et al., 2023]. We evaluate our method on the Replica [Straub et al., 2019] dataset and the TUM RGB-D dataset [Sturm et al., 2012]. To isolate the impact of scene representation from system differences, we replaced MonoGS’s representation with 2DGS while keeping all other system configurations identical. For **Offline Non-Rigid Reconstruction ablation**, we report the average geometry and appearance rendering metrics on subsets of the DeepDeform [Božič et al., 2020b], KillingFusion [Slavcheva et al., 2017], and iPhone datasets [Gao et al., 2022], which are used in [Wang et al., 2024a]. Since [Wang et al., 2024a] primarily focuses on object shape completion, metrics are calculated only within the given segmentation mask. The camera pose is provided by the dataset, and pose optimization is disabled to focus solely on reconstruction performance. We perform 30000 iteration for training, which takes approximately 30 mins.

Baseline Methods For quantitative non-rigid SLAM evaluation, we compare our method with SurfelWarp [Gao and Tedrake, 2018], the only non-rigid RGB-D SLAM method with publicly available code. For component-wise ablation analysis, we compare against MonoGS [Matsuki et al., 2024a] for static SLAM evaluation and Morpheus [Wang et al., 2024a] for offline reconstruction.

Implementation Details Our SLAM system runs on a desktop equipped with an Intel Core i9-12900K (3.50GHz) processor and a single NVIDIA GeForce RTX 4090 GPU. The camera pose

jacobian for 2DGS, described in Section 6.3.2, is implemented using a CUDA rasterizer, similar to other gradients in Gaussian Splatting. For real-world data capture, we used the Realsense D455.

System Details and Hyper parameters

Non-Rigid SLAM: We set the learning weights as follows: $\lambda_p = 0.9$, $\lambda_g = 0.1$, $\lambda_{iso} = 10.0$ and $\lambda_n = 0.002$. For the ARAP regularization [Luiten et al., 2024], we use a nearest neighbor count of 20, a radius of 0.05, and an exponential decay weight of 500. Keyframes are selected with $N = 1$. For the MLP, we use an 8-layer architecture with 256 neurons per layer. Frequency encoding is set to 1 for time and 4 for position. MLP is implemented with CUDA-optimized CutlassMLP in tiny-cuda-nn [Müller, 2021] for the fast optimization.

Static SLAM Ablation: We followed the same hyperparameters as MonoGS [Matsuki et al., 2024a], but we use normal loss L_n with the weight $\lambda_n = 0.01$ for the entire mapping process and $\lambda_g = 0.5$ for the final refinement. For the Replica 3D reconstruction evaluation, we have used the script introduced in [Sandström et al., 2023].

Offline Reconstruction Ablation: Camera poses are provided by the dataset and remain fixed during training. For the MLP, we adopt the same architecture described in [Yang et al., 2024a], consisting of an 8-layer network with 256 dimensions per layer, where a concatenated feature vector is input to the fourth layer. The positional encoding frequencies are set to 6 for time and 10 for position. Following the approach in [Cai et al., 2022, Wang et al., 2024a], we evaluate the geometric and appearance metrics against the input views and report the average values.

Method	Category	Metric	curtain	flag	mercedes	modular_vehicle	rhino	shoe_rack	water_effect	wave_toy
SurfelWarp [Gao and Tedrake, 2018]	Trajectory	ATE RMSE[cm]↓	6.10	31.9	5.21	4.21	2.81	2.16	2.60	1.45
	Geometry	L1 Depth[cm]↓	49.1	50.8	5.2	10.3	44.9	4.25	46.7	47.5
	Appearance	PSNR [dB] ↑	15.78	11.04	25.7	19.45	16.76	26.3	17.3	16.4
		SSIM ↑	0.468	0.343	0.779	0.362	0.188	0.795	0.325	0.364
		LPIPS ↓	0.56	0.659	0.483	0.638	0.665	0.397	0.587	0.555
Ours	Trajectory	ATE RMSE[cm]↓	0.15	1.00	0.08	0.24	0.12	0.18	0.29	0.22
	Geometry	L1 Depth[cm]↓	0.85	3.58	0.51	1.24	1.05	0.80	2.51	3.03
	Appearance	PSNR [dB] ↑	29.6	21.53	33.56	31.09	24.13	32.7	28.03	27.18
		SSIM ↑	0.867	0.648	0.934	0.811	0.762	0.918	0.824	0.793
		LPIPS ↓	0.076	0.141	0.131	0.200	0.230	0.08	0.808	0.077

Table 6.1: Non-rigid SLAM Evaluation on Sim4D Dataset.

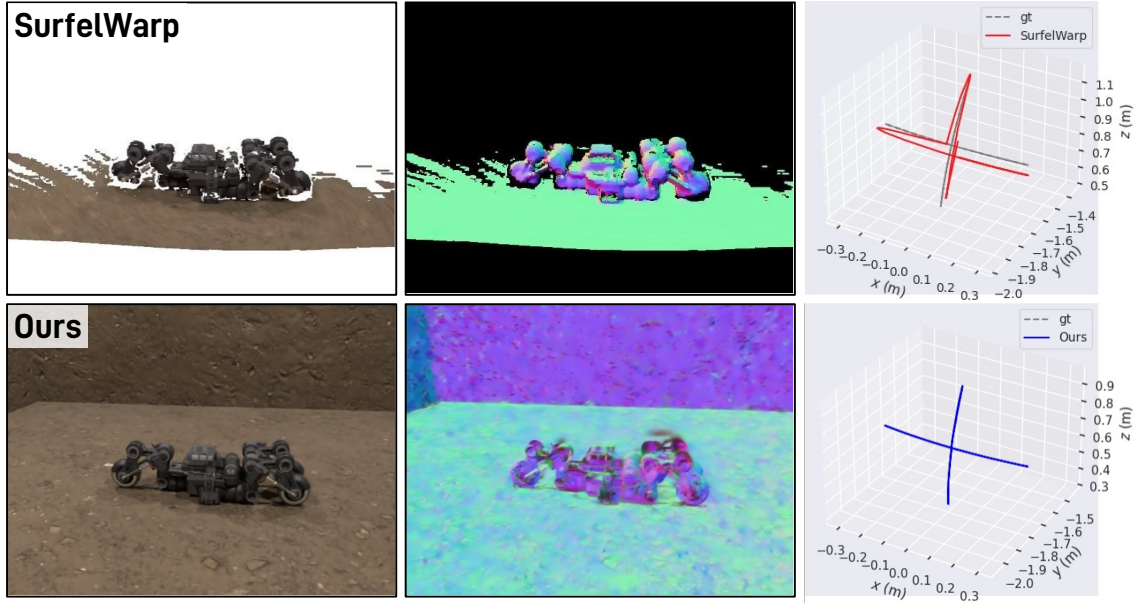


Figure 6.6: **Qualitative comparison to SurfelWarp.** Left: Rendered image, Middle: Rendered normal map, Right: Estimated camera trajectory

6.4.2 Quantitative Evaluation

Table 6.1 compares our method with SurfelWarp [Gao and Tedrake, 2018]. Our method outperforms SurfelWarp across all metrics. To analyze this further, Fig. 6.6 provides qualitative visualizations and trajectory plots for the modular_vehicle sequence. Since SurfelWarp relies on a foreground mask, its reconstruction lacks scene completeness. In contrast, our method reconstructs the entire scene within a joint optimization framework, providing more comprehensive coverage. Additionally, compared to SurfelWarp’s back-projection and Surfel fusion scheme, our differentiable rendering-based optimization enforces multi-view consistency over time, resulting in superior camera tracking and consistent 3D reconstruction. Our method achieves camera pose estimation at approximately 1.2 fps and completes the final global optimization in 1 minute.

6.4.3 Qualitative Evaluation

Fig. 6.7 presents qualitative reconstruction results on real-world dynamic scenes. Our method successfully reconstructs dynamic scenes with non-rigid deformations, whereas MonoGS fails to handle such complexities.

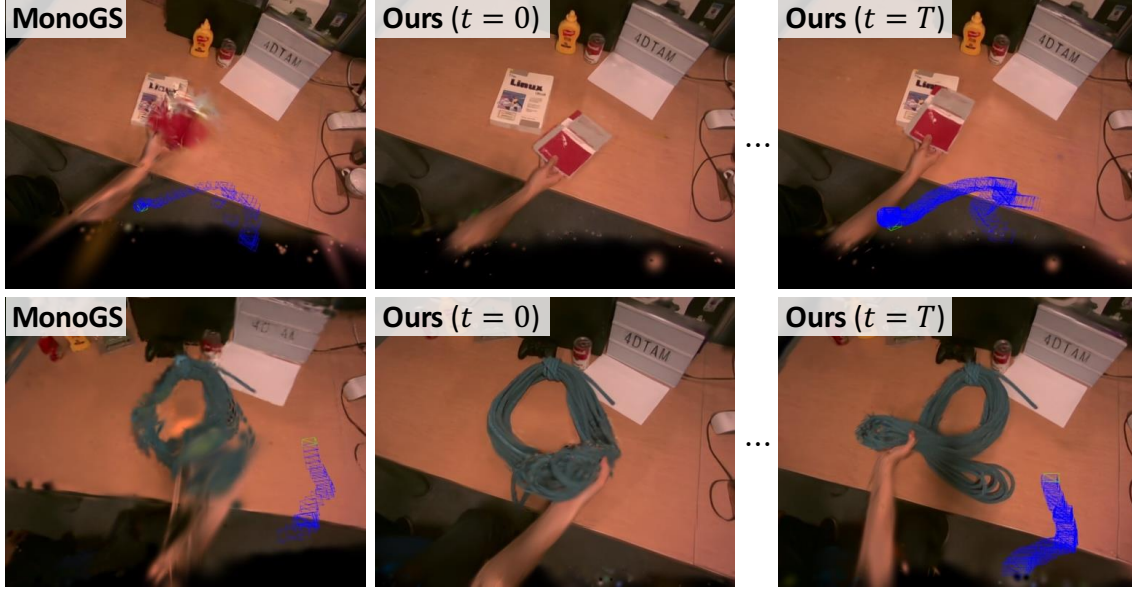


Figure 6.7: Qualitative Results on Real-World Dataset.

6.4.4 Ablation Study

Normal Rigidity Loss Table 6.2 presents the quantitative results demonstrating the effect of the normal rigidity loss defined in Equation 6.30. The normal rigidity loss improves the overall geometric metrics, such as camera ATE and L1 Depth, for the benchmark sequences by preserving the local geometric consistency of 2D Gaussians.

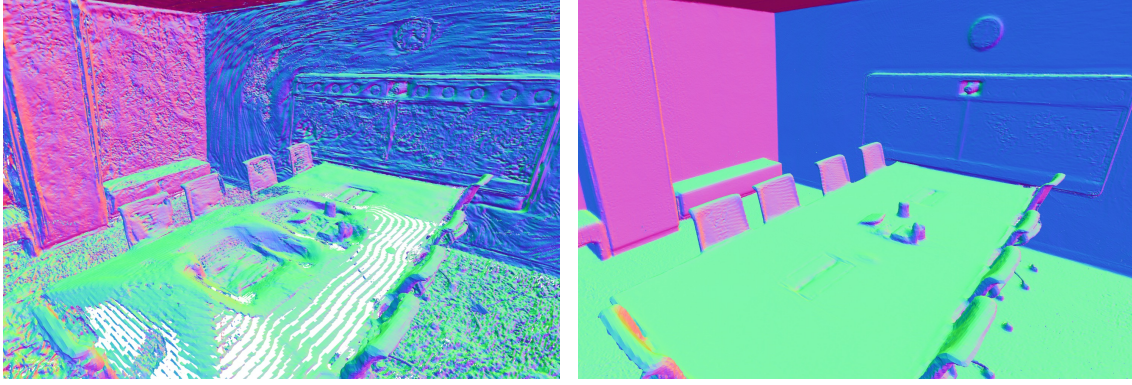
	ATE RMSE	L1 Depth	PSNR	SSIM	LPIPS
Ours full	0.28	1.71	28.47	0.820	0.12
w/o L_{ARAP-n}	0.52	2.00	29.04	0.853	0.13

Table 6.2: Ablation Study on L_{ARAP-n} . We report the average number of Sim4D dataset.

Monocular Depth Prior While our method was primarily tested with RGB-D camera input, we conducted an ablation study using depth input from the state-of-the-art monocular prediction network [Wang et al., 2024c], as shown in Table 6.3. The results demonstrate performance competitive with SurfelWarp, highlighting the potential for purely monocular non-rigid SLAM.

Method	Category	Metric	curtain	flag	mercedes	modular_vehicle	rhino	shoe_rack	water_effect	wave_toy
Ours (Monocular)	Trajectory	ATE RMSE[cm]↓	6.23	16.29	4.90	1.86	3.17	8.02	5.52	7.21
	Geometry	L1 Depth[cm]↓	74.2	155	59.2	38.0	37.7	89.8	72.4	80.8
	Appearance	PSNR [dB] ↑	17.73	16.22	20.72	26.28	21.48	17.49	18.86	17.98
		SSIM ↑	0.461	0.455	0.636	0.578	0.253	0.448	0.390	0.441
		LPIPS ↓	0.297	0.517	0.282	0.380	0.339	0.391	0.258	0.281

Table 6.3: Non-rigid SLAM Evaluation on Sim4D Dataset with Monocular Depth Prior.

Figure 6.8: **3D Reconstruction Result on Replica Office4.** Left: MonoGS. Right: Ours.

	Metric	r0	r1	r2	o0	o1	o2	o3	o4	avg
MonoGS	ATE RMSE[cm]↓	0.44	0.32	0.31	0.44	0.52	0.23	0.17	2.25	0.59
	Depth L1[cm]↓	3.00	3.47	4.66	3.10	6.08	6.15	4.77	4.94	4.52
	Precision[%]↑	39.0	28.8	28.9	39.7	15.8	28.0	32.5	25.5	29.7
	Recall[%]↑	44.2	34.5	32.8	47.6	24.3	30.0	35.4	28.5	34.6
	F1[%]↑	41.5	31.4	30.7	43.3	19.1	29.0	33.9	26.9	31.9
MonoGS-2D	ATE RMSE[cm]↓	0.42	0.43	0.35	0.19	0.19	0.22	0.27	0.80	0.36
	Depth L1[cm]↓	0.45	0.28	0.57	0.37	0.59	0.85	0.62	0.63	0.54
	Precision[%]↑	97.0	97.0	97.0	97.1	97.9	95.8	94.8	83.9	95.0
	Recall[%]↑	85.5	86.0	84.8	89.4	85.1	81.8	81.5	72.4	83.3
	F1[%]↑	90.9	91.3	90.5	93.1	91.1	88.2	87.6	77.7	88.8

Table 6.4: **Static SLAM Ablation on Replica.**

Static SLAM Table 6.4 provides the camera ATE and 3D reconstruction evaluation results. Our 2DGS-based implementation shows competitive performance and achieves the best result in 6 out of 8 sequences for camera ATE, and consistently better result on rendering and 3D reconstruction metrics. The reconstruction is visualized in Fig 6.8 which shows the comparison of the mesh generated by TSDF Fusion between MonoGS and MonoGS-2D. Table 6.5 provides the camera ATE and rendering metrics evaluation on TUM dataset. Our method shows on par camera ATE but shows the increased geometric reconstruction quality.

Method	Metric	fr1/desk	fr2/xyz	fr3/office	avg.
MonoGS	ATE RMSE[cm]↓	1.50	1.44	1.49	1.47
	Depth L1[cm]↓	6.2	13.0	13.0	10.7
MonoGS-2D	ATE RMSE[cm]↓	1.58	1.2	1.83	1.57
	Depth L1[cm]↓	3.0	2.3	4.3	3.2

Table 6.5: **Static SLAM Ablation on TUM**

Table 6.6 presents the average memory usage on the TUM dataset sequences. Due to the geometrically accurate alignment, 2D Gaussians require fewer primitives to represent the scene, resulting in reduced memory consumption.

Memory Usage [MB]	
MonoGS-2D	MonoGS
2.73MB	3.97MB

Table 6.6: Memory Analysis on TUM RGB-D dataset.

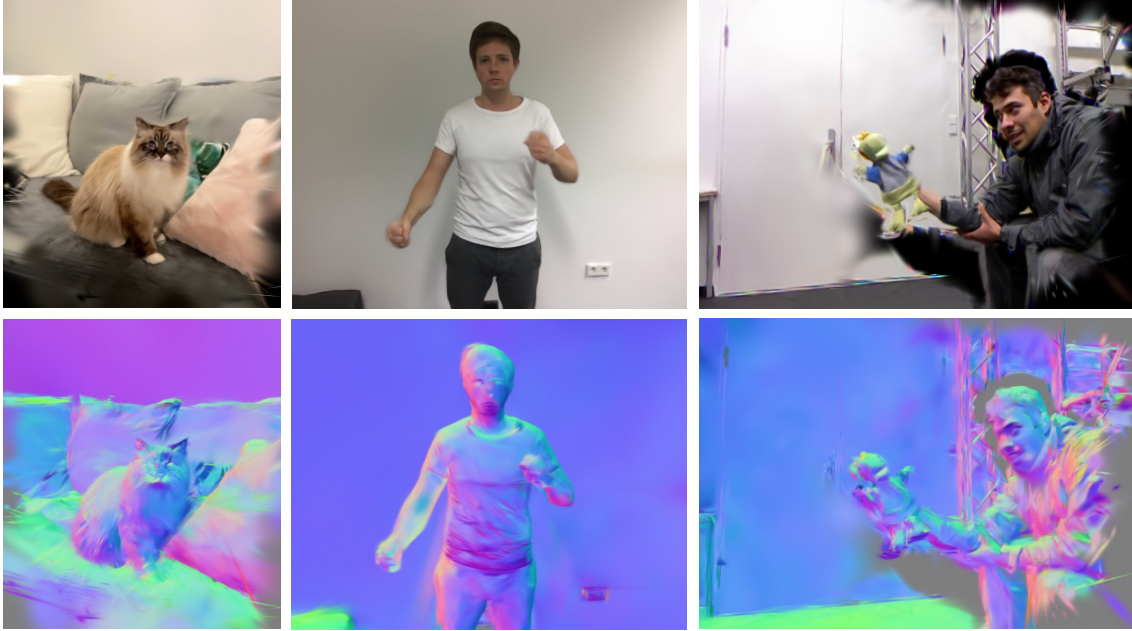


Figure 6.9: **Non-rigid Reconstruction Results.** Our method flexibly reconstructs dynamic objects with non-rigid deformations across various object categories, without requiring any shape templates or foreground/background separation.

Offline Non-rigid RGB-D Surface Reconstruction Table 6.7 reports offline reconstruction results, where camera poses are given. Our 2DGS+MLP deformation model shows competitive rendering performance compared to NeRF based methods. Note that Gaussian Splatting has the additional advantage of its rendering speed. We further provide qualitative visualizations in Fig. 6.9.

Method	Metric	KillingFusion	DeepDeform	iPhone
Morpheus [Wang et al., 2024a]	Depth L1[cm]↓	3.2	1.9	2.4
	PSNR [dB] ↑	27.02	26.81	25.28
	SSIM ↑	0.77	0.81	0.46
	LPIPS ↓	0.40	0.38	0.63
Ours	Depth L1[cm]↓	4.9	1.1	0.57
	PSNR [dB] ↑	31.13	24.15	27.54
	SSIM ↑	0.93	0.90	0.79
	LPIPS ↓	0.13	0.27	0.26

Table 6.7: **Offline Non-Rigid Reconstruction Ablation:** Rendering Error Metrics on Real-world Dataset.

6.5 Conclusion

We presented the first tracking and mapping method for non-rigid surface reconstruction using Surface Gaussian Splatting. Our approach integrates a 2DGS + MLP warp-field SLAM framework with camera pose estimation and regularization, leveraging RGB-D input. To support further research, we also introduced a novel dataset for dynamic scene reconstruction with reliable ground truth. Experimental results demonstrate that our method outperforms traditional non-rigid SLAM approaches.

Limitations:

- Our method has primarily been tested on small-scale scenes. Extending it to more complex real-world scenarios could benefit from incorporating 2D image priors, such as point tracking and optical flow.
- The current implementation runs at 1.2 fps, limiting its applicability in real-time scenarios. Developing lighter scene representations for interactive dynamic scene scanning remains a promising direction for future work.

Conclusion and Future Directions

7.1 Conclusion

In this thesis, we explored *Differentiable Rendering for Dense Visual SLAM*, including 3D scene representation and the way to optimize it from the run-time 2D image observations in real-time.

In Chapter 3, we introduced the NeRF based monocular mapping approach. We demonstrated the domain-agnostic dense reconstruction in real-time by making use of the smoothness inductive bias inherent in multi-layer perceptron network and differentiable rendering framework. The method works solely based on run-time observation, and therefore it can be deployed from small desk-scale objects to outdoor scene reconstruction.

To address the limitations of a single, world-centric NeRF representation, we extended our NeRF-based approach in Chapter 4, enabling large-scale mapping with frequent loop closures. By dynamically allocating multiple NeRF models based on run-time observations, we demonstrated a promising direction for online SLAM.

Next, in Chapter 5, we proposed Gaussian Splatting SLAM, arguing that 3D Gaussians are better suited to SLAM-type online state estimation tasks than NeRF due to their flexible spatial allocation and fast rendering. We presented the first 3DGS-based SLAM system that achieves superior performance even without direct depth measurements from external sensors.

Finally, in Chapter 6, we introduced 4DTAM, which models dynamic scenes and non-rigid deformations through a simple 3D scene representation and motion field optimized by differentiable rendering. This work points to exciting future directions for SLAM in more natural, dynamically-changing environments.

We demonstrated that differentiable rendering brings unique advantages for SLAM, including complex light-field and motion-field capture, high-fidelity rendering, and unified optimization through end-to-end processes. Particularly, 3DGS offers major advantages in flexible resource allocation and rendering speed. We believe that 3DGS has become a core scene representation for SLAM with differentiable rendering. At the same time, several technical limitations exist, which will be addressed in future research. Here, we discuss the current challenges and possible future research directions.

Real-time performance: One of the major issue of NeRF/3DGS-based SLAM is its slow processing time. While classical dense SLAM method such as ElasticFusion [Whelan et al., 2015b] runs in hard real-time 30fps including tracking, mapping and rendering with desktop GPU at the time of 2015 (GTX 780Ti), most of the NeRF/3DGS-based SLAM runs slower than 1fps even with modern GPU with much more FLOPs and VRAM (e.g. NVIDIA RTX4090). This makes current NeRF/3DGS-based SLAM approaches impractical for real-world online applications on embodied devices but only performs on offline server GPU. Since SLAM by definition is a real-time system, this issue is critical and needs to be addressed in the future research. Since 3DGS has shown promising real-time rendering capabilities, reducing the optimization time is the next big challenge for SLAM. Possible directions include developing a compact representation that reduces the number of primitives or utilizing higher-order optimizers instead of standard first-order gradient descent. Another approach could involve leveraging learned geometric priors to reduce the number of iterations required during test-time optimization.

Exploiting Learned Prior: The research presented in this thesis primarily focuses on purely test-time optimization, meaning the methods do not rely on any strong priors predicted by neural networks. In practical system development, these test-time optimization approaches can be combined with available learned priors, such as monocular depth or surface normals. The insights and techniques described here will be valuable for refining initial prior predictions at test time when integrated. Such an integration is both practically useful and an interesting research direction in robotics. Another direction could involve using rendering-based SLAM methods to train other networks or priors. Since these methods are differentiable and can propagate gradients end-to-end, they can be used as components to train larger networks. This is a relatively unexplored but promising research topic.

Bibliography

- [Pol,] Poly haven. <https://polyhaven.com/textures/fabric>. Accessed: 2024-11-01. 121
- [Ske,] Sketchfab. <https://sketchfab.com/>. Accessed: 2024-11-01. 120, 121
- [Apple.,] Apple. Apple vision pro. <https://www.apple.com/apple-vision-pro/>. 2
- [Barron et al., 2022] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5, 58, 62, 63, 64
- [Bloesch et al., 2018] Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J. (2018). CodeSLAM — learning a compact, optimisable representation for dense visual SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2, 19, 39, 59
- [Božič et al., 2020a] Božič, A., Palafox, P., Zollhöfer, M., Thies, J., Dai, A., and Nießner, M. (2020a). Neural deformation graphs for globally-consistent non-rigid reconstruction. *arXiv preprint arXiv:2012.01451*. 112
- [Božič et al., 2020b] Božič, A., Zollhöfer, M., Theobalt, C., and Nießner, M. (2020b). Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 113, 123
- [Burri et al., 2016] Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The EuRoC micro aerial vehicle datasets. *International Journal of Robotics Research (IJRR)*, 35(10):1157–1163. 46

- [Butler et al., 2012] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. [113](#)
- [Cai et al., 2022] Cai, H., Feng, W., Feng, X., Wang, Y., and Zhang, J. (2022). Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*. [112](#), [124](#)
- [Campos et al., 2021a] Campos, C., Elvira, R., Gómez, J. J., Montiel, J. M. M., and Tardós, J. D. (2021a). ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics (T-RO)*, 37(6):1874–1890. [94](#)
- [Campos et al., 2021b] Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. (2021b). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890. [52](#)
- [Chen and Zhang, 2019] Chen, Z. and Zhang, H. (2019). Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948. [23](#), [58](#)
- [Civera et al., 2008] Civera, J., Davison, A. J., and Montiel, J. M. (2008). Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945. [61](#)
- [Community, 2018] Community, B. O. (2018). Blender - a 3d modelling and rendering package. [113](#), [120](#)
- [Czarnowski et al., 2020a] Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. J. (2020a). DeepFactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters*, 5(2):721–728. [19](#)
- [Czarnowski et al., 2020b] Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. J. (2020b). Deepfactors: Real-time probabilistic dense monocular SLAM. *IEEE Robotics and Automation Letters*, 5(2):721–728. [78](#)
- [Dai et al., 2017a] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017a). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [47](#), [67](#)
- [Dai et al., 2017b] Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017b). Bundldefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 36(4):1. [38](#), [59](#), [70](#), [73](#), [78](#)

- [Davison et al., 2007] Davison, A. J., Molton, N. D., Reid, I., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067. 2, 3, 38
- [Deitke et al., 2023] Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., Vanderbilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., and Farhadi, A. (2023). Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153. 113, 120, 121
- [Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee. 23
- [Deng et al., 2021] Deng, K., Liu, A., Zhu, J.-Y., and Ramanan, D. (2021). Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*. 42
- [Dexheimer and Davison, 2023] Dexheimer, E. and Davison, A. J. (2023). Learning a Depth Covariance Function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 88, 90
- [Duisterhof et al., 2024] Duisterhof, B. P., Mandi, Z., Yao, Y., Liu, J.-W., Seidenschwarz, J., Shou, M. Z., Deva, R., Song, S., Birchfield, S., Wen, B., and Ichnowski, J. (2024). DeformGS: Scene flow in highly deformable scenes for deformable object manipulation. *WAFR*. 112
- [Dyson.,] Dyson. Robot vacuums. <https://www.dyson.com/vacuum-cleaners/robot>. 2
- [Engel et al., 2017] Engel, J., Koltun, V., and Cremers, D. (2017). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 2, 3, 4, 59, 78, 84, 88, 90, 120
- [Engel et al., 2014] Engel, J., Schoeps, T., and Cremers, D. (2014). LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2, 19, 38
- [Forster et al., 2014] Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. 78, 94
- [Fridovich-Keil et al., 2022] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. (2022). Plenoxels: Radiance fields without neural networks. In *Proceedings of*

- the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510. [28](#), [79](#)
- [Gallup et al., 2007] Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q., and Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE. [47](#)
- [Gao et al., 2022] Gao, H., Li, R., Tulsiani, S., Russell, B., and Kanazawa, A. (2022). Monocular dynamic view synthesis: A reality check. In *NeurIPS*. [113](#), [123](#)
- [Gao and Tedrake, 2018] Gao, W. and Tedrake, R. (2018). Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Proceedings of Robotics: Science and Systems (RSS)*. [112](#), [123](#), [124](#), [125](#)
- [Gao et al., 2023] Gao, Y., Cao, Y.-P., and Shan, Y. (2023). Surfelnerf: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 108–118. [28](#)
- [Garg et al., 2013] Garg, R., Roussos, A., and Agapito, L. (2013). Dense variational reconstruction of non-rigid surfaces from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [111](#)
- [Guédon and Lepetit, 2024] Guédon, A. and Lepetit, V. (2024). Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [111](#)
- [Habermann et al., 2020] Habermann, M., Xu, W., Zollhofer, M., Pons-Moll, G., and Theobalt, C. (2020). Deepcap: Monocular human performance capture using weak supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5052–5063. [112](#)
- [He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969. [23](#)
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. [23](#)
- [Ho et al., 2018] Ho, B.-J., Sodhi, P., Teixeira, P., Hsiao, M., Kusnur, T., and Kaess, M. (2018). Virtual occupancy grid map for submap-based pose graph slam and planning in 3d environ-

- ments. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 20
- [Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*. 21
- [Huang et al., 2024a] Huang, B., Yu, Z., Chen, A., Geiger, A., and Gao, S. (2024a). 2d gaussian splatting for geometrically accurate radiance fields. In *Proceedings of SIGGRAPH*. 110, 111, 113
- [Huang et al., 2021] Huang, J., Huang, S.-S., Song, H., and Hu, S.-M. (2021). Di-fusion: On-line implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 88, 90
- [Huang et al., 2024b] Huang, Y.-H., Sun, Y.-T., Yang, Z., Lyu, X., Cao, Y.-P., and Qi, X. (2024b). Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 112
- [Innmann et al., 2016] Innmann, M., Zollhöfer, M., Nießner, M., Theobalt, C., and Stamminger, M. (2016). VolumeDeform: Real-time Volumetric Non-rigid Reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 112
- [Ionescu et al., 2013] Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2013). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339. 112
- [Johari et al., 2023] Johari, M. M., Carta, C., and Fleuret, F. (2023). ESLAM: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 79, 88, 90, 100, 111
- [Kähler et al., 2016] Kähler, O., Prisacariu, V. A., and Murray, D. W. (2016). Real-time large-scale dense 3d reconstruction with loop closure. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VIII 14*, pages 500–516. Springer. 59
- [Kähler et al., 2016] Kähler, O., Prisacariu, V. A., and Murray, D. W. (2016). Real-time large-scale dense 3d reconstruction with loop closure. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 110

- [Kato et al., 2018a] Kato, H., Ushiku, Y., and Harada, T. (2018a). Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4, 24
- [Kato et al., 2018b] Kato, H., Ushiku, Y., and Harada, T. (2018b). Neural 3D mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916. 111
- [Keetha et al., 2024] Keetha, N., Karhade, J., Jatavallabhula, K. M., Yang, G., Scherer, S., Ramanan, D., and Luiten, J. (2024). Splatam: Splat, track and map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 111
- [Keller et al., 2013] Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., and Kolb, A. (2013). Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion. In *Proceedings of the International Conference on 3D Vision (3DV)*. 78, 80
- [Kerbl et al., 2023] Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*. 4, 5, 24, 30, 31, 76, 77, 87, 93, 111
- [Kerl et al., 2013a] Kerl, C., Sturm, J., and Cremers, D. (2013a). Dense visual slam for rgb-d cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*. 19
- [Kerl et al., 2013b] Kerl, C., Sturm, J., and Cremers, D. (2013b). Dense visual slam for rgb-d cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 20, 38
- [Keselman and Hebert, 2022] Keselman, L. and Hebert, M. (2022). Approximate differentiable rendering with algebraic surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 30, 79
- [Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 81
- [Klein and Murray, 2007] Klein, G. and Murray, D. W. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 2, 3, 38

- [Kobayashi et al., 2022] Kobayashi, S., Matsumoto, E., and Sitzmann, V. (2022). Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330. [30](#)
- [Koestler et al., 2021] Koestler, L., Yang, N., Zeller, N., and Cremers, D. (2021). Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *Proceedings of the 5th Conference on Robot Learning*. [4](#)
- [Kopanas et al., 2021] Kopanas, G., Philip, J., Leimkühler, T., and Drettakis, G. (2021). Point-based neural rendering with per-view optimization. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 40(4). [30](#)
- [Labbé and Michaud, 2019] Labbé, M. and Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics (JFR)*. [20](#)
- [Li et al., 2023] Li, H., Gu, X., Yuan, W., Yang, L., Dong, Z., and Tan, P. (2023). Dense rgb slam with neural implicit maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*. [88](#)
- [Liu et al., 2020] Liu, L., Gu, J., Lin, K. Z., Chua, T.-S., and Theobalt, C. (2020). Neural sparse voxel fields. *NeurIPS*. [79](#)
- [Liu et al., 2019] Liu, S., Li, T., Chen, W., and Li, H. (2019). Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the International Conference on Computer Vision (ICCV)*. [4](#), [24](#)
- [Lombardi et al., 2019] Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., and Sheikh, Y. (2019). Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*. [24](#)
- [Loper et al., 2015] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16. [112](#)
- [Loper and Black, 2014] Loper, M. M. and Black, M. J. (2014). Opendr: An approximate differentiable renderer. In *Proceedings of the European Conference on Computer Vision (ECCV)*. [22](#), [24](#)
- [Luiten et al., 2024] Luiten, J., Kopanas, G., Leibe, B., and Ramanan, D. (2024). Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *3DV*. [80](#), [112](#), [120](#), [124](#)

- [Ma and Karaman, 2018] Ma, F. and Karaman, S. (2018). Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 47, 50, 51
- [Ma et al., 2021] Ma, R., Wang, R., Zhang, Y., Pizer, S., McGill, S. K., Rosenman, J., and Frahm, J.-M. (2021). Rnnslam: Reconstructing the 3d colon to visualize missing regions during a colonoscopy. *Medical image analysis*, 72:102100. 112
- [Matsuki et al., 2025] Matsuki, H., Bae, G., and Davison, A. (2025). 4dtam: Non-rigid tracking and mapping via dynamic surface gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6, 7
- [Matsuki et al., 2024a] Matsuki, H., Murai, R., Kelly, P. H. J., and Davison, A. J. (2024a). Gaussian Splatting SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6, 7, 111, 118, 120, 123, 124
- [Matsuki et al., 2021] Matsuki, H., Scona, R., Czarnowski, J., and Davison, A. J. (2021). Codemapping: Real-time dense mapping for sparse slam using compact scene representations. *IEEE Robotics and Automation Letters*, 6(4):7105–7112. 7, 36, 39, 40
- [Matsuki et al., 2023] Matsuki, H., Sucar, E., Laidow, T., Wada, K., Scona, R., and Davison, A. J. (2023). imode: Real-time incremental monocular dense mapping using neural field. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4171–4177. IEEE. 5, 7, 36
- [Matsuki et al., 2024b] Matsuki, H., Tateno, K., Niemeyer, M., and Tombari, F. (2024b). Newton: Neural view-centric mapping for on-the-fly large-scale slam. *IEEE Robotics and Automation Letters*. 6, 7, 55
- [Mazur et al., 2023] Mazur, K., Sucar, E., and Davison, A. (2023). Feature-realistic neural fusion for real-time, open set scene understanding. In *2023 International Conference on Robotics and Automation (ICRA)*. 30
- [McCormac et al., 2017] McCormac, J., Handa, A., Davison, A., and Leutenegger, S. (2017). Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 79
- [Mescheder et al., 2019] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of*

- the *IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470. [23](#), [58](#)
- [Meuleman et al., 2023] Meuleman, A., Liu, Y.-L., Gao, C., Huang, J.-B., Kim, C., Kim, M. H., and Kopf, J. (2023). Progressively optimized local radiance fields for robust view synthesis. In *CVPR*. [58](#)
- [Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*. [4](#), [24](#), [27](#), [28](#), [36](#), [37](#), [39](#), [43](#), [58](#), [61](#), [63](#), [79](#), [111](#), [118](#)
- [Mitra et al., 2004] Mitra, N. J., Gelfand, N., Pottmann, H., and Guibas, L. J. (2004). Registration of Point Cloud Data from a Geometric Optimization Perspective. In *Proceedings of the Symposium on Geometry Processing*. [95](#)
- [Müller, 2021] Müller, T. (2021). tiny-cuda-nn. [124](#)
- [Müller et al., 2022] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (TOG)*. [28](#), [58](#), [60](#), [63](#), [79](#), [111](#), [118](#)
- [Mur-Artal and Tardós, 2015] Mur-Artal, R. and Tardós, J. D. (2015). Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *Robotics: Science and Systems*, volume 2015. Rome. [42](#)
- [Mur-Artal and Tardós, 2017a] Mur-Artal, R. and Tardós, J. D. (2017a). ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5):1255–1262. [2](#), [3](#), [78](#), [90](#)
- [Mur-Artal and Tardós, 2017b] Mur-Artal, R. and Tardós, J. D. (2017b). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262. [38](#), [41](#), [57](#), [59](#), [60](#), [67](#)
- [Murez et al., 2020] Murez, Z., van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., and Rabinovich, A. (2020). Atlas: End-to-end 3d scene reconstruction from posed images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. [59](#)
- [Newcombe, 2012] Newcombe, R. A. (2012). *Dense Visual SLAM*. PhD thesis, Imperial College London. [95](#)

- [Newcombe et al., 2015] Newcombe, R. A., Fox, D., and Seitz, S. M. (2015). Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [112](#)
- [Newcombe et al., 2011a] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE. [4](#), [21](#), [58](#)
- [Newcombe et al., 2011b] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011b). KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. [3](#), [4](#), [78](#), [79](#), [110](#)
- [Newcombe et al., 2011c] Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011c). DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. [38](#)
- [Niemeyer et al., 2020a] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020a). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, pages 3501 – 3512, Piscataway, NJ. IEEE. [24](#), [58](#)
- [Niemeyer et al., 2020b] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020b). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515. [39](#)
- [Niemeyer et al., 2020c] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020c). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [79](#)
- [Niemeyer et al., 2020d] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A. (2020d). Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [111](#)

- [Oechsle et al., 2021] Oechsle, M., Peng, S., and Geiger, A. (2021). Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599. [29](#), [30](#), [44](#)
- [Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174. [23](#), [58](#)
- [Park et al., 2021] Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. (2021). Nerfies: Deformable neural radiance fields. *ICCV*. [112](#)
- [Peng et al., 2020] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. [58](#)
- [Peng et al., 2021] Peng, S., Zhang, Y., Xu, Y., Wang, Q., Shuai, Q., Bao, H., and Zhou, X. (2021). Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9054–9063. [112](#)
- [Pizzoli et al., 2014] Pizzoli, M., Forster, C., and Scaramuzza, D. (2014). Remode: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616. IEEE. [47](#)
- [Prisacariu et al., 2014] Prisacariu, V. A., Kähler, O., Cheng, M., Ren, C. Y., Valentin, J. P. C., Torr, P. H. S., Reid, I. D., and Murray, D. W. (2014). A framework for the volumetric integration of depth images. *arXiv preprint arXiv:1410.0925*. [78](#)
- [Prisacariu et al., 2017] Prisacariu, V. A., Kähler, O., Golodetz, S., Sapienza, M., Cavallari, T., Torr, P. H., and Murray, D. W. (2017). Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*. [21](#)
- [Pumarola et al., 2020] Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. (2020). D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*. [112](#)
- [Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660. [23](#)

- [Qin et al., 2019] Qin, T., Cao, S., Pan, J., and Shen, S. (2019). A general optimization-based framework for global pose estimation with multiple sensors. [94](#)
- [Reiser et al., 2021] Reiser, C., Peng, S., Liao, Y., and Geiger, A. (2021). Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the International Conference on Computer Vision (ICCV)*. [28](#)
- [Riegler et al., 2017] Riegler, G., Osman Ulusoy, A., and Geiger, A. (2017). Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586. [23](#)
- [Rodriguez et al., 2023] Rodriguez, J. J. G., Montiel, J. M. M., and Tardos, J. D. (2023). Nr-slam: Non-rigid monocular slam. *IEEE Transactions on Robotics (T-RO)*. [112](#)
- [Roessle et al., 2021] Roessle, B., Barron, J. T., Mildenhall, B., Srinivasan, P. P., and Nießner, M. (2021). Dense depth priors for neural radiance fields from sparse input views. *arXiv preprint arXiv:2112.03288*. [50](#)
- [Rosinol et al., 2023] Rosinol, A., Leonard, J. J., and Carlone, L. (2023). Nerf-slam: Real-time dense monocular slam with neural radiance fields. *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. [56](#), [57](#), [59](#), [66](#), [68](#), [70](#)
- [Rückert et al., 2022] Rückert, D., Franke, L., and Stamminger, M. (2022). Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)*, 41(4):1–14. [30](#)
- [Rünz and Agapito, 2017] Rünz, M. and Agapito, L. (2017). Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. [111](#)
- [Sandström et al., 2023] Sandström, E., Li, Y., Van Gool, L., and R. Oswald, M. (2023). Point-slam: Dense neural point cloud-based slam. In *Proceedings of the International Conference on Computer Vision (ICCV)*. [79](#), [88](#), [90](#), [94](#), [100](#), [111](#), [123](#), [124](#)
- [Schöps et al., 2019] Schöps, T., Sattler, T., and Pollefeys, M. (2019). Surfelmeshing: Online surfel-based mesh reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. [19](#)
- [Schöps et al., 2019] Schöps, T., Sattler, T., and Pollefeys, M. (2019). Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [3](#), [19](#), [38](#), [59](#), [70](#), [78](#), [79](#), [90](#), [110](#)

- [Scona et al., 2018] Scona, R., Jaimez, M., Petillot, Y. R., Fallon, M., and Cremers, D. (2018). StaticFusion: Background reconstruction for dense rgb-d slam in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 111
- [Scona et al., 2022] Scona, R., Matsuki, H., and Davison, A. (2022). From scene flow to visual odometry through local and global regularisation in markov random fields. *IEEE Robotics and Automation Letters*, 7(2):4299–4306. 7
- [Seidenschwarz et al., 2024] Seidenschwarz, J., Zhou, Q., Duisterhof, B., Ramanan, D., and Leal-Taixé, L. (2024). Dynomo: Online point tracking by dynamic online monocular gaussian reconstruction. 112
- [Silberman et al., 2012] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from RGBD images. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 47
- [Sitzmann et al., 2019a] Sitzmann, V., Zollhöfer, M., and Wetzstein, G. (2019a). Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*. 24
- [Sitzmann et al., 2019b] Sitzmann, V., Zollhöfer, M., and Wetzstein, G. (2019b). Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*. 58
- [Slavcheva et al., 2017] Slavcheva, M., Baust, M., Cremers, D., and Ilic, S. (2017). Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 112, 113, 123
- [Slavcheva et al., 2018] Slavcheva, M., Baust, M., and Ilic, S. (2018). Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 112
- [Solà et al., 2018] Solà, J., Deray, J., and Atchuthan, D. (2018). A micro Lie theory for state estimation in robotics. *arXiv:1812.01537*. 81, 117
- [Straub et al., 2019] Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., et al. (2019). The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*. 46, 72, 87, 123

- [Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. [46](#), [87](#), [123](#)
- [Sucar et al., 2021] Sucar, E., Liu, S., Ortiz, J., and Davison, A. J. (2021). iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. [2](#), [27](#), [37](#), [39](#), [46](#), [56](#), [57](#), [59](#), [66](#), [68](#), [72](#), [79](#), [87](#), [88](#), [90](#), [91](#), [98](#), [99](#), [111](#)
- [Sucar et al., 2020] Sucar, E., Wada, K., and Davison, A. (2020). Nodeslam: Neural object descriptors for multi-view shape reconstruction. In *2020 International Conference on 3D Vision (3DV)*, pages 949–958. IEEE. [39](#)
- [Sun et al., 2022a] Sun, C., Sun, M., and Chen, H. (2022a). Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [79](#)
- [Sun et al., 2022b] Sun, C., Sun, M., and Chen, H.-T. (2022b). Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5459–5469. [28](#)
- [Sun et al., 2021] Sun, J., Xie, Y., Chen, L., Zhou, X., and Bao, H. (2021). NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [47](#), [58](#)
- [Tancik et al., 2022] Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P., Barron, J. T., and Kretzschmar, H. (2022). Block-NeRF: Scalable large scene neural view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [58](#), [66](#)
- [Tancik et al., 2020] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547. [46](#)
- [Tancik et al., 2023] Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., and Kanazawa, A. (2023). Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*. [63](#), [66](#), [67](#), [72](#)

- [Tang et al., 2024] Tang, J., Ren, J., Zhou, H., Liu, Z., and Zeng, G. (2024). Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *Proceedings of the International Conference on Learning Representations (ICLR)*. 80
- [Tateno et al., 2017] Tateno, K., Tombari, F., Laina, I., and Navab, N. (2017). CNN-SLAM: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 39, 46
- [team, 2022] team, N. (2022). nerfstudio documentation. https://docs.nerf.studio/en/latest/nerfology/model_components/visualize_spatial_distortions.html#scene-contraction. 63
- [Teed and Deng, 2021] Teed, Z. and Deng, J. (2021). DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. *Neural Information Processing Systems (NeurIPS)*, 34:16558–16569. 59, 68, 78, 88, 90
- [Torresani et al., 2008] Torresani, L., Hertzmann, A., and Chris Bregler, C. (2008). Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(5). 111
- [Tretschk et al., 2021] Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., and Theobalt, C. (2021). Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 112
- [Tschernezki et al., 2022] Tschernezki, V., Laina, I., Larlus, D., and Vedaldi, A. (2022). Neural feature fusion fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*. 30
- [Vespa et al., 2018] Vespa, E., Nikolov, N., Grimm, M., Nardi, L., Kelly, P. H., and Leutenegger, S. (2018). Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151. 3, 21, 58
- [Wang et al., 2022] Wang, A., Wang, P., Sun, J., Kortylewski, A., and Yuille, A. (2022). Voge: a differentiable volume renderer using gaussian ellipsoids for analysis-by-synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 79
- [Wang et al., 2023] Wang, H., Wang, J., and Agapito, L. (2023). Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 27, 88, 90, 91, 98, 99, 111

- [Wang et al., 2024a] Wang, H., Wang, J., and Agapito, L. (2024a). Morpheus: Neural dynamic 360deg surface reconstruction from monocular rgb-d video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20965–20976. [112](#), [113](#), [123](#), [124](#), [128](#)
- [Wang et al., 2021a] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021a). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*. [29](#), [30](#)
- [Wang et al., 2021b] Wang, Q., Wang, Z., Genova, K., Srinivasan, P. P., Zhou, H., Barron, J. T., Martin-Brualla, R., Snavely, N., and Funkhouser, T. A. (2021b). Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [58](#)
- [Wang et al., 2024b] Wang, Q., Ye, V., Gao, H., Austin, J., Li, Z., and Kanazawa, A. (2024b). Shape of motion: 4d reconstruction from a single video. In *arXiv preprint arXiv:2407.13764*. [112](#)
- [Wang et al., 2024c] Wang, R., Xu, S., Dai, C., Xiang, J., Deng, Y., Tong, X., and Yang, J. (2024c). Moge: Unlocking accurate monocular geometry estimation for open-domain images with optimal training supervision. [126](#)
- [Wang et al., 2020] Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., and Scherer, S. (2020). Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916. IEEE. [113](#)
- [Wei et al., 2021] Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., and Zhou, J. (2021). Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619. [50](#)
- [Whelan et al., 2015a] Whelan, T., Kaess, M., Johannsson, H., Fallon, M. F., Leonard, J. J., and McDonald, J. B. (2015a). Real-time large scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626. [67](#), [78](#), [90](#), [110](#)
- [Whelan et al., 2015b] Whelan, T., Leutenegger, S., Salas-Moreno, R. F., Glocker, B., and Davison, A. J. (2015b). Elasticfusion: Dense SLAM without a pose graph. In *Proceedings of Robotics: Science and Systems (RSS)*. [3](#), [19](#), [38](#), [78](#), [79](#), [80](#), [94](#), [110](#), [132](#)

- [Whelan et al., 2016] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). Elasticfusion: Real-time dense slam and light source estimation. *International Journal of Robotics Research (IJRR)*. 19, 20
- [Wizadwongsa et al., 2021] Wizadwongsa, S., Phongthawee, P., Yenphraphai, J., and Suwajanakorn, S. (2021). Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 58
- [Wu et al., 2024] Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., and Wang, X. (2024). 4d gaussian splatting for real-time dynamic scene rendering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 80
- [Xu et al., 2022] Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., and Neumann, U. (2022). Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448. 28, 29
- [Yan et al., 2024] Yan, C., Qu, D., Xu, D., Zhao, B., Wang, Z., Wang, D., and Li, X. (2024). Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*. 111
- [Yang et al., 2020] Yang, N., Stumberg, L. v., Wang, R., and Cremers, D. (2020). D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 39
- [Yang et al., 2022] Yang, X., Li, H., Zhai, H., Ming, Y., Liu, Y., and Zhang, G. (2022). Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 88, 90, 111
- [Yang et al., 2024a] Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., and Jin, X. (2024a). Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 112, 117, 124
- [Yang et al., 2024b] Yang, Z., Yang, H., Pan, Z., Zhu, X., and Zhang, L. (2024b). Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *Proceedings of the International Conference on Learning Representations (ICLR)*. 80, 112, 117
- [Yariv et al., 2021a] Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021a). Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815. 29, 37, 39

- [Yariv et al., 2021b] Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021b). Volume rendering of neural implicit surfaces. In *Neural Information Processing Systems (NeurIPS)*. 58
- [Yi et al., 2024] Yi, T., Fang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., and Wang, X. (2024). Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 80
- [Yu et al., 2021a] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. (2021a). Plen-Octrees for real-time rendering of neural radiance fields. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 28
- [Yu et al., 2021b] Yu, A., Ye, V., Tancik, M., and Kanazawa, A. (2021b). pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 58
- [Yu et al., 2024] Yu, Z., Sattler, T., and Geiger, A. (2024). Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*. 111
- [Zeng et al., 2013] Zeng, M., Zhao, F., Zheng, J., and Liu, X. (2013). Octree-based fusion for realtime 3d reconstruction. *Graphical Models*. 21
- [Zhang et al., 2020] Zhang, K., Riegler, G., Snively, N., and Koltun, V. (2020). Nerf++: Analyzing and improving neural radiance fields. *arxiv*, 2010.07492. 58
- [Zhi et al., 2021] Zhi, S., Laidlow, T., Leutenegger, S., and Davison, A. (2021). In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 29, 30, 51
- [Zhi et al., 2022] Zhi, S., Sucar, E., Mouton, A., Haughton, I., Laidlow, T., and Davison, A. J. (2022). ilabel: Revealing objects in neural fields. *IEEE Robotics and Automation Letters*, 8(2):832–839. 29
- [Zhu et al., 2024] Zhu, Z., Peng, S., Larsson, V., Cui, Z., Oswald, M. R., Geiger, A., and Pollefeys, M. (2024). Nicer-slam: Neural implicit scene encoding for rgb slam. *International Conference on 3D Vision (3DV)*. 79
- [Zhu et al., 2022] Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., and Pollefeys, M. (2022). Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings*

of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [27](#), [56](#), [57](#), [59](#), [66](#), [67](#), [68](#), [70](#), [72](#), [79](#), [88](#), [90](#), [91](#), [111](#)

[Zienkiewicz et al., 2016] Zienkiewicz, J., Davison, A., and Leutenegger, S. (2016). Real-time height map fusion using differentiable rendering. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. [22](#)

[Zwicker et al., 2002] Zwicker, M., Pfister, H., van Baar, J., and Gross, M. (2002). Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238. [31](#), [81](#)