Diss. ETH No. 29858

Neural Scene Representations for 3D Reconstruction and Scene Understanding

A thesis submitted to attain the degree of
Doctor of Sciences of **ETH Zurich**
(Dr. sc. ETH Zurich)

presented by

Songyou Peng
Erasmus Mundus MSc in Vision and Robotics (VIBOT)
Heriot-Watt University / Universitat de Girona / Université de Bourgogne
Born on 10.07.1992
Citizen of China

accepted on the recommendation of
Prof. Dr. Marc Pollefeys, examiner
Prof. Dr. Andreas Geiger, co-examiner
Prof. Dr. Vincent Sitzmann, co-examiner
Prof. Dr. Leonidas J. Guibas, co-examiner

2023

Abstract

In an era where machines are increasingly integrated into our daily lives, their ability to perceive and understand the three-dimensional world becomes of great importance. Central to this capability is the scene representation, which translates sensory data into compact, detailed, and holistic descriptions of the environment. While deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized many facets of computer vision, its primary focus remains on 2D information. This thesis delves into the challenges and potential of transitioning these technologies to 3D environments, aiming to bridge the gap between machine perception and human-like spatial understanding.

Our primary objective is to pioneer the development of neural scene representations tailored for accurate 3D reconstruction and comprehensive 3D scene understanding. We start by introducing a scalable scene representation tailored for deep-learning-based 3D reconstruction. This representation is capable of capturing 3D shapes in a continuous, resolution-agnostic fashion, effectively addressing the constraints of traditional explicit-based methods. Next, by incorporating a differentiable point-to-mesh layer, we present a lightweight representation that ensures high-quality reconstruction with rapid inference, addressing the need for speed in real-world applications. Furthermore, we explore the realm of dense visual Simultaneous Localization and Mapping (SLAM) with a system that employs hierarchical neural implicit representations. This approach enables detailed reconstruction in large-scale indoor scenarios, pushing the boundaries of what's achievable with current SLAM systems. Lastly, our research culminates in the development of a unified scene representation for a broad spectrum of 3D scene understanding tasks, bypassing the need for costly 3D labeled data.

In conclusion, this thesis presents a series of advancements in neural scene representations, offering solutions that not only enhance 3D reconstruction capabilities but elevate the level of 3D scene understanding, bringing us a step closer to achieving machine perception that mirrors human cognition.

Zusammenfassung

In einer Zeit, in der Maschinen zunehmend in unseren Alltag integriert werden, spielt ihre Fähigkeit, die dreidimensionale Welt wahrzunehmen und zu begreifen, eine bedeutende Rolle. Die Szenendarstellung ist von zentraler Bedeutung für diese Fähigkeit, da sie Sensordaten in kompakte, detaillierte und holistische Beschreibungen der Umgebung umwandelt. Obwohl Deep Learning, insbesondere Convolutional Neural Networks (CNNs), viele Facetten der Computer Vision revolutioniert hat, liegt der Schwerpunkt nach wie vor auf 2D-Daten. Diese Dissertation befasst sich mit den Herausforderungen und dem Potenzial der Übertragung dieser Technologien auf die 3D-Welt, um die Lücke zwischen maschineller Wahrnehmung und menschenähnlichem Raumverständnis zu schliessen.

Unser primäres Ziel ist es, Pionierarbeit bei der Entwicklung neuronaler Szenendarstellungen zu leisten, die auf eine genaue 3D-Rekonstruktion und ein umfassendes 3D-Szenenverständnis zugeschnitten sind. Wir beginnen mit der Implementierung einer skalierbaren Szenendarstellung, die für Deep-Learning-basierte 3D-Rekonstruktion ausgerichtet ist. Dies ermöglicht es, 3D-Formen in einer kontinuierlichen, auflösungsunabhängigen Art und Weise zu erfassen, wodurch die Beschränkungen traditioneller, explizit-basierter Methoden effektiv angegangen werden. Als Nächstes präsentieren wir durch die Einbeziehung einer differenzierbaren Punkt-zu-Mesh-Schicht eine leichtgewichtige Darstellungsmethode, die eine qualitativ hochwertige Rekonstruktion mit schneller Inferenz gewährleistet und so dem Bedarf an Geschwindigkeit in realen Anwendungen gerecht wird. Zusatzlicht untersuchen wir den Bereich der dichten visuellen Simultaneous Localization and Mapping (SLAM), mit einer Methode, die hierarchische neuronale implizite Repräsentationen verwendet. Dieser Ansatz ermöglicht eine detaillierte Rekonstruktion in

grossflächigen Innenraumszenarien und erweitert die Grenzen dessen, was mit aktuellen SLAM-Methoden erreicht werden kann. Schliesslich kulminiert unsere Forschung in der Entwicklung einer einheitlichen Szenendarstellung für ein breites Spektrum von 3D-Szenenverständnisaufgaben und umgeht die Notwendigkeit kostspieliger 3D-beschrifteter Daten.

Insgesamt präsentiert diese Arbeit eine Reihe von Fortschritten in der neuronalen Szenendarstellung. Dabei bietet sie Lösungen, die nicht nur die 3D-Rekonstruktionsfähigkeiten verbessern, sondern auch das Niveau des 3D-Szenenverständnisses anheben, womit sie uns einen Schritt näher an eine maschinelle Wahrnehmung bringt, die die menschliche Kognition widerspiegelt.

Acknowledgments

As I reflect on my PhD journey, it is the remarkable individuals and groups that stand out – those with whom I have shared unforgettable memories.

Advisors. I wish to express my deepest gratitude to my PhD supervisors Prof. Marc Pollefeys and Prof. Andreas Geiger for offering me the privileged opportunity to pursue my PhD at two of the world's top research groups. Marc, your trust in me and the autonomy you granted inspired me to independently approach my research with creativity. Andreas, you not only have a remarkable vision of identifying key research questions, but also your insightful critiques consistently ensure our research stays on track, ultimately achieving the highest quality.

Collaborators. I am deeply grateful to Michael Niemeyer and Lars Mescheder for the invaluable guidance and continuous support on my first own project, which set the foundation for my entire PhD. Michael, your patience in guiding me was exceptional. Every discussion with you was a lesson for me. Throughout my PhD, having you as a brilliant collaborator, role model, office mate, and a cherished friend, has truly been a privilege. Lars, you are incredibly smart yet so humble, and always a joy to chat with. Thank you so much for all the sincere advice, both in and beyond research.

I am also very grateful to Chiyu "Max" Jiang, my favored collaborator and treasured friend. Whenever I am in doubt, you always come through with multiple solutions. And during those tough times, your comforting words and encouragement have been a guiding light, turning challenges into opportunities.

To Zihan Zhu, thanks for being so accepting as I took initial advisory steps. It is my great honor to work with such a talented student like you.

Special thanks also go to Shaohui Liu and Zhaopeng Cui, for bringing me into your project at the very beginning of my PhD and opening doors to a promising area that I have passionately pursued ever since.

My internships were enriched and made truly unforgettable due to the enormous support from numerous collaborators, especially Tom Funkhouser and Kyle Genova from Google Research, as well as Michael Zollhoefer and Shunsuke Saito from Meta Reality Labs Research.

Big thanks to Zehao Yu, Lixiang Lin, Victor Larsson, Martin R. Oswald, Yiyi Liao, Michael Oechsle, Christian Reiser, Rui Huang, Haiwen Huang, and many other brilliant collaborators. Truly, You are the reason my collaborative spirit thrives. It was also very privileged to (co-)mentor some talented and highly motivated students. I have learned so much and got inspired by Lei Li, Weining Ren, Mirlan Karimov, Gonca Yilmaz, Shengqu Cai, Severin Pfister, Weirong Chen.

I owe a profound debt of gratitude to Prof. Peter Sturm, who brought me to the world of computer vision and showed me the beauty of research. Your influence is the reason for my decision to dive into research and embark on this PhD journey. Without you, my path would have been vastly different. You are a role model for me in every way. I truly appreciate our friendship and your mentorship.

Colleagues, Friends, and Family. Reflecting on these 4 years of my PhD journey, doing research is only a part of the story. I would not have come this far without the support of incredible colleagues, dear friends, and loving family.

My heartfelt gratitude is reserved for my peers at AVG – Aditya, Anpei, Apratim, Axel, Bernhard, Bozidar, Caro, Christian, Chuqiao, Despoina, Gege, Haiwen, Haofei, Haoyu, Joo Ho, Kashyap, Katja, Katrin, Kerstin, Lynn, Markus, Niklas, Naama, Shaofei, Stefano, Takeru, Xu, Zehao. Our countless brainstorming sessions, inspiring discussions, coffee breaks, late-night drinkings, and of course tree tree-climbing and epic fun in Greece are memories I will treasure forever. AVG for me has been more than a group, but a home where I have a special bond.

My deep gratitude extends to the current and past members of CVG – Ayse, Boyang, Daniel B, Daniel T, Denys, Fangjinhua, Francis, Hermann, Iago, Ian, Iro, Jonas, Julia, Katarína, Linfei, Luca, Marcel, Mihai, Paul-Edouard, Peidong, Petr, Philipp, Rémi, Sandro, Silvan, Taein, Yao, Zador, Zuoyue, and Zuria. I really enjoyed all the social Thursdays, group lunches, and ski retreats we had together. Furthermore, I would like to send a hearty thanks to the fun individuals at VLG, IGL, AIT, and CVL. Your presence made my time in Zurich vibrant and memorable.

Outside the lab, my life would have felt incomplete without the incredible groups of friends from Zurich, Tübingen, and the US. I cherish all the precious moments shared with the AVG Bachelors group, MPI Chinese crew, Research4Food, the lively Chitchat WeChat group, the Chopsticks fan club, the D-Lab gangster, and Googleplex dinner squad. Also, to Feiyi, thank you for being by my side through the up-and-down carousel of a significant part of my PhD, and I have grown as a better individual because of our time together. I sincerely wish you all the best.

Finally, to my parents, your love and faith in me have been my strongest strength. Your kindness, tolerance, and patience have shaped me into the person I am. To my grandpa, watching from the heavens, I hope I made you proud.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

With the rapid advancements in science and technology, machines have seamlessly integrated into our daily lives. Now we find ourselves living alongside machines capable of driving cars, organizing our homes, and even assisting in medical surgeries. Central to these advances is the machine's ability to perceive and understand the surrounding environment.

For machines to effectively perceive the three-dimensional world, they need to model the surroundings from sensory data. In particular, accurately representing and reconstructing detailed geometry to their real-life counterparts is vital for applications in AR/VR, autonomous driving, robotics, etc. Yet, creating detailed geometry from scratch is a labor-intensive task, demanding specialized expertise. Despite the emergence of advanced software and user-friendly modeling tools, challenges like scalability and speed prohibit their

large-scale deployment. How to accurately construct geometric details for large scenes at speed is a primary focus of this thesis.

Once the 3D environment is constructed accurately, it is equally important to understand the semantics, affordances, functions, and physical properties of the reconstructed subjects. This kind of holistic understanding is pivotal for machines to really interact intelligently with humans in daily scenarios. However, traditional methods are often tailored for specific tasks, such as 3D semantic segmentation for a limited set of classes, leaving other tasks unaddressed. Achieving a broad understanding of 3D scenes is another objective of this thesis.

Scene representation, i.e. translating observations of an environment, either visual, haptic, auditory, or otherwise, into a concise model of the environment [157, 194], is naturally crucial for machines aiming to tackle complex tasks like accurately reconstructing a realistic scene and having a comprehensive understanding of our world. Recent advances in deep learning, particularly the emergence of Convolutional Neural Networks (CNNs), offer a promising way of deriving robust and powerful scene representations, termed here as *neural scene representations*.

CNNs have revolutionized many computer vision tasks, notably in areas like image classification and depth estimation, showcasing the potential of deep learning in processing visual information. However, much of their prowess is centered on processing 2D information. Transitioning these 2D-focused technologies to 3D environments poses distinct challenges. To effectively model and understand the complex world, it is essential for machines to learn 3D scene representations, enabling a deeper spatial understanding akin to how humans perceive the world.

The goal of this thesis is to pioneer the development of neural scene representations, specifically tailored to accurately reconstruct and comprehensively understand the 3D world. Our roadmap is marked with clear milestones that are all tied together. First, we want to

develop a scalable scene representation capable of faithfully reconstructing detailed 3D geometry, spanning from objects to large-scale scenes. Next, with the integration of a novel differentiable point-to-mesh layer, we can represent detailed shapes using just lightweight point clouds, and speed up the 3D reconstruction process. Third, we also investigate a hierarchical neural scene representation that empowers dense RGB-D SLAM applications, specifically for large indoor scenarios. Once obtaining the 3D reconstruction of a scene, the final piece of the thesis is to produce 3D neural scene representations for a plethora of 3D scene understanding tasks, leveraging only a 2D pre-trained model, thus bypassing the need for any costly 3D labeled data.

Overall, this thesis investigates various neural scene representations to produce detailed 3D scene reconstruction efficiently, and subsequently pushes the boundary of 3D scene understanding to another level. In the next section, we will delve into the actual problems and challenges.

1.2 Research Questions and Challenges

In this thesis, we are interested in developing neural scene representations for two different but closely related topics: 3D reconstruction and 3D scene understanding. We present the following research questions that we try to address in this thesis:

Research Question 1: ***What shape representation is scalable and suitable for detailed 3D reconstruction?***

Shape representations are pivotal for learning-based 3D reconstruction. Explicit shape representations, such as voxels, point clouds, or meshes, have been traditionally favored due to their simplicity. However, each has its limitations: voxels are limited in terms of resolution due to large memory requirements, point clouds discard topological relationships, and predicting mesh-based representations di-

rectly via neural networks is challenging. The recent neural implicit representations define shapes implicitly as the level set of a continuous function, parameterized with neural networks [29, 144, 166]. They can model dense surfaces in arbitrary topologies, but often fall short when reconstructing comparably simple objects. Our aim is to advance the neural implicit representations, enabling them to encode complex geometries across diverse topologies and scale to large scenes.

Research Question 2: ***Can we find a representation that is interpretable, lightweight, and facilitates rapid inference?***

As mentioned before, neural implicit representations gained popularity due to their expressiveness and flexibility. However, their reliance on heavy neural networks for encoding surface details often results in slow surface extraction as they require numerous network evaluations in 3D space. This significantly limits its feasibility for applications demanding fast inference. On the other hand, explicit representations like point clouds, require only a few parameters to represent the geometry, and it is very fast to predict. Therefore, our target is to benefit the best from both worlds, leading to a lightweight representation that ensures high-quality reconstruction at low inference times.

Research Question 3: ***How can neural implicit representations be employed for dense SLAM in large scenes?***

While our first two research questions explore optimal shape representations for 3D reconstruction from input point clouds, A more realistic scenario for 3D reconstruction is to densely model a scene solely from an unposed RGB(-D) sequence. This falls into the category of dense visual SLAM. Traditional dense visual SLAM systems are often unable to estimate plausible geometry for unobserved regions. Although recent SLAM approaches using neural implicit representations attain a certain level of predictive power, they are typically confined to smaller scenes due to their reliance on suboptimal neural scene representations. We want to circumvent this limitation

by introducing a novel hybrid representation, enabling the neural-implicit-based SLAM system for large-scale scenes.

Research Question 4: ***How to generate a unified neural representation for a broad set of 3D scene understanding tasks without any 3D supervision?***

Upon addressing the first three research questions, we can assume having obtained the 3D geometry of a scene. One natural downstream application is the understanding of this reconstructed scene. Previous learning-based methods usually handle one single 3D scene understanding task at a time, in a fully-supervised learning manner. Our aspiration instead is to develop a zero-shot method, producing a neural scene representation capable of inferring 3D semantics, affordances, physical properties, and beyond.

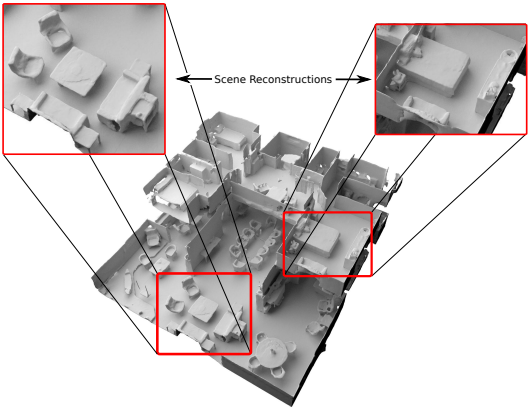
1.3 Contributions

This thesis addresses the research questions outlined earlier and contributes to the instigation of learning neural scene representations for 3D reconstruction as well as 3D scene understanding. Specific contributions are detailed as follows.

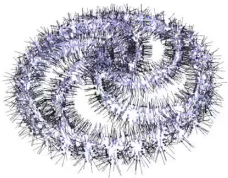
1.3.1 3D Reconstruction with Scalable Neural Representations

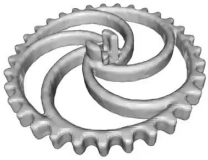
Neural implicit representations have emerged as a popular choice for learning-based 3D reconstruction since they can capture 3D shapes in a continuous, resolution-independent, and topologically flexible manner. However, most implicit-based approaches struggle with complex geometries and larger scenes. This limitation often stems from their simple fully-connected network architecture which does not allow for integrating local information in the observations or incorporating inductive biases such

as translational equivariance. To address this, we propose *Convolutional Occupancy Networks*, a novel flexible implicit representation for detailed reconstruction of objects and 3D scenes. Our model incorporates inductive biases by combining convolutional encoders with implicit occupancy decoders, enabling structured reasoning in 3D space. Our evaluations show that our method enables fine-grained implicit 3D reconstruction of single objects, scales to large indoor scenes, and generalizes well from synthetic to real data.



1.3.2 3D Reconstruction with a Differentiable Poisson Solver

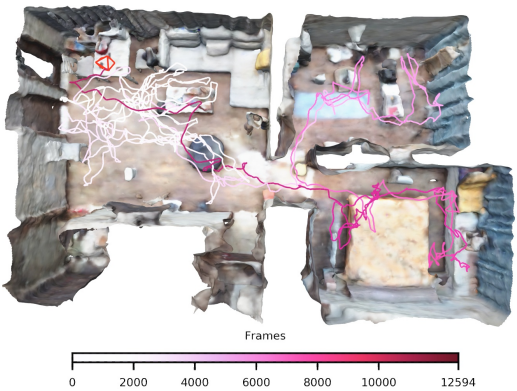




While our scalable neural implicit representations show promising results in detailed reconstruction, the inference process remains time-consuming due to the numerous network evaluations for extracting surfaces. To address this problem, we revisit the classic yet ubiquitous point cloud representation and introduce a differentiable point-to-mesh layer using a differentiable formulation of Poisson Surface Reconstruction (PSR), which allows for a GPU-accelerated fast solution of the indicator function given an oriented point cloud. The differentiable PSR layer bridges the explicit 3D point representation with the 3D mesh via the implicit indicator field, enabling end-to-end optimization. This duality between

points and meshes hence allows us to represent shapes as oriented point clouds, which are explicit, lightweight, and expressive. Our *Shape-As-Points* (SAP) model is interpretable, lightweight, and accelerates inference time by one order of magnitude compared to neural implicit representations, but could still produce topology-agnostic, high-fidelity watertight surfaces.

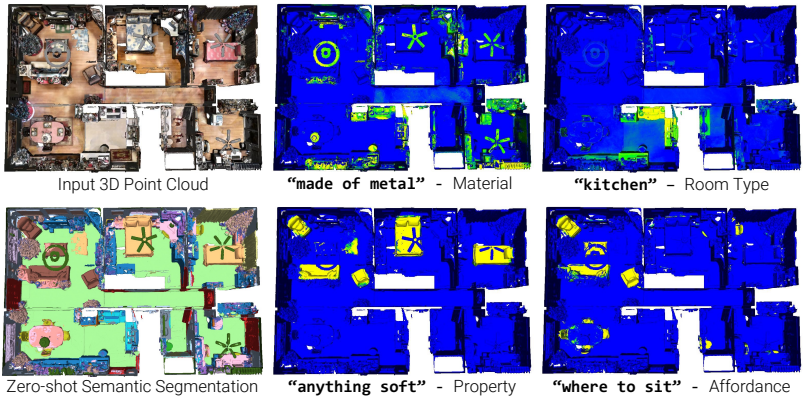
1.3.3 SLAM with Scalable Neural Representations



While our earlier contributions focused on 3D reconstruction from point clouds, a more practical setting for 3D reconstruction is to reconstruct 3D dense scene geometry given only some unposed RGB(-D) sequences with a hand-held camera, i.e. dense visual SLAM. To this end, we present *NICE-SLAM*, a dense SLAM system that employs a hierarchical neural implicit representation. Optimizing this representation with pre-trained geometric priors enables detailed reconstruction on large indoor scenes, outperforming recent neural implicit SLAM systems in scalability, efficiency, and robustness.

1.3.4 3D Scene Understanding with Large Vision Language Models

Once we obtain the realistic 3D reconstruction of a scene, the aim for the last part of this thesis is high-level perception tasks, such as



3D scene understanding. Traditional 3D scene understanding approaches have largely depended on supervision from benchmark datasets tailored for specific tasks, such as 3D semantic segmentation, often confined to a closed set of classes. Such specialized models, while adept in their designated task, are impractical for many real-world applications as the models lack the flexibility to continuously adapt to new concepts/classes in the scene.

Addressing this challenge, recent advancements, including our work *OpenScene* discussed in Chapter 6 emphasize open-vocabulary 3D scene understanding. This approach allows segmentation and understanding of arbitrary concepts, independent of any fixed closed set of classes. Specifically, given an arbitrary query like a text description or an image of an object, the goal is to segment those parts in the 3D scene that are described by the query. For example, within a reconstructed house as shown above, we are interested in understanding which surfaces are part of "`a bed`" (semantics), "`made of metal`" (materials), "`kitchen`" (room types), "`where to sit`", and which surfaces are "`soft`" (physical property). Such capabilities not only offer a richer understanding but are also pivotal for applications such as facilitating robot navigation in unfamiliar settings or enhanc-

ing AR/VR experiences in complicated indoor scenarios, especially when specific annotated labels are sparse.

1.4 Outline

This thesis is divided into 7 chapters.

Chapter 2 provides a background review of research related to this thesis.

Chapter 3 introduces our exploration into scalable neural implicit representations and their application in detailed 3D reconstruction. This chapter is based on our publication at ECCV 2020 [171].

Chapter 4 presents a differentiable Poisson solver that enables representing shapes as lightweight point clouds and speeds up the reconstruction process. This chapter is based on our paper presented at NeurIPS 2021 [170].

Chapter 5 explores a hierarchical neural scene representation for dense RGB-D SLAM in large scenes. This chapter is based on our showcased at CVPR 2022 [281].

Chapter 6 presents a zero-shot method for a range of novel 3D scene understanding tasks with open vocabularies. The content of this chapter is rooted in our publication at CVPR 2023 [169].

Chapter 7 concludes the thesis by summarizing its contributions. This chapter also reflects on the nature and potential role of learning neural scene representations, and provides a discussion of promising future directions.

1.5 Publications

This thesis includes the following 4 publications [169–171, 281].

Convolutional Occupancy Networks
Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, Andreas Geiger
European Conference on Computer Vision (ECCV) 2020 (**Spotlight**)

Shape As Points: A Differentiable Poisson Solver
Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, Andreas Geiger
Advances in Neural Information Processing Systems (NeurIPS), 2021 (**Oral**)

NICE-SLAM: Neural Implicit Scalable Encoding for SLAM
Zihan Zhu*, **Songyou Peng***, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, Marc Pollefeys
Conference on Computer Vision and Pattern Recognition (CVPR), 2022

OpenScene: 3D Scene Understanding with Open Vocabularies
Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser
Conference on Computer Vision and Pattern Recognition (CVPR), 2023

---

13 papers that I contributed to during PhD but are not included in this thesis, 8 published papers [16, 125, 127, 130, 165, 177, 260, 280] and 5 papers in submission:

NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM
Zihan Zhu*, **Songyou Peng***, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, Marc Pollefeys
International Conference on 3D Vision (3DV), 2024 (**Oral**)

FastHuman: Reconstructing High-Quality Clothed Human in Minutes
Lixiang Lin, **Songyou Peng**, Qijun Gan, Jianke Zhu
International Conference on 3D Vision (3DV), 2024

DiffDreamer: Towards Consistent Unsupervised Single-view Scene Extrapolation with Conditional Diffusion Models
Shengqu Cai, Eric R. Chan, **Songyou Peng**, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, Gordon Wetzstein
International Conference on Computer Vision (ICCV), 2023

MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction
Zehao Yu, **Songyou Peng**, Michael Niemeyer, Torsten Sattler, Andreas Geiger
Advances in Neural Information Processing Systems (NeurIPS), 2022

UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction
Michael Oechsle, **Songyou Peng**, Andreas Geiger
International Conference on Computer Vision (ICCV), 2021 (**Oral**)

KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs
Christian Reiser, **Songyou Peng**, Yiyi Liao, Andreas Geiger
International Conference on Computer Vision (ICCV), 2021

Dynamic Plane Convolutional Occupancy Networks
Stefan Lionar*, Daniil Emtsev*, Dusan Svilarkovic*, **Songyou Peng**
Winter Conference on Applications of Computer Vision (WACV), 2021

DIST: Rendering Deep Implicit Signed Distance Function with Differentiable Sphere Tracing
Shaohui Liu, Yinda Zhang, **Songyou Peng**, Boxin Shi, Marc Pollefeys, Zhaopeng Cui
Conference on Computer Vision and Pattern Recognition (CVPR), 2020

NeRF On-the-go: Exploiting Uncertainty for Distractor-free NeRFs in the Wild
Weining Ren*, Zihan Zhu*, Boyang Sun, Jiaqi Chen, Marc Pollefeys, **Songyou Peng**
In Submission

3D Neural Edge Reconstruction
Lei Li, **Songyou Peng**, Zehao Yu, Shaohui Liu, Rémi Pautrat, Xiaochuan Yin, Marc Pollefeys
In Submission

RENOVATE: Renaming Classes for Open-Vocabulary Segmentation
Haiwen Huang, **Songyou Peng**, Dan Zhang, Andreas Geiger
In Submission

Segment3D: Learning Fine-Grained Class-Agnostic 3D Segmentation without Manual Labels
Rui Huang, **Songyou Peng**, Ayça Takmaz, Federico Tombari, Marc Pollefeys, Shiji Song, Gao Huang, Francis Engelmann
In Submission

Ternary-type Opacity and Hybrid Odometry for RGB-only NeRF-SLAM
Junru Lin, Asen Nachkov, **Songyou Peng**, Luc Van Gool, Danda Pani Paudel
In Submission

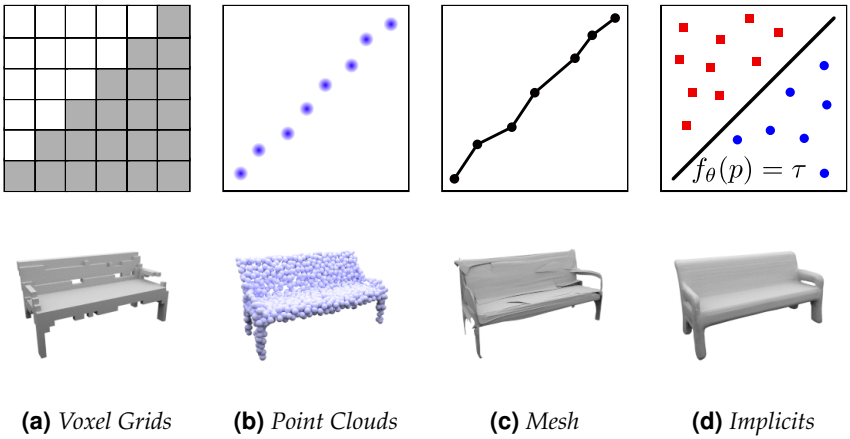
CHAPTER 2

Background

In this chapter, we provide a background overview of the research fields related to this thesis. We first provide an overview of the development in 3D shape representations in Sec. 2.1. We then discuss in Sec. 2.2 and Sec. 2.3 the development of 3D reconstruction from point clouds and multi-view images. Finally, we provide a comprehensive review of the topic of 3D scene understanding, or more specifically, 3D semantic segmentation in Sec. 2.4.

2.1 3D Shape Representations

In this section, we explore the predominant 3D shape representations employed in learning-based 3D reconstruction. Specifically, the focus lies on four main representations: voxel grids, polygon meshes, point clouds, and neural implicit representations. While



(a) *Voxel Grids* **(b)** *Point Clouds* **(c)** *Mesh* **(d)** *Implicits*

Figure 2.1: ***Comparison on 3D Shape Representation.*** *Image taken from [144].*

many other representations exist [27], this discussion will concentrate exclusively on the aforementioned four (see Fig. 2.1).

2.1.1 Voxel Grids

Voxel grids are an extension of the pixel-based image representation into the 3D domain. Just as an image uses a grid structure to store RGB color values, voxel representations use a three-dimensional grid to store binary values, like occupancy, or scalar values, such as signed distance functions (SDFs), to implicitly define 3D structures. Neural networks can easily predict a grid of voxels with implicit field values, which can subsequently be processed using meshing or iso-surfacing algorithms to extract the mesh.

The simplest way to generate voxels is to use a 3D CNN network to predict a 3D grid. Some works [40, 240, 241, 272] use an encoder-decoder structure where a 2D CNN encoder encodes the input image into a latent code, and a 3D CNN decoder decodes the latent code into a voxel grid. This concept was later extended to reconstructing

3D geometry from multiple input images, as showcased by [86, 96, 167].

However, a limitation arises when we consider the space complexity of voxel grids, which is of $O(N3)$. Given hardware memory limitations, generating a voxel grid with a sufficiently high resolution becomes challenging. To address this, many have turned to the octree representation, a technique that adaptively subdivides voxels. This representation has been applied in many works, such as HSP (Hierarchical Surface Prediction) [70], OctNetFusion [179], OGN [209], and Dual OCNN [226]. While octrees offer a flexible structure that can allocate varying capacities across the 3D space, implicit field values are still stored in grids. As a consequence, the resolution of the represented geometry remains bounded by the chosen grid resolution and maximum depth.

2.1.2 Point Clouds

A point cloud comprises a set of individual points in 3D space, each representing a segment of an object's surface. While each point is primarily identified by its 3D coordinates, some might carry additional attributes, such as color. Their ability to represent any shape lends point clouds a significant degree of versatility. However, their inherently unstructured and unordered format posed challenges in the realm of neural networks, primarily as conventional convolutional neural networks were tailored for structured data.

Qi et al. [172] pioneered point clouds as a representation of discriminative deep learning tasks. They attain permutation invariance by individually applying a fully connected neural network to each point, subsequently paired with a global pooling operation. This pioneering method became a foundation upon which numerous enhancements emerged, such as enhancing the convolution operation [4, 117, 242, 250] or integrating hierarchical structures [173,229,274]. In the domain of 3D reconstruction from images,

Fan et al. [52] take in images and use a neural network to output 3D point clouds.

While point clouds bypass the need for careful topology design, this absence of inherent connectivity poses limitations. Specifically, without details on point relationships or connections, point clouds fall short in certain graphic applications like rendering and collision detection, where such relationships are crucial.

2.1.3 Polygon Meshes

Meshes serve as another 3D shape representation. Essentially, they represent the surface of a 3D object through a collection of interconnected geometric shapes, predominantly triangles or quadrilaterals, termed polygons. These polygons are defined by vertices that represent distinct points in 3D space, connected by edges. The collection of polygons forms the mesh that describes the 3D object's surface. Meshes extend the point cloud representation with connectivity information, facilitating geodesic path information traversal.

In the expanding horizon of learning-based 3D reconstruction techniques, meshes have been considered as the output representation. Such methods essentially directly regress the vertices and faces of a mesh. One pioneer work was Deep Marching Cubes (DMC) [121]. DMC has an encoder-decoder structure, where the encoder converts inputs into latent codes, while the decoder, a 3D CNN, generates grids of inside-outside signs and vertex positions. An explicit triangle mesh can be extracted by applying the Marching Cubes algorithm, and the positions of the mesh vertices are given by the predicted grid of vertex positions. While this initiative sparked a series of work with similar ideas [28, 30, 62, 65, 95, 124, 224, 232], most of these approaches often yield self-intersecting meshes. Moreover, they are either only able to generate meshes with simple topology [28, 30, 62, 65, 224], demand reference templates from similar object class [95, 108], or cannot guarantee closed surfaces [62, 65].

Undoubtedly, mesh-based representations find resonance in diverse applications. Yet, their inherent need for 3D space discretization, combined with the challenges they pose for neural network predictions, can restrict topology and harm generalizability.

2.1.4 Neural Implicit Representations

Neural implicit representations, now widely known as *neural fields* [245], have been rapidly making a mark in areas such as 3D reconstruction, novel view synthesis, and generative modeling. This burgeoning interest can be traced back to three pivotal works unveiled at CVPR 2019: Occupancy Networks [144], DeepSDF [166], and IM-Net [29].

At its core, a neural implicit representation is an intricate multi-layer perceptron (MLP) designed to take a point's coordinate as input and output its corresponding inside-outside sign (occupancy) [29,144] or signed distance [166]. The MLP itself hence implicitly represents a 3D shape. To generate different output shapes based on input, the MLP is tailored using various conditioning techniques, such as concatenating a latent shape code with the input point coordinates before feeding to the MLP, or modulating MLP weights via a hypernetwork [195]. Soon after, to capture detailed shape geometries and generalize better, methods have been proposed to replace global latent codes with local ones for conditioning the MLP. PIFu [185], PIFuHD [186], DISN [248] employ pixel-aligned local features from 2D image encoders for 3D reconstruction from images. On the other hand, our works ConvONet [171] and DPConvONet [127] among others like LIG [90] and IF-Nets [36] use local features extracted from point cloud or voxels to perform object or large-scale scene reconstruction. With the recent strides in differentiable renderers for neural implicit representations, these representations can be effectively learned with only 2D observations. Noteworthy methods like DIST [130], DVR [161] and IDR [256] can reconstruct objects

from images but they assume given object masks. UNISURF [165], NeuS [225], and VolSDF [255] further relax the need of object mask.

Neural implicit representations differ from other aforementioned representations due to their continuous nature, thus they can potentially represent geometries at infinite resolution, and naturally handle complicated shape topologies. Another highlight is its compact memory footprint compared to voxel-based solutions, since the memory requirements for MLP-based representations scale only according to the number of network parameters and are in general much lower. Yet, a significant downside remains in its inherently slow inference time. To output the entire shape, a grid of points needs to be sampled in space and the MLP needs to evaluate every point to produce a grid of implicit field values. To speed up the process, in Chapter 4 we propose a method that efficiently solves the Poisson Equation during inference.

Given the rapid evolution of this field, we touch only on some representative works. For a more comprehensive overview, we recommend the survey paper [245].

2.2 3D Reconstruction from Point Clouds

Reconstructing 3D shapes from point clouds is a pivotal task in computer vision and graphics. This section reviews methods that undertake this task, considering point clouds both with and without point normals Note that, state-of-the-art reconstruction accuracy is predominantly achieved using *implicit representations*. Hence, our discussion will be centered on works along this line.

Broadly speaking, the methods can be categorized into optimization-based and learning-based approaches. While optimization-based techniques excel in capturing geometric details, they often suffer from extended optimization times and sensitivity to input point cloud noise. Conversely, learning-based methods, trained on large datasets, showcase resilience to varied input noise and boast faster

reconstruction speeds. However, they occasionally compromise on the granularity of the final reconstruction.

2.2.1 Optimization-Based Approaches

PSR-Based Approaches. The groundbreaking Poisson Surface Reconstruction (PSR) [98] formulates surface reconstruction as solving a Poisson problem. The Poisson problem admits a hierarchy of locally supported functions, and therefore its solution reduces to a well-conditioned sparse linear system. The solution to the problem is global since they consider all points simutaneously, eliminating the need for heuristic partitioning or blending. Its resilience to data noise and enabling high-detail reconstruction make it a favored choice for surface reconstruction from point clouds. Enhancing the original PSR, Screened Poisson Surface Reconstruction (SPSR) [97] explicitly incorporate the points as interpolation constraints, and introduces algorithmic improvements that reduce the time complexity of the solver to linear in the number of points. Sellán and Jacobson [191] also introduce a statistical extension of PSR. Instead of directly outputting an implicit function, they represent the reconstructed shape through a modified Gaussian Process, facilitating statistical queries.

However, a common limitation among these methods is their reliance on accurate point normals. Inaccuracies can drastically degrade performance. Addressing this, iPSR [75] iteratively refines normals using PSR. They take as input point samples with normals directly computed from the surface obtained in the preceding iteration, and then generate a new surface with better quality. Our work, Shape-As-Points (Chapter 4), introduces a differentiable PSR variant, which not only reduces the dependency on point normals, but also allows for a GPU-accelerated fast solution of Poisson equation.

Neural Implicit-Based Approaches. Diverging from traditional methods, neural implicit-based techniques optimize an MLP using

point clouds, with or without normals. SAL [2] proposes an unsigned similarity function and a geometric network initialization to learn a neural unsigned distance field from dense point clouds. SALD [3] enhances SAL by incorporating derivatives in the regression loss. This leads to a lower sample complexity, and consequently better fitting. IGR [64] presents different optimization objectives that encourage the neural network to vanish on the input point cloud and to have a unit norm gradient. SIREN [195] improves the representation capability of MLPs by using periodic activation in MLPs, so it can quickly overfit a neural implicit from point clouds. Neural Splines [237] employs a kernel method to obtain implicit field from a set of points and their normals, based on kernels arising from infinitely wide shallow ReLU networks. Note that, although many of these methods [2, 3, 64, 195] support reconstruction from unoriented point clouds, we notice that optimal performance is typically achieved when point normals are available.

2.2.2 Learning-Based Approaches

While optimization-based techniques excel in capturing geometric nuances, their prolonged optimization process and sensitivity to noise can be limiting. In contrast, learning-based methods, trained on extensive datasets, offer robustness against varied noise levels and efficient reconstructions through straightforward feedforward passes.

Note that a significant amount of learning-based approaches use a global shape latent code to encode the shape from point clouds [59, 65,89,144,166]. However, as highlighted in Sec. 2.1, such global codes often fall short of representing detailed shape geometries. Thus, our focus will be on works that consider local features.

ConvONet [171] (Chapter 3) and IF-Nets [36] previously discussed in Sec. 2.1, stand out for their ability to reconstruct shapes from point clouds without normals. Our ConvONet [171] and its most direct follow-up works [127, 208] encode the input point clouds to either

2D feature planes or 3D grids, which are later processed by CNNs for further feature aggregation. They condition the occupancy prediction network with the local features extracted from the feature planes/grids. They show for the first time large-scale scene reconstruction with neural implicit representations. IF-Nets [36], while conceptually similar, adopts CNNs to process multi-scale 3D feature grids. GIFS [258] does not predict the inside/outside status of each query point, but rather predicts whether two points are separated by any surface. This modification enables them to represent non-watertight shapes, but also requires a modified Marching Cubes algorithm.

Points2Surf [51] is purely based on point cloud encoders without any CNN. For a query point in space, it adopts a PointNet [172] to encode points sampled at the neighborhood of the query point into a local feature code, and another PointNet to encode the points sampled at the entire input point cloud into a global feature code. These features are then used by the decoder to predict the signed distance of a query point. POCO [11], on the other hand, leverages point cloud convolutions to compute latent vectors at each input point. For query points, it performs a learning-based interpolation on nearest neighbors in input points to retrieve a weighted-averaged feature vector, and the feature vector is processed by an MLP to predict the occupancy.

Hybrid Approaches. There are also methods that attempt to combine the learning-based pipeline with online optimization. LIG [90] and DeepLS [20] leverage the shape priors learned from local patches. They first train an autoencoder to learn the latent code of local crops of 3D shapes. During inference, the MLP decoder is fixed, and they divide the space into a grid of overlapping cubes and optimize to obtain a latent code for each patch of the input point cloud. However, their reliance on point normals and the time-intensive reconstruction process can be limiting. SAIL-S3 [273] while conceptually similar, achieves reconstruction without the need for normals, by adopting Sign Agnostic Learning [2].

NKF [235], an extension of Neural Spline [237] mentioned before,

instead of using the fixed point properties (normals), introduces a kernel ridge regression that fits the input points on-the-fly by solving a simple positive definite linear system using the learned kernel. NKSR [81] builds upon NKF and develops a new gradient-based kernel formulation, ensuring robustness to noise. Moreover, it uses an explicit voxel hierarchy structure and compactly supported kernels to be capable of handling large inputs while still producing high-fidelity outputs. However, NKF and NKSR still require point normals as input. Deep IMLS [132] processes sparse, unoriented point clouds as its input. Utilizing a U-Net-inspired autoencoder, it predicts an octree structure. Within this structure, each octree node encompasses a set number of predicted points, each accompanied by normals. These predicted points, equipped with their normals, serve as the foundation for constructing an implicit field using the implicit moving least-squares (IMLS) surface formulation [107].

Worth mentioning that for our proposed SAP [170] (Chapter 4), besides the optimzation-based setting discussed in Sec. 2.2.1, we also incorporate a differentiable Poisson solver within the learning-based framework. Given a noisy, unoriented point cloud, our model is trained to predict a refined, oriented point cloud. This enhanced point cloud then facilitates the derivation of a watertight mesh, achieved by solving the Poisson equation using the differentiable solver.

2.3 3D Reconstruction from Multi-view Images

Going from reconstructing shapes from point clouds, our focus now shifts to the reconstruction of objects and scenes using posed multi-view images. It is important to note that only a select few methods [56,63,151,239,265] employ explicit mesh representations for this task. Consequently, our discussion will center on works that utilize implicit representations, similar to Sec. 2.2.

In this domain, A limited number of studies have adopted a learning-

based approach, where they learn priors from a set of training shapes. These methods have various techniques for aggregating data. 3D-R2N2 [40] uses recurrent neural networks to combine global shape latent codes from multiple input images, Pix2Vox [244] aggregates spatial features decoded from global shape latent codes, Pixel2Mesh++ [232] directly aggregates image features from multiple input images, while EVolT [221] aggregates image features from multiple inputs together with 3D embeddings of spatial locations using a Transformer.

However, a majority of the methods tend to overfit or optimize a single shape or scene based on multiple input images. This optimization often leverages methods rooted in differentiable rendering algorithms on implicit representations, or alternatively, are based on the volume rendering formula as presented in NeRF [148].

2.3.1 Approaches with Surface Rendering

Methods discussed in this section utilize a differentiable surface rendering formula for implicit representations. They operate under the assumption that an object segmentation mask is provided for each input image and that each ray intersects the surface only once, allowing for a single intersection point per ray for gradient propagation.

Pioneering this approach, SDFDiff [91], DIST [130], DVR [161], and IDR [256] introduced unique formulations of differentiable rendering on implicit surfaces. Specifically, SDFDiff [91] employs a regular grid SDF, while others opt for neural implicit representations. When it comes to color modeling, SDFDiff assumes the absence of textures in the target shape and does not predict textures for the reconstructed shape. Both DIST and DVR integrate Texture fields [164], leveraging an MLP to predict the RGB color for each surface point. However, this method falls short in modeling view-dependent effects. IDR employs an MLP to approximate the bidirectional reflectance distribution function (BRDF) for each surface point.

Neural Lumigraph Rendering (NLR) [99] demonstrates that the extracted mesh, when combined with unstructured lumigraph rendering [13], can facilitate real-time rendering. MVSDF [267] capitalizes on stereo matching and feature consistency to refine the neural implicit SDF representation. RegSDF [266] harnesses reconstructed point clouds from input images to guide and regularize neural field learning. Finally, the Reparameterization SDF renderer [6] offers a technique to compute accurate gradients concerning network parameters in neural SDF renderers.

2.3.2 Approaches with Volume Rendering

Following the introduction of NeRF [148], numerous methods have embraced the NeRF-style volume rendering for multi-view reconstruction. Central to these methods is the principle that each point sampled along a ray possesses both density (termed ”opacity”) and radiance (or ”view-dependent RGB color”). Predicted by an MLP, the final pixel color is derived from the accumulated radiance of all sampled points wrt. their density, similar to alpha-compositing. This approach has enhanced both the applicability and scalability of these methods since it eliminates the need for object masks.

The pioneer works in this domain are UNISURF [165], NeuS [225], and VolSDF [255], each introducing unique formulations of volume rendering on implicit surfaces. For instance, instead of learning the density, UNISURF learn an occupancy field, and then re-formulate the volume rendering process accordingly. In contrast, both NeuS and VolSDF model density by adapting the learnable SDF field.

UNISURF [165], NeuS [225], and VolSDF [255] are pioneers that propose different formulations of volume rendering on implicit surfaces. Specifically, instead of outputting density, UNISURF learns to output an occupancy field, and re-formulate the volume rendering process. NeuS and VolSDF instead model the density by transforming the learnable SDF field.

Building upon these foundational works, subsequent research has introduced various enhancements. Azinovic et al. [5] introduces depth supervision into the optimization process. NeuralWarp [47] emphasizes photo-consistency across different views during optimization, ensuring the accuracy of the implicit geometry. ManhattanSDF [67] integrates planar constraints, refining the geometry in areas like floors and walls. Geo-Neus [55] optimizes multi-view geometry by harnessing sparse geometry from structure-from-motion and photometric consistency in multi-view stereo. SparseNeuS [135] focuses on 3D reconstruction from sparse images, introducing geometry encoding volumes for universal surface prediction. HF-NeuS [227] adopts a decomposition approach for the SDF, using a coarse-to-fine strategy to amplify high-frequency details. Sun et al. [202] extend the principles of NeuS and NeRF-W [141] to reconstruct scenes from diverse Internet photo collections, accommodating varying illumination. MonoSDF [260] leverages depth and normal maps predicted by pretrained monocular estimator networks for 2D images, to enhance reconstruction quality and reduce optimization time.

All methods outlined in Sec. 2.3.1 and Sec. 2.3.2 operate under the assumption that camera poses are provided. This assumption is crucial as accurate camera poses can significantly influence the quality of the reconstruction. However, in real-world scenarios, this requirement might not always be met due to inaccuracies in camera estimation. Furthermore, while these methods have demonstrated impressive reconstruction results, they come with the caveat of extensive offline optimization time, often spanning hours. Such a long training phase can hinder their real-time applicability in practical scenarios. As a result, there's a growing emphasis on advancing this domain towards online 3D reconstruction, where both reconstruction and tracking occur simultaneously.

SLAM, especially those based on neural-implicit approaches, offer promising solutions in this direction. We will delve deeper into the intricacies and advancements of neural-implicit based online SLAM methods in Chapter 5.

2.4 3D Scene Understanding

With a reconstructed 3D scene in hand, gaining a comprehensive understanding of its varied properties becomes crucial. Traditional methods for 3D scene understanding have predominantly relied on benchmark datasets, often designed for specific tasks like 3D semantic segmentation for a close-set of 20 classes. While these models excel in their specific domains, their narrow scope restricts their utility in dynamic real-world scenarios where adaptability to evolving scene elements is imperative.

Addressing this limitation, the forefront of research is now pivoting towards open-vocabulary 3D scene understanding. This progressive approach facilitates segmentation and comprehension of a myriad of concepts, decoupled from any fixed class set. For instance, within a reconstructed house, this method could discern aspects ranging from chair-associated surfaces to metal materials, encompassing diverse queries from room types to tactile properties. This can enable a wide range of new applications, especially when there are no specific annotated labels.

In this section, we will begin by shedding light on the 2D vision-language foundation models in Sec. 2.4.1, and subsequently delve into the latest advances in open-vocabulary 3D scene understanding in Sec. 2.4.2.

2.4.1 Vision-Language Foundation Models

The emergence of open-vocabulary scene understanding in both 2D and 3D domains can be attributed to the advancements in vision-language foundation models. Notable models in this category include CLIP [175], OpenCLIP [35], ALIGN [87], and Flamingo [1]. CLIP (Contrastive Language-Image Pre-Training), for instance, is representative of this shift. Trained on vast col-

lections of Internet-derived image-caption pairs, CLIP employs an image encoder and a text encoder, as shown in Fig. 2.2. These encoders collaboratively map inputs into a shared embedding space. Through a contrastive training approach, corresponding images and captions are aligned in the embedding space, while non-related pairs are strategically separated. These models leverage large-scale pretraining, granting them the capability for zero-shot transfer across diverse downstream tasks, from object recognition and classification.

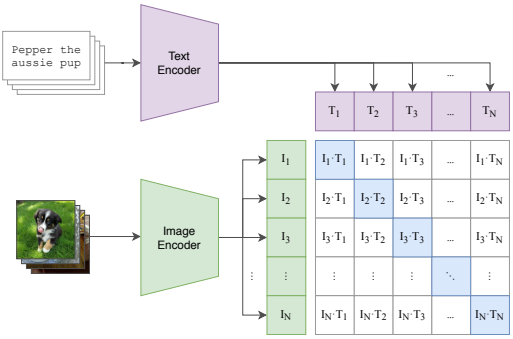


Figure 2.2: ***Overview of CLIP.*** *The diagram is taken from the original paper [175].*

Building on the advancements of large-scale model pre-training, there has been a surge of interest in foundational models focused on images [61, 66, 110, 113, 119, 176, 246, 247, 276]. Diverging from the traditional route of obtaining per-image global features as shown in Fig. 2.3 (a), these works emphasize extracting versatile class-agnostic features from images, illustrated in Fig. 2.3 (b) and (c). This shift equips users with the ability to define arbitrary text labels for tasks more than just classification, but also dense prediction tasks like detection, or segmentation during test phases.

Delving deeper, LSeg [113] introduces a method where a visual encoder generates per-pixel embeddings from an input image. These embeddings then align with the text representations of their specific pixel class labels using 2D semantic segmentation datasets, as seen in Fig. 2.3 (b). However, a significant hurdle arises: the pixel-wise class annotations are resource-intensive. To alleviate this, alternative approaches like OpenSeg [61], OVSeg [119], and ODISE [247] have emerged. These methods first derive a collection of class-agnostic

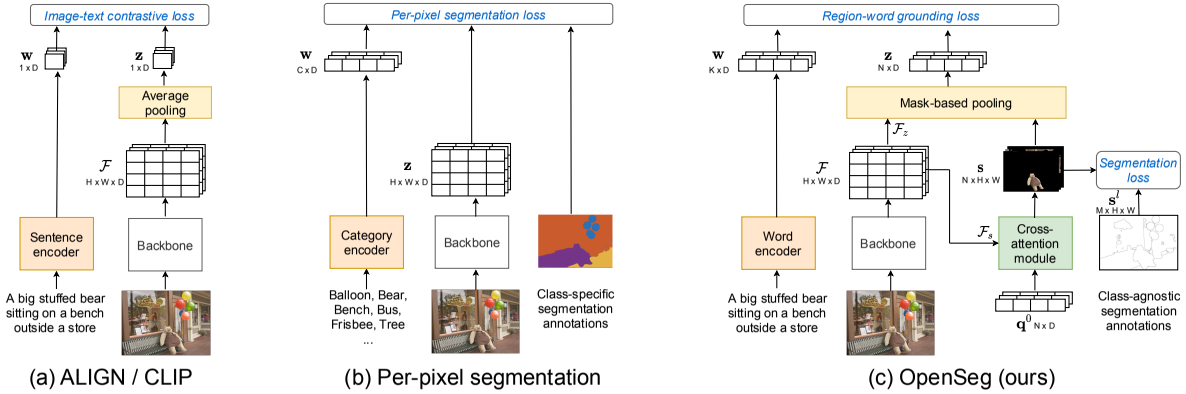


Figure 2.3: ***Comparison of Different 2D Vision-Language Foundation Models.*** *Image taken from OpenSeg [61].*

segmentation masks and their associated features. They then align each word in an image caption to one or multiple anticipated masks using region-word grounding losses, as illustrated in Fig. 2.3 (c). This strategy eliminates the need for costly pixel-wise class labels, leveraging only weak labels. Such an approach is pivotal for scaling up training data and expanding vocabulary sizes. Furthermore, these methods harness the inherent generalization prowess of CLIP, adeptly handling new classes that were absent during training.

2.4.2 Open-Vocabulary 3D Scene Understanding

The recent success of 2D open-vocabulary segmentation models [61, 113,119,247] highlighted in Sec. 2.4.1, has motivated the 3D scene understanding community to consider the setting of open vocabulary.

2D Lifting. A line of research investigates how to lift such 2D open-vocabulary information to 3D. In Semantic Abstraction [68], 2D-based CLIP features are unprojected to 3D space via relevancy maps, which are extracted from an input RGB-D stream. Though results look promising, they only address minor partial scenes and depend on ground truth data for supervision. In ConceptFusion [85] multi-modal 3D semantic representations are inferred from an input RGB-D stream using 2D foundation models. They use point clouds as the 3D representation. Similarly, approaches such as DFF [105], LERF [100], VL-Fields [215], and 3D-OVS [129] harness the NeRF [148] interpolation potential through volume-rendering of CLIP embeddings. By Supervising these embeddings across training views multi-view consistency is obtained. Once trained, they can produce comprehensive 3D relevancy maps from a broad range of language prompts interactively. However, all these methods rely on 2D images as a surrogate for understanding 3D scenes.

Text-to-3D Learning. Another line of work attempts to learn to bridge text with 3D. In essence, post-training, these models enable

querying of novel 3D spaces without the need for 2D imagery. ScanNet200 [182] pioneered this trajectory by utilizing the CLIP's versatility to segment directly from 3D point clouds. However, they only pre-train 3D networks to text encoded anchors of the corresponding semantic labels without a contrastive loss, but still fine-tune with 3D GT annotations afterwards. Their focus is on using the CLIP embedding to achieve better supervised 3D semantic segmentation, rather than open-vocabulary queries. PLA [49] and its follow-up work RegionPLC [253] build 3D-caption pairs and employ contrastive learning to train the 3D model. However, they focus on discovering unseen classes in indoor scenes. Similarly, CLIP^2 [264] collects a large dataset for text-image-3D labels, and performs cross-modal contrastive pertaining, aiming to learn a 3D CLIP model fit for recognition tasks. 3D-CLR [73] starts by aligning a per-scene neural field of LSeg features with corresponding 3D point clouds. Next, they train a relation network with a dataset captured in Habitat simulator [140] for 3D multi-view visual Q&A. Lastly, OpenScene (see Chapter 6) unprotects per-pixel 2D open-vocabulary features to 3D point clouds and uses them as a supervision signal to train a 3D CNN, thus facilitating numerous 3D scene understanding tasks straight from 3D point clouds.

CHAPTER 3

3D Reconstruction with Scalable Neural Representations

In this chapter, we delve into the challenges and potentials of neural scene representations for large-scale 3D reconstruction in learning-based systems. Although recent advances in neural implicit representations have showcased promising outcomes in 3D reconstruction, their primary focus has been on the simpler geometry of simple objects. This limitation primarily stems from the inherent architecture of these representations—relying predominantly on simple fully-connected networks—which consequently restricts the integration of local information into observations and the incorporation of inductive biases, such as translational equivariance. To address these constraints, this chapter presents an approach that combines convolutional encoders with implicit occupancy decoders. This not only facilitates structured reasoning in 3D space, but also promotes fine-

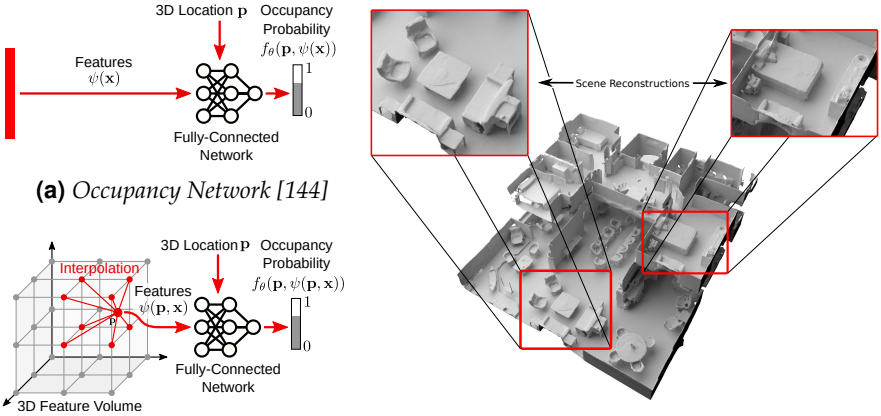
grained implicit 3D reconstruction from single objects to expansive indoor scenes. Moreover, it also showcases robust adaptability from synthetic to real data.

3.1 Introduction

3D reconstruction is a fundamental problem in computer vision with numerous applications. An ideal representation of 3D geometry should have the following properties: a) encode complex geometries and arbitrary topologies, b) scale to large scenes, c) encapsulate local and global information, and d) be tractable in terms of memory and computation.

Unfortunately, current representations for 3D reconstruction do not satisfy all of these requirements. Volumetric representations [142] are limited in terms of resolution due to their large memory requirements. Point clouds [52] are lightweight but discard topological relations. Mesh-based representations [65] are often hard to predict using neural networks.

Recently, several works [29, 144, 146, 166] have introduced deep implicit representations which represent 3D structures using learned occupancy or signed distance functions. In contrast to explicit representations, implicit methods do not discretize 3D space during training, thus resulting in continuous representations of 3D geometry without topology restrictions. While inspiring many follow-up works [58, 59, 131, 133, 160, 162, 164, 197], all existing approaches are limited to single objects and do not scale to larger scenes. The key limiting factor of most implicit models is their simple fully-connected network architecture [144, 166] which neither allows for integrating local information in the observations, nor for incorporating inductive biases such as translation equivariance into the model. This prevents these methods from performing *structured reasoning* as they only act globally and result in overly smooth surface reconstructions.



(a) *Occupancy Network [144]*

(b) *Conv. Occupancy Network*

(c) *Reconstruction on Matterport3D [23]*

Figure 3.1: ***Convolutional Occupancy Networks.*** *Traditional implicit models (a) are limited in their expressiveness due to their fully-connected network structure. We propose Convolutional Occupancy Networks (b) which exploit convolutions, resulting in scalable and equivariant implicit representations. We query the convolutional features at 3D locations* $\mathbf{p} \in \mathbb{R}^3$ *using linear interpolation. In contrast to Occupancy Networks (ONet) [144], the proposed feature representation* $\psi(\mathbf{p}, \mathbf{x})$ *therefore depends on both the input* $\mathbf{x}$ *and the 3D location* $\mathbf{p}$*. Fig. (c) shows a reconstruction of a two-floor building from a noisy point cloud on the Matterport3D dataset [23].*

In contrast, translation equivariant convolutional neural networks (CNNs) have demonstrated great success across many 2D recognition tasks including object detection and image segmentation. Moreover, CNNs naturally encode information in a hierarchical manner in different network layers [262, 269]. Exploiting these inductive biases is expected to not only benefit 2D but also 3D tasks, e.g., reconstructing 3D shapes of multiple similar chairs located in the same room. In this work, we seek to combine the complementary strengths of convolutional neural networks with those of implicit representations.

Towards this goal, we introduce *Convolutional Occupancy Networks*, a novel representation for accurate large-scale 3D reconstruction with continuous implicit representations (Fig. 3.1). We demonstrate that

this representation not only preserves fine geometric details, but also enables the reconstruction of complex indoor scenes at scale. Our key idea is to establish rich input features, incorporating inductive biases and integrating local as well as global information. More specifically, we exploit convolutional operations to obtain translation equivariance and exploit the local self-similarity of 3D structures. We systematically investigate multiple design choices, ranging from canonical planes to volumetric representations. Our contributions are summarized as follows:

- We identify major limitations of current implicit 3D reconstruction methods.
- We propose a flexible translation equivariant architecture which enables accurate 3D reconstruction from object to scene level.
- We demonstrate that our model enables generalization from synthetic to real scenes as well as to novel object categories and scenes.

3.2 Method

Our goal is to make implicit 3D representations more expressive. An overview of our model is provided in Fig. 3.2. We first encode the input $\mathbf{x}$ (e.g., a point cloud) into a 2D or 3D feature grid (left). These features are processed using convolutional networks and decoded into occupancy probabilities via a fully-connected network. We investigate planar representations (a+c+d), volumetric representations (b+e) as well as combinations thereof in our experiments. In the following, we explain the encoder (Sec. 3.2.1), the decoder (Sec. 3.2.2), the occupancy prediction (Sec. 3.2.3) and the training procedure (Sec. 3.2.4) in more detail.

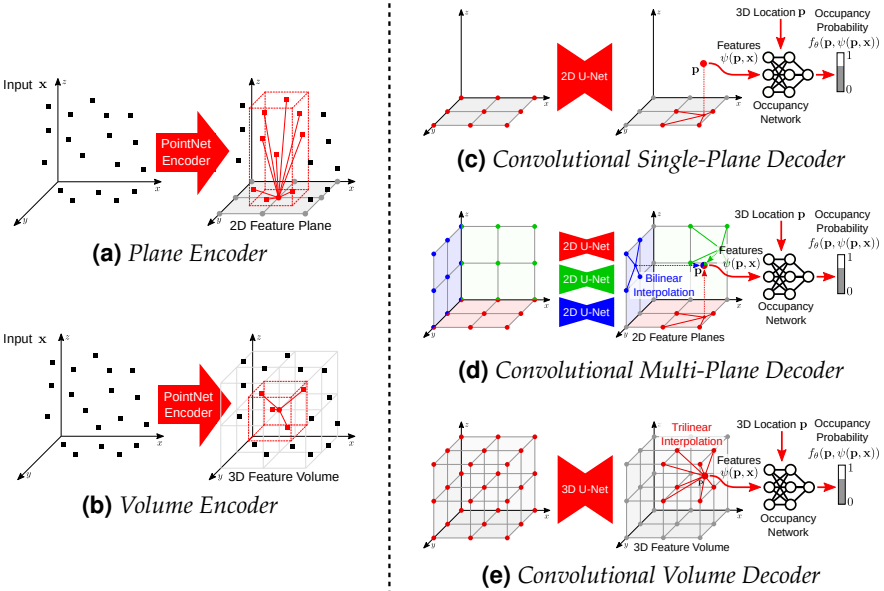


Figure 3.2: ***Model Overview.*** *The* ***encoder (left)*** *first converts the 3D input* $\mathbf{x}$ *(e.g., noisy point clouds or coarse voxel grids) into features using task-specific neural networks. Next, the features are projected onto one or multiple planes (Fig. 3.2a) or into a volume (Fig. 3.2b) using average pooling. The* ***convolutional decoder (right)*** *processes the resulting feature planes/volume using 2D/3D U-Nets to aggregate local and global information. For a query point* $\mathbf{p} \in \mathbb{R}^3$*, the point-wise feature vector* $\psi(\mathbf{x}, \mathbf{p})$ *is obtained via bilinear (Fig. 3.2c and Fig. 3.2d) or trilinear (Fig. 3.2e) interpolation. Given feature vector* $\psi(\mathbf{x}, \mathbf{p})$ *at location* $\mathbf{p}$*, the occupancy probability is predicted using a fully-connected network* $f_\theta(\mathbf{p}, \psi(\mathbf{p}, \mathbf{x}))$*.*

3.2.1 Encoder

While our method is independent of the input representation, we focus on 3D inputs to demonstrate the ability of our model in recov-

ering fine details and scaling to large scenes. More specifically, we assume a noisy sparse point cloud (e.g., from structure-from-motion or laser scans) or a coarse occupancy grid as input $\mathbf{x}$.

We first process the input $\mathbf{x}$ with a task-specific neural network to obtain a feature encoding for every point or voxel. We use a one-layer 3D CNN for voxelized inputs, and a shallow PointNet [172] with local pooling for 3D point clouds. Given these features, we construct planar and volumetric feature representations in order to encapsulate local neighborhood information as follows.

Plane Encoder. As illustrated in Fig. 3.2a, for each input point, we perform an orthographic projection onto a canonical plane (i.e., a plane aligned with the axes of the coordinate frame) which we discretize at a resolution of $H \times W$ pixel cells. For voxel inputs, we treat the voxel center as a point and project it to the plane. We aggregate features projecting onto the same pixel using average pooling, resulting in planar features with dimensionality $H \times W \times d$, where d is the feature dimension.

In our experiments, we analyze two variants of our model: one variant where features are projected onto the ground plane, and one variant where features are projected to all three canonical planes. While the former is computationally more efficient, the latter allows for recovering richer geometric structure in the z dimension.

Volume Encoder. While planar feature representations allow for encoding at large spatial resolution (128^2 pixels and beyond), they are restricted to two dimensions. Therefore, we also consider volumetric encodings (see Fig. 3.2b) which better represent 3D information, but are restricted to smaller resolutions (typically 32^3 voxels in our experiments). Similar to the plane encoder, we perform average pooling, but this time over all features falling into the same *voxel* cell, resulting in a feature volume of dimensionality $H \times W \times D \times d$.

3.2.2 Decoder

We endow our model with translation equivariance by processing the feature planes and the feature volume from the encoder using 2D and 3D convolutional hourglass (U-Net) networks [41,181] which are composed of a series of down- and upsampling convolutions with skip connections to integrate both local and global information. We choose the depth of the U-Net such that the receptive field becomes equal to the size of the respective feature plane or volume.

Our single-plane decoder (Fig. 3.2c) processes the ground plane features with a 2D U-Net. The multi-plane decoder (Fig. 3.2d) processes each feature plane separately using 2D U-Nets with shared weights. Our volume decoder (Fig. 3.2e) uses a 3D U-Net. Since convolution operations are translational equivariant, our output features are also translation equivariant, enabling structured reasoning. Moreover, convolutional operations are able to "inpaint" features while preserving global information, enabling reconstruction from sparse inputs.

3.2.3 Occupancy Prediction

Given the aggregated feature maps, our goal is to estimate the occupancy probability of any point $\mathbf{p}$ in 3D space. For the single-plane decoder, we project each point $\mathbf{p}$ orthographically onto the ground plane and query the feature value through bilinear interpolation (Fig. 3.2c). For the multi-plane decoder (Fig. 3.2d), we aggregate information from the 3 canonical planes by summing the features of all 3 planes. For the volume decoder, we use trilinear interpolation (Fig. 3.2e).

Denoting the feature vector for input $\mathbf{x}$ at point $\mathbf{p}$ as $\psi(\mathbf{p}, \mathbf{x})$, we predict the occupancy of $\mathbf{p}$ using a small fully-connected network:

$$f_\theta(\mathbf{p}, \psi(\mathbf{p}, \mathbf{x})) \rightarrow [0, 1] \tag{3.1}$$

The network comprises multiple ResNet blocks. We use the network architecture of [162], adding ψ to the input features of every ResNet block instead of the more memory intensive batch normalization operation proposed in earlier works [144]. In contrast to [162], we use a feature dimension of 32 for the hidden layers. Details about the network architecture can be found in Sec. 3.2.5.

3.2.4 Training and Inference

At training time, we uniformly sample query points $\mathbf{p} \in \mathbb{R}^3$ within the volume of interest and predict their occupancy values. We apply the binary cross-entropy loss between the predicted $\hat{o}_\mathbf{p}$ and the true occupancy values $o_\mathbf{p}$:

$$\mathcal{L}(\hat{o}_\mathbf{p}, o_\mathbf{p}) = -[o_\mathbf{p} \cdot \log(\hat{o}_\mathbf{p}) + (1 - o_\mathbf{p}) \cdot \log(1 - \hat{o}_\mathbf{p})] \tag{3.2}$$

We implement all models in PyTorch [168] and use the Adam optimizer [101] with a learning rate of 10^{-4}. During inference, we apply *Multiresolution IsoSurface Extraction* (MISE) [144] to extract meshes given an input $\mathbf{x}$. As our model is fully-convolutional, we are able to reconstruct large scenes by applying it in a "sliding-window" fashion at inference time. We exploit this property to obtain reconstructions of entire apartments (see Fig. 3.1).

3.2.5 Network Architectures

Here we provide a detailed description of our network architectures.

Point Cloud Encoder. We first use a fully-connected layer followed by a fully-connected ResNet [72] block to map the three-dimensional input point coordinates into the feature space. Next, unlike PointNet [172] which pools over all points to acquire a global feature, we perform the pooling operation locally. Depending on the defined plane/volume feature maps, we perform max-pooling only over the

features falling into the same pixel/voxel cell. The locally pooled features are concatenated with the features before pooling, and then fed into the next ResNet block. We use 5 of these ResNet blocks with intermediate pooling to obtain the final point-wise features.

Voxel Encoder. Given an occupancy grid as input, we use a single 3D convolutional layer with convolution kernel size $3 \times 3 \times 3$ to extract voxel-wise features with dimension of 32.

U-Net. We use a U-Net [41, 181] to process the plane or volume features. We follow [181] and adapt a modified implementation from [80] for our 2D variants. For our 3D variant, we adapt the 3D U-Net [41] implementation from [238]. We set the input and output feature dimensions to 32. Note that we choose the depth of the U-Net such that the receptive field is equal or larger than the size of the feature plane or volume. For example, when considering a 3D feature volume of 32^3 or a 2D feature plane of 128^2, the depth is set to 3 or 5, respectively.

Occupancy Prediction Decoder. To predict the occupancy probability of query points, we use the network of [162] comprising a stack of fully-connected ResNet blocks. Table 3.5 provides an overview of the number of ResNet blocks and hidden dimensions. For all experiments, we use a hidden feature dimension of 32 and 5 ResNet blocks for the occupancy prediction network.

Architecture Comparison with ONet [144]. For point cloud inputs, ONet uses a PointNet [172] as point cloud encoder and 5 fully-connected ResNet blocks as occupancy decoder. Both networks have a hidden dimension of 512, resulting in 10.4 million parameters in total. In contrast, our method uses shallow variants for both networks with a hidden dimension of 32: as discussed, we use a shallow local PointNet and consider the less memory-intense conditioning in the decoder from [161]. Combined, our shallow PointNet and our decoder have 43k parameters. Our 2D/3D U-Net has roughly 1 million parameters depending on the depth. Thus, our final model is

more memory-efficient than ONet. Moreover, we perform batch-processing over instances as well as points. Hence, the decoder is queried more often than the encoder. As we are able to use a shallow decoder, this further reduces memory consumption in practice.

3.2.6 Implementation Details of Fully-Convolutional Model

For very large scenes, such as vast mansions in Matterport3D with numerous rooms, it is not realistic to reconstruct the entire place with one forward pass due to the memory constraint. To address this challenge, we need to modify our pipeline and fully exploit the translation equivariant property of convolution networks. This allows our method to scale to scenes with arbitrary size represented in metric real-world units (i.e., in meters). Importantly, this fully-convolutional model should not depend on a global coordinate system, but only on relative local coordinates.

To train such a model capable of reconstructing very large-scale scenes, we use our synthetic indoor scene dataset. A scene consists of multiple objects from the ShapeNetCore [24] dataset, see Sec. 3.3 for details. While no real-world units are provided in this dataset, we find that the synthetic scenes roughly correlate to a real-world unit of 4.4m $\times$ 4.4m $\times$ 4.4m. The voxel size s is a hyperparameter of our model and determines the granularity of the convolutional part. In all experiments, we set $s = 0.02$m. Therefore, each scene is contained in a regular grid of size $220 \times 220 \times 220$ voxels.

For training the network, we predict the occupancy of query points inside grid volumes cropped randomly within the scene. Specifically, at each iteration, we randomly sample one point within the scene as the center of the crop. The crop size for query points is $H \times W \times D$, which is defined as $25 \times 25 \times 25$ voxels in our experiments. To effectively handle the boundary, we take a bigger input crop. Since the receptive field of our network is $r = 64$, the corresponding input crop has a size of $(H + 63) \times (W + 63) + (D + 63) = 88 \times 88 \times 88$

voxels. We use the point cloud encoder described in Sec. 3.2.5 to encode the input point clouds inside each input crop. We use a batch size of 2 crops in practice to fit in a single Nvidia RTX 2080-Ti GPU.

Similarly, at inference time, we split the scene into overlapping input crops so that we can perform occupancy prediction of every crop in a sliding-window manner. The crop size is determined to fit into GPU memory. Note that the input crops overlap, such that no padding is needed to explicitly handle the boundary between crops.

3.3 Experiments

We conduct three types of experiments to evaluate our method. First, we perform **object-level reconstruction** on ShapeNet [24] chairs, considering noisy point clouds and low-resolution occupancy grids as inputs. Next, we compare our approach against several baselines on the task of **scene-level reconstruction** using a synthetic indoor dataset of various objects. Finally, we demonstrate **synthetic-to-real generalization** by evaluating our model on real indoor scenes [23, 44].

Datasets.

ShapeNet [24]: We use all 13 classes of the ShapeNet subset, voxelizations, and train/val/test split from Choy et al. [40]. Per-class results can be found in supplementary.

Synthetic Indoor Scene Dataset: We create a synthetic dataset with multiple objects from ShapeNet (chair, sofa, lamp, cabinet, table). We consider scenes with 4 to 8 objects and for each type, we generate 1000 scenes, so there are 5000 scenes in total.

For a single scene, we sample the x-y ratio of the ground plane uniformly between 0.3 and 1.0. For each object in the scene, we sample a rotation angle around the z-axis and a scaling factor uniformly from an interval that depends on how many objects are in the scene in

total. We place the objects randomly in the scene via rejection sampling. We draw 4 samples from a Bernoulli distribution to decide whether to add a wall to the respective border of the scene. We sample the wall height uniformly from the interval between 0.2 and 0.4. We further adhere to the object-level splits from [40] to not have similar objects in scenes of different splits.

ScanNet v2 [44]: This dataset contains 1513 real-world rooms captured with an RGB-D camera. We sample point clouds from the provided meshes for testing.

Matterport3D [23]: Matterport3D contains 90 buildings with multiple rooms on different floors captured using a Matterport Pro Camera. Similar to ScanNet, we sample point clouds for evaluating our model on Matterport3D.

Baselines.

ONet [144]: Occupancy Networks is a state-of-the-art implicit 3D reconstruction model. It uses a fully-connected network architecture and a global encoding of the input. We compare against this method in all of our experiments.

PointConv: We construct another simple baseline by extracting point-wise features using PointNet++ [173], interpolating them using Gaussian kernel regression and feeding them into the same fully-connected network used in our approach. While this baseline uses local information, it does not exploit convolutions. We adopt the PyTorch implementation from [251].

More specifically, we first calculate the Euclidean distance between a query point and all points in the input point cloud. The weights are then computed using a Gaussian kernel with 0.1 as the defined variance. After performing weight normalization, we acquire interpolated point-wise features for query points and estimate their occupancy probability with an occupancy network as discussed before. We train PointConv end-to-end by backpropagating through the convolutional operations and the Gaussian kernel regression.

SPSR [97]: Screened Poisson Surface Reconstruction (SPSR) is a traditional 3D reconstruction technique which operates on oriented point clouds as input. Note that in contrast to all other methods, SPSR requires additional surface normals which are often hard to obtain for real-world scenarios.

Training Details.

All methods are trained for at least 300000 iterations. We use the Adam optimizer [101] with a learning rate of 10^{-4} for all methods. We perform evaluations on the validation set every 10000 iterations and pick the model for testing which performs best wrt. volumetric IoU on the validation set.

Object-Level Reconstruction. For the reconstruction from point cloud experiments, we use a batch size of 32 for all our methods including ONet [144], and 24 for the baseline PointConv. For the voxel super-resolution tasks, we train all methods with a batch size of 64.

Scene-Level Reconstruction. We use the official implementation[1] of ONet [144] but change the batch size to 12 in order to fit into GPU memory. For the baseline PointConv the batch size is set to 16. Our lightweight architectures allow us to set the batch size to 32 for our plane encoder for a resolution of 128^2 and 3×128^2, as well as the volumetric encoder for a resolution of 32^3. For our variant combining 2D and 3D features, the batch size is 24, while we use a batch size of 6 for our volumetric approach with a resolution of 64^3.

Metrics.

Following [144], we consider Volumetric IoU, Chamfer Distance, Normal Consistency for evaluation. We further report F-Score [211] with the default threshold value of 1% unless otherwise specified.

Volumetric Intersection over Union (IoU). Let $\mathcal{M}_{\text{pred}}$ and $\mathcal{M}_{\text{GT}}$ be

[1] https://github.com/autonomousvision/occupancy_networks

the set of all points that are inside or on the surface of the predicted and ground truth mesh, respectively. The volumetric IoU is the volume of two meshes' intersection divided by the volume of their union:

$$\text{IoU}(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) \equiv \frac{|\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{GT}}|}{|\mathcal{M}_{\text{pred}} \cup \mathcal{M}_{\text{GT}}|}. \tag{3.3}$$

We randomly sample 100k points from the bounding boxes and determine if the points lie inside or outside $\mathcal{M}_{\text{pred}}$ and $\mathcal{M}_{\text{GT}}$, respectively.

Chamfer-L_1. Define accuracy and completeness of $\mathcal{M}_{\text{pred}}$ wrt. $\mathcal{M}_{\text{GT}}$:

$$\text{Accuracy}(\mathcal{M}_{\text{pred}}|\mathcal{M}_{\text{GT}}) \equiv \frac{1}{|\partial\mathcal{M}_{\text{pred}}|} \int_{\partial\mathcal{M}_{\text{pred}}} \min_{\mathbf{q}\in\partial\mathcal{M}_{\text{GT}}} \|\mathbf{p}-\mathbf{q}\| \mathrm{d}\mathbf{p} \tag{3.4}$$

$$\text{Complete.}(\mathcal{M}_{\text{pred}}|\mathcal{M}_{\text{GT}}) \equiv \frac{1}{|\partial\mathcal{M}_{\text{GT}}|} \int_{\partial\mathcal{M}_{\text{GT}}} \min_{\mathbf{p}\in\partial\mathcal{M}_{\text{pred}}} \|\mathbf{p}-\mathbf{q}\| \mathrm{d}\mathbf{q} \tag{3.5}$$

where $\partial\mathcal{M}_{\text{pred}}$ and $\partial\mathcal{M}_{\text{GT}}$ denote the surfaces of the two meshes. Then, the Chamfer-L_1 distance between two meshes is defined as below:

$$\begin{aligned}&\text{Chamfer-}L_1(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) = \\ &\frac{1}{2}(\text{Accuracy}(\mathcal{M}_{\text{pred}}|\mathcal{M}_{\text{GT}}) + \text{Completeness}(\mathcal{M}_{\text{pred}}|\mathcal{M}_{\text{GT}}))\end{aligned} \tag{3.6}$$

Normal Consistency. we define the normal consistency score as

$$\begin{aligned}\text{Normal-Con.}(\mathcal{M}_{\text{pred}}, \mathcal{M}_{\text{GT}}) \equiv \\ \frac{1}{2\,|\partial\mathcal{M}_{\text{pred}}|} \int_{\partial\mathcal{M}_{\text{pred}}} |\langle n(\mathbf{p}), n(\text{proj}_2(\mathbf{p}))\rangle| \mathrm{d}\mathbf{p} \\ + \frac{1}{2\,|\partial\mathcal{M}_{\text{GT}}|} \int_{\partial\mathcal{M}_{\text{GT}}} |\langle n(\text{proj}_1(\mathbf{q})), n(\mathbf{q})\rangle| \mathrm{d}\mathbf{q}\end{aligned} \tag{3.7}$$

where $\langle\cdot,\cdot\rangle$ indicates the inner product, $n(\mathbf{p})$ and $n(\mathbf{q})$ the (unit) normal vectors on the mesh surface $\partial\mathcal{M}_{\text{pred}}$ and $\partial\mathcal{M}_{\text{GT}}$, respectively and

$\text{proj}_2(\mathbf{p})$ and $\text{proj}_1(\mathbf{q})$ denote the projections of $\mathbf{p}$ and $\mathbf{q}$ onto $\partial\mathcal{M}_{\text{GT}}$ and $\partial\mathcal{M}_{\text{pred}}$ respectively.

F-Score. We first define recall and precision. As discussed in [211], recall counts how many points on the GT mesh lie within a certain distance to the reconstruction. Precision counts the percentage of points on the reconstructed mesh that lie within a certain distance to the GT. The F-Score is then defined as the harmonic mean between precision and recall:

$$\text{F-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.8}$$

3.3.1 Object-Level Reconstruction

We first evaluate our method on the single object reconstruction task on ShapeNet [24]. We consider three different types of 3D input: noisy point clouds, noisy partial point clouds, and low-resolution voxels. For the first case, we sample 3000 points from the mesh and apply Gaussian noise with zero mean and standard deviation 0.05. For the partial point clouds, we sample 3000 points from the cropped GT mesh, where we randomly cut out parts of the original mesh. As for the last case, we use coarse 32^3 voxelizations from [144]. For the query points (i.e., for which supervision is provided), we follow [144] and uniformly sample 2048 and 1024 points for noisy (partial) point clouds and low-resolution voxels, respectively. Due to the different encoder architectures for these tasks, we set the batch size to 32 for point cloud reconstruction and 64 for voxel super-resolution, respectively.

Reconstruction from Point Clouds. Table 3.1, Fig. 3.3, Fig. 3.4, Fig. 3.5, and Fig. 3.6 show quantitative and qualitative results. Compared to the baselines, all variants of our method achieve equal or better results on all three metrics. As evidenced by the training progression plot on the right, our method reaches a high validation

	GPU Memory	IoU	Chamfer-L_1	Normal C.	F-Score
PointConv	5.1G	0.689	0.126	0.858	0.644
ONet [144]	7.7G	0.761	0.087	0.891	0.785
Ours-2D (64^2)	1.6G	0.833	0.059	0.914	0.887
Ours-2D (3×64^2)	2.4G	**0.884**	**0.044**	**0.938**	**0.942**
Ours-3D (32^3)	5.9G	0.870	0.048	0.937	0.933

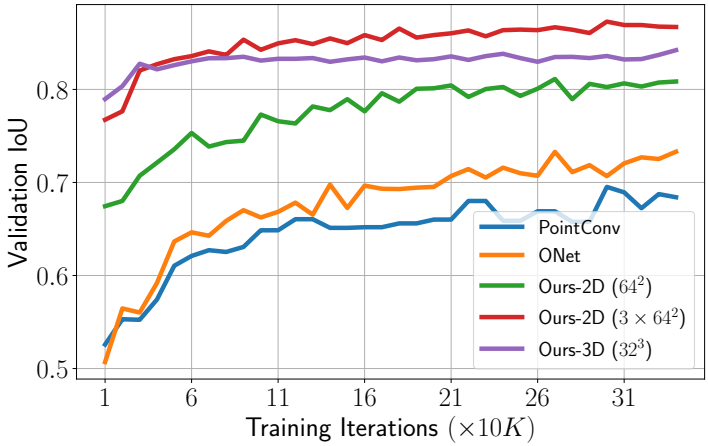


Table 3.1: ***Object-Level 3D Reconstruction from Point Clouds.*** *Top: We report GPU memory, IoU, Chamfer-L_1 distance, Normal Consistency, and F-Score for our approach (2D plane and 3D voxel grid dimensions in brackets), the baselines ONet [144] and PointConv on ShapeNet (mean over all 13 classes). Bottom: The training progression plot shows that our method converges faster than the baselines.*

IoU after only a few iterations. This verifies our hypothesis that leveraging convolutions and local features benefits 3D reconstruction in both accuracy and efficiency. The results show that, compared to PointConv which directly aggregates features from point clouds, projecting point features to planes or volumes followed by 2D/3D CNNs is more effective. In addition, decomposing 3D representations from volumes into three planes with higher resolution (64^2 vs. 32^3) improves performance while at the same time requiring less GPU memory.

Figure 3.3: ***Object-Level 3D Reconstruction from Point Clouds (Part 1).*** *Comparison of our convolutional representation to ONet and PointConv on ShapeNet objects.*

Figure 3.4: *Object-Level 3D Reconstruction from Point Clouds (Part 2).* *Comparison of our convolutional representation to ONet and PointConv on ShapeNet objects.*

Figure 3.5: ***Object-Level 3D Reconstruction from Point Clouds (Part 3).*** *Comparison of our convolutional representation to ONet and PointConv on ShapeNet objects.*

Figure 3.6: ***Object-Level 3D Reconstruction from Point Clouds (Part 4).*** *Comparison of our convolutional representation to ONet and PointConv on ShapeNet objects.*

	GPU Memory	IoU	Chamfer-L_1	Normal C.	F-Score
Input	-	0.631	0.136	0.810	0.440
ONet [144]	4.8G	0.703	0.110	0.879	0.656
Ours-2D (64^2)	2.4G	0.652	0.145	0.861	0.592
Ours-2D (3×64^2)	4.0G	**0.752**	0.092	0.905	**0.735**
Ours-3D (32^3)	10.8G	**0.752**	**0.091**	**0.912**	0.729

Table 3.2: ***Voxel Super-Resolution.*** *3D reconstruction results from low resolution voxelized inputs (32^3 voxels) on the ShapeNet dataset (mean over 13 classes).*

Reconstruction from Partial Point Clouds. We also investigate the ability of our method to reconstruct shapes from partial point clouds. To this end, we first randomly select one axis of the x, y, z directions and calculate its coordinate range r. Then, we uniformly sample an offset between $[0.7r,\ r]$ and filter out all points with coordinates larger than the offset along that axis. The offset is always a positive value, so e.g. for the z axis, we always crop from the top. Finally, 3000 points are uniformly sampled from the cropped point clouds. Fig. 3.7 shows our qualitative results.

Voxel Super-Resolution. Besides noisy point clouds, we also evaluate the task of voxel super-resolution. Here, the goal is to recover high-resolution details from coarse (32^3) voxelizations of the shape. Table 3.2 and Fig. 3.8 show that our method with three planes achieves comparable results over our volumetric method while requiring only 37% of the GPU memory. In contrast to reconstruction from point clouds, our single-plane approach fails at this task. We hypothesize that a single plane is not sufficient for resolving ambiguities in the coarse but regularly structured voxel input.

Generalization. In the last experiment for the object-level reconstruction, we want to investigate the generalizability of our proposed method. To this end, we train only on the "chair" category and test on "table". In contrast to baselines, our method degrades gracefully

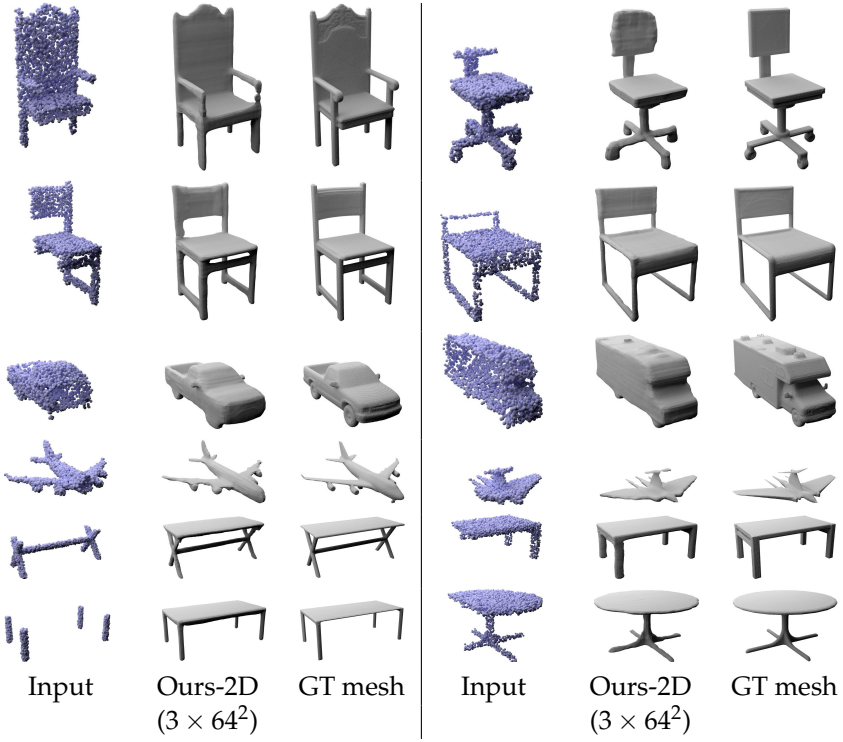


Figure 3.7: ***Object-Level 3D Reconstruction from Partial Point Clouds.*** *We show qualitative results on the ShapeNet "plane", "car", "chair" and "table" categories. Our method correctly reconstruct 3D shapes from partial point clouds. Note that the models are trained in all classes.*

(Fig. 3.9). This emphasizes the importance of equivariant representations and geometric reasoning using both local and global features.

3.3.2 Scene-Level Reconstruction

To analyze whether our approach can scale to larger scenes, we now reconstruct 3D geometry from point clouds on our synthetic indoor

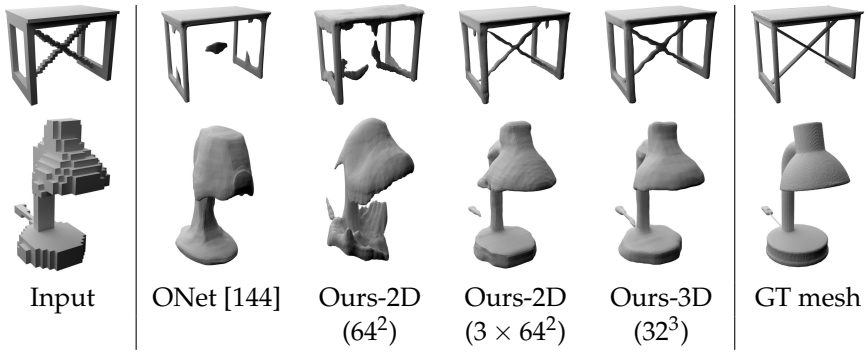


Figure 3.8: ***Voxel Super-Resolution.*** *Qualitative comparison between our method and ONet using coarse voxelized inputs at resolution 32^3 voxels.*

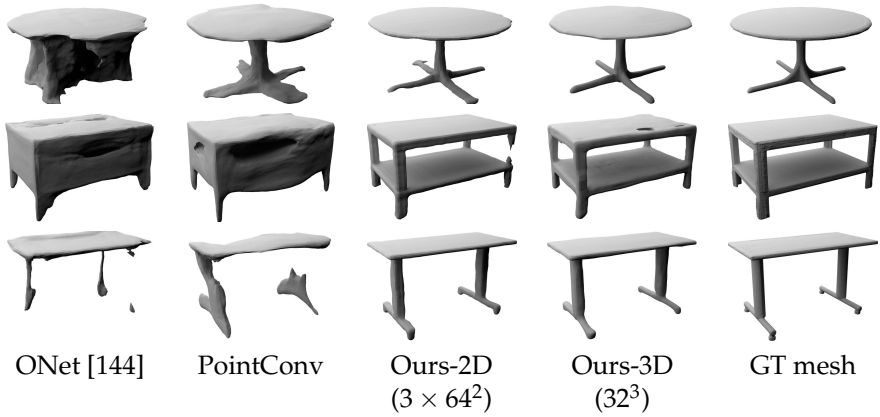


Figure 3.9: ***Generalization (Chair $\rightarrow$ Table).*** *We analyze the generalization performance of our method and the baselines by training them on the ShapeNet "chair" category and evaluating them on the "table" category.*

scene dataset. Due to the increasing complexity of the scene, we uniformly sample 10000 points as input point cloud and apply Gaussian noise with a standard deviation of 0.05. During training, we sample 2048 query points, similar to object-level reconstruction. For our plane-based methods, we use a resolution of 128^2. For our volumet-

	IoU	Chamfer-L_1	Normal C.	F-Score
ONet [144]	0.475	0.203	0.783	0.541
PointConv	0.523	0.165	0.811	0.790
SPSR [97]	-	0.223	0.866	0.810
SPSR [97] (trimmed)	-	0.069	0.890	0.892
Ours-2D (128^2)	0.795	0.047	0.889	0.937
Ours-2D (3×128^2)	0.805	0.044	0.903	0.948
Ours-3D (32^3)	0.782	0.047	0.902	0.941
Ours-3D (64^3)	**0.849**	**0.042**	**0.915**	**0.964**
Ours-2D-3D ($3 \times 128^2 + 32^3$)	0.816	0.044	0.905	0.952

Table 3.3: ***Scene-Level Reconstruction on Synthetic Rooms.*** *Quantitative comparison for reconstruction from noisy point clouds on synthetic rooms. We do not report IoU for SPSR because SPSR generates only a single surface for walls and the ground plane. To ensure a fair comparison to SPSR, we compare all methods with only a single surface for walls/ground planes when calculating Chamfer-L_1 and F-Score.*

ric approach, we investigate both 32^3 and 64^3 resolutions. Assuming that the plane and volumetric features are complementary, we also test the combination of the multi-plane and volumetric variants.

Table 3.3 and Fig. 3.10 show our results. All variants of our method are able to reconstruct geometric details of the scenes and lead to smooth results. In contrast, ONet and PointConv suffer from low accuracy while SPSR leads to noisy surfaces. While high-resolution canonical plane features capture fine details they are prone to noise. Low-resolution volumetric features are instead more robust to noise, yet produce smoother surfaces. Combining complementary volumetric and plane features improves results compared to considering them in isolation. This confirms our hypothesis that plane-based and volumetric features are complementary. However, the best results in this setting are achieved when increasing the resolution of the volumetric features to 64^3.

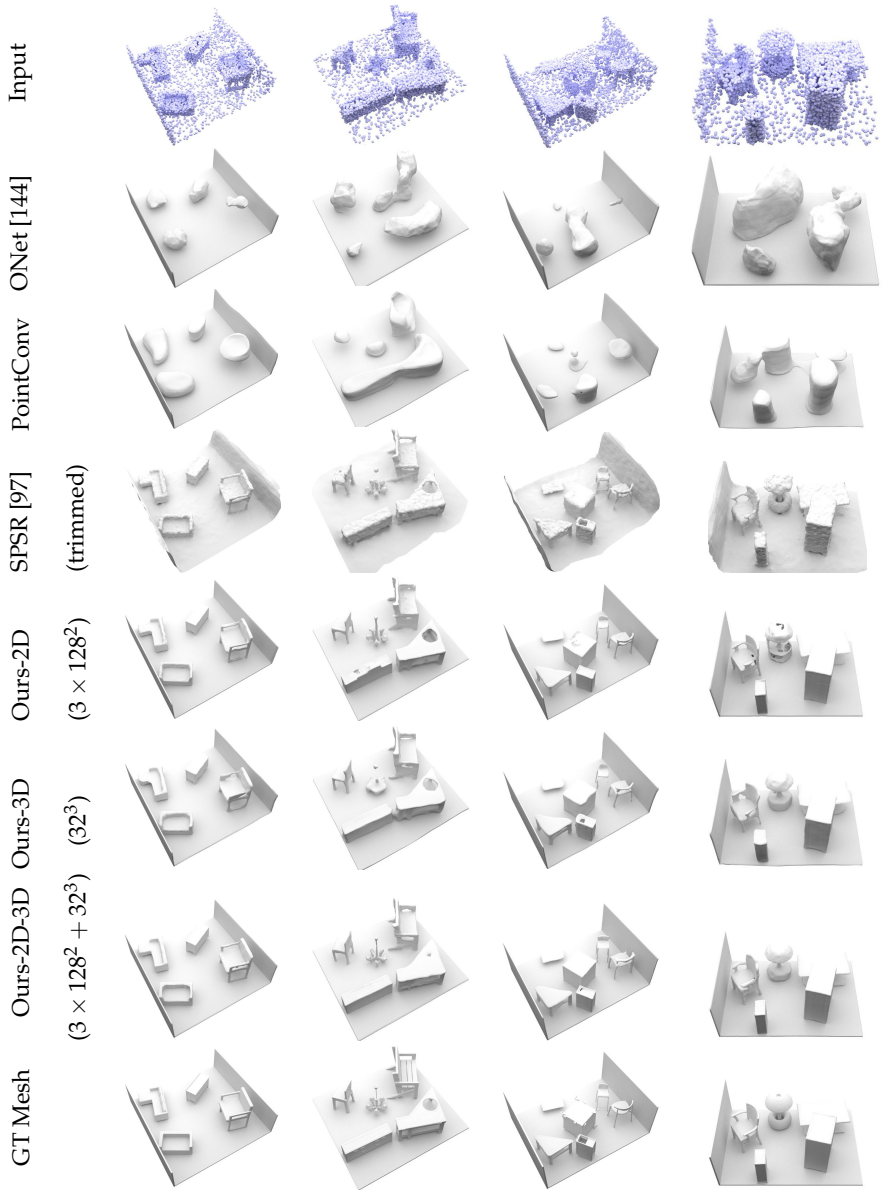


Figure 3.10: ***Scene-Level Reconstruction on Synthetic Rooms.*** *Qualitative comparison for point-cloud based reconstruction on the synthetic indoor scene dataset.*

3.3.3 Ablation Study

In this section, we investigate on our synthetic indoor scene dataset different feature aggregation strategies at similar GPU memory consumption as well as different feature interpolation strategies.

Performance at Similar GPU Memory. Table 3.4a shows a comparison of different feature aggregation strategies at similar GPU memory utilization. Our multi-plane approach slightly outperforms the single plane and the volumetric approach in this setting. Moreover, the increase in plane resolution for the single plane variant does not result in a clear performance boost, demonstrating that higher resolution does not necessarily guarantee better performance.

Feature Interpolation Strategy. To analyze the effect of the feature interpolation strategy in the convolutional decoder of our method, we compare nearest neighbor and bilinear interpolation for our multi-plane variant. The results in Table 3.4b clearly demonstrate the benefit of bilinear interpolation.

Network Architecture. Table 3.5 provides an ablation study of the number of ResNet blocks and hidden dimensions. To balance the memory efficiency and performance, we use a hidden feature dimension of 32 and 5 ResNet blocks for the occupancy prediction network for all experiments.

3.3.4 Reconstruction on Real-World Datasets

Next, we investigate the generalizability of our method. Towards this goal, we evaluate our models trained on the synthetic indoor scene dataset on the real world datasets ScanNet v2 [44] and Matterport3D [23]. Similar to our previous experiments, we use 10000 points sampled from the meshes as input.

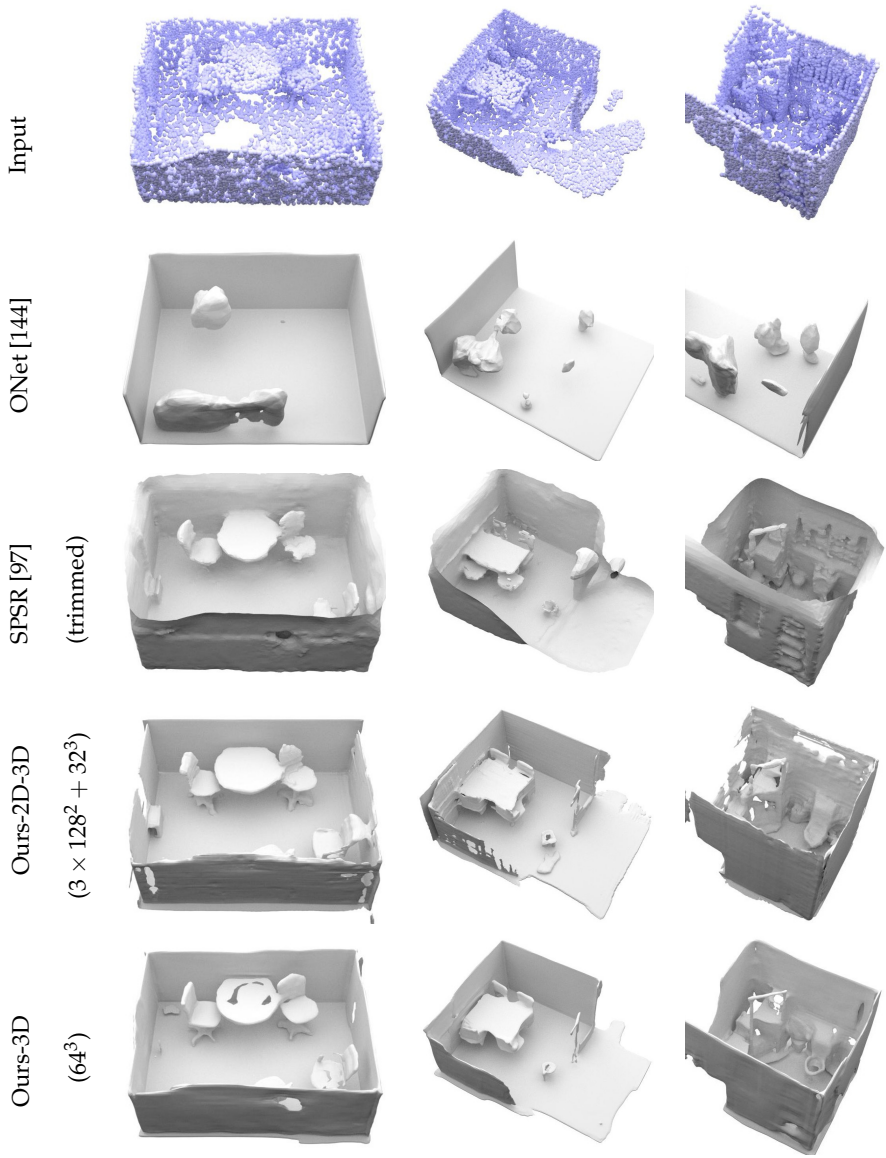


Figure 3.11: ***Scene-Level Reconstruction on ScanNet.*** *Qualitative results for point-based reconstruction on ScanNet [44]. All learning-based methods are trained on the synthetic room dataset and evaluated on ScanNet.*

	GPU Memory	IoU	Chamfer-L_1	Normal C.	F-Score
Ours-2D (192^2)	9.5GB	0.773	0.047	0.889	0.937
Ours-2D (3×128^2)	9.3GB	**0.805**	**0.044**	**0.903**	**0.948**
Ours-3D (32^3)	8.5GB	0.782	0.047	0.902	0.941

(a) *Performance at similar GPU Memory*

	IoU	Chamfer-L_1	Normal C.	F-Score
Nearest Neighbor	0.766	0.052	0.885	0.920
Bilinear	**0.805**	**0.044**	**0.903**	**0.948**

(b) *Interpolation Strategy*

Table 3.4: ***Ablation Study on Synthetic Rooms.*** *We compare the performance of different feature aggregation strategies at similar GPU memory in Table 3.4a and evaluate two different sampling strategies in Table 3.4b.*

Reconstruction for ScanNet. Our results in Table 3.6 show that among all our variants, the volumetric-based models perform best, indicating that the plane-based approaches are more affected by the domain shift. We find that 3D CNNs are more robust to noise as they aggregate features from all neighbors which results in smooth outputs. Furthermore, all variants outperform learning-based baselines by a significant margin.

The qualitative comparison in Fig. 3.11 shows that our model is able to smoothly reconstruct scenes with geometric details at various scales. While Screened PSR [97] also produces reasonable reconstructions, it tends to close the resulting meshes and hence requires a carefully chosen trimming parameter. In contrast, our method does not require additional hyperparameters.

Reconstruction for Large Matterport3D Scene. Finally, we investigate the scalability of our method to larger scenes that comprise multiple rooms and multiple floors. For this experiment, we exploit the Matterport3D dataset [23]. Unlike before, we implemented a fully

No. Blocks	Hidden Dim.	GPU Memory	IoU	Chamfer-L_1	Normal C.
3	32	2.2G	0.857	0.050	0.936
5	32	2.4G	0.861	0.048	0.937
5	256	3.8G	**0.864**	**0.047**	**0.941**

Table 3.5: ***Ablation Study on Network Architecture.*** *We train our 3-plane method with a resolution of 64^2 on the ShapeNet "chair" class with different numbers of ResNet blocks and hidden feature dimensions.*

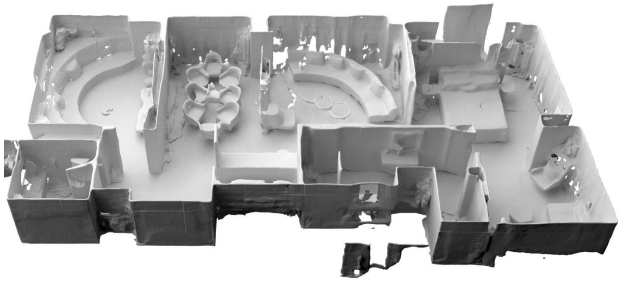
	Chamfer-L_1	F-Score
ONet [144]	0.398	0.390
PointConv	0.316	0.439
SPSR [97]	0.293	0.731
SPSR [97] (trimmed)	0.086	0.847

	Chamfer-L_1	F-Score
Ours-2D (128^2)	0.139	0.747
Ours-2D (3×128^2)	0.142	0.776
Ours-3D (32^3)	0.095	0.837
Ours-3D (64^3)	**0.077**	**0.886**
Ours-2D-3D ($3 \times 128^2 + 32^3$)	0.099	0.847

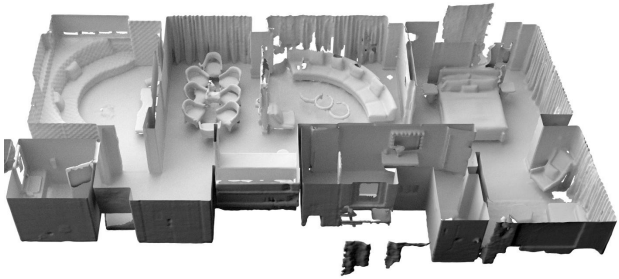
Table 3.6: ***Scene-Level Reconstruction on ScanNet.*** *Evaluation of point-based reconstruction on the real-world ScanNet dataset. As ScanNet does not provide watertight meshes, we trained all methods on the synthetic indoor scene dataset. Remark: In ScanNet, walls/floors are only observed from one side. To not incorrectly penalize methods for predicting walls and floors with thickness (0.01 in our training set), we chose an F-Score threshold of 1.5% for this experiment.*

convolutional version of our 3D model that can be scaled to any size by running on overlapping crops of the input point cloud in a sliding window fashion. The overlap is determined by the size of the receptive field to ensure the correctness of the results. Details are provided in Sec. 3.2.6.

Fig. 3.1, Fig. 3.12, Fig. 3.13 show the resulting 3D reconstruction. Our method reconstructs the details inside each room while adhering to the room layout. Given a reasonable amount of surface points, we can see that our method is able to reconstruct scenes of different sizes, ranging from apartments to entire buildings. Note that the geometry and point distribution of the Matterport3D dataset differs significantly from the synthetic indoor scene dataset on which our model is trained. This demonstrates that our method is able to gen-



(a) Ours

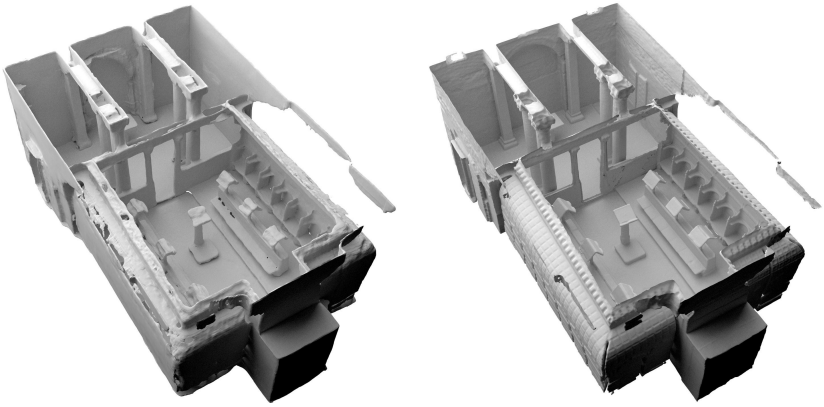


(b) GT Mesh

Figure 3.12: ***Scene-Level Reconstruction on Matterport3D.*** *Scene size:* $18.5m \times 9.6m \times 2.2m$*. No. points in input point cloud: 60K.*

eralize not only to unseen classes, but also to novel room layouts and sensor characteristics.

We further show the comparison over SPSR [97] in Fig. 3.14 and Fig. 3.15. Note that SPSR requires additional surface normals as input, whereas our method only needs raw point clouds. Moreover, SPSR requires a carefully chosen trimming factor. In contrast, our method does not require any such hyperparameter tuning. Our results indicate that our method better preserves details and the reconstructions contain fewer artifacts.



(a) Ours (b) GT Mesh

Figure 3.13: ***Scene-Level Reconstruction on Matterport3D.*** *Scene size:* $11.3m \times 6.6m \times 4.0m$. *No. points in input point cloud: 100K.*

3.4 Discussion

We introduced Convolutional Occupancy Networks, a novel shape representation that combines the expressiveness of convolutional neural networks with neural implicit representations. We analyzed the tradeoffs between 2D and 3D feature representations and found that incorporating convolutional operations facilitates generalization to unseen classes, novel room layouts and large-scale indoor spaces. Our 3-plane model is memory-efficient and excels in synthetic scenes with higher feature resolutions. Conversely, our 3D volumetric model performs better in real-world situations but uses more memory.

Limitations and Future Works. Our method is only translation equivariant for multiples of the voxel size and not rotation equivariant. Moreover, there is still a performance gap between synthetic and real data. While the focus of this work was on learning-based 3D reconstruction, in future work, we plan to apply our novel rep-

resentation to other domains such as implicit appearance modeling and 4D reconstruction.

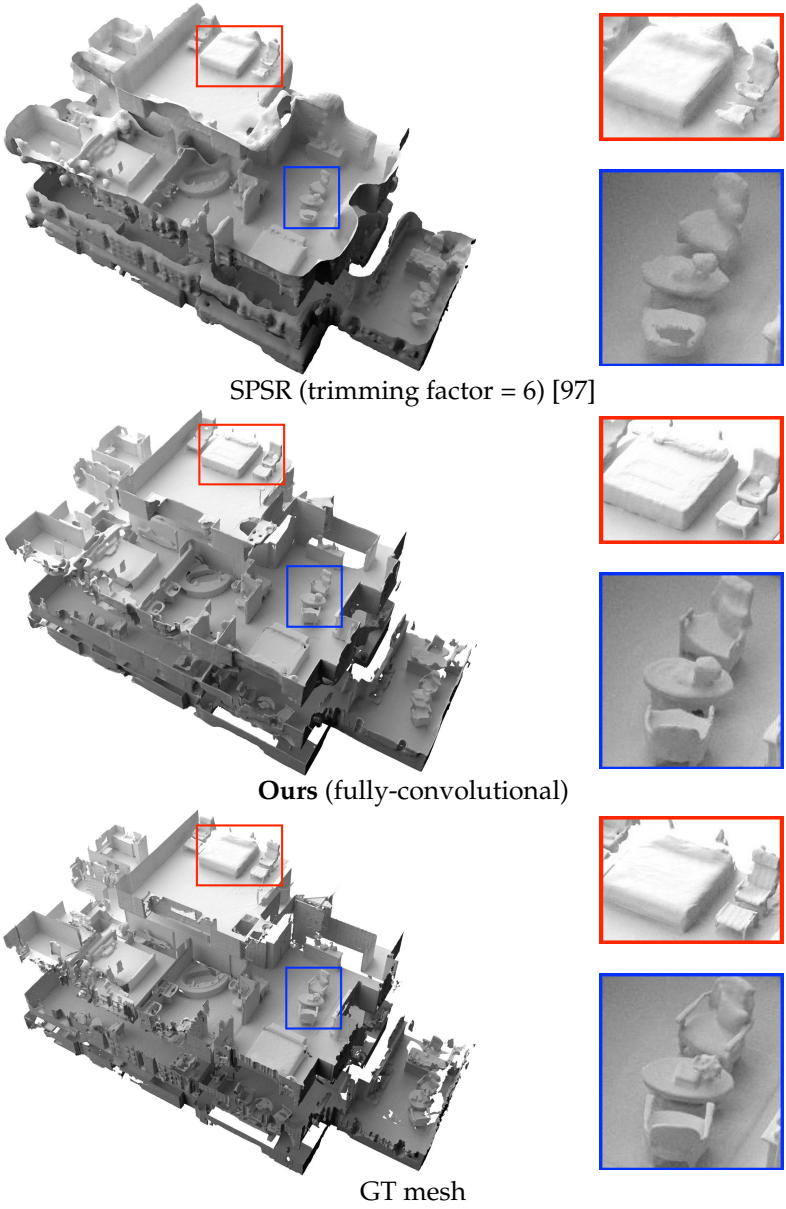


Figure 3.14: ***Comparison of Building-Level Reconstruction on Matterport3D.*** *Scene size:* $19.7m \times 10.9m \times 9.4m$. *200K points are sampled from the GT mesh and used as the input to SPSR and our method.*

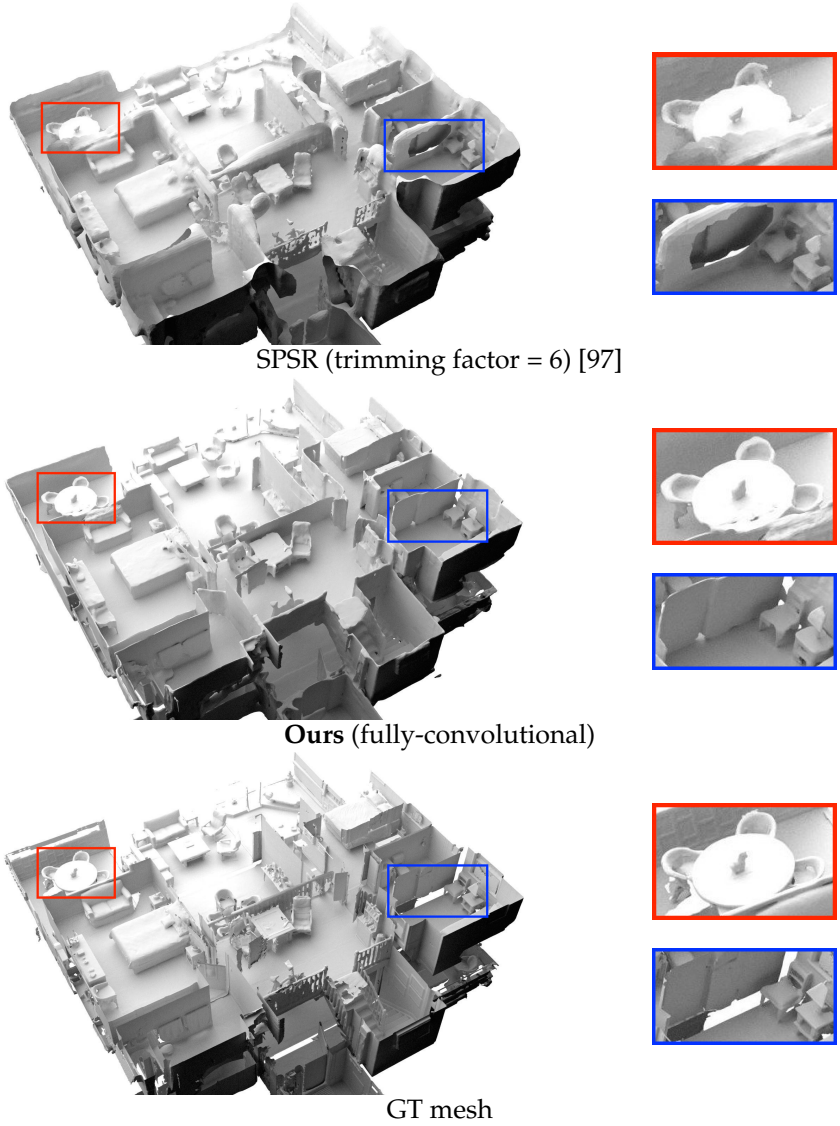


Figure 3.15: ***Comparison of Building-Level Reconstruction on Matterport3D.*** *Scene size:* $15.7m \times 12.3m \times 4.5m$. *200K points are sampled from the GT mesh and used as the input to SPSR and our method.*

CHAPTER 4

3D Reconstruction with a Differentiable Poisson Solver

In the previous chapter, we showed the potential of neural implicit representations in facilitating detailed 3D reconstruction. However, their implicit nature results in slow inference speed and requires careful initialization, which limits their real-world scenarios. In light of this, this chapter pivots to the classic yet widely adopted point cloud representation. We introduce a differentiable point-to-mesh layer, leveraging a differentiable formulation of Poisson Surface Reconstruction (PSR). This enables GPU-accelerated fast solution of the indicator function given an oriented point cloud. The inherent duality between points and meshes allows us to represent shapes as oriented point clouds, which are explicit, lightweight, and expressive. Our novel hybrid scene representations offer some benefits: Compared to neural implicit representations, not only do they provide enhanced interpretability and quicker inference (at a scale of an order of magnitude faster than neural implicit representations),. but

they also produce topology-agnostic, watertight manifold surfaces, setting them apart from other explicit representations like points clouds, patches, and meshes.

4.1 Introduction

As we already know, shape representations are central to many of the recent advancements in 3D computer vision and computer graphics, ranging from neural rendering [145, 148, 164, 177, 196] to shape reconstruction [29, 90, 144, 161, 166, 171, 256]. While conventional representations such as point clouds and meshes are efficient and well-studied, they also suffer from several limitations: Point clouds are lightweight and easy to obtain, but do not directly encode surface information. Meshes, on the other hand, are usually restricted to fixed topologies. More recently, neural implicit representations [29, 144, 166] have shown promising results for representing geometry due to their flexibility in encoding varied topologies, and their easy integration with differentiable frameworks. However, since such representations implicitly encode surface information, extracting the underlying surface is typically slow, as they require numerous network evaluations in 3D space for extracting complete surfaces using marching cubes [29, 144, 166], or along rays for intersection detection in the context of volumetric rendering [148, 161, 165, 256].

In this work, we introduce a novel Poisson solver that performs fast GPU-accelerated Differentiable Poisson Surface Reconstruction (DPSR) and solves for an indicator function from an oriented point cloud in a few milliseconds. Thanks to the differentiablility of our Poisson solver, gradients from a loss on the output mesh or a loss on the intermediate indicator grid can be efficiently backpropagated to update the oriented point cloud representation. This differential bridge between points, indicator functions, and meshes allows us to represent shapes as oriented point clouds. We, therefore, call

this shape representation *Shape-As-Points* (SAP). Compared to existing shape representations, Shape-As-Points has the following advantages (see also Table 4.1):

Efficiency: SAP has a low memory footprint as it only requires storing a collection of oriented point samples at the surface, rather than volumetric quantities (voxels) or a large number of network parameters for neural implicit representations. Using spectral methods, the indicator field can be computed efficiently (12 ms at 128^3 resolution[1]), compared to the typical rather slow query time of neural implicit networks (330 ms using [144] at the same resolution). **Accuracy:** The resulting mesh can be generated at high resolutions, is guaranteed to be watertight, free from self-intersections and also topology-agnostic. **Initialization:** It is easy to initialize SAP with a given geometry such as template shapes or noisy observations. In contrast, neural implicit representations are harder to initialize, except for few simple primitives such as spheres [2]. See supplementary for more discussions.

To investigate the aforementioned properties, we perform a set of controlled experiments. Moreover, we demonstrate state-of-the-art performance in reconstructing surface geometry from unoriented point clouds in two settings: an optimization-based setting that does not require training and is applicable to a wide range of shapes, and a learning-based setting for conditional shape reconstruction that is robust to noisy point clouds and outliers. In summary, the main contributions of this chapter are as follows.

- We present Shape-As-Points, a novel shape representation that is interpretable, lightweight, and yields high-quality watertight meshes at low inference times.
- The core of the Shape-As-Points representation is a versatile, differentiable and generalizable Poisson solver that can be used for a range of applications.

[1] On average, our method requires 12 ms for computing a 128^3 indicator grid from 15K points on a single NVIDIA GTX 1080Ti GPU. Computing a 256^3 indicator grid requires 140 ms.

Representations		Points [52]	Voxels [40]	Meshes [224]	Patches [65]	Implicits [144]	**SAP** (Ours)	GT
								
Efficiency	Grid Eval Time (128^3)	n/a	n/a	n/a	n/a	0.33s	0.012s	
Priors	Easy Initialization	✔	✔	✔	✘	✘	✔	
	Watertight	✘	✔	✔	✘	✔	✔	
Quality	No Self-intersection	n/a	n/a	✘	✘	✔	✔	
	Topology-Agnostic	✔	✔	✘	✔	✔	✔	

Table 4.1: ***Overview of Different Shape Representations.*** Shape-As-Points *produces higher quality geometry compared to other explicit representations [40, 52, 65] and requires significantly less inference time for extracting geometry compared to neural implicit representations [144].*

- We study various properties inherent to the Shape-As-Points representation, including inference time, sensitivity to initialization, and topology-agnostic representation capacity.
- We demonstrate state-of-the-art reconstruction results from noisy unoriented point clouds at a significantly reduced computational budget compared to existing methods.

4.2 Method

At the core of the Shape-As-Points representation is a differentiable Poisson solver, which can be used for both optimization-based and learning-based surface estimation. We first introduce the Poisson solver in Sec. 4.2.1. Next, we investigate two applications using our solver: optimization-based 3D reconstruction (Sec. 4.2.2) and learning-based 3D reconstruction (Sec. 4.2.3).

4.2.1 Differentiable Poisson Solver

The key step in Poisson Surface Reconstruction [97,98] involves solving the Poisson Equation. Let $\mathbf{x} \in \mathbb{R}^3$ denote a spatial coordinate and $\mathbf{n} \in \mathbb{R}^3$ denote its corresponding normal. The Poisson Equation arises from the insight that a set consisting of point coordinates and normals $\{\mathbf{p} = (\mathbf{c}, \mathbf{n})\}$ can be viewed as samples of the gradient of the underlying implicit indicator function $\chi(\mathbf{x})$ that describes the solid geometry. We define the normal vector field as a superposition of pulse functions $\mathbf{v}(\mathbf{x}) = \sum_{(\mathbf{c}_i, \mathbf{n}_i) \in \{\mathbf{p}\}} \delta(\mathbf{x} - \mathbf{c}_i, \mathbf{n}_i)$, where $\delta(\mathbf{x}, \mathbf{n}) = \{\mathbf{n} \text{ if } \mathbf{x} = 0 \text{ and } 0 \text{ otherwise}\}$. By applying the divergence operator, the variational problem transforms into the standard Poisson equation:

$$\nabla^2 \chi := \nabla \cdot \nabla \chi = \nabla \cdot \mathbf{v} \tag{4.1}$$

In order to solve this set of linear Partial Differential Equations (PDEs), we discretize the function values and differential operators.

Without loss of generality, we assume that the normal vector field $\mathbf{v}$ and the indicator function χ are sampled at r uniformly spaced locations along each dimension. Denote the spatial dimensionality of the problem to be d. Without loss of generality, we consider the three dimensional case where $n := r \times r \times r$ for $d = 3$. We have the indicator function $\chi \in \mathbb{R}^n$, the point normal field $\mathbf{v} \in \mathbb{R}^{n \times d}$, the gradient operator $\nabla : \mathbb{R}^n \mapsto \mathbb{R}^{n \times d}$, the divergence operator $(\nabla \cdot) : \mathbb{R}^{n \times d} \mapsto \mathbb{R}^n$, and the derived laplacian operator $\nabla^2 := \nabla \cdot \nabla : \mathbb{R}^n \mapsto \mathbb{R}^n$. Under such a discretization scheme, solving for the indicator function amounts to solving the linear system by inverting the divergence operator subject to boundary conditions of surface points having zero level set. Following [98], we fix the overall scale to $m = 0.5$ at $\mathbf{x} = 0$:

$$\chi = (\nabla^2)^{-1} \nabla \cdot \mathbf{v} \quad \text{s.t.} \quad \chi|_{\mathbf{x} \in \{\mathbf{c}\}} = 0 \quad \text{and} \quad \text{abs}(\chi|_{\mathbf{x}=0}) = m \tag{4.2}$$

Point Rasterization. We obtain the uniformly discretized point normal field $\mathbf{v}$ by rasterizing the point normals onto a uniformly sampled voxel grid. We can differentiably perform point rasterization via inverse trilinear interpolation, similar to the approach in [97, 98]. We scatter the point normal values to the voxel grid vertices, weighted by the trilinear interpolation weights. The point rasterization process has $\mathcal{O}(n)$ space complexity, linear with respect to the number of grid cells, and $\mathcal{O}(N)$ time complexity, linear with respect to the number of points. See appendix A.1.1 for details.

Spectral Methods for Solving PSR. In contrast to the finite-element approach taken in [97, 98], we solve the PDEs using spectral methods [18]. While spectral methods are commonly used in scientific computing for solving PDEs and in some cases applied to computer vision problems [114], we are the first to apply them in the context of Poisson Surface Reconstruction. Unlike finite-element approaches that depend on irregular data structures such as octrees or tetrahedral meshes for discritizing space, spectral methods can be efficently solved over a uniform grid as they leverage highly optimized Fast Fourier Transform (FFT) operations that are well supported for GPUs, TPUs, and mainstream deep learning frameworks. Spectral

methods decompose the original signal into a linear sum of functions represented using the sine / cosine basis functions whose derivatives can be computed analytically. This allows us to easily approximate differential operators in spectral space. We denote the spectral domain signals with a tilde symbol, i.e., $\tilde{\mathbf{v}} = \text{FFT}(\mathbf{v})$. We first solve for the unnormalized indicator function χ', not accounting for boundary conditions.

$$\chi' = \text{IFFT}(\tilde{\chi}) \quad \tilde{\chi} = \tilde{g}_{\sigma,r}(\mathbf{u}) \odot \frac{i\mathbf{u} \cdot \tilde{\mathbf{v}}}{-2\pi\|\mathbf{u}\|^2} \quad \tilde{g}_{\sigma,r}(\mathbf{u}) = \exp\Big(-2\frac{\sigma^2\|\mathbf{u}\|^2}{r^2}\Big) \tag{4.3}$$

where the spectral frequencies are denoted as $\mathbf{u} := (u, v, w) \in \mathbb{R}^{n\times d}$ corresponding to the x, y, z spatial dimensions, and $\text{IFFT}(\tilde{\chi})$ represents the inverse fast Fourier transform of $\tilde{\chi}$. $\tilde{g}_{\sigma,r}(\mathbf{u})$ is a Gaussian smoothing kernel of bandwidth σ at grid resolution r in the spectral domain. The Gaussian kernel is used to mitigate ringing effects as a result of the Gibbs phenomenon from rasterizing the point normals. We denote the element-wise product as $\odot : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$, the L2-norm as $\|\cdot\|^2 : \mathbb{R}^{n\times d} \mapsto \mathbb{R}^n$, and the dot product as $(\cdot) : \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times d} \mapsto \mathbb{R}^n$. Finally, we subtract by the mean of the indicator function in the point set and scale the indicator function to obtain the solution to the PSR problem in Eqn. 4.2:

$$\chi = \underbrace{\frac{m}{\text{abs}(\chi'|_{\mathbf{x}=0})}}_{\text{scale}} \underbrace{\Big(\chi' - \frac{1}{|\{\mathbf{c}\}|}\sum_{\mathbf{c}\in\{\mathbf{c}\}} \chi'|_{\mathbf{x}=\mathbf{c}}\Big)}_{\text{subtract by mean}} \tag{4.4}$$

A detailed derivation of our differentiable PSR solver is provided in the appendix A.1.2.

4.2.2 SAP for Optimization-based 3D Reconstruction

We can use the proposed differentiable Poisson solver for various applications. First, we consider the classical task of surface reconstruction from unoriented point clouds. The overall pipeline for this

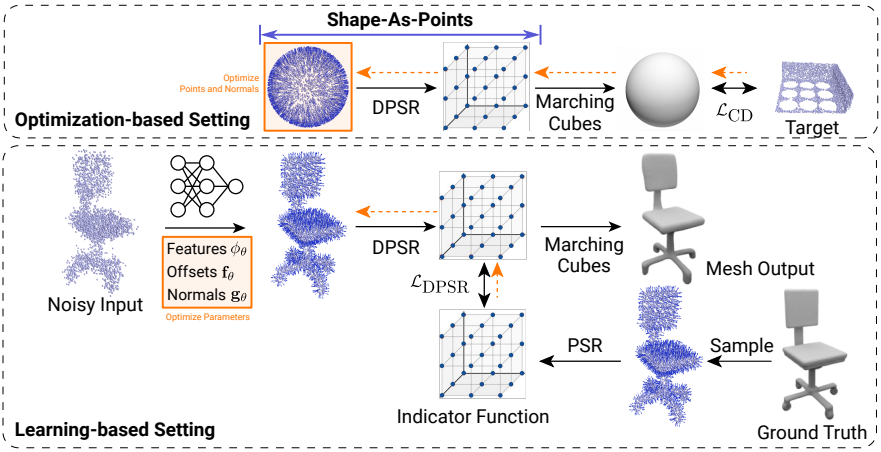


Figure 4.1: ***Model Overview.*** *Top: Pipeline for optimization-based single object reconstruction. The Chamfer loss on the target point cloud is backpropagated to the source point cloud w/ normals for optimization. Bottom: Pipeline for learning-based surface reconstruction. Unlike the optimization-based setting, here we provide supervision at the indicator grid level, since we assume access to watertight meshes for supervision, as is common practice in learning-based single-object reconstruction.*

setting is illustrated in Fig. 4.1 (top). We now provide details about each component.

Forward pass. It is natural to initialize the oriented 3D point cloud serving as 3D shape representation using the noisy 3D input points and corresponding (estimated) normals. However, to demonstrate the flexibility and robustness of our model, we purposefully initialize our model using a generic 3D sphere with radius r in our experiments. Given the orientated point cloud, we apply our Poisson solver to obtain an indicator function grid, which can be converted to a mesh using Marching Cubes [136].

Backward pass. For every point $\mathbf{p}_{\text{mesh}}$ sampled from the mesh $\mathcal{M}$, we calculate a bi-directional L2 Chamfer Distance $\mathcal{L}_{\text{CD}}$ with respect to the input point cloud. To backpropagate the loss $\mathcal{L}_{\text{CD}}$ through

$\mathbf{p}_{\text{mesh}}$ to point $\mathbf{p}$ in our source oriented point cloud, we decompose the gradient using the chain rule:

$$\frac{\partial \mathcal{L}_{\text{CD}}}{\partial \mathbf{p}} = \frac{\partial \mathcal{L}_{\text{CD}}}{\partial \mathbf{p}_{\text{mesh}}} \frac{\partial \mathbf{p}_{\text{mesh}}}{\partial \chi} \frac{\partial \chi}{\partial \mathbf{p}} \tag{4.5}$$

All terms in (4.5) are differentiable except for the middle one $\frac{\partial \mathbf{p}_{\text{mesh}}}{\partial \chi}$ which involves Marching Cubes. However, this gradient can be effectively approximated by the inverse surface normal [178]:

$$\frac{\partial \mathbf{p}_{\text{mesh}}}{\partial \chi} = -\mathbf{n}_{\text{mesh}} \tag{4.6}$$

where $\mathbf{n}_{\text{mesh}}$ is the normal of the point $\mathbf{p}_{\text{mesh}}$. Different from MeshSDF [178] that uses the gradients to update the latent code of a pretrained implicit shape representation, our method updates the source point cloud using the proposed differentiable Poisson solver.

Resampling. To increase the robustness of the optimization process, we uniformly resample points and normals from the largest mesh component every 200 iterations, and replace all points in the original point clouds with the resampled ones. This resampling strategy eliminates outlier points that drift away during the optimization, and enforces a more uniform distribution of points.

Coarse-to-fine. To further decrease run-time, we consider a coarse-to-fine strategy during optimization. More specifically, we start optimizing at an indicator grid resolution of 32^3 for 1000 iterations, from which we obtain a coarse shape. Next, we sample from this coarse mesh and continue optimization at a resolution of 64^3 for 1000 iterations. We repeat this process until we reach the target resolution (256^3) at which we acquire the final output mesh.

4.2.3 SAP for Learning-based 3D Reconstruction

We now consider the learning-based 3D reconstruction setting in which we train a conditional model that takes a noisy, unoriented

point cloud as input and outputs a 3D shape. More specifically, we train the model to predict a clean oriented point cloud, from which we obtain a watertight mesh using our Poisson solver and Marching Cubes. We leverage the differentiability of our Poisson solver to learn the parameters of this conditional model. Following common practice, we assume watertight meshes as ground truth and consequently supervise directly with the ground truth indicator grid obtained from these meshes. Fig. 4.1 (bottom) illustrates the pipeline of our architecture for the learning-based surface reconstruction task.

Architecture. We first encode the unoriented input point cloud coordinates $\{\mathbf{c}\}$ into a feature ϕ. The resulting feature should encapsulate both local and global information about the input point cloud. We utilize the convolutional point encoder proposed in [171] for this purpose. Note that in the following, we will use $\phi_\theta(\mathbf{c})$ to denote the features at point $\mathbf{c}$, dropping the dependency of ϕ on the remaining points $\{\mathbf{c}\}$ for clarity. Also, we use θ to refer to network parameters in general.

Given their features, our objective is to estimate both offsets and normals for every input point $\mathbf{c}$ in the point cloud $\{\mathbf{c}\}$. We use a shallow Multi-Layer Perceptron (MLP) $\mathbf{f}_\theta$ to predict the offset for $\mathbf{c}$:

$$\Delta\mathbf{c} = \mathbf{f}_\theta(\mathbf{c}, \phi_\theta(\mathbf{c})) \tag{4.7}$$

where $\phi(\mathbf{c})$ is obtained from the feature volume using trilinear interpolation. We predict the k offsets per input point, where $k \geq 1$. We add the offsets $\Delta\mathbf{c}$ to the input point position $\mathbf{c}$ and call the updated point position $\hat{\mathbf{c}}$. Additional offsets allow us to densify the point cloud, leading to enhanced reconstruction quality. We choose $k = 7$ for all learning-based reconstruction experiments (see ablation study in Table 4.6). For each updated point $\hat{\mathbf{c}}$, we use a second MLP $\mathbf{g}_\theta$ to predict its normal:

$$\hat{\mathbf{n}} = \mathbf{g}_\theta(\hat{\mathbf{c}}, \phi_\theta(\hat{\mathbf{c}})) \tag{4.8}$$

We use the same decoder architecture as in [171] for both $\mathbf{f}_\theta$ and $\mathbf{g}_\theta$. The network comprises 5 layers of ResNet blocks with a hidden dimension of 32. These two networks $\mathbf{f}_\theta$ and $\mathbf{g}_\theta$ do not share weights.

Training and Inference. During training, we obtain the estimated indicator grid $\hat{\chi}$ from the predicted point clouds $(\hat{\mathbf{c}}, \hat{\mathbf{n}})$ using our differentiable Poisson solver. Since we assume watertight and noise-free meshes for supervision, we acquire the ground truth indicator grid by running PSR on densely sampled point clouds of the ground truth meshes with the corresponding ground truth normals. This avoids running Marching Cubes at every iteration and accelerates training. We use the Mean Square Error (MSE) loss on the predicted and ground truth indicator grid:

$$\mathcal{L}_{\text{DPSR}} = \|\hat{\chi} - \chi\|^2 \tag{4.9}$$

We implement all models in PyTorch [168] and use the Adam optimizer [101] with a learning rate of 5e-4. During inference, we use our trained model to predict normals and offsets, use DPSR to solve for the indicator grid, and run Marching Cubes [136] to extract meshes.

4.3 Experiments

Following the exposition in the previous section, we conduct two types of experiments to evaluate our method. First, we perform single object reconstruction from unoriented point clouds. Next, we apply our method to learning-based surface reconstruction on ShapeNet [24], using noisy point clouds with or without outliers as inputs.

Datasets. We use the following datasets for optimization-based reconstruction: 1) Thingi10K [279], 2) Surface reconstruction benchmark (SRB) [236] and 3) D-FAUST [10]. Similarly to previous work, we used 5 objects per dataset [64,71,236]. For learning-based object-level reconstruction, we consider all 13 classes of the ShapeNet [24] subset, using the train/val/test split from [40].

Metrics. As in Chapter 3, we consider Chamfer Distance, Normal

Consistency and F-Score with the default threshold of 1% for evaluation, and also report optimization & inference time.

Baselines. In the optimization-based reconstruction setting, we compare to network-based methods IGR [64] and Point2Mesh [71], as well as Screened Poisson Surface Reconstruction[2] (SPSR) [97] on plane-fitted normals. To ensure that the predicted normals are consistently oriented for SPSR, we propagate the normal orientation using the minimum spanning tree [278]. For learning-based surface reconstruction, we compare against point-based Point Set Generation Networks (PSGN) [52], patch-based AtlasNet [65], voxel-based 3D-R2N2 [40], and ConvONet [171], which has recently reported state-of-the-art results on this task. We use ConvONet in their best-performing setting (3-plane encoders). SPSR is also used as a baseline. In addition, to evaluate the importance of our differentiable PSR optimization, we design another point-based baseline. This baseline uses the same network architecture to predict points and normals. However, instead of passing them to our Poisson solver and calculating $\mathcal{L}_{\text{DPSR}}$ on the indicator grid, we directly supervise the point positions with a bi-directional Chamfer distance, and an L1 Loss on the normals as done in [139]. During inference, we also feed the predicted points and normals to our PSR solver and run Marching Cubes to obtain meshes.

Implementation Details.

Optimization-based 3D reconstruction. We use the official implementation of IGR[3] [64]. We optimize IGR for 15000 iterations on each object until convergence. For Point2Mesh [71], we follow the official implementation[4] and use 6000 iterations for each object. We generate the initial mesh required by Point2Mesh following the description of the original paper. Specifically, the initial mesh is provided as the convex hull of the input point cloud for objects with a genus of zero.

[2] https://github.com/mkazhdan/PoissonRecon.
[3] https://github.com/amosgropp/IGR
[4] https://github.com/ranahanocka/point2mesh

If the genus is larger than zero, we apply the watertight manifold algorithm [83] using a low-resolution octree reconstruction on the output mesh of SPSR to obtain a coarse initial mesh.

For our method, besides following the coarse-to-fine and resampling strategy described in Sec. 4.2.2, we gradually increase the Gaussian smoothing parameter σ in Eq. (4.3) when increasing the grid resolution: $\sigma = 2$ for a grid resolution of 32^3 and 64^3, $\sigma = 3$ when the grid resolution is 128^3. At the final resolution of 256^3, we use $\sigma = 3$ for objects with more details (e.g. objects in SRB [236] and D-FAUST [10], and $\sigma = 5$ for the input points with noises (Thingi10K [279]) to smooth the output mesh as well as to stabilze the optimization process. We use the Adam optimizer [102] with a learning rate decay. The learning rate is set to 2×10^{-3} at the initial resolution of 32^3 with a decay of 0.7 after every increase of the grid resolution. Moreover, we run 1000 iterations at every grid resolution of 32^3, 64^3 and 128^3, and 200 iterations for 256^3. 20000 source points and normals are used by our method to represent the final shapes for all objects.

Learning-based 3D reconstruction. For AtlasNet [65], we use the official implementation[5] with 25 parameterizations. We change the number of input points from 2500 (default) to 3000 for our setting. Depending on the experiment, we add different noise levels or outlier points. We train ConvONet [171], PSGN [52], and 3D-R2N2 [40] for at least 300000 iterations, and use Adam optimizer [101] with a learning rate of 10^{-4} for all methods.

We train our method as well as Ours (w/o $\mathcal{L}_{\text{DPSR}}$) for all 3 noise levels for 300000 iterations (roughly 2 days with 2 GTX 1080Ti GPUs) and use Adam optimizer with a learning rate of 5×10^{-4}. We consider a batch size of 32. To generate the ground truth PSR indicator field χ in Eq. (4.9), first we sample 100000 points and the corresponding point normals from the ground truth mesh, and input to our DPSR at a grid resolution of 128^3.

[5] https://github.com/ThibaultGROUEIX/AtlasNet

Dataset	Method	Chamfer-L_1 (↓)	F-Score (↑)	Normal C. (↑)	Time (s)
Thingi10K	IGR [64]	0.440	0.505	0.692	1842.3
	Point2Mesh [71]	0.109	0.656	0.806	3714.7
	SPSR [97]	0.223	0.787	0.896	9.3
	Ours	**0.054**	**0.940**	**0.947**	370.1
SRB	IGR [64]	0.178	0.755	–	1847.6
	Point2Mesh [71]	0.116	0.648	–	4707.9
	SPSR [97]	0.232	0.735	–	9.2
	Ours	**0.076**	**0.830**	–	326.0
D-FAUST	IGR [64]	0.235	0.805	0.911	1857.2
	Point2Mesh [71]	0.071	0.855	0.905	3678.7
	SPSR [97]	0.044	**0.966**	**0.965**	4.3
	Ours	**0.043**	**0.966**	0.959	379.9

Table 4.2: ***Optimization-based 3D Reconstruction.*** *Quantitative comparison on 3 datasets. Normal consistency cannot be evaluated on SRB as the provided GTs are unoriented point clouds. Optimization time is evaluated on a single GTX 1080Ti GPU.*

4.3.1 Optimization-based 3D Reconstruction

In this part, we investigate whether our method can be used for the single-object surface reconstruction task from unoriented point clouds or scans. We consider three different types of 3D inputs: point clouds sampled from synthetic meshes [279] with Gaussian noise, real-world scans [236], and high-resolution raw scans of humans with comparably little noise [10].

Table 4.2 shows that our method achieves superior performance compared to both classical methods and network-based approaches. Note that the objects considered in this task are challenging due to their complex geometry, thin structures, noisy and incomplete observations. While some of the baseline methods fail completely on these challenging objects, our method achieves robust performance across all datasets.

In particular, Fig. 4.2, Fig. 4.3, and Fig. 4.4 show that IGR occasionally creates meshes in free space, as this is not penalized by its optimization objective when point clouds are unoriented. Both, Point2Mesh

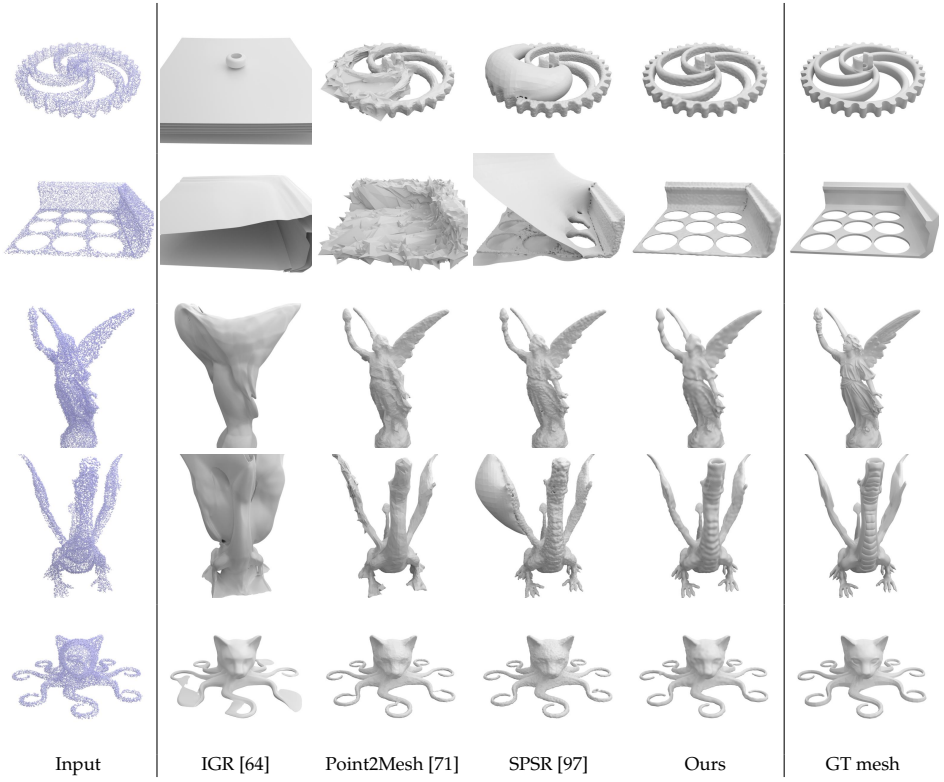


Figure 4.2: ***Optimization-based 3D Reconstruction on Thingi10K Dataset [279].*** *Input point clouds are downsampled for visualization.*

and our method alleviate this problem by optimizing for the Chamfer distance between the estimated mesh and the input point clouds. However, Point2Mesh requires an initial mesh as input of which the topology cannot be changed during optimization. Thus, it relies on SPSR to provide an initial mesh for objects with a genus larger than 0 and suffers from inaccurate initialization [71]. Furthermore, compared to both IGR and Point2Mesh, our method converges faster.

While SPSR is even more efficient, it suffers from incorrect normal es-

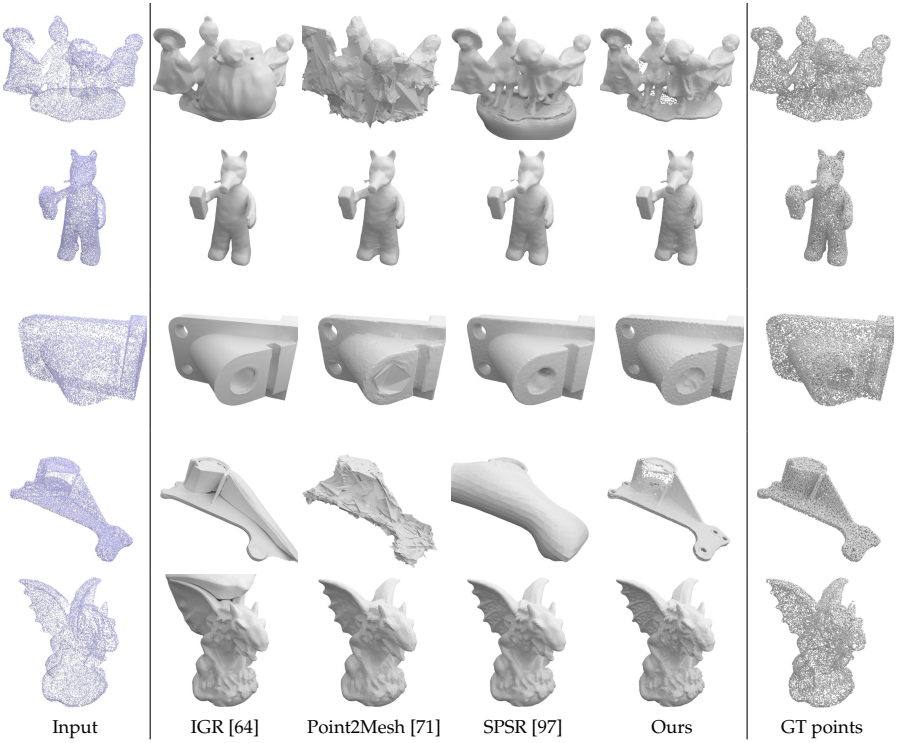


Figure 4.3: ***Optimization-based 3D Reconstruction on SRB Dataset [236].*** *Input point clouds are downsampled for visualization.*

timation on noisy input point clouds, which is a non-trivial task on its own. In contrast, our method demonstrates more robust behavior as we optimize points and normals guided by the Chamfer distance. Note that in this *single* object reconstruction task, our method is not able to complete large unobserved regions (e.g., the bottom of the person's feet in Fig. 4.4 is unobserved and hence not completed). This limitation can be addressed using learning-based object-level reconstruction as discussed next.

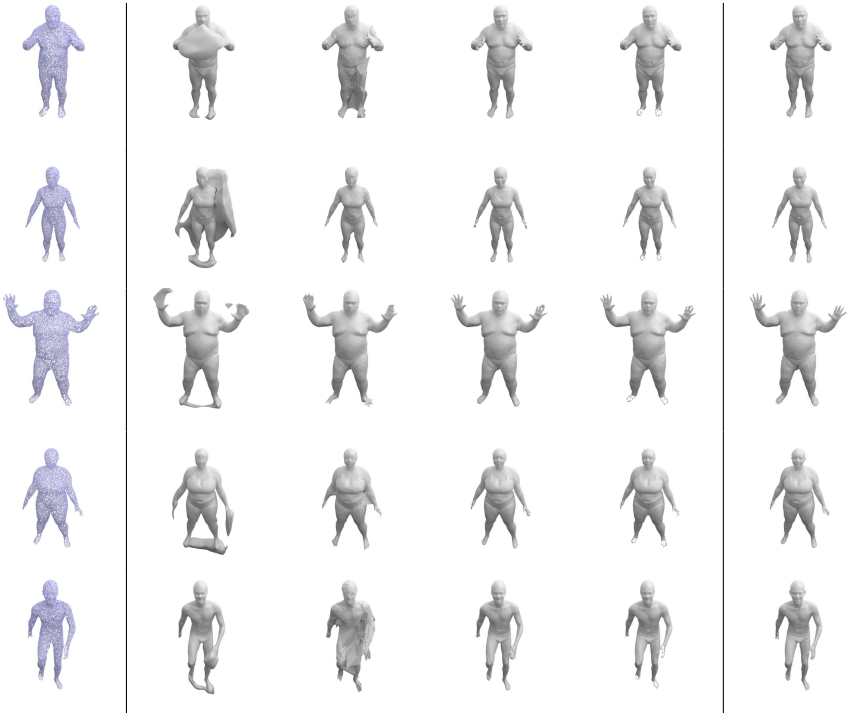


Figure 4.4: ***Optimization-based 3D Reconstruction on D-FAUST Dataset [10].*** *Input point clouds are downsampled for visualization.*

4.3.2 Ablation Study for Optimization-based Setting

Ablation Study of Point Resampling Strategy. In Table 4.3 and Fig. 4.5, we compare the reconstructed shapes with and without the proposed resampling strategy. Our method is able to produce reasonable reconstructions even without the resampling strategy, but the shapes are much noisier. Since we directly optimize the source point positions and normals without any additional constraints, the optimized point clouds can be unevenly distributed as

Dataset	Method	Chamfer-L_1 (↓)	F-Score (↑)	Normal C. (↑)
Thingi10K	Ours (w/o resampling)	0.061	0.897	0.902
	Ours	**0.053**	**0.941**	**0.947**
DGP	Ours (w/o resampling)	0.077	0.813	–
	Ours	**0.067**	**0.848**	–
D-FAUST	Ours (w/o resampling)	0.044	0.964	0.952
	Ours	**0.043**	**0.965**	**0.959**

Table 4.3: ***Ablation Study of Resampling Strategy***. *On all datasets, our resampling strategy leads to improved results. For D-FAUST, the increase is the lowest because the supervision point clouds are noise free. Note that normal consistency cannot be evaluated on SRB as this dataset provides only unoriented point clouds.*

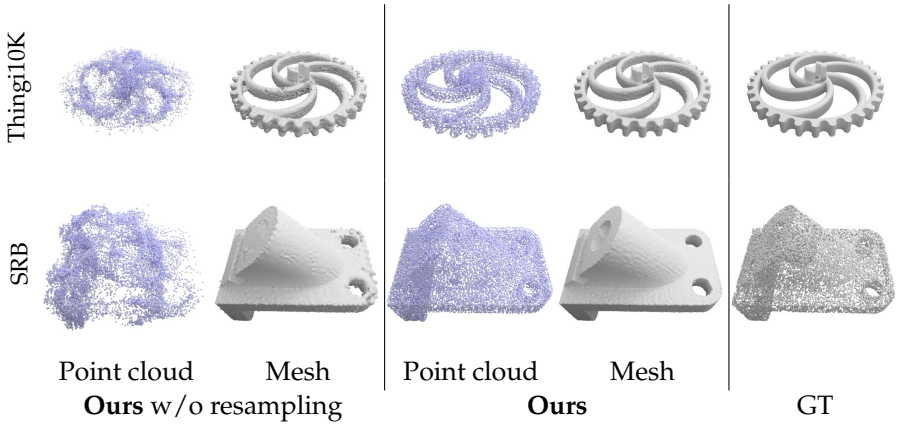


Figure 4.5: ***Ablation Study of Resampling Strategy***. *We show the optimized point cloud and the reconstructed mesh without and with the resampling strategy. Using the point resampling strategy leads to a more uniformly distributed point cloud and better shape reconstruction.*

shown in Fig. 4.5. This limits the representational expressivity of the point clouds given the same number of points. The resampling strategy acts as a regularization to enforce a uniformly distributed point cloud, which leads to better surface reconstruction.

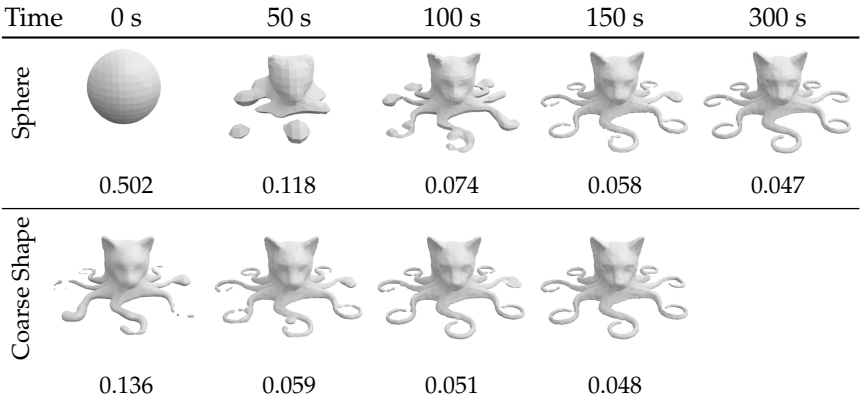


Figure 4.6: ***Ablation Study of Different Geometric Initialization under Optimization Setting****. We compare the reconstructions of SAP initialized from a sphere and the coarse geometry. The number below each image indicates the Chamfer Distance to GT mesh.*

Ablation Study of Geometric Initialization. As mentioned in Sec. 4.1, it is easy to initialize SAP with a given geometry such as template shapes or noisy observations. Here we provide further discussions with some experiments under both optimization and learning settings of SAP.

From all the results that we have shown so far under the optimization setting, we chose to start from a sphere, since we intended to demonstrate that if our method is able to produce decent 3D reconstruction even starting from a sphere, we can also faithfully reconstruct from a coarse or noisy shape since it is a simpler task, and it should converge faster. In Fig. 4.6, we show a comparison between initialization from a sphere and coarse shape. As can be observed, when starting from points and normals sampled from a coarse shape, our method indeed converges faster. In terms of accuracy, both results are equivalent, i.e., there is no better local minima attained by the optimization process.

Iterations	10K	50K	100K	200K	Best
ConvONet [171]	0.082	0.058	0.055	0.050	0.044
Ours	**0.041**	**0.036**	**0.035**	**0.034**	**0.034**

Table 4.4: ***Training Progress****. We show the Chamfer distance at different training iterations evaluated in the Shapenet test set with 3K input points ((noise level=0.005). Our method uses geometric initialization and converges much faster than ConvONet.*

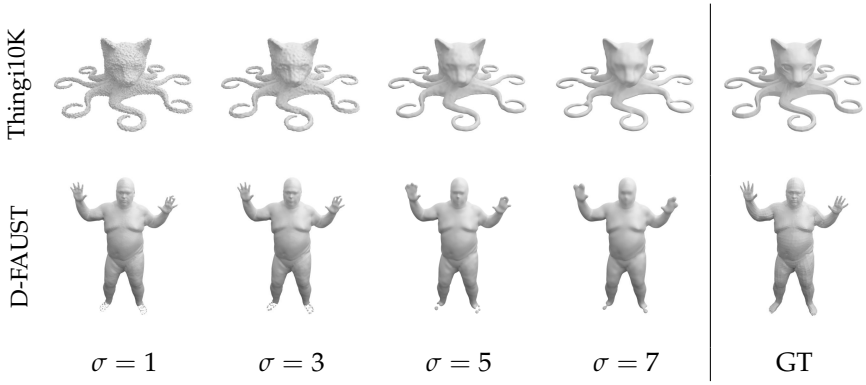


Figure 4.7: ***Ablation Study of the Gaussian Smoothing Parameter*** σ*. Low* σ *preserves details better but is prone to noise, while high* σ *results in smooth shapes, but also leads to the loss of detail.*

Why Do We Need a Gaussian? The Gaussian serves as a regularizer for the smoothness of the solved implicit function. Not using a Gaussian is equivalent to using a Gaussian kernel with $\sigma = 0$. Fig. 4.7 motivates the use of our sigma parameter.

Why Use a Gaussian in the Spectral Domain? First, the FFT of a Gaussian remains a Gaussian. Second, convolution of a Gaussian in the physical domain is equivalent to a dot product with a Gaussian in the spectral domain and a dot product in the spectral domain is more efficient than convolution in the physical domain: $\mathcal{O}(N \log N)$ vs $\mathcal{O}(N^2)$, where n is the resolution of a regular grid and $N = n^3$.

Ablation Study of the Gaussian Smoothing Parameter σ. We study the effect of the Gaussian smoothing parameter σ at a resolution of 256^3. As visualized in Fig. 4.7, we can obtain faithful reconstructions given different σ values. Nevertheless, we can notice that lower σ can preserve details better but also is prone to noise, while high σ results in smooth shapes but can also lead to the loss of details. In practice, σ can be chosen according to the noise level of the target point cloud. In the results depicted above, we choose $\sigma = 3$ for SRB (Fig. 4.3) and D-FAUST dataset (Fig. 4.4) and $\sigma = 5$ for Thingi10K dataset (Fig. 4.2).

4.3.3 Learning-based Reconstruction

To analyze whether our proposed differentiable Poisson solver is also beneficial for learning-based reconstruction, we evaluate our method on the single object reconstruction task using noise and outlier-augmented point clouds from ShapeNet as input to our method. We investigate the performance for three different noise levels: (a) Gaussian noise with zero mean and standard deviation 0.005, (b) Gaussian noise with zero mean and standard deviation 0.025, (c) 50% points have the same noise as in a) and the other 50% points are outliers uniformly sampled inside the unit cube.

Fig. 4.8 and Table 4.5 show our results. Compared to the baselines, our method achieves similar or better results on all three metrics. The results show that, in comparison to directly using Chamfer loss on point positions and L1 loss on point normals, our DPSR loss can produce better reconstructions in all settings as it directly supervises the indicator grid which implicitly determines the surface through the Poisson equation. SPSR fails when the noise level is high or when there are outliers in the input point cloud. We achieve significantly better performances than other representations such as point clouds, meshes, voxel grids and patches. Moreover, we find that our method is robust to strong outliers.

	(a) Noise=0.005			(b) Noise=0.025			(c) Noise=0.005, Outliers=50%			
	Chamfer-L_1	F-Score	Normal C.	Chamfer-L_1	F-Score	Normal C.	Chamfer-L_1	F-Score	Normal C.	Runtime
SPSR [97]	0.298	0.612	0.772	0.499	0.324	0.604	1.317	0.164	0.636	-
PSGN [52]	0.147	0.259	-	0.151	0.247	-	0.736	0.007	-	0.010 s
3D-R2N2 [40]	0.172	0.400	0.715	0.173	0.418	0.710	0.202	0.387	0.709	0.015 s
AtlasNet [65]	0.093	0.708	0.855	0.117	0.527	0.821	1.822	0.057	0.609	0.025 s
ConvONet [171]	0.044	0.942	0.938	0.066	0.849	0.913	0.052	0.916	0.929	0.327 s
Ours (w/o $\mathcal{L}_{\text{DPSR}}$)	0.044	0.942	0.935	0.067	0.841	0.907	0.085	0.819	0.903	0.064 s
Ours	**0.034**	**0.975**	**0.944**	**0.054**	**0.896**	**0.917**	**0.038**	**0.959**	**0.936**	0.064 s

Table 4.5: ***3D Reconstruction from Point Clouds on ShapeNet.*** *Quantitative comparison between our learning-based method and baselines on the ShapeNet dataset (mean over 13 classes).*

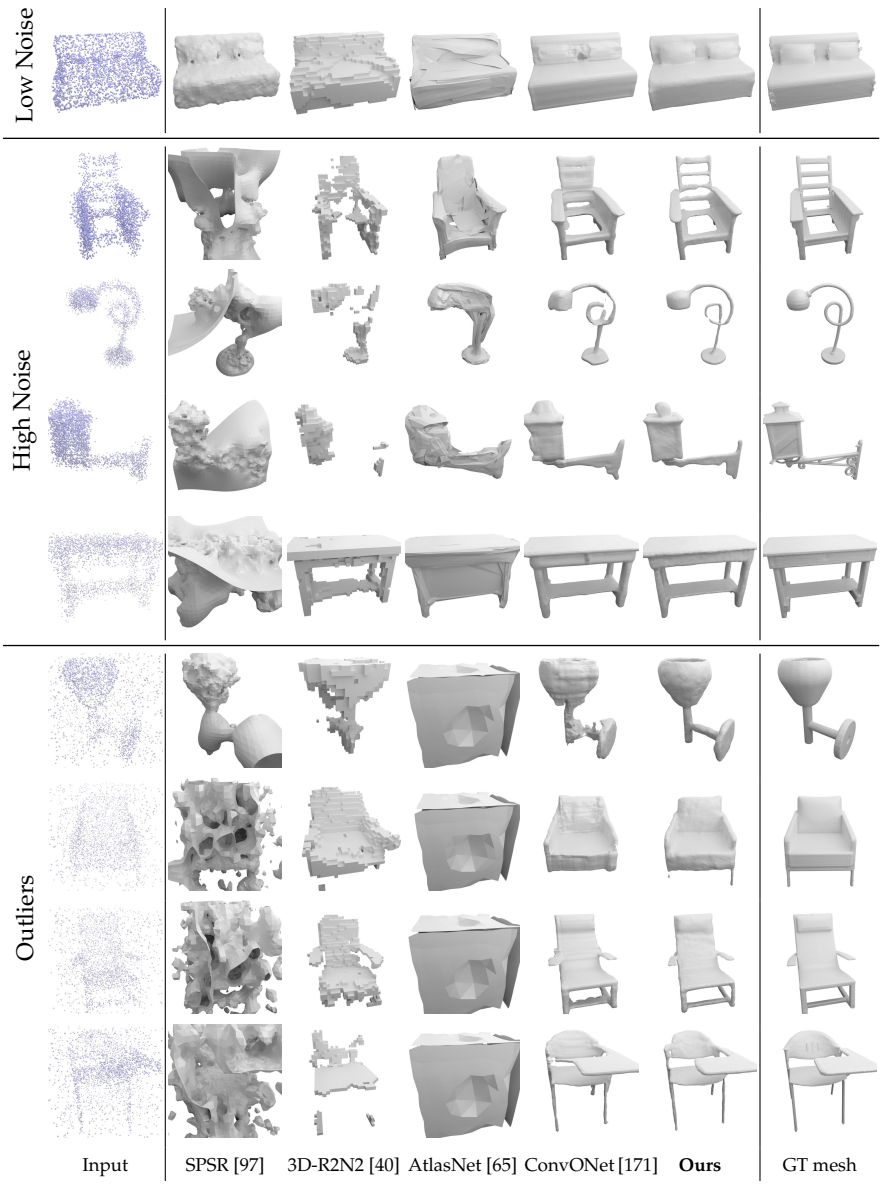


Figure 4.8: ***3D Reconstruction from Point Clouds on ShapeNet.*** *Comparison of SAP to baselines on 3 different setups.*

Table 4.5 also reports the runtime for setting (a) for all GPU-accelerated methods using a single NVIDIA GTX 1080Ti GPU, averaged over all objects of the ShapeNet test set. The baselines [40,52,65] demonstrate fast inference time but suffer in terms of reconstruction quality while the neural implicit model [171] attains high quality reconstructions but suffers from slow inference. In contrast, our method is able to produce competitive reconstruction results at reasonably fast inference time. In addition, since ConvONet and our method share a similar reconstruction pipeline, we provide a more detailed breakdown of the runtime at a resolution of 128^3 and 256^3 voxels in Table 4.6. We use the default setup from ConvONet, except that we use the Marching Cubes implementation from [218] for both ConvONet and ours for consistency. As we can see from Table 4.6, the difference in terms of point encoding and Marching Cubes is marginal, but we gain more than $20\times$ speed-up over ConvONet in evaluating the indicator grid. In total, we are roughly $5\times$ and $8\times$ faster regarding the total inference time at a resolution of 128^3 and 256^3 voxels, respectively.

4.3.4 Ablation Study for Learning-based Setting

we first investigate different architecture choices in the context of learning-based reconstruction. We conduct our ablation experiments on ShapeNet for the third setup (most challenging).

Number of Offsets. From Table 4.6 (left) we notice that predicting more offsets per input point leads to better performance. This can be explained by the fact that with more points near the object surface, geometric details can be better preserved.

Point Cloud Encoder. In Table 4.6 (right), we compare two different point encoder architectures proposed in [171]: a 2D encoder using 3 canonical planes at a resolution of 64^2 pixels and a 3D encoder using a feature volume with a resolution of 32^3 voxels. We find that the 3D

	128^3				256^3			
	Enc.	Grid	MC	Total	Enc.	Grid	MC	Total
ConvONet	**0.010**	0.280	**0.037**	0.327	**0.010**	3.798	**0.299**	4.107
Ours	0.013	**0.012**	0.039	**0.064**	0.019	**0.140**	0.374	**0.533**

	Chamfer	F-Score	NormalC
Offset 1×	0.041	0.952	0.928
Offset 3×	0.039	0.958	0.934
Offset 5×	0.039	0.957	0.934
Offset 7×	**0.038**	**0.959**	**0.936**
2D Enc.	0.043	0.939	0.928
3D Enc.	**0.038**	**0.959**	**0.936**

Table 4.6: ***Ablation Study for Learning-based Setting.*** *Top: Runtime breakdown (encoding, grid evaluation, marching cubes) for ConvONet vs. ours in seconds. Bottom: Ablation over the number of offsets and 2D vs. 3D encoders.*

encoder works best in this setting and hypothesize that this is due to the representational alignment with the 3D indicator grid.

How SAP Handles Noise and Outlier.

Here we visualize how our trained models handle noise and outliers during inference. For Gaussian noises present in input point clouds, we can see from the top row of Fig. 4.9 that, compared to the input point cloud, the updated SAP points are densified because we predict $k = 7$ offsets per input point. More importantly, all SAP points are located roughly on the surface, which leads to enhanced reconstruction quality.

We also visualize how SAP handles outlier points at the bottom row of Fig. 4.9. The arrows' length represents the magnitude of the predicted normals. There are two interesting observations: a) A large amount of outlier points in the input are moved near to the surface. b) Some outlier points still remain outliers. For these points, the network learns to predict normals with a very small magnitude/norm

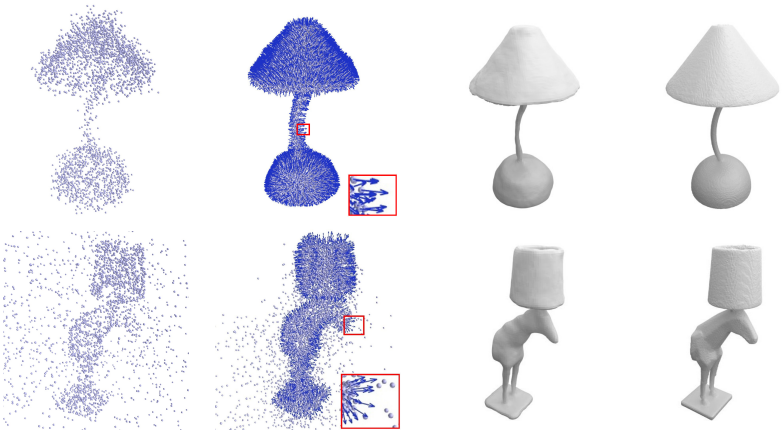


Figure 4.9: ***Visualization of SAP Handling Noise and Outliers.*** *The length of arrows represents the magnitude of normals. SAP point clouds are downsampled for better visualization.*

as shown in the zoom-in view (we do not normalize the point normals to unit length). In this way, those outlier points are "muted" when being passed to the DPSR layer such that they do not contribute to the final reconstruction.

4.4 Conclusion and Discussion

We introduce Shape-As-Points, a novel shape representation which is lightweight, interpretable and produces watertight meshes efficiently. We demonstrate its effectiveness for 3D surface reconstruction from unoriented point clouds in both optimization-based and learning-based settings.

Limitations and Future Works. First, our method is currently limited to small scenes due to the cubic memory requirements with

respect to the indicator grid resolution. We believe that processing scenes in a sliding-window manner and space-adaptive data structures (e.g., octrees) will enable extending our method to larger scenes. Moreover, we currently only consider input as point clouds. A natural next step is to benefit the task of multi-view reconstruction with our DPSR. One recent attempt [125] has shown our DPSR can indeed significantly speed up the reconstruction process of human body from only RGB images.

CHAPTER 5

SLAM with Scalable Scene Representations

In Chapter 3, we showed the aptitude of neural implicit representations for detailed 3D reconstructions. Building on this, Chapter 4 optimized inference speed via the integration of a differentiable solver. The models we discussed so far are confined to reconstructing shapes solely from point clouds. To bridge this gap and cater to real-world applications—where typically only RGB-D sequences serve as input—we introduce in this chapter a dense SLAM system. This system pioneers a hierarchical scene representation, incorporating multi-level local information, and is designed for joint camera tracking and 3D reconstruction, especially of expansive indoor scenes.

5.1 Introduction

Dense visual Simultaneous Localization and Mapping (SLAM) is a fundamental problem in 3D computer vision with many applications in autonomous driving, indoor robotics, mixed reality, etc. In order to make a SLAM system truly useful for real-world applications, the following properties are essential. First, we desire the SLAM system to be real-time. Next, the system should have the ability to make reasonable predictions for regions without observations. Moreover, the system should be able to scale up to large scenes. Last but not least, it is crucial to be robust to noisy or missing observations.

In the scope of real-time dense visual SLAM system, many methods have been introduced for RGB-D cameras in the past years. Traditional dense visual SLAM systems [156, 188, 233, 234] fulfil the real-time requirement and can be used in large-scale scenes, but they are unable to make plausible geometry estimation for unobserved regions. On the other hand, learning-based SLAM approaches [9, 43, 201,275] attain a certain level of predictive power since they typically train on task-specific datasets. Moreover, learning-based methods tend to better deal with noises and outliers. However, these methods are typically only working in small scenes with multiple objects. Recently, Sucar et al. [200] applied a neural implicit representation in the real-time dense SLAM system (called iMAP), and they showed decent tracking and mapping results for room-sized datasets. Nevertheless, when scaling up to larger scenes, e.g., an apartment consisting of multiple rooms, significant performance drops are observed in both the dense reconstruction and camera tracking accuracy.

The key limiting factor of iMAP [200] stems from its use of a single multi-layer perceptron (MLP) to represent the entire scene, which can only be updated globally with every new, potentially partial RGB-D observations. In contrast, our work in Chapter 3 and some other recent efforts [203, 270] demonstrate that establishing grid-based features can help to preserve geometric details and enable re-

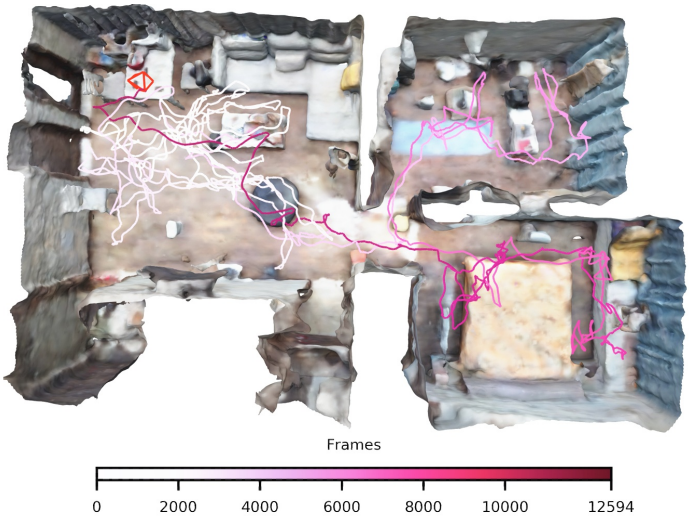


Figure 5.1: ***Multi-room Apartment 3D Reconstruction using NICE-SLAM****. A hierarchical feature grid jointly encodes geometry and color information and is used for both mapping and tracking. We depict the final mesh and camera tracking trajectory.*

constructing complex scenes, but these are offline methods without real-time capability.

In this chapter, we seek to combine the strengths of hierarchical scene representations with those of neural implicit representations for the task of dense RGB-D SLAM. To this end, we introduce *NICE-SLAM*, a dense RGB-D SLAM system that can be applied to large-scale scenes while preserving the predictive ability. Our key idea is to represent the scene geometry and appearance with hierarchical feature grids and incorporate the inductive biases of neural implicit decoders pre-trained at different spatial resolutions. With the rendered depth and color images from the occupancy and color decoder outputs, we can optimize the features grids only within the viewing frustum by minimizing the re-rendering losses. We perform extensive evaluations on a wide variety of indoor RGB-D sequences and demonstrate the

scalability and predictive ability of our method. Overall, we make the following contributions in this chapter:

- We present NICE-SLAM, a dense RGB-D SLAM system that is real-time capable, scalable, predictive, and robust to various challenging scenarios.
- The core of NICE-SLAM is a hierarchical, grid-based neural implicit encoding. In contrast to global neural scene encodings, this representation allows for local updates, which is a prerequisite for large-scale approaches.
- We conduct extensive evaluations on various datasets which demonstrate competitive performance in both mapping and tracking.

5.2 Related Work

5.2.1 Dense Visual SLAM

Most modern methods for visual SLAM follow the overall architecture introduced in the seminal work by Klein et al. [104], decomposing the task into mapping and tracking. The map representations can be generally divided into two categories: view-centric and world-centric. The first anchors 3D geometry to specific keyframes, often represented as depth maps in the dense setting. One of the early examples of this category was DTAM [156]. Because of its simplicity, DTAM has been widely adapted in many recent learning-based SLAM systems. For example, [216,277] regress both depth and pose updates. DeepV2D [212] similarly alternates between regressing depth and pose estimation but uses test-time optimization. BA-Net [207] and DeepFactors [43] simplify the optimization problem by using a set of basis depth maps. There are also some methods, e.g. CodeSLAM [9], SceneCode [275] and NodeSLAM [201], which optimize a latent representation that decodes into the keyframe or object depth maps. DROID-SLAM [213] uses regressed optical flow to define geometrical residuals for its refinement. TANDEM [106]

combines multi-view stereo with DSO [50] for a real-time dense SLAM system. On the other hand, the world-centric map representation anchors the 3D geometry in uniform world coordinates, and can be further divided into surfels [188, 234] and voxel grids, typically storing occupancies or TSDF values [42]. Voxel grids have been used extensively in RGB-D SLAM, e.g. KinectFusion [155] among other works [14, 45, 94, 163].

In our proposed pipeline we also adopt the voxel-grid representation. In contrast to previous SLAM approaches, we store implicit latent codes of the geometry and directly optimize them during mapping. This richer representation allows us to achieve more accurate geometry at lower grid resolutions.

5.2.2 SLAM with Neural Implicit Representations

Recently, neural implicit representations demonstrated promising results for object geometry representation [29, 130, 144, 161, 165, 166, 170, 185, 225, 249, 255, 256], scene completion [20, 90, 171], novel view synthesis [141, 149, 177, 268] and also generative modelling [22, 158, 159, 190]. A few recent papers [5, 12, 37, 153, 203, 231, 252] attempt to predict scene-level geometry with RGB-(D) inputs, but they all assume given camera poses. Another set of works [123, 230, 259] tackle the problem of camera pose optimization, but they need a rather long optimization process, which is not suitable for real-time applications.

The most related work to our method in this chapter is iMAP [200]. Given an RGB-D sequence, they introduce a real-time dense SLAM system that uses a single multi-layer perceptron (MLP) to compactly represent the entire scene. Nevertheless, due to the limited model capacity of a single MLP, iMAP fails to produce detailed scene geometry and accurate camera tracking, especially for larger scenes. In contrast, we provide a scalable solution akin to iMAP, that combines learnable latent embeddings with a pretrained continuous implicit decoder. In this way, our method can reconstruct complex geometry

and predict detailed textures for larger indoor scenes, while maintaining much less computation and guaranteeing faster convergence. Notably, the works [90, 171] also combine traditional grid structures with learned feature representations for scalability, but neither of them is real-time capable. Moreover, DI-Fusion [82] also optimizes a feature grid given an RGB-D sequence, but their reconstruction often contain holes and their camera tracking is not robust for the pure surface rendering loss.

5.3 Method

We provide an overview of our method in Fig. 5.2. We represent the scene geometry and appearance using four feature grids and their corresponding decoders (Sec. 5.3.1). We trace the viewing rays for every pixel using the estimated camera calibration. By sampling points along a viewing ray and querying the network, we can render both depth and color values of this ray (Sec. 5.3.2). By minimizing the re-rendering losses for depth and color, we are able to optimize both the camera pose and the scene geometry in an alternating fashion (Sec. 5.3.3). We also discuss how to initialize the feature grids (Sec. 5.3.4) and select keyframes (Sec. 5.3.5).

5.3.1 Hierarchical Scene Representation

We now introduce our hierarchical scene representation that combines multi-level grid features with pre-trained decoders for occupancy predictions. The geometry is encoded into three feature grids ϕ_θ^l and their corresponding MLP decoders f^l, where $l \in \{0, 1, 2\}$ is referred to coarse, mid and fine-level scene details. In addition, we also have a single feature grid ψ_ω and decoder $\mathbf{g}_\omega$ to model the scene appearance. Here θ and ω indicate the optimizable parameters for geometry and color, i.e., the features in the grid and the weights in the color decoder.

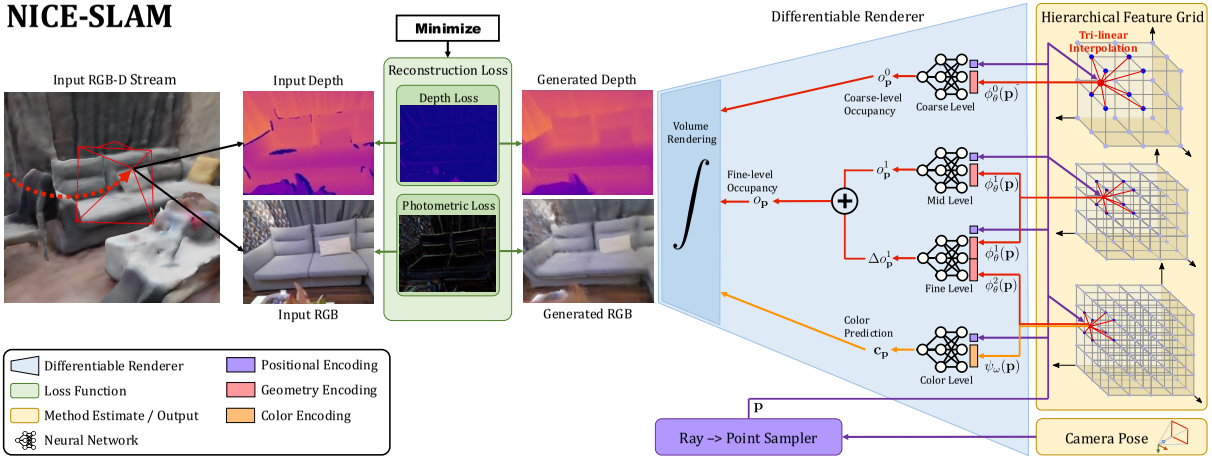


Figure 5.2: ***System Overview.*** *Our method takes an RGB-D image stream as input and outputs both the camera pose as well as a learned scene representation in form of a hierarchical feature grid. From right-to-left, our pipeline can be interpreted as a generative model which renders depth and color images from a given scene representation and camera pose. At test time we estimate both the scene representation and camera pose by solving the inverse problem via backpropagating the image and depth reconstruction loss through a differentiable renderer (left-to-right). Both entities are estimated within an alternating optimization:* ***Mapping:*** *The backpropagation only updates the hierarchical scene representation;* ***Tracking:*** *The backpropagation only updates the camera pose. For better readability we joined the fine-scale grid for geometry encoding with the equally-sized color grid and show them as one grid with two attributes (red and orange).*

Mid-&Fine-level Geometric Representation. The observed scene geometry is represented in the mid- and fine-level feature grids. In the reconstruction process we use these two grids in a coarse-to-fine approach where the geometry is first reconstructed by optimizing the mid-level feature grid, followed by a refinement using the fine-level. In the implementation we use voxel grids with side lengths of 32cm and 16cm respectively, except for TUM RGB-D [199] we use 16cm and 8cm. For the mid-level, the features are directly decoded into occupancy values using the associated MLP f^1. For any point $\mathbf{p} \in \mathbb{R}^3$, we get the occupancy as

$$o^1_{\mathbf{p}} = f^1(\mathbf{p}, \phi^1_\theta(\mathbf{p})), \tag{5.1}$$

where $\phi^1_\theta(\mathbf{p})$ denotes that the feature grid is tri-linearly interpolated at the point $\mathbf{p}$. The relatively low-resolution allow us to efficiently optimize the grid features to fit the observations. To capture smaller high-frequency details in the scene geometry we add in the fine-level features in a residual manner. In particular, the fine-level feature decoder takes as input both the corresponding mid-level feature and the fine-level feature and outputs an offset from the mid-level occupancy, i.e.,

$$\Delta o^1_{\mathbf{p}} = f^2(\mathbf{p}, \phi^1_\theta(\mathbf{p}), \phi^2_\theta(\mathbf{p})), \tag{5.2}$$

where the final occupancy for a point is given by

$$o_{\mathbf{p}} = o^1_{\mathbf{p}} + \Delta o^1_{\mathbf{p}}. \tag{5.3}$$

Note that we fix the pre-trained decoders f^1 and f^2, and only optimize the feature grids ϕ^1_θ and ϕ^2_θ throughout the entire optimization process. We demonstrate that this helps to stabilize the optimization and learn consistent geometry.

Coarse-level Geometric Representation. The coarse-level feature grid aims to capture the high-level geometry of the scene (e.g., walls, floor, etc), and is optimized independently from the mid- and fine-level. The goal of the coarse-grid is to be able to predict approximate occupancy values outside of the observed geometry (which is

encoded in the mid/fine-levels), even when each coarse voxel has only been partially observed. For this reason we use a very low resolution, with a side-length of 2m in the implementation. Similarly to the mid-level grid, we decode directly into occupancy values by interpolating the features and passing through the MLP f^0, i.e.,

$$o^0_{\mathbf{p}} = f^0(\mathbf{p}, \phi^0_\theta(\mathbf{p})). \tag{5.4}$$

During tracking, the coarse-level occupancy values are only used for predicting previously unobserved parts. This *forecasted* geometry allows us to track even when a large portion of the current image is previously unseen.

Why is the Mid-level Output not a Residual to the Coarse-level Output? The coarse grid has a significantly larger voxel size (side of > 1 meter) than the mid and fine levels, so updating the coarse-level feature would affect a large area. To ensure small local updates for efficiency, we disconnect coarse level from mid and fine levels, and only use coarse level for prediction.

Pre-training Feature Decoders. In our framework we use three different fixed MLPs to decode the grid features into occupancy values. The coarse and mid-level decoders are pre-trained as part of our 3D model in ConvONet [171] (Chapter 3), which consists of a CNN encoder and an MLP decoder. We train both the encoder/decoder using the binary cross-entropy loss between the predicted and the ground-truth value, same as in [171]. After training, we only use the decoder MLP, as we will directly optimize the features to fit the observations in our reconstruction pipeline. In this way the pre-trained decoder can leverage resolution-specific priors learned from the training set, when decoding our optimized features.

The same strategy is used to pre-train the fine-level decoder, except that we simply concatenate the feature $\phi^1_\theta(\mathbf{p})$ from the mid-level together with the fine-level feature $\phi^2_\theta(\mathbf{p})$ before inputting to the decoder.

Color Representation. While we are mainly interested in the scene geometry, we also encode the color information allowing us to render RGB images which provides additional signals for tracking. To encode the color in the scene, we apply another feature grid ψ_ω and decoder $\mathbf{g}_\omega$:

$$\mathbf{c_p} = \mathbf{g}_\omega(\mathbf{p}, \psi_\omega(\mathbf{p})), \tag{5.5}$$

where ω indicates learnable parameters during optimization. Different from the geometry that has strong prior knowledge, we empirically found that jointly optimizing the color features ψ_ω and decoder $\mathbf{g}_\omega$ improves the tracking performance (c.f. Table 5.7). Note that, similarly to iMAP [200], this can lead to forgetting problems and the color is only consistent locally. If we want to visualize the color for the entire scene, it can be optimized globally as a post-processing step.

Network Design. For all MLP decoders, we use a hidden feature dimension of 32 and 5 fully-connected blocks [161, 171]. Except for the coarse-level geometric representation, we apply a learnable Gaussian positional encoding [200,205] to $\mathbf{p}$ before input to MLP decoders. We observe this allows the discovery of high-frequency details for both geometry and appearance.

5.3.2 Depth and Color Rendering

Inspired by the recent success of volume rendering in NeRF [149], we propose to also use a differentiable rendering process which integrates the predicted occupancy and colors from our scene representation in Sec. 5.3.1.

Given camera intrinsic parameters and current camera pose, we can calculate the viewing direction $\mathbf{r}$ of a pixel coordinate. We first sample along this ray N_{strat} points for stratified sampling, and also uniformly sample N_{imp} points near to the depth[1]. In total we sam-

[1] We empirically define the sampling interval as $\pm 0.05D$, where D is the depth value of the current ray.

ple $N = N_{\text{strat}} + N_{\text{imp}}$ points for each ray. More formally, let $\mathbf{p}_i = \mathbf{o} + d_i\mathbf{r}, i \in \{1, \cdots, N\}$ denote the sampling points on the ray $\mathbf{r}$ given the camera origin $\mathbf{o}$, and d_i corresponds to the depth value of $\mathbf{p}_i$ along this ray. For every point $\mathbf{p}_i$, we can calculate their coarse-level occupancy probability $o^0_{\mathbf{p}_i}$, fine-level occupancy probability $o_{\mathbf{p}_i}$, and color value $\mathbf{c}_{\mathbf{p}_i}$ using Eq. (5.4), Eq. (5.3), and Eq. (5.5). Similar to [165], we model the ray termination probability at point $\mathbf{p}_i$ as $w^c_i = o^0_{\mathbf{p}_i} \prod_{j=1}^{i-1}(1 - o^0_{\mathbf{p}_j})$ for coarse level, and $w^f_i = o_{\mathbf{p}_i} \prod_{j=1}^{i-1}(1 - o_{\mathbf{p}_j})$ for fine level.

Finally for each ray, the depth at both coarse and fine level, and color can be rendered as:

$$\hat{D}^c = \sum_{i=1}^{N} w^c_i d_i, \quad \hat{D}^f = \sum_{i=1}^{N} w^f_i d_i, \quad \hat{I} = \sum_{i=1}^{N} w^f_i \mathbf{c}_i. \tag{5.6}$$

Moreover, we also calculate depth variances along the ray:

$$\hat{D}^c_{var} = \sum_{i=1}^{N} w^c_i (\hat{D}^c - d_i)^2 \quad \hat{D}^f_{var} = \sum_{i=1}^{N} w^f_i (\hat{D}^f - d_i)^2. \tag{5.7}$$

5.3.3 Mapping and Tracking

In this section, we provide details on the optimization of the scene geometry θ and appearance ω parameters of our hierarchical scene representation, and of the camera poses.

Mapping. To optimize the scene representation mentioned in Sec. 5.3.1, we uniformly sample total M pixels from the current frame and the selected keyframes. Next, we perform optimization in a staged fashion to minimize the geometric and photometric losses.

The geometric loss is simply an L_1 loss between the observations and predicted depths at coarse or fine level:

$$\mathcal{L}^l_g = \frac{1}{M} \sum_{m=1}^{M} \left| D_m - \hat{D}^l_m \right|, \quad l \in \{c, f\}. \tag{5.8}$$

The photometric loss is also an L_1 loss between the rendered and observed color values for M sampled pixel:

$$\mathcal{L}_p = \frac{1}{M} \sum_{m=1}^{M} \left| I_m - \hat{I}_m \right| . \tag{5.9}$$

At the first stage, we optimize only the mid-level feature grid ϕ_θ^1 using the geometric loss $\mathcal{L}_g^f$ in Eq. (5.8). Next, we jointly optimize both the mid and fine-level $\phi_\theta^1, \phi_\theta^2$ features with the same fine-level depth loss $\mathcal{L}_g^f$. Finally, we conduct a local bundle adjustment (BA) to jointly optimize feature grids at all levels, the color decoder, as well as the camera extrinsic parameters $\{\mathbf{R}_i, \mathbf{t}_i\}$ of K selected keyframes:

$$\min_{\theta, \omega, \{\mathbf{R}_i, \mathbf{t}_i\}} \left(\mathcal{L}_g^c + \mathcal{L}_g^f + \lambda_p \mathcal{L}_p \right) , \tag{5.10}$$

where λ_p is the loss weighting factor.

This multi-stage optimization scheme leads to better convergence as the higher-resolution appearance and fine-level features can rely on the already refined geometry coming from mid-level feature grid.

Note that we parallelize our system in three threads to speed up the optimization process: one thread for coarse-level mapping, one for mid-&fine-level geometric and color optimization, and last one for camera tracking.

Camera Tracking. In addition to optimizing the scene representation, we also run in parallel camera tracking to optimize the camera poses of the current frame, i.e., rotation and translation $\{\mathbf{R}, \mathbf{t}\}$. To this end, we sample M_t pixels in the current frame and apply the same photometric loss in Eq. (5.9) but use a modified geometric loss:

$$\mathcal{L}_{g\_var} = \frac{1}{M_t} \sum_{m=1}^{M_t} \frac{\left| D_m - \hat{D}_m^c \right|}{\sqrt{\hat{D}_{var}^c}} + \frac{\left| D_m - \hat{D}_m^f \right|}{\sqrt{\hat{D}_{var}^f}}. \tag{5.11}$$

The modified loss down-weights less certain regions in the reconstructed geometry [200, 254], e.g., object edges. The camera tracking

is finally formulated as the following minimization problem:

$$\min_{\mathbf{R},\mathbf{t}} \left(\mathcal{L}_{g\_var} + \lambda_{pt} \mathcal{L}_p \right). \tag{5.12}$$

The coarse feature grid is able to perform short-range predictions of the scene geometry. This extrapolated geometry provides a meaningful signal for the tracking as the camera moves into previously unobserved areas. Making it more robust to sudden frame loss or fast camera movement. We provide experiments in the ablation study in Sec. 5.4.4.

Robustness to Dynamic Objects. To make the optimization more robust to dynamic objects during tracking, we filter pixels with large depth/color re-rendering loss. In particular, we remove any pixel from the optimization where the loss Eq. (5.12) is larger than $10\times$ the median loss value of all pixels in the current frame. Fig. 5.8 shows an example where a dynamic object is ignored since it is not present in the rendered RGB and depth image. Note that for this task, we only optimize the scene representation during the mapping. Jointly optimizing camera parameters and scene representations under dynamic environments is non-trivial, and we consider it as an interesting future direction.

5.3.4 Initialization for Hierarchical Feature Grids

Coarse-level Feature Grid. The coarse-level feature grid is randomly initialized in all experiments.

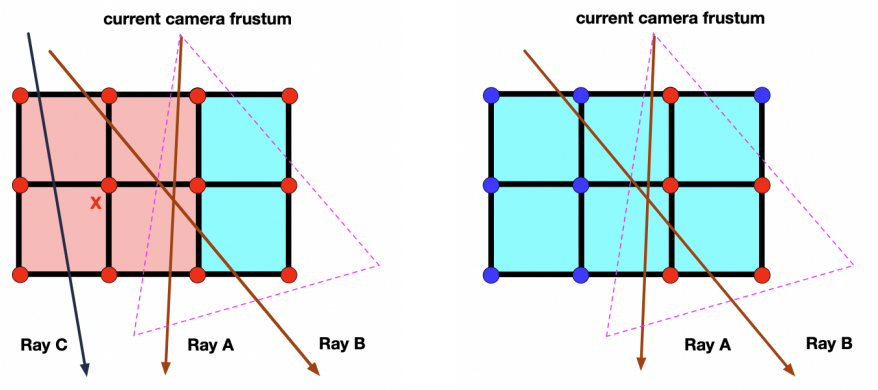
Mid-level Feature Grid. The mid-level feature grid is also randomly initialized in all experiments, except for the result shown in Fig. 5.7, where it is initialized to free space to better visualize the predictions from the coarse-level grid. Empirically we find that the random initialization gives slightly better convergence compared to initializing from a fixed feature vector corresponding to the free space.

Fine-level Feature Grid. The fine-level feature grid is initialized to ensure the output of the fine-level decoder f^2 as zero, as it is added in a residual manner onto the occupancy predicted from the mid-level features. This guarantees a smooth energy transition in the coarse-to-fine optimization. During the training of the fine-level decoder from ConvONet [171], we add additional regularization loss to enforce that, if the fine-level feature is zero, no matter what the concatenated mid-level feature is, the output residual should always be zero. This regularization allows us to zero-initialize the fine-level grid at runtime.

5.3.5 Keyframe Selection

Similar to other SLAM systems, we continuously optimize our hierarchical scene representation with a set of selected keyframes. We maintain a global keyframe list in the same spirit of iMAP [200], where we incrementally add new keyframes based on the information gain. However, in contrast to iMAP [200], we only include keyframes which have visual overlap with the current frame when optimizing the scene geometry. This is possible since we are able to make local updates to our grid-based representation, and we do not suffer from the same forgetting problems as [200]. This keyframe selection strategy not only ensures the geometry outside of the current view remains static, but also results in a very efficient optimization problem as we only optimize the necessary parameters each time.

In practice, we first randomly sample pixels and back-project the corresponding depths using the optimized camera pose. Then, we project the point cloud to every keyframe in the global keyframe list. From those keyframes that have points projected onto, we randomly select $K-2$ frames. In addition, we also include the most recent keyframe and the current frame in the scene representation optimization, forming a total number of K active frames. Refer to Sec. 5.4.4 for an ablation study on the keyframe selection strategy.



(a) *Interpolation problem.* **(b)** *Feature selection.*

Figure 5.3: ***2D Illustration of Feature Grids.*** *The lattice points correspond to features. The optimized and fixed features are shown in red and blue respectively.*

5.3.6 Frustum Feature Selection

The grid-based representation allows us to only optimize the geometry within the current viewing frustum while keeping the rest of the scene geometry fixed. However, naive optimization for all voxels will affect features even just slightly outside the viewing frustum because of trilinear interpolation. This is illustrated in Fig. 5.3a. The rays A and B are viewing rays from the current frame and an active keyframe, respectively. Including these rays in the optimization will update the feature at X (marked in the figure) due to trilinear interpolation. However, updating this feature will also affect the ray C coming from an inactive keyframe.

To solve the problem, we propose to only update features fully inside the current viewing frustum during the optimization, see Fig. 5.3b. In this way, it will not only preserve the previously reconstructed geometry, but also significantly reduce the number of parameters during optimization.

5.4 Experiments

We evaluate our SLAM framework on a wide variety of datasets, both real and synthetic, of varying size and complexity. We also conduct a comprehensive ablation study that supports our design choices.

5.4.1 Experimental Setup

Datasets. We consider 5 versatile datasets: Replica [198], ScanNet [44], TUM RGB-D dataset [199], Co-Fusion dataset [183], as well as a self-captured large apartment with multiple rooms. We follow the same pre-processing step for TUM RGB-D as in [214].

Baselines. We compare with TSDF-Fusion [42] with our camera poses with a voxel grid resolution of 256^3 (results of higher resolutions are reported in the supp. material), DI-Fusion [82] using their official implementation[2], as well as our faithful iMAP [200] re-implementation: iMAP*. Our re-implementation has a similar performance as the original iMAP in both scene reconstruction and camera tracking.

Metrics. We use both 2D and 3D metrics to evaluate the scene geometry. For the 2D metric, we evaluate the L1 loss on 1000 randomly sampled depth maps from both reconstructed and ground truth meshes. For a fair comparison, we apply the bilateral solver [7] to DI-Fusion [82] and TSDF-Fusion to fill depth holes before calculating the average L1 loss. For 3D metrics, we follow [200] and consider *Accuracy* [cm], *Completion* [cm], and *Completion Ratio* [$<$ 5cm %], except that we remove unseen regions that are not inside any camera's viewing frustum. Regarding the evaluation of camera tracking, we use ATE RMSE [199]. If not specified otherwise, by default we report the average results of 5 runs.

[2] https://github.com/huangjh-pub/di-fusion

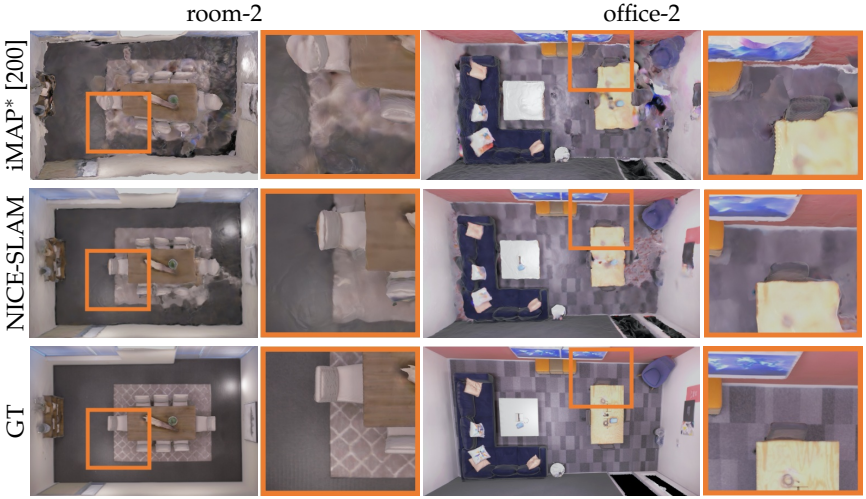


Figure 5.4: ***Reconstruction Results on the Replica Dataset [198].*** *iMAP* refers to our iMAP re-implementation.*

Implementation Details. We run our SLAM system on a desktop PC with a 3.80GHz Intel i7-10700K CPU and an NVIDIA RTX 3090 GPU. In all our experiments, we use the number of sampling points on a ray $N_{\text{strat}} = 32$ and $N_{\text{imp}} = 16$, photometric loss weighting $\lambda_p = 0.2$ and $\lambda_{pt} = 0.5$. For small-scale synthetic datasets (Replica and Co-Fusion), we select $K = 5$ keyframes and sample $M = 1000$ and $M_t = 200$ pixels. For large-scale real datasets (ScanNet and our self-captured scene), we use $K = 10$, $M = 5000$, $M_t = 1000$. As for the challenging TUM RGB-D dataset, we use $K = 10$, $M = 5000$, $M_t = 5000$. For our re-implementation iMAP*, we follow all the hyperparameters mentioned in [200] except that we set the number of sampling pixels to 5000 since it leads to better performance in both reconstruction and tracking.

	TSDF-Fusion [42]	iMAP* [200]	DI-Fusion [82]	**NICE-SLAM**
Mem. (MB) ↓	67.10	**1.04**	3.78	12.02
Depth L1 ↓	7.57	7.64	23.33	**3.53**
Acc. ↓	**1.60**	6.95	19.40	2.85
Comp. ↓	3.49	5.33	10.19	**3.00**
Comp. Ratio ↑	86.08	66.60	72.96	**89.33**

Table 5.1: ***Reconstruction Results for the Replica Dataset [198] (average over 8 scenes).*** *iMAP* is our re-implementation of iMAP. TSDF-Fusion uses camera poses from NICE-SLAM.*

5.4.2 Evaluation of Mapping and Tracking

Evaluation on Replica [198]. To evaluate Replica [198], we use the same rendered RGB-D sequence provided by the authors of iMAP. With the hierarchical scene representation, our method is able to reconstruct the geometry precisely within limited iterations. As shown in Table 5.1 as well as the per-scene results in Table 5.3, NICE-SLAM significantly outperforms baseline methods on almost all metrics, while keeping a reasonable memory consumption. Qualitatively, we can see from Fig. 5.4 that our method produces sharper geometry and less artifacts.

Evaluation on TUM RGB-D [199]. We also evaluate the camera tracking performance on the small-scale TUM RGB-D dataset. As shown in Table 5.2, our method outperforms iMAP and DI-Fusion even though ours is by design more suitable for large scenes. As can be noticed, the state-of-the-art approaches for tracking (e.g. BAD-SLAM [188], ORB-SLAM2 [152]) still outperforms the methods based on implicit scene representations (iMAP [200] and ours). Nevertheless, our method significantly reduces the gap between these two categories, while retaining the representational advantages of implicit representations.

Evaluation on ScanNet [44]. We select multiple large scenes from

	fr1/desk	fr2/xyz	fr3/office
iMAP [200]	4.9	2.0	5.8
iMAP* [200]	7.2	2.1	9.0
DI-Fusion [82]	4.4	2.3	15.6
NICE-SLAM	2.7	1.8	3.0
BAD-SLAM [188]	1.7	1.1	1.7
Kintinuous [233]	3.7	2.9	3.0
ORB-SLAM2 [152]	**1.6**	**0.4**	**1.0**

Table 5.2: ***Camera Tracking Results on TUM RGB-D [199].*** *ATE RMSE [cm] (↓) is used as the evaluation metric. NICE-SLAM reduces the gap between SLAM methods with neural implicit representations and traditional approaches. We report the best out of 5 runs for all methods in this table. The numbers for iMAP, BAD-SLAM, Kintinuous, and ORB-SLAM2 are taken from [200].*

ScanNet [44] to benchmark the scalability of different methods. For the geometry shown in Fig. 5.5, we can clearly notice that NICE-SLAM produces sharper and more detailed geometry over TSDF-Fusion, DI-Fusion and iMAP*. In terms of tracking, as can be observed, iMAP* and DI-Fusion either completely fail or introduce large drifting, while our method successfully reconstructs the entire scene. Quantitatively speaking, our tracking results are also significantly more accurate than both DI-Fusion and iMAP* as shown in Table 5.4.

Evaluation on a Larger Scene. To evaluate the scalability of our method we captured a sequence in a large apartment with multiple rooms. Fig. 5.1 and Fig. 5.6 show the reconstructions obtained using NICE-SLAM, DI-Fusion [82] and iMAP* [200]. For reference, we also show the 3D reconstruction using the offline tool Redwood [38] in Open3D [278]. We can see that NICE-SLAM has comparable results with the offline method, while iMAP* and DI-Fusion fail to reconstruct the full sequence.

		room-0	room-1	room-2	office-0	office-1	office-2	office-3	office-4	Avg.
TSDF-Fusion Res. = 512 (536.87MB)	**Depth L1** [cm] ↓	6.38	5.33	6.84	4.74	4.62	11.32	9.89	6.49	6.95
	Acc. [cm] ↓	1.87	2.48	1.69	1.14	0.96	1.63	2.08	1.74	1.70
	Comp. [cm] ↓	3.60	3.20	2.85	1.72	2.31	3.66	3.69	3.91	3.12
	Comp. Ratio [< 5cm %] ↑	88.33	89.82	90.38	93.55	90.35	86.74	85.35	86.31	88.85
TSDF-Fusion Res. = 256 (67.10MB)	**Depth L1** [cm] ↓	6.69	5.47	7.47	4.97	5.28	12.30	11.17	7.20	7.57
	Acc. [cm] ↓	1.76	2.11	1.59	1.15	0.97	1.56	1.98	1.66	**1.60**
	Comp. [cm] ↓	3.85	3.36	3.33	1.93	2.68	4.17	4.22	4.37	3.49
	Comp. Ratio [< 5cm %] ↑	86.29	88.44	86.63	91.73	87.88	82.95	81.31	83.38	86.08
iMAP* **[200]** (1.04MB)	**Depth L1** [cm] ↓	5.70	4.93	6.94	6.43	7.41	14.23	8.68	6.80	7.64
	Acc. [cm] ↓	5.66	5.31	5.64	7.39	11.89	8.12	5.62	5.98	6.95
	Comp. [cm] ↓	5.20	5.16	5.04	4.35	5.00	6.33	5.47	6.10	5.33
	Comp. Ratio [< 5cm %] ↑	67.67	66.41	69.27	71.97	71.58	58.31	65.95	61.64	66.60
DI-Fusion [82] (3.78MB)	**Depth L1** [cm] ↓	6.66	96.82	36.09	7.36	5.05	13.73	11.41	9.55	23.33
	Acc. [cm] ↓	1.79	49.00	26.17	70.56	1.42	2.11	2.11	2.02	19.40
	Comp. [cm] ↓	3.57	39.40	17.35	3.58	2.20	4.83	4.71	5.84	10.19
	Comp. Ratio [< 5cm %] ↑	87.77	32.01	45.61	87.17	91.85	80.13	78.94	80.21	72.96
NICE-SLAM (12.02MB)	**Depth L1** [cm] ↓	2.11	1.68	2.90	1.83	2.46	8.92	5.93	2.38	**3.53**
	Acc. [cm] ↓	2.73	2.58	2.65	2.26	2.50	3.82	3.50	2.77	2.85
	Comp. [cm] ↓	2.87	2.47	3.00	2.02	2.36	3.57	3.83	3.84	**3.00**
	Comp. Ratio [< 5cm %] ↑	90.93	92.80	89.07	94.93	92.61	85.20	82.98	86.14	**89.33**

Table 5.3: ***Reconstruction Results for the Replica Dataset of Each Scene.*** *We provide results for each scene, an average of 5 runs.*

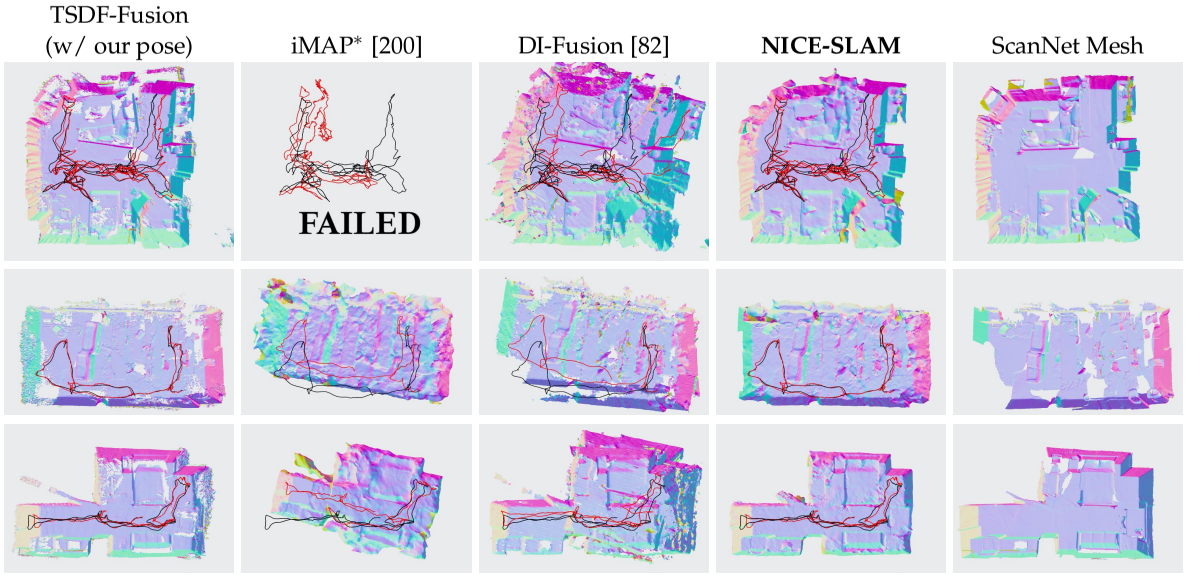


Figure 5.5: ***3D Reconstruction and Tracking on ScanNet [44]****. The black trajectory is from ScanNet [44], the red trajectory is the methods' tracking result. We tried various hyperparameters for iMAP* and present the best results which are mostly inferior.*

Scene ID	0000	0059	0106	0169	0181	0207	Avg.
iMAP* [200]	55.95	32.06	17.50	70.51	32.10	11.91	36.67
DI-Fusion [82]	62.99	128.00	18.50	75.80	87.88	100.19	78.89
NICE-SLAM	**8.64**	**12.25**	**8.09**	**10.28**	**12.93**	**5.59**	**9.63**

Table 5.4: ***Camera Tracking Results on ScanNet [44].*** *Our approach yields consistently better results on this dataset. ATE RMSE (↓) is used as the evaluation metric.*

5.4.3 Performance Analysis

Besides the evaluation of scene reconstruction and camera tracking on various datasets, in the following, we also evaluate other characteristics of the proposed pipeline.

Computation Complexity. First, we compare the number of floating point operations (FLOPs) needed for querying color and occupancy/volume density of one 3D point, see Table 5.5. Our method requires only 1/4 FLOPs of iMAP. It is worth mentioning that FLOPs in our approach remain the same even for very large scenes. In contrast, due to the use of a single MLP in iMAP, the capacity limit of the MLP might require more parameters, which results in more FLOPs.

Runtime. We also compare in Table 5.5 the runtime for tracking and mapping using the same number of pixel samples ($M_t = 200$ for tracking and $M = 1000$ for mapping). We can notice that our method is over $2\times$ and $3\times$ faster than iMAP in tracking and mapping. This indicates the advantage of using feature grids with shallow MLP decoders over a single heavy MLP.

Geometry Forecast and Hole Filling. As illustrated in Fig. 5.7, we are able to complete unobserved scene regions thanks to the use of coarse-level scene prior. In contrast, the unseen regions reconstructed by iMAP* are very noisy since no scene prior knowledge is encoded in iMAP*.

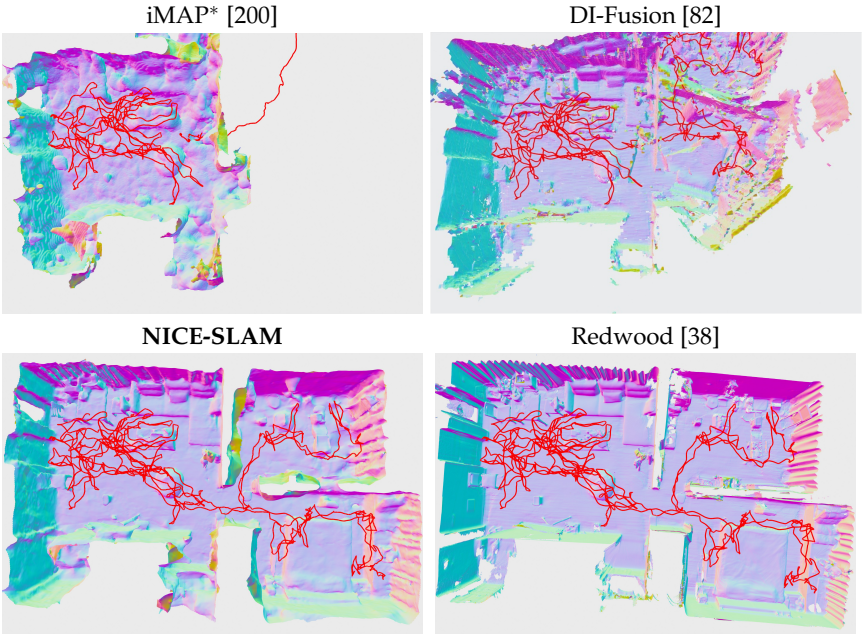


Figure 5.6: ***3D Reconstruction and Tracking on a Multi-room Apartment.*** *The camera tracking trajectory is shown in red. iMAP* and DI-Fusion failed to reconstruct the entire sequence. We also show the result of an offline method [38] for reference.*

	FLOPs $[\times 10^3]\downarrow$	Tracking [ms]$\downarrow$	Mapping [ms]$\downarrow$
iMAP [200]	443.91	101	448
NICE-SLAM	**104.16**	**47**	**130**

Table 5.5: ***Computation & Runtime.*** *Our scene representation does not only improve the reconstruction and tracking quality, but is also faster. The runtimes for iMAP are taken from [200].*

Robustness to Dynamic Objects. Here we consider the Co-Fusion dataset [183] which contains dynamically moving objects. As illustrated in Fig. 5.8, our method correctly identifies and ignores pixel

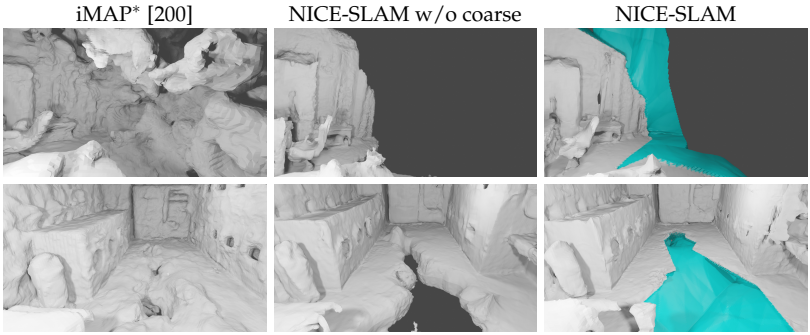


Figure 5.7: ***Geometry Forecast and Hole Filling.*** *The white-colored area is the region with observations, and cyan indicates the unobserved but predicted region. Thanks to the use of coarse-level scene prior, our method has better prediction capability compared to iMAP*. This in turn also improves our tracking performance.*

Figure 5.8: ***Robustness to Dynamic Objects.*** *We show the sampled pixels overlaid on an image with a dynamic object in the center (left), our rendered RGB (middle) and our rendered depth (right) to illustrate the ability of handling dynamic environments. The masked pixel samples during tracking are colored in black, while the used ones are shown in red.*

samples falling into the dynamic object during optimization, which leads to better scene representation modelling (see the rendered RGB and depths). Furthermore, we also compare with iMAP* on the same sequence for camera tracking. The ATE RMSE scores of ours and iMAP* is 1.6cm and 7.8cm respectively, which clearly demonstrates our robustness to dynamic objects.

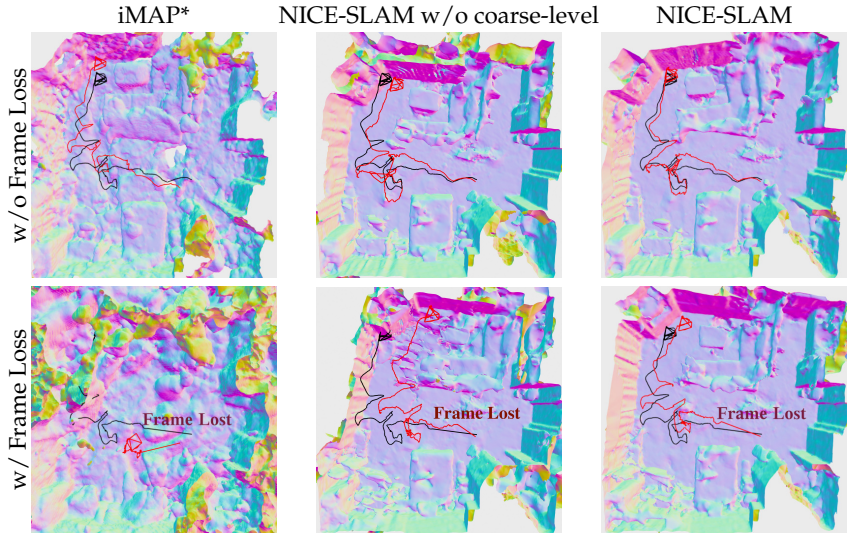


Figure 5.9: ***Robustness to Frame Loss.*** *We show the results at frame 2100 after frame loss at frame 2000. The black trajectory is the ground truth from ScanNet [44], and the red trajectory indicates tracking results. The missing frames correspond to the straight line in the middle.*

Frame Loss Robustness. We simulate extreme frame loss on ScanNet `scene0000_00` by skipping 100 frames from frame ID 2001 to 2100. As visualized in Fig. 5.9, iMAP* struggles to recover camera poses and scene geometry, even given 1500 iterations. In contrast, our NICE-SLAM is able to recover the camera pose using only 300 iterations. This is due to the use of coarse-level geometric representation which improves the prediction capability.

5.4.4 Ablation Study

Hierarchical Architecture. Fig. 5.10 compares our hierarchical architecture against: a) one feature grid with the same resolution as our

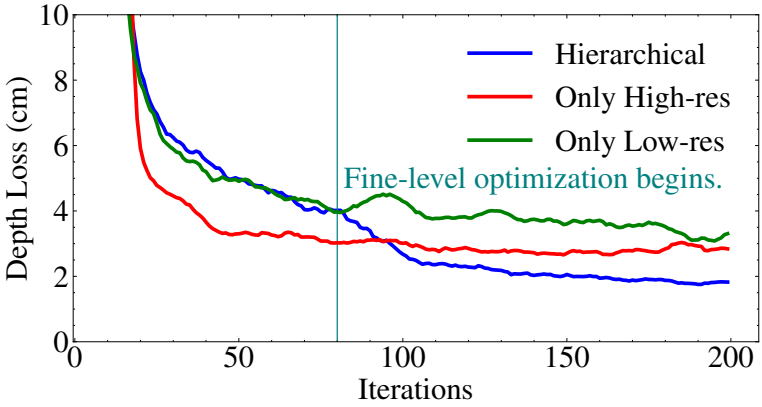


Figure 5.10: ***Hierarchical Architecture Ablation.*** *Geometry optimization on a single depth image on Replica [198] with different architectures. The curves are smoothed for visualization.*

fine-level representation (Only High-res); b) one feature grid with mid-level resolution (Only Low-res). Our hierarchical architecture can quickly add geometric details when the fine-level representation participates in the optimization, which also leads to better convergence. The hierarchical architecture guarantees a good balance between the quality and real-time capability / memory consumption (only 12 MB for Replica scenes).

We also conduct another ablation study on the number of levels of feature grids in Table 5.6. It shows that the 3-level feature grid is a good balance between the reconstruction quality and computational efficiency.

Local BA. We verify the effectiveness of local bundle adjustment on ScanNet [44]. If we do not jointly optimize camera poses for K keyframes together with the scene representation (w/o Local BA in Table 5.7), the camera tracking is not only significantly less accurate, but also less robust.

Levels	**2**	**3**	**4**
FLOPs [$\times 10^3$] $\downarrow$	58.45	104.16	155.95
Depth L1 [cm] $\downarrow$	**1.86**	1.87	1.96
Acc. [cm] $\downarrow$	2.87	**2.78**	3.15
Comp. [cm] $\downarrow$	2.76	2.76	**2.40**
Comp. Ratio [$<$ 5cm %] $\uparrow$	91.24	91.37	**93.60**

Table 5.6: ***Ablation on the Levels of Feature Grids.*** *Reconstruction results on Replica room-0 with ground truth camera pose.*

ATE RMSE ($\downarrow$)	w/o Local BA	w/o $\mathcal{L}_p$	w/ iMAP keyframes	Full
Mean	37.74	32.02	12.10	9.63
Std.	30.97	21.98	3.38	0.62

Table 5.7: ***Ablation Study on LocalBA, Color Representation, and Keyframe Selection.*** *We investigate the usefulness of local BA, color representation, as well as our keyframe selection strategy. We run each scene 5 times and calculate their mean and standard deviation of ATE RMSE ($\downarrow$). We report the average values over 6 scenes in ScanNet [44].*

Color Representation. In Table 5.7 we compare our method without the photometric loss $\mathcal{L}_p$ in Eq. (5.9). It shows that, although our estimated colors are not perfect due to the limited optimization budget and the lack of sampling points, learning such a color representation still plays an important role for accurate camera tracking.

Keyframe Selection. We test our method using iMAP's keyframe selection strategy (w/ iMAP keyframes in Table 5.7) where they select keyframes from the entire scene. This is necessary for iMAP to prevent their simple MLP from forgetting the previous geometry. Nevertheless, it also leads to slow convergence and inaccurate tracking.

Mapping and Tracking Iterations. We show in Fig. 5.11 how the number of tracking and mapping iterations affects the tracking per-

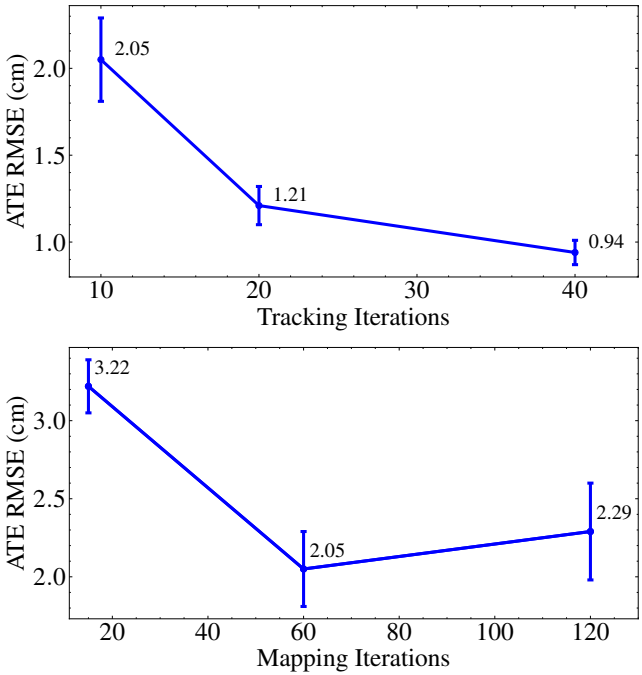


Figure 5.11: ***Ablation on the Tracking Performance.*** *ATE RMSE (cm) is used as the metric.*

formance. We also give ground truth camera pose and evaluate reconstruction with different mapping iterations in Table 5.8.

5.5 Conclusion and Discussion

In this chapter, we presented NICE-SLAM, a dense visual SLAM approach that combines the advantages of neural implicit representations with the scalability of a hierarchical grid-based scene representation. Compared to a scene representation with a single big MLP, our experiments demonstrate that our representation (tiny MLPs +

Mapping Iterations	**15**	**30**	**60**	**120**	**240**
Depth L1 [cm] ↓	2.31	2.03	1.87	1.74	1.59
Acc. [cm] ↓	2.90	2.84	2.78	2.80	2.78
Comp. [cm] ↓	3.14	2.91	2.76	2.65	2.50
Comp. Ratio [$<$ 5cm %] ↑	89.15	90.55	91.37	91.94	92.76

Table 5.8: ***Ablation on Mapping Iterations.*** *Reconstruction results on Replica room-0 with ground truth camera poses.*

multi-res feature grids) not only guarantees fine-detailed mapping and high tracking accuracy, but also faster speed and much less computation due to the benefit of local scene updates. Besides, our network is able to fill small holes and extrapolate scene geometry into unobserved regions which in turn stabilizes the camera tracking.

Note that, one concurrent work named Instant-NGP [150] also shows the benefit of hierarchical feature grids for fast training of neural fields, which draw much attention and inspire follow-up works in many research area. This further demonstrates the usefulness of our proposed representation.

Limitations. The predictive ability of our method is restricted to the scale of the coarse representation. In addition, our method does not perform loop closures, which is an interesting future direction. Finally, although traditional methods lack some of the features, there is still a performance gap in the learning-based approaches that needs to be closed.

CHAPTER 6

3D Scene Understanding with Large Vision Language Models

Building upon our exploration of scene representations for 3D reconstruction from either point clouds or RGB-D sequences in previous chapters, this chapter delves deeper into the realm of high-level perception tasks, focusing on 3D scene understanding of the reconstructed scenes. Contrary to traditional 3D scene understanding approaches that rely on labeled 3D datasets for supervised training on a single task, we introduce a zero-shot approach that facilitates task-agnostic training and supports open-vocabulary queries. Remarkably, our model, trained without any labeled 3D data, can effectively identify objects, materials, affordances, activities, and room types in complex 3D scenes.

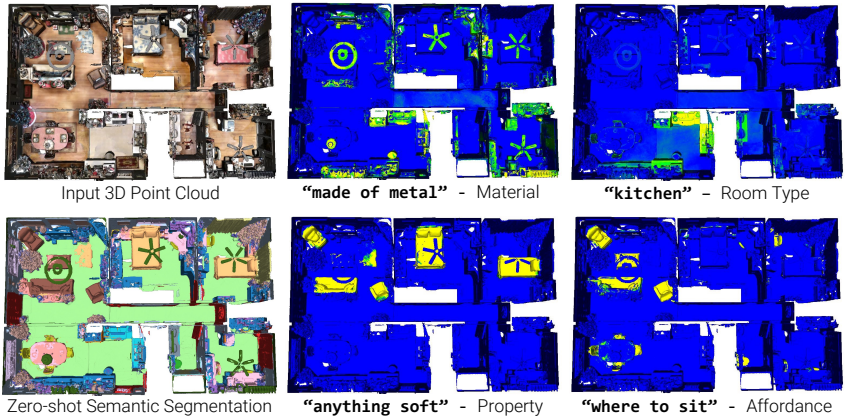


Figure 6.1: ***Open-vocabulary 3D Scene Understanding.*** *We propose OpenScene, a zero-shot approach to 3D scene understanding that co-embeds dense 3D point features with image pixels and text. The examples above show a 3D scene with surface points colored by how well they match a user-specified query – yellow is highest, green is middle, blue is low. Harnessing the power of language-based features, OpenScene answers a wide variety of example queries, without labeled 3D data.*

6.1 Introduction

3D scene understanding stands as a cornerstone in the expansive field of computer vision. Having a set of posed RGB images together with 3D mesh or point cloud obtained from the previous chapters, the goal now becomes inferring the semantics, affordances, functions, physical properties of every 3D point in a scene. For example, given the house shown in Fig. 6.1, we would like to predict which surfaces are part of a fan (semantics), made of metal (materials), within a kitchen (room types), where a person can sit (affordances), where a person can work (functions), and which surfaces are soft (physical properties). Answers to these queries can help a robot interact intelligently with the scene or help a person understand it through interactive query and visualization.

Achieving this broad scene-understanding goal is challenging due to the diversity of possible queries. Traditional 3D scene understanding systems are trained with supervision from benchmark datasets designed for specific tasks (e.g., 3D semantic segmentation for a closed set of 20 classes [23, 44]). They are designed to answer one type of query, but provide little assistance for related queries where training data are scarce (e.g., segmenting rare objects) or other queries with no 3D supervision (e.g., estimating material properties). While traditional methods have made significant strides in addressing specific tasks, there's a growing realization that leveraging pre-trained models like text-image embedding models could fill the gaps left by conventional approaches.

In this chapter, we investigate how to use pre-trained text-image embedding models (e.g., CLIP [175]) to assist in 3D scene understanding. These models have been trained from large datasets of captioned images to co-embed visual and language concepts in a shared feature space. Recent work has shown that these models can be used to increase the flexibility and generalizability of 2D image semantic segmentation [61,113,176,246,261,276]. However, nobody has investigated how to use them to improve the diversity of queries possible for 3D scene understanding.

We present *OpenScene*, a simple yet effective zero-shot approach for open-vocabulary 3D scene understanding. Our key idea is to compute dense features for 3D points that are co-embedded with text strings and image pixels in the CLIP feature space (Fig. 6.2). To achieve this, we establish associations between 3D points and pixels from posed images in the 3D scene, and train a 3D network to embed points using CLIP pixel features as supervision. This approach brings 3D points in alignment with pixels in the feature space, which in turn are aligned with text features, and thus enable open vocabulary queries on the 3D points.

Our 3D point embedding algorithm includes both 2D and 3D convolutions. We first back-project the 3D position of the point into every image and aggregate the features from the associated pixels using

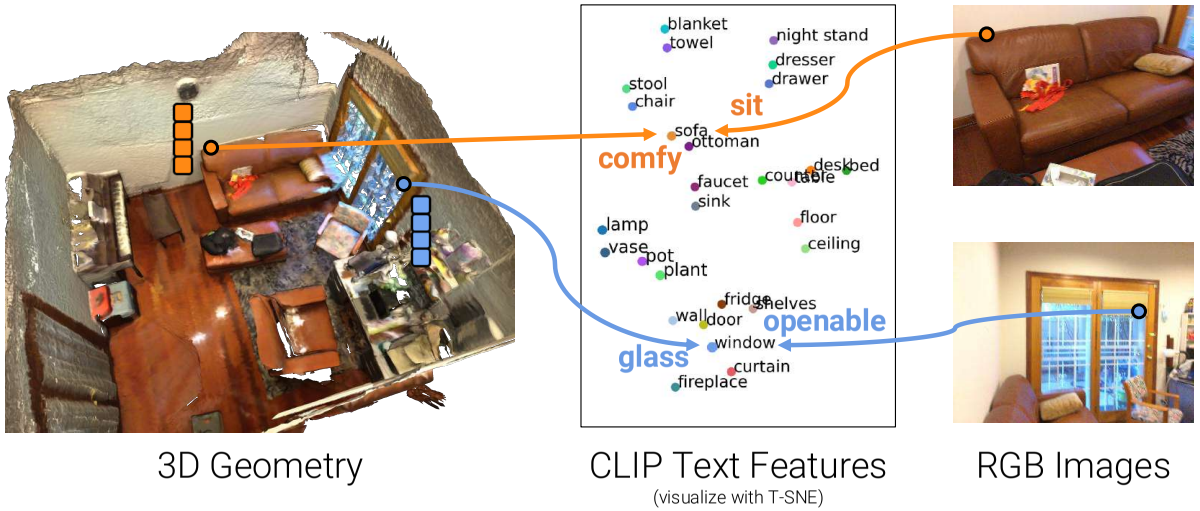


Figure 6.2: ***Key Idea.*** *We co-embed 3D points with text and image pixels in the CLIP feature space (visualized with T-SNE [217]) which has structure learned from large image and text repositories.*

multi-view fusion. Next, we train a sparse 3D convolutional network to perform feature extraction from only the 3D point cloud geometry with a loss that minimizes differences to the aggregated pixel features. Finally, we ensemble the features produced by the 2D fusion and the 3D network into a single feature for each 3D point. This hybrid 2D-3D feature strategy enables the algorithm to take advantage of salient patterns in both 2D images and 3D geometry, and thus is more robust and descriptive than features from either domain alone.

Once we have computed features for every 3D point, we can perform a variety of 3D scene understanding queries. Since the CLIP model is trained with natural language captions, it captures concepts beyond object class labels, including affordances, materials, attributes, and functions (Fig. 6.1). For example, computing the similarity of 3D features with the embedding for "soft" produces the result shown in the bottom-left image of Fig. 6.1, which highlights couches, beds, and comfy chairs as the best matches. Since our approach is zero-shot (i.e. no use of labeled data for the target task), it does not perform as well as fully-supervised approaches on the limited set of tasks for which there is sufficient training data in traditional benchmarks (e.g., 3D semantic segmentation with 20 classes). However, it achieves significantly stronger performance on other tasks. For example, it beats a fully-supervised approach on indoor 3D semantic segmentation with 40, 80, or 160 classes. It also performs better than other zero-shot baselines, and can be used without any retraining on novel datasets even if they have different label sets. It works for indoor RGBD scans as well as outdoor driving captures. Our approach, to our knowledge, is the first to integrate text-image embedding models like CLIP into 3D scene understanding, opening avenues for multifaceted and complex queries.

Overall, our contributions are summarized as follows:

- We introduce open vocabulary 3D scene understanding tasks where arbitrary text queries are used for semantic segmentation, affordance estimation, room type classification, 3D object search, and 3D scene exploration.

- We propose OpenScene, a zero-shot method for extracting 3D features from an open vocabulary embedding space with multi-view fusion and 3D convolution.
- We show that the extracted features can be used for 3D semantic segmentation with performance better than fully supervised methods for rare classes.

6.2 Related Work

Closed-set 3D Scene Understanding. There is a long history of work on 3D scene understanding for vision and robotics applications. Most prior work focuses on training models with ground-truth 3D labels [39, 69, 77, 78, 84, 115, 154, 173, 180, 189, 226]. These works have yielded network architectures and training protocols that have significantly pushed the boundary of several 3D scene understanding benchmarks, including 3D object classification [243], 3D object detection and localization [15, 26, 57, 204], 3D semantic and instance segmentation [8,23,44,79,122], 3D affordance prediction [48,116,223], and so on. The most closely related work to ours of this type is [182], since they use the CLIP embedding to pre-train a model for 3D semantic segmentation. However, they only use the text embedding for point encoder pretraining, and then train the point decoder with 3D GT annotations afterwards. Their focus is on using the CLIP embedding to achieve better supervised 3D semantic segmentation, rather than open-vocabulary queries.

Another line of research performs 3D scene understanding experiments with only 2D ground truth supervisions [60, 109, 143, 187, 219, 228]. For example, [60] generates pseudo 3D annotation by backprojecting and fusing the 2D predicted labels, from which they learn the 3D segmentation task. However, their 2D network is trained with

ground truth 2D labels. A few works [134, 187] pretrain the 3D segmentation network using point-pixel pairs via contrastive learning between 2D and 3D features. We also utilize 2D image features as our pseudo-supervision when training the 3D network and no labels are needed.

All these approaches have mainly been applied with small predefined labelsets containing common object categories. They do not work as well when the number of object categories increases, as tail classes have few training examples. In contrast, we are able to segment with arbitrary labelsets without any re-training, and we show strong ability of understanding different contents, ranging from rare object types to even materials or physical properties, which is impossible for previous methods.

Open-Vocabulary 2D Scene Understanding. The recent advances of large visual language models [1, 87, 175] have enabled a remarkable level of robustness in zero-shot 2D scene understanding tasks, including recognizing long-tail objects in images. However, the learned embeddings are often at the image level, thus not applicable for dense prediction tasks requiring pixel-level information. Many recent efforts [61, 66, 110, 113, 120, 138, 176, 192, 246, 261, 276] attempt to correlate the dense image features with the embedding from large language models. In this way, given an image at test time, users can define arbitrary text labels to classify, detect, or segment the image.

More recently, Ha and Song [68] take a step forward and perform open-vocabulary partial scene understanding and completion given a single RGB-D frame as input. This method is limited to small partial scenes and requires ground truth training data for supervision. In contrast, in this work, we solely rely on pretrained open-vocabulary 2D models and perform a series of 3D scene-level understanding tasks, without the need for any ground truth training data in 2D or 3D. Moreover,in the absence of 2D images, our method can perform 3D-only open-vocabulary scene understanding tasks based on a 3D point network distilled from an open-vocabulary 2D image model through 3D fusion.

Zero-shot Learning for 3D Point Clouds. While there have been a number of studies on zero-shot learning for 2D images, their application to 3D is still recent and scarce. A handful of works [31–34, 271] attempt to address the 3D point classification and generation tasks. More recently, [128, 147] investigated zero-shot learning for semantic segmentation for 3D point clouds. They train with supervision of 3D ground truth labels for a predefined set of seen classes and then evaluate on new unseen classes. However, these methods are still limited to the closed-set segmentation setting and still require GT training data for the majority of the 3D dataset. Our method *does not* require any labeled 3D data for training, and it handles a broad range of queries supported by a large language model.

6.3 Method

An overview of our approach is illustrated in Fig. 6.3. We first compute per-pixel features for every image using a model pre-trained for open-vocabulary 2D semantic segmentation. We then aggregate the pixel features from multiple views onto every 3D point to form a per-point fused feature vector Sec. 6.3.1. We next distill a 3D network to reproduce the fused features using only the 3D point cloud as input Sec. 6.3.2. Next, we ensemble the fused 2D features and distilled 3D features into a single per-point feature Sec. 6.3.3 and use it to answer open-vocabulary queries Sec. 6.3.4.

6.3.1 Image Feature Fusion

The first step in our approach is to extract dense per-pixel embeddings for each RGB image from a 2D visual-language segmentation model, and then back-project them onto the 3D surface points of a scene.

Image Feature Extraction. Given RGB images with a resolution of

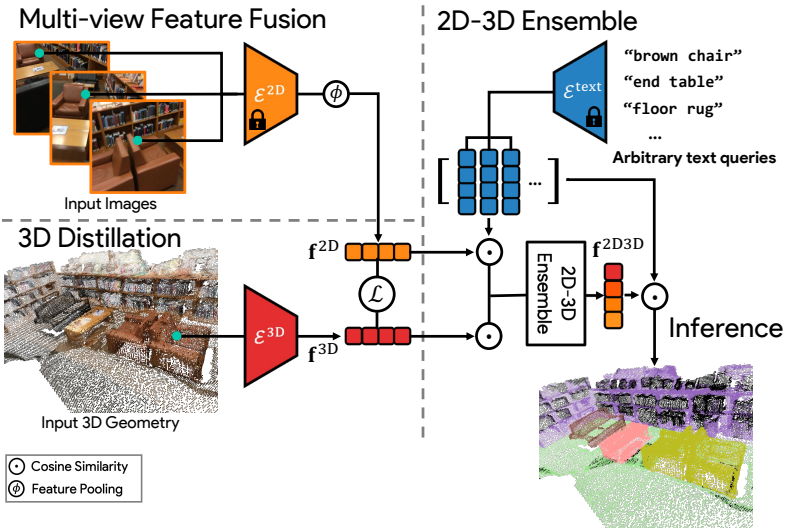


Figure 6.3: ***Method Overview.*** *Given a 3D model (mesh or point cloud) and a set of posed images, we train a 3D network $\mathcal{E}^{3D}$ to produce dense features for 3D points $\mathbf{f}^{3D}$ with a distillation loss $\mathcal{L}$ to multi-view fused features $\mathbf{f}^{2D}$ for projected pixels. We ensemble $\mathbf{f}^{2D}$ and $\mathbf{f}^{3D}$ based on cosine similarities to CLIP embeddings for an arbitrary set of queries to form $\mathbf{f}^{2D3D}$. During inference, we can use the similarity scores between per-point features and given CLIP features to perform open-vocabulary 3D scene understanding tasks.*

$H \times W$, we can simply compute the per-pixel embeddings from the (frozen) segmentation model $\mathcal{E}^{2D}$, denoted as $\mathbf{I}_i \in \mathbb{R}^{H \times W \times C}$, where C is the feature dimension, and i is the index spanning the total number of images. For $\mathcal{E}^{2D}$, we experiment with two pretrained image segmentation models OpenSeg [61] and LSeg [113].

2D-3D Pairing. Given a 3D surface point $\mathbf{p} \in \mathbb{R}^3$ in the point clouds $\mathbf{P} \in \mathbb{R}^{M \times 3}$ of a scene with M points, we calculate its corresponding pixel $\mathbf{u} = (u, v)$ when the intrinsic matrix I_i and world-to-camera extrinsic matrix E_i of that frame i are provided. We only consider the pinhole camera model in this chapter so the projection can be represented as $\tilde{\mathbf{u}} = I_i \cdot E_i \cdot \tilde{\mathbf{p}}$, where $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{p}}$ are the homogeneous co-

ordinates of $\mathbf{u}$ and $\mathbf{p}$, respectively. Note that for indoor datasets like ScanNet and Matterport3D where the depth images are provided, we also conduct occlusion tests to guarantee the pixel $\mathbf{u}$ are only paired with a visible surface point $\mathbf{p}$.

Fusing Per-Pixel Features. With the 2D-3D pairing, the corresponding 2D features in frame i for point $\mathbf{p}$ can be written as $\mathbf{f}_i = \mathbf{I}_i(\mathbf{u}) \in \mathbb{R}^C$. Now, assume a total number of K views can be associated with point $\mathbf{p}$, we can then fuse such 2D pixel embeddings to obtain a single feature vector for this point $\mathbf{f}^{2D} = \phi(\mathbf{f}_1, \cdots, \mathbf{f}_K)$, where $\phi : \mathbb{R}^{K \times C} \mapsto \mathbb{R}^C$ is an average pooling operator for multi-view features. An ablation study on different fusion strategies are discussed in supplemental. After repeating the fusion process for each point, we can build a feature point cloud $\mathbf{F}^{2D} = \{\mathbf{f}_1^{2D}, \cdots, \mathbf{f}_M^{2D}\} \in \mathbb{R}^{M \times C}$.

6.3.2 3D Distillation

The feature cloud $\mathbf{F}^{2D}$ can be directly used for language-driven 3D scene understanding *when* images are present. Nevertheless, such fused features could lead to noisy segmentation due to potentially inconsistent 2D predictions. Moreover, some tasks only provide 3D point clouds or meshes. Therefore, we can distill such 2D visual-language knowledge into a 3D point network that only takes 3D point positions as input.

Specifically, given an input point cloud $\mathbf{P}$, we seek to learn an encoder that outputs per-point embeddings:

$$\mathbf{F}^{3D} = \mathcal{E}^{3D}(\mathbf{P}), \quad \mathcal{E}^{3D} : \mathbb{R}^{M \times 3} \mapsto \mathbb{R}^{M \times C} \tag{6.1}$$

where $\mathbf{F}^{3D} = \{\mathbf{f}_1^{3D}, \cdots, \mathbf{f}_M^{3D}\}$. To enforce the output of the network $\mathbf{F}^{3D}$ to be consistent with the fused features $\mathbf{F}^{2D}$, we use a cosine similarity loss:

$$\mathcal{L} = 1 - \cos(\mathbf{F}^{2D}, \mathbf{F}^{3D}) \tag{6.2}$$

We use MinkowskiNet18A [39] as our 3D backbone $\mathcal{E}^{3D}$, and change the dimension of outputs to C.

Since the open-vocabulary image embeddings from [61, 113] are co-embedded with CLIP features, the output of our distilled 3D model naturally lives in the same embedding space as CLIP. Therefore, even without any 2D observations, such text-3D co-embeddings $\mathbf{F}^{3D}$ allow 3D scene-level understanding given arbitrary text prompts. We show such results in the ablation study in Sec. 6.4.2.

6.3.3 2D-3D Feature Ensemble

Although one can already perform open-vocabulary queries with the 2D fused features $\mathbf{F}^{2D}$ or 3D distilled features $\mathbf{F}^{3D}$, here we introduce a 2D-3D ensemble method to obtain a hybrid feature to yield better performance.

The inspiration comes from the observation that 2D fused features specialize in predicting small objects (e.g. a mug on the table) or ones with ambiguous geometry (e.g. a painting on the wall), while 3D features yield good predictions for objects with distinctive shapes (e.g. walls and floors). We aim to combine the best of both.

Our ensemble method leverages a set of text prompts, either provided at inference or offline (e.g. predefined classes from public benchmarks like ScanNet, or arbitrary classes defined by users). We first compute the embeddings for all the text prompts using the CLIP [175] text encoder $\mathcal{E}^{\text{text}}$, denoted as $\mathbf{T} = \{\mathbf{t}_1, \cdots, \mathbf{t}_N\} \in \mathbb{R}^{N \times C}$, where N is the number of text prompts and C the feature dimension. Next, for each 3D point, we obtain its 2D fused and 3D distilled embeddings $\mathbf{f}^{2D}$ and $\mathbf{f}^{3D}$ (dropping the subscript for simplicity). We can now correlate text features with these two sets of features via cosine similarity, respectively:

$$\mathbf{s}_n^{2D} = \cos(\mathbf{f}^{2D}, \mathbf{t}_n), \qquad \mathbf{s}_n^{3D} = \cos(\mathbf{f}^{3D}, \mathbf{t}_n) \tag{6.3}$$

Once having the similarity scores wrt every text prompt $\mathbf{t}_n$, we can use the max value $\mathbf{s}^{2D} = \max_n(\mathbf{s}_n^{2D})$ and $\mathbf{s}^{3D} = \max_n(\mathbf{s}_n^{3D})$ among all N prompts as the *ensemble scores* for both features. Our final 2D-3D

ensemble feature $\mathbf{f}^{2D3D}$ is simply the feature with the highest ensemble score.

6.3.4 Inference

With any per-point feature described in the previous subsections ($\mathbf{f}^{2D}$, $\mathbf{f}^{3D}$, or $\mathbf{f}^{2D3D}$) and CLIP features from an arbitrary set of text prompts, we can estimate their similarities by simply calculating the cosine similarity score between them. We use this similarity score for all of our scene understanding tasks. For example, for the zero-shot 3D semantic segmentation using 2D-3D ensemble features, the final segmentation for each 3D point is computed point-wise by $\text{argmax}_n\{\cos(\mathbf{f}^{2D3D}, \mathbf{t}_n)\}$.

6.3.5 Implementation Details

Details of 3D Distillation. We implement our pipeline in PyTorch [168]. To distill $\mathcal{E}^{3D}$, we use Adam [101] as the optimizer with an initial learning rate of $1e{-}4$ and train for 100 epochs. For MinkowskiNet we use a voxel size of 2cm for ScanNet and Matterport3D experiments, and 5cm for nuScenes. For indoor datasets, we input all points of a scene to the 3D backbone to have the full contexts, but for the distillation loss (Eq. (6.2)) we only supervise with 20K uniformly sampled point features at every iteration due to the memory constraints. For nuScenes, we input all Lidar points within the half-second segments, and only train with point features at the last time stamp. We use a batch size of 8 for ScanNet and Matterport3D with a single NVIDIA A100 (40G). For nuScenes, we use a batch size of 16 with 4 A100 GPUs. It takes around 24 hours to train, and 0.1 seconds for inference. Moreover, for all dataset we only take in the 3D point position as input to the MinkowskiNet during distillation.

Details of Feature Fusion. For Matterport3D and nuScenes, we use all images of each scene for fusion, while for ScanNet, we sample 1 out of every 20 video frames. As for the occlusion test, for dataset like ScanNet and Matterport3D where the depth map is provided, we do occlusion test to guarantee that a pixel is only paired with a *visible* surface point. For every surface point, we first find its corresponding pixel in an image, and we can obtain the distance between that pixel and 3D point. The 3D points and pixel are only paired when the difference between the distance and the depth value of that pixel is smaller than a threshold σ. The threshold σ is proportional to the depth value D. We use $\sigma = 0.2D$ for ScanNet due to the highly noisy depths and $\sigma = 0.02D$ for Matterport. For pixels with "invalid" regions of the depth map, we do not project their features to 3D points.

For nuScenes Lidar points, since no depth images are provided, no occlusion test is conducted, and we only use the synchronized images and the corresponding Lidar points on the last timestamp of a 0.5 second segment.

Prompt Engineering. Given a set of text prompts, we use a simple prompt engineering before extract CLIP text features. For each object class "XX" (except for "other") we modify the prompts to "a XX in a scene", for instance "a chair in a scene". With such a simple modification, we observe +2.3 mIoU performance boost with our LSeg ensemble model for ScanNet evaluation. We apply the trick for all our benchmark comparison experiments.

6.4 Experiments

We ran a series of experiments to test how well the proposed methods work for a variety of 3D scene understanding tasks. We start by evaluating on traditional closed-set 3D semantic segmentation benchmarks (in order to be able to compare to previous work), and later demonstrate the more novel and exciting open-vocabulary applications in the next section.

	mIoU					mAcc				
	Bookshelf	Desk	Sofa	Toilet	Mean	Bookshelf	Desk	Sofa	Toilet	Mean
3DGenZ [147]	6.3	3.3	13.1	8.1	7.7	13.4	5.9	49.6	26.3	23.8
MSeg Voting	47.8	40.3	56.5	68.8	53.4	50.1	67.7	69.8	81.0	67.2
Ours - LSeg	**67.1**	**46.4**	**60.2**	**77.5**	**62.8**	**85.5**	69.5	79.0	90.0	81.0
Ours - OpenSeg	64.1	27.4	49.6	63.7	51.2	73.7	**73.4**	**92.5**	**95.3**	**83.7**

Table 6.1: ***Comparison on Zero-shot 3D Semantic Segmentation.*** *We show quantitative comparison between our method and the most recent zero-shot 3D segmentation approach [147] and a multi-view fusion baseline utilizing MSeg [112]. Following [147], we take 4 classes (bookself, desk, sofa, toilet) out of 20 classes from ScanNet validation set for evaluation. Unlike [147], which requires training on 16 seen classes, our approach does not train with any 2D or 3D ground labels on any classes. Still, both of our variants show significantly better performance in both mIoU and mAcc.*

Datasets. To test our method in a variety of settings, we evaluate on three popular public benchmarks: ScanNet [44, 182], Matterport3D [23], and nuScenes Lidarseg [15]. These three datasets span a broad gamut of situations – the first two provide RGBD images and 3D meshes of indoor scenes, and the last provides Lidar scans of outdoor scenes. We use all three datasets to compare to alternative methods. Moreover, Matterport3D is a complex dataset with highly detailed scenes, and thus provides the opportunity to stress open-vocabulary queries. Finally, to evaluate 3D scene exploration performance, we also conduct an experiment on the 3DSSG dataset [220] that has annotations in object-level material estimation.

6.4.1 Comparisons

Zero-shot 3D semantic segmentation. We first compare OpenScene to the most closely related work on zero-shot 3D semantic segmentation: MSeg [112] Voting and 3DGenz [147]. MSeg Voting is a baseline method that we introduce, which predicts a semantic segmentation for each image using MSeg [112] with mapping to the corresponding label sets. For each 3D point, we perform majority voting of the *logits*

from multi-view images. MSeg supports a unified taxonomy of 194 classes. We use their official image semantic segmentation code[1] and their pretrained MSeg-3m-1080p model. MSeg already provided the mapping from some of 194 classes to 20 ScanNet classes, so we directly use the mapping. For Matterport3D, we simply add the mapping from "ceiling" in the MSeg labelset. As for nuScenes, we manually define the mapping from MSeg to nuScenes 16 labelsets. However, for the "construction vehicles", "traffic cone", and 'other flat', there is no mapping at all, so we set them to unknown.

3DGenZ [147] divides the 20 classes of the ScanNet dataset into 16 seen and 4 unseen classes, and trains a network utilizing the ground truth supervision on the seen classes to generate features for both sets.

Following the experimental setup in [147], we report the mIoU and mAcc values on their 4 unseen classes in Table 6.1. Our results on those classes is significantly better than [147] (7.7% vs 62.8% mIoU), even though 3DGenz [147] utilizes ground truth data for 16 seen classes and ours does not. We also outperform MSeg Voting. In this case, the difference is mainly because our method (regress CLIP features and then classify) naturally models the similarities and differences between classes, where as the MSeg Voting approach (classify and then vote) treats every class as equally distinct from all other classes (a couch and a love seat are just as different as a couch and an airplane in their model).

Comparison on 3D semantic segmentation benchmarks. In Table 6.2 we compare our approach with both fully-supervised and zero-shot methods on all classes of the nuScenes [15] validation set, ScanNet [44] validation set, and Matterport3D [23] test set. Again, we outperform the zero-shot baseline (MSeg Voting) on both mIoU and mAcc metrics all three datasets. Although we have noticeable gap to the state-of-the-art fully-supervised approaches, our zero-shot method is surprisingly competitive with fully-supervised ap-

[1] https://github.com/mseg-dataset/mseg-semantic

Figure 6.4: ***Qualitative comparisons.*** *Images of 3D semantic segmentation results on public indoor and outdoor benchmarks.*

	nuScenes		ScanNet		Matterport	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Fully-supervised methods						
TangentConv [210]	-	-	40.9	-	-	46.8
TextureNet [84]	-	-	54.8	-	-	63.0
ScanComplete [46]	-	-	56.6	-	-	44.9
DCM-Net [189]	-	-	65.8	-	-	66.2
Mix3D [154]	-	-	**73.6**	-	-	-
VMNet [78]	-	-	73.2	-	-	**67.2**
LidarMultiNet [257]	**82.0**	-	-	-	-	-
MinkowskiNet [39]	78.0	83.7	69.0	77.5	54.2	64.6
Zero-shot methods						
MSeg Voting	31.0	36.9	45.6	54.4	33.4	39.0
Ours - LSeg	36.7	42.7	**54.2**	66.6	**43.4**	53.5
Ours - OpenSeg	**42.1**	**61.8**	47.5	**70.7**	42.6	**59.2**

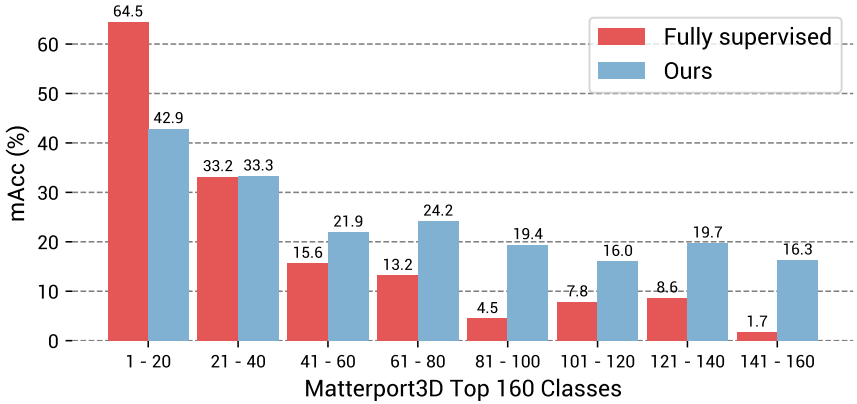
Table 6.2: ***Comparisons on Indoor and Outdoor Benchmarks.*** *We compare our method with both zero-shot and fully-supervised baselines for semantic segmentation of one outdoor dataset (nuScenes) and two indoor datasets (ScanNet and Matterport). Note that our zero-shot method performs worse than SOTA approaches trained on this data, but comparable to supervised approaches from a few years ago, and better than the previous SOTA zero-shot approach. Except for [39], the numbers for fully-supervised methods (in gray) are taken from previous papers.*

proaches from a few years ago [46, 84, 210]. Among all 3 datasets our approach has the smallest gap (only -11.6 mIoU and -8.0 mAcc) to the SOTA fully-supervised approach on Matterport3D. We conjecture the reason is that Matterport3D is more diverse, which makes the fully-supervised training harder.

Visual comparisons of semantic segmentations are shown in Fig. 6.4. They show that some of the predictions marked wrong in our results are actually either incorrect or ambiguous ground truth annotations. For example, in the first row in Fig. 6.4, we successfully segment the

	$K = 21$	$K = 40$	$K = 80$	$K = 160$
Fully-supervision [39]	**64.5**	50.8	33.4	18.4
Ours	59.2	**50.9**	**34.6**	**23.1**

(a) Results on different numbers of classes in mAcc



(b) Window evaluation on groups of 20 classes

Table 6.3: ***Impact of Increasing the Number of Object Classes.*** *Here we show (a) mAcc on Matterport3D [23] with different numbers of classes K, and (b) mAcc within a window of 20 classes ranked by decreasing numbers of examples in training set, i.e. the right-most bars represent average of the 20 classes with fewest examples (e.g., only 5 instances). Even though the fully-supervised approach [39] is trained on each labelset separately, while our model is fixed for all label sets, we can handle the less-common / long-tail classes much better.*

picture on the wall, while the GT misses it. And in the nuScenes results, the truck composed of a trailer and the truck head is segmented correctly, but the annotation is not fine-grained enough to separate the parts.

Impact of increasing the number of object classes. Besides the standard benchmarks with only a small set of classes, we also show comparisons as the number of object classes increases. We evalu-

ate on the most frequent K classes[2] of Matterport3D, where $K = 21, 40, 80, 160$. For the baseline, we train a separate MinkowskiNet for each K. However, for ours we use the *same* model for all K, since it is class agnostic.

As shown in Table 6.3 (a), when trained on only 21 classes, the fully-supervised method performs much better due to the rich 3D labels in the most common classes (wall, floor, chair, etc.). However, with the increase of the number of classes, our zero-shot approach overtakes the fully-supervised approach, especially when K gets large. The reason is demonstrated in Table 6.3 (b), where we show the mean accuracy for groups of 20 classes ranked by frequency. Fully-supervised suffers severely in segmenting tail classes because there are only a few instances available in the training dataset. In contrast, we are more robust to such rare objects since we do not rely upon any 3D labeled data.

Comparison on material estimation on 3DSSG dataset [220]. We further conduct an experiment on the 3DSSG dataset that has annotations in object-level material estimation. In Table 6.4, we compare material class predictions for the 3DSSG test set using variants of our approach trained on ScanNet and a fully-supervised MinkowskiNet. Our findings here are aligned with previous expriments: 1) 2D-3D ensembling is our best variant, 2) it underperforms fully-supervised methods for classes with abundant examples, and 3) it excels for classes with fewer examples.

6.4.2 Ablation Studies & Analysis

Does it matter which 2D features are used? We tested our method with features extracted from both OpenSeg [61] and LSeg [113]. In most experiments, we found the accuracy and generalizability of OpenSeg features to be better than LSeg (Table 6.1, Table 6.2, and

[2] $K = 21$ was from original Matterport3D benchmark. For $K = 40, 80, 160$ we use most frequent K classes of the NYU label set provided with the benchmark.

	mIoU	mAcc
Fully supervision [39]	**23.5**	30.6
Ours - 2D Fusion	18.6	31.9
Ours - 3D Model distilled on ScanNet	15.3	26.4
Ours - 2D-3D Ensemble	20.1	**35.6**

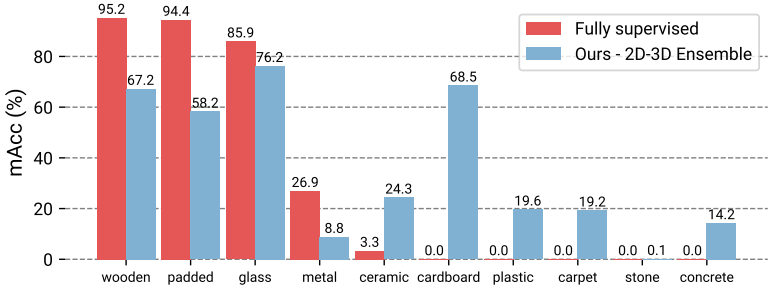


Table 6.4: ***Comparison on 3DSSG [220] in Material Estimation.*** *We report the average of 10 material classes in test set. Classes are sorted left-to-right by the number of training examples.*

		ScanNet [44]		Matterport3D [23]	
		mIoU	mAcc	mIoU	mAcc
Ours LSeg	2D Fusion	50.0	62.7	32.3	40.0
	3D Distill	52.9	63.2	41.9	51.2
	2D-3D Ens.	**54.2**	66.6	**43.4**	53.5
Ours OpenSeg	2D Fusion	41.4	63.6	32.4	45.0
	3D Distill	46.0	66.3	41.3	55.1
	2D-3D Ens.	47.5	**70.7**	42.6	**59.2**

Table 6.5: ***Ablation Study.*** *Comparison of semantic segmentation performance of different 3D features computed by our method.*

Table 6.5), so we use OpenSeg for all experiments unless explicitly stated otherwise.

Is our 2D-3D ensemble method effective? In Table 6.5, we ablate the

	3D-only model		2D-3D ensemble model	
Evaluation Datasets	ScanNet20	Matterport40	ScanNet20	Matterport40
	mIoU (%)	mAcc (%)	mIoU (%)	mAcc (%)
Distill w/ ScanNet images	46.1	37.6	47.7	46.4
Distill w/ Matterport images	38.0	47.1	47.1	50.9

Table 6.6: ***Domain Transfer with Open Vocabularies.****. These results show that it is possible to apply our models trained on ScanNet [44] to a novel 3D semantic segmentation task with a different labelset in Matterport3D [23], and vice versa. Since our trained models are task-agnostic (they predict only CLIP features), they can be applied to arbitrary label sets without retraining.*

performance for predicting features on 3D points including only image feature fusion (Sec. 6.3.1), only running the distilled MinkowskiNet (Sec. 6.3.2), and our full 2D-3D ensemble model (Sec. 6.3.3). As can be seen, on all scenarios (different datasets, metrics, and 2D features), our proposed 2D-3D ensemble model performs the best. This suggests that leveraging patterns in both 2D and 3D domains makes the ensemble features more robust and descriptive.

Can we transfer to another dataset with different labelsets? Here we investigate the ability of our trained models to handle domain transfer between 3D segmentation benchmarks with different labelsets. We train on one dataset (e.g., ScanNet20) and then test on another (e.g., Matterport40) without any retraining (Table 6.6). Since our trained model is task agnostic (it predicts only CLIP features), it does not over-fit to the classes of the training set, and thus can transfer to other datasets with different classes directly. Doing the same using a fully-supervised approach would require a sophisticated domain-transfer algorithm (e.g., [53]).

What features does our 2D-3D ensemble method use most? Here we study how our ensemble model selects among the 2D and 3D features, and investigate how it changes with increasing numbers of classes in the label set. As shown in Table 6.7, we find that the

	$K = 21$	$K = 40$	$K = 80$	$K = 160$
2D Features	28.56%	29.96%	31.58%	32.46%
3D Features	71.44%	70.04%	68.42%	67.54%

Table 6.7: ***Behavior of Ensemble Model.*** *Each entry indicates the percentage of points for which the Ensemble Model selects 2D or 3D features for semantic segmentation on Matterport3D for different numbers of classes K in the labelset.*

majority of predictions ($\sim$ 70%) select the 3D features, corroborating the value of our 3D distillation model. However, the percentage of predictions coming from 2D features increases with the number of classes, suggesting that the 2D features are more important for long-tailed classes, which tend to be smaller in both size and number of training examples.

To further study our ensemble model on how to select 2D and 3D features based on different labelsets, we visualize the results on a Matterport3D house in Fig. 6.5. First, we can notice that our ensemble model uses 3D features for those large areas like floors and walls, while 2D features are preferrable for smaller objects. Second, when comparing the feature selections using 21 and 160 classes, we can see that when the number of classes increases, our ensemble model selects more 2D features for the segmentation. The possible reason is that 2D image features can better understand those fine-grained concept than purely from 3D point clouds. For example, on the bottom-right there is a pool table there. When using 21 class labels, it is segmented as a table, so 3D features are preferrable. When using 160 class labels for 2D-3D ensemble, it is much easier to understand the concept of "pool table" using 2D images than 3D point clouds.

Ablation on multi-view fusion strategy. We ablate different multi-view feature fusion strategies in Table 6.8. Random means that having multiple features corresponding to one surface point, we randomly assign one feature to the 3D point. For Median, we take the

	Random	Median	Mean
mIoU	38.2	40.1	**41.4**
mAcc	60.1	62.2	**63.6**

Table 6.8: ***Ablation on Multi-view Fusion Strategy.*** *We report mIoU and mAcc on ScanNet [44] with our OpenSeg feature fusion.*

feature that has the smallest Euclidean distance in the feature space to all other features. As can be seen, the simple average pooling yields the best results, and we use it for all our experiments.

6.5 Applications

This section investigates new 3D scene understanding applications enabled by our approach. Since the feature vectors estimated for every 3D point are co-embedded with text and images, it becomes possible to extract information about a scene using arbitrary text and image queries. The following are just a few example applications.[3]

Open-vocabulary 3D object search. We first investigate whether it is possible to query a 3D scene database to find examples based on their names – e.g., "find a teddy bear in the Matterport3D test set." To do so, we ask a user to enter an arbitrary text string as a query, encode the CLIP embedding vector for the query, and then compute the cosine-similarity of that query vector with the features of every 3D point in the Matteport3D test set (containing 18 buildings with 406 indoor and outdoor regions) to discover the best matches. In our implementation, we return at most one match per region (i.e., room, as defined in the dataset) to ensure diversity of the retrieval results.

[3]Please note that all of these applications are zero-shot – i.e., none of them leverage any labeled data from any 3D scene understanding dataset.

Class Name	# All	# Test	# Found	# Missed	# New
Fire Extinguisher	25	3	3	0	0
Telephone	21	4	15	2	13
Exit Sign	15	5*	8	0	3
Piano	15	1	2	0	1
Ball	15	1*	4	0	3
Hat	15	1*	1	0	0
Bulletin Board	6	0	1	0	1
Globe	5	2	5	0	3
Teddy Bear	2	0	1	0	1
Toy Giraffe	1	1	1	0	0
Yellow Egg-Shaped Vase	1	0	1	0	1

Table 6.9: ***Open-vocabulary 3D Search Results.*** *Each row depicts a search of the Matterport3D test set for a class given as a text query. The columns list the # of instances in the ground truth for the whole dataset (# All), the # in the test set (# Test, counting clusters of nearby objects as one when marked with a '*'), the # of top matches found with 100% precision (# Found), the # of GT instances missed amongst those top matches (# Missed), and the # newly discovered that were not in the GT (# New).*

Fig. 6.6 shows a few example top-1 results. Most other specific text queries yield nearly perfect results. To evaluate that observation quantitatively, we chose a sampling of 10 raw categories from the ground truth set of Matterport3D, retrieved the best matching 3D points from the test set, and then visually verified the correctness of the top matches. For each query, Table 6.9 reports the numbers of instances in the test set (# Test) along with the number of instances found with 100% precision before the first mistake in the ranked list. The results are very encouraging. In all of these queries, only two ground truth instances were missed (two telephones). On the other hand, 26 instances were found among these top matches that were not correctly labeled in the ground truth, including 13 telephones. Overall, these results suggest that our open-vocabulary retrieval ap-

plication identifies these relatively rare classes at least as well as the manually labeling process did.

Fig. 6.7-Fig. 6.10 show the full set of results, where we display ranked retrieval lists, with the best match first and expected matches in parentheses. A red wireframe sphere highlights the top match. Green/red-bordered images show correct/incorrect matches, with incorrect ones ranked lower than the last ground truth instance. Gray-bordered images demonstrate near misses, not expected as matches due to their rank exceeding labeled ground truths.

We can see that the algorithm is able to retrieve very specific objects from the database with great precision. For example, when queried with "yellow egg-shaped vase," its top match is indeed a match (which was not labeled in the ground truth), and the following retrieval results are tan vase, a pumpkin, and a white egg-shaped vase with gold decorations. Similarly, when queried with "teddy bear," it retrieves two teddy bears (neither labeled in the ground truth), a stuffed monkey, and a stuffed lion among the top four matches. Among all the queries in all of the experiments, The only false positive occurred with "telephone" where a bowl of stones ranked 15th, while two ground truth instances ranked 25th and 29th. In this case, 29 of the top 30 matches were correct (20 are shown in Fig. 6.9).

These results suggest that the open-vocabulary features computed with our 2D-3D ensemble are very effective at retrieving object types with specific names. Further experiments are required to understand the limitations.

Image-based 3D object detection. We next investigate whether it is possible to query a 3D scene database to retrieve examples based on similarities to a given input image – e.g., "find points in a Matterport3D building that match this image." Given a set of query images, we encode them with CLIP image encoder, compute cosine-similarities to 2D-3D ensemble features for 3D points, and then threshold to produce a 3D object detection and mask, see Fig. 6.11.

Note that the pool table and dining table are identified correctly, even though both are types of "tables."

Open-vocabulary 3D scene understanding and exploration. Finally, we investigate whether it is possible to query a 3D scene to understand properties that extend beyond category labels. Since the CLIP embedding space is trained with a massive corpus of text, it can represent far more than category labels – it can encode physical properties, surface materials, human affordances, potential functions, room types, and so on. We hypothesize that we can use the co-embedding our 3D points with the CLIP features to *discover* these types of information about a scene.

Fig. 6.12 shows some example results for querying about physical properties, surface materials, and potential sites of activities. From these examples, we find that the OpenScene is indeed able to relate words to relevant areas of the scene – e.g., the beds, couches, and stuffed chairs match "Comfy," the oven and fireplace match "Hot," and the piano keyboard matches "Play." This diversity of 3D scene understanding would be difficult to achieve with fully supervised methods without massive 3D labeling efforts. In the authors' opinion, this is the most interesting result of this chapter of the thesis.

We add additional results in 3D scene exploration with open vocabularies. Fig. 6.13-6.18 show results for a broad range of queries, including ones that describe object categories in Fig. 6.13, room types in Fig. 6.14, activities in Fig. 6.15, colors in Fig. 6.17, materials in Fig. 6.16, and abstract concepts in Fig. 6.18. Please note the power of using language models learned via CLIP to reason about scene attributes and abstract concepts that would be difficult to label in a supervised setting. For example, searching for "store" highlights 3D points mainly on closets and cabinets (middle-right of Fig. 6.15), and searching for "cluttered" yields points in a particularly busy closet (top-right of Fig. 6.18). These examples demonstrate the power of the proposed approach for scene *understanding*, which goes far beyond semantic segmentation.

6.6 Conclusion and Discussion

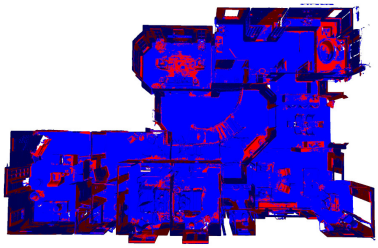
This chapter introduces a task-agnostic method to co-embed 3D points in a feature space with text and image pixels and demonstrates its utility for zero-shot, open-vocabulary 3D scene understanding. It achieves state-of-the-art for zero-shot 3D semantic segmentation on standard benchmarks, outperforms supervised approaches in 3D semantic segmentation with many class labels, and enables new open-vocabulary applications where arbitrary text and image queries can be used to query 3D scenes, all without using any labeled 3D data. These results suggest a new direction for 3D scene understanding, where foundation models trained from massive multi-modal datasets guide 3D scene understanding systems rather than training them only with small labeled 3D datasets.

Limitation and Future Works. There are several limitations of our work and still much to do to realize the full potential of the proposed approach. First, the inference algorithm could probably take better advantage of pixel features when images are present at test time using earlier fusion (we tried this with limited success). Second, the experiments could be expanded to investigate the limits of open-vocabulary 3D scene understanding on a wider variety of tasks. We evaluated extensively on closed-set 3D semantic segmentation, but provide only qualitative results for other tasks since 3D benchmarks with ground truth are scarce. In future work, it will be interesting to design experiments to quantify the success of open vocabulary queries for tasks where ground truth is not available.

Input 3D Geometry

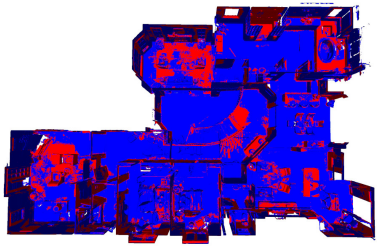
160-class label sets

Our top 21-class semantic segmentation results



2D-3D features selected for 21-class segmentation

Our top 160-class semantic segmentation results



2D-3D features selected for 160-class segmentation

Figure 6.5: ***Study of Our 2D-3D Ensemble Model.*** *We show semantic segmentation results and the feature selection of our ensemble model on a Matterport3D house. We show the comparison between the 21-class and 160-class prediction. As can be seen, when the number of classes increases, our ensemble model selects more 2D features for the segmentation. The reason can be that, when involving more fine-grained or long-tailed classes, 2D image features can better understand those fine-grained concept than purely from 3D point clouds. Points using 2D features for final segmentation are marked as red, while points with 3D features are marked as blue.*

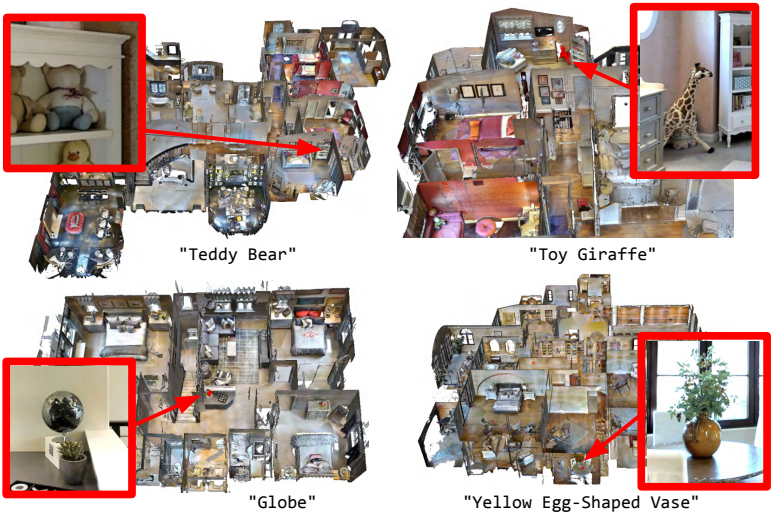


Figure 6.6: ***Open-vocabulary 3D Search.*** *These images show the 3D point within a database of 3D house models that best matches a text query. The inset image shows a zoomed view of the match.*

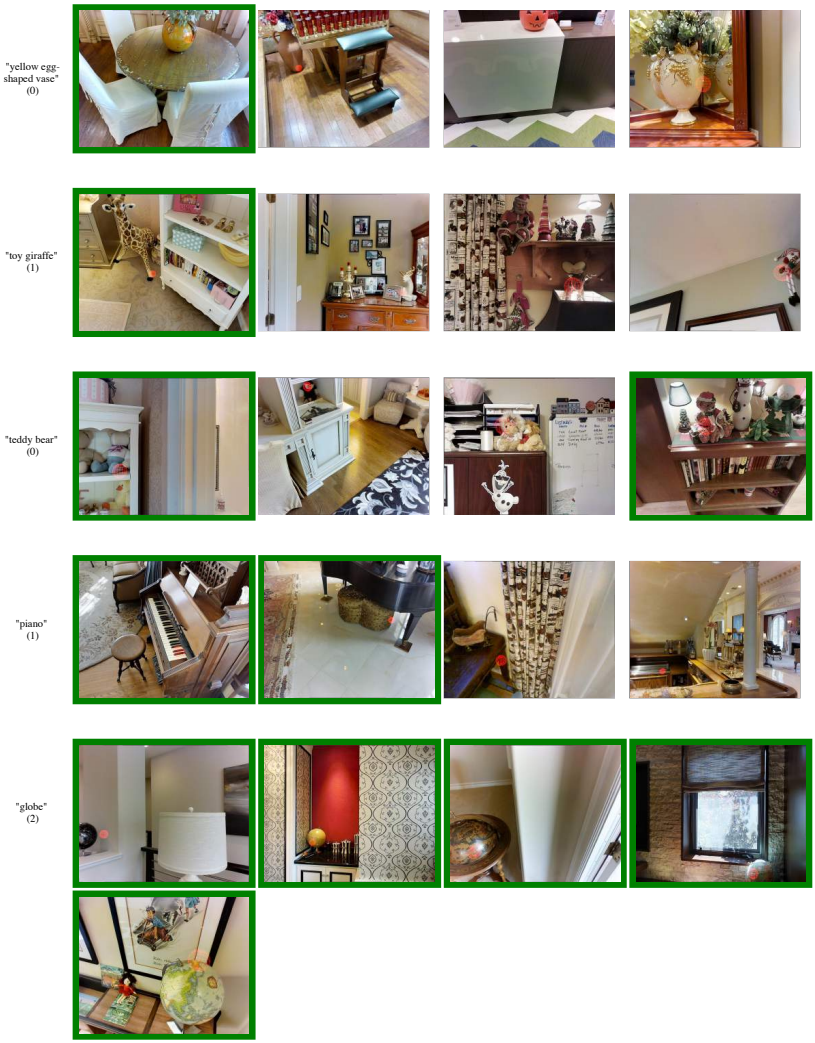


Figure 6.7: ***Example object retrieval results (page 1 of 4).*** *The query text is in the left column, with the number of ground truth instances in the Matterport test set listed in parentheses below. The images show top matching 3D points in the Matterport test set ranked from left to right (note the red wireframe sphere around the matching point in each image). Correct matches are marked with green borders. The one incorrect match is marked with a red border (in page 3 of 4). Others marked with gray borders are not wrong (since there are no further objects matching the query according to the ground truth), but are shown as examples of near matches.*

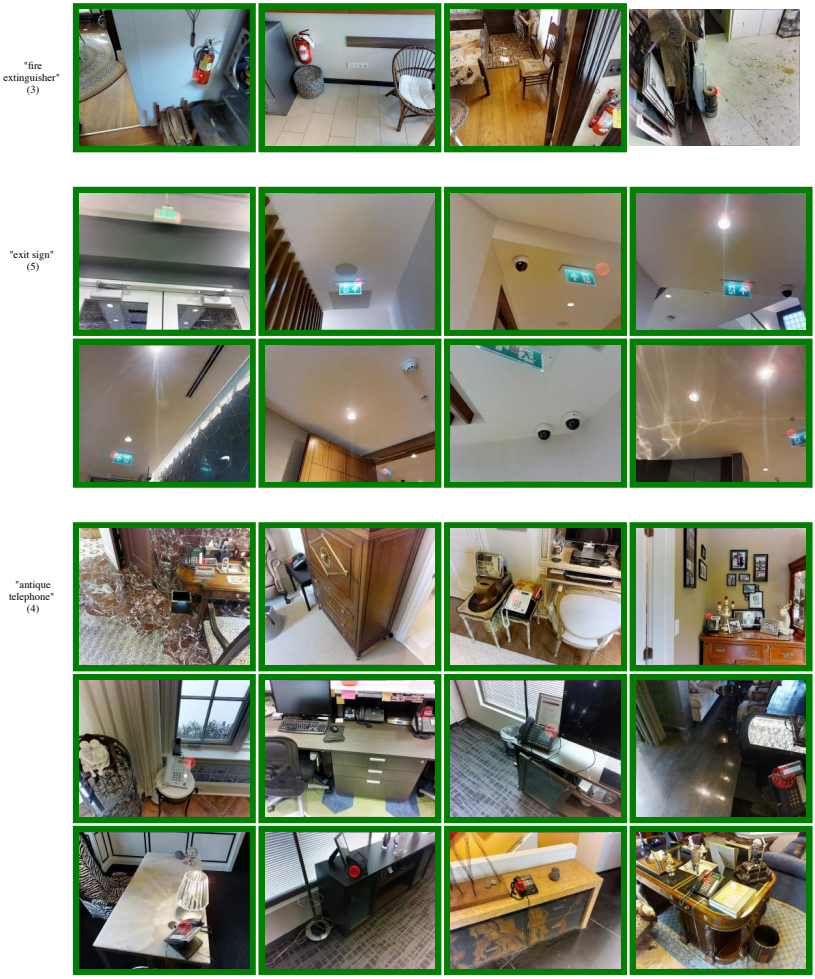


Figure 6.8: ***Example object retrieval results (page 2 of 4)****. See caption of Figure 6.7 for details.*

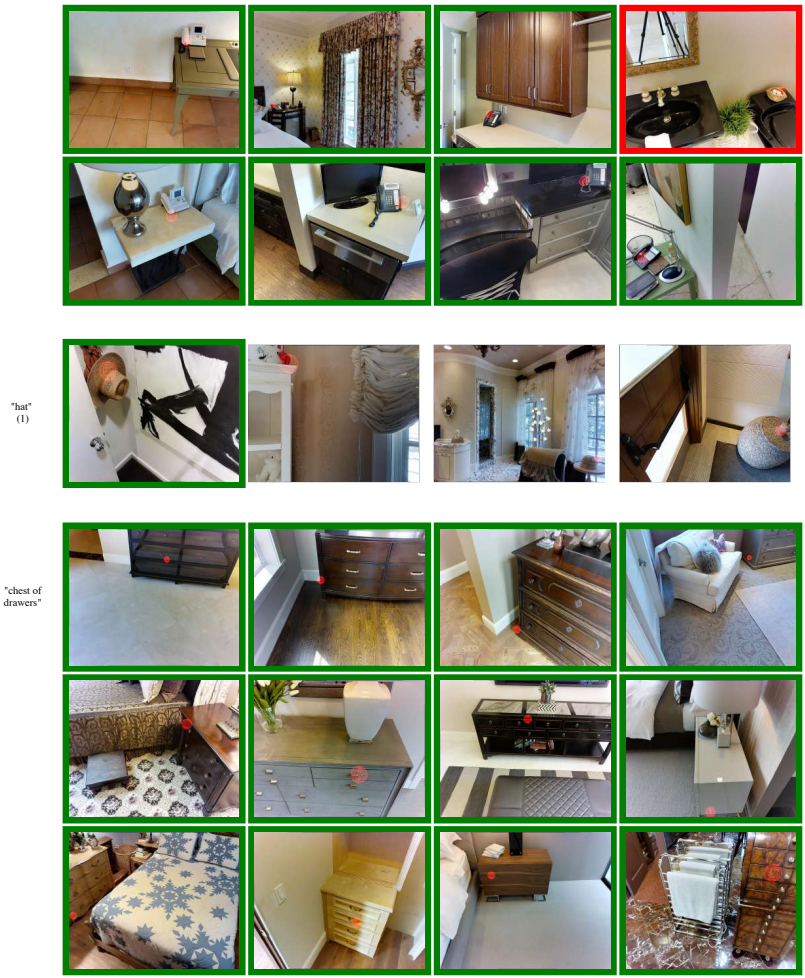


Figure 6.9: ***Example object retrieval results (page 3 of 4).*** *See caption of Figure 6.7 for details.*

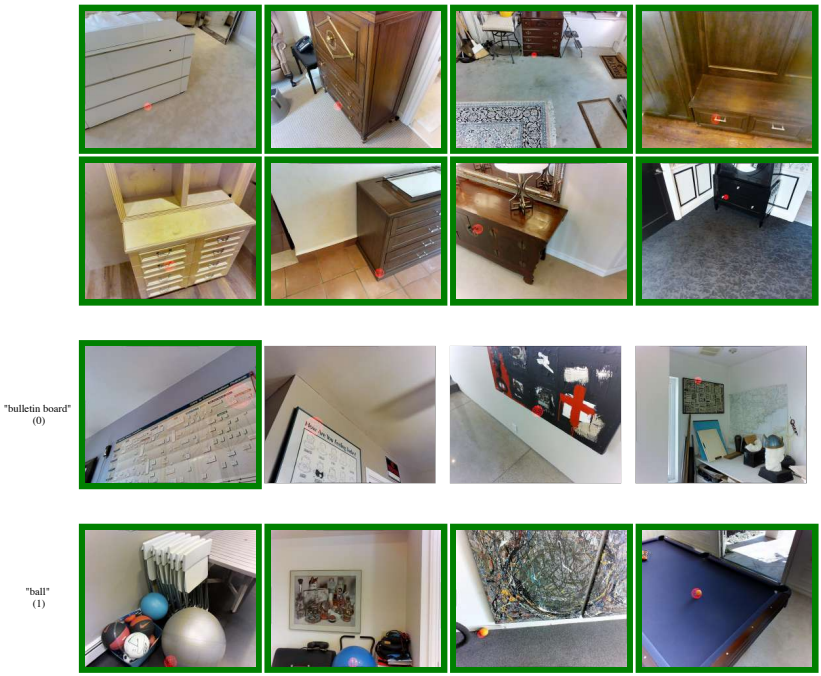


Figure 6.10: ***Example object retrieval results (page 4 of 4)***. *See caption of Figure 6.7 for details.*

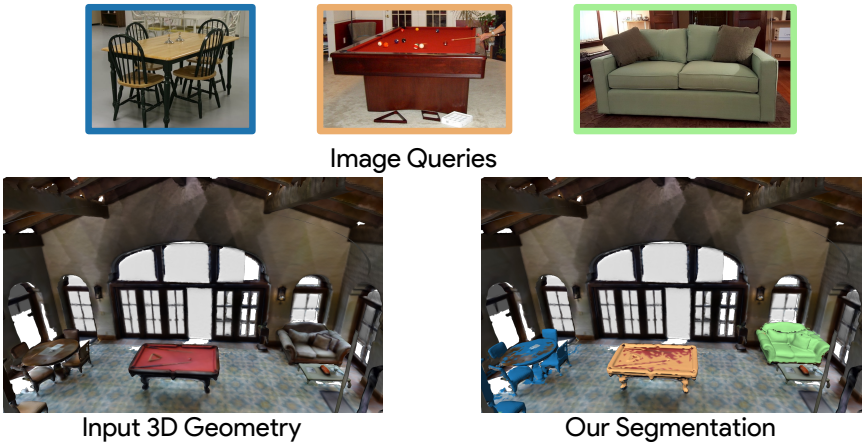


Figure 6.11: ***Image-based 3D Object Detection.*** *A 3D scene (bottom left) can be queried with images from Internet (top) to find matching 3D points (bottom right). The colors of the image query outlines indicate the corresponding matches in the 3D point cloud. All 3 images are under Creative Commons licenses.*

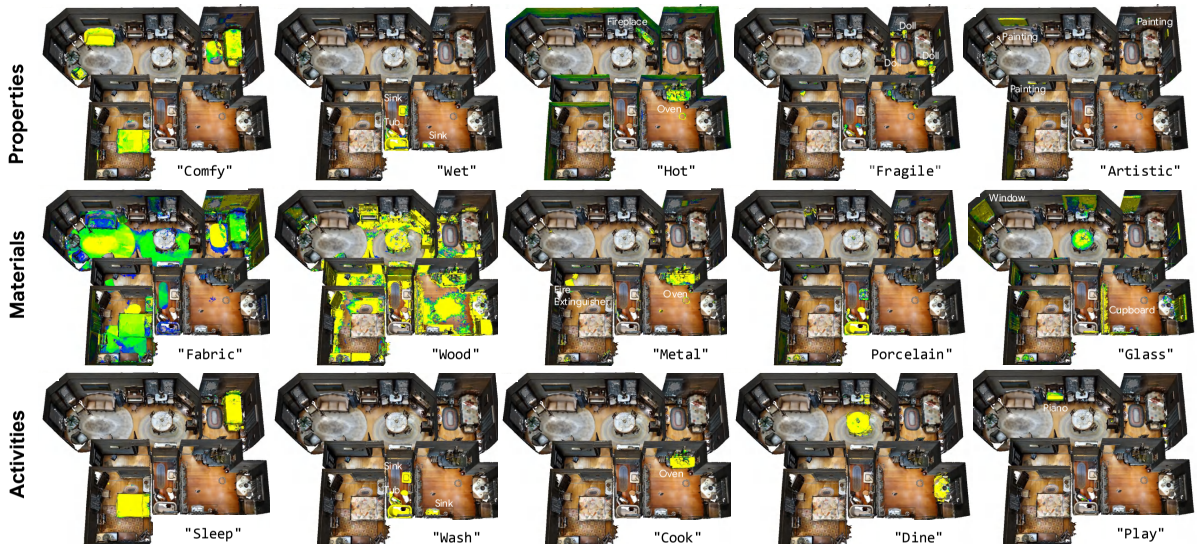


Figure 6.12: ***Open-vocabulary 3D Scene Exploration****. Examples of discovering properties, surface materials, and activity sites within a scene using open-vocabulary queries. For each example, the query text is listed below (e.g., "Comfy"), and the 3D points are colored based on their cosine similarity to the clip embedding for the query text – yellow is highest, green is middle, blue is low, and uncolored is lowest.*

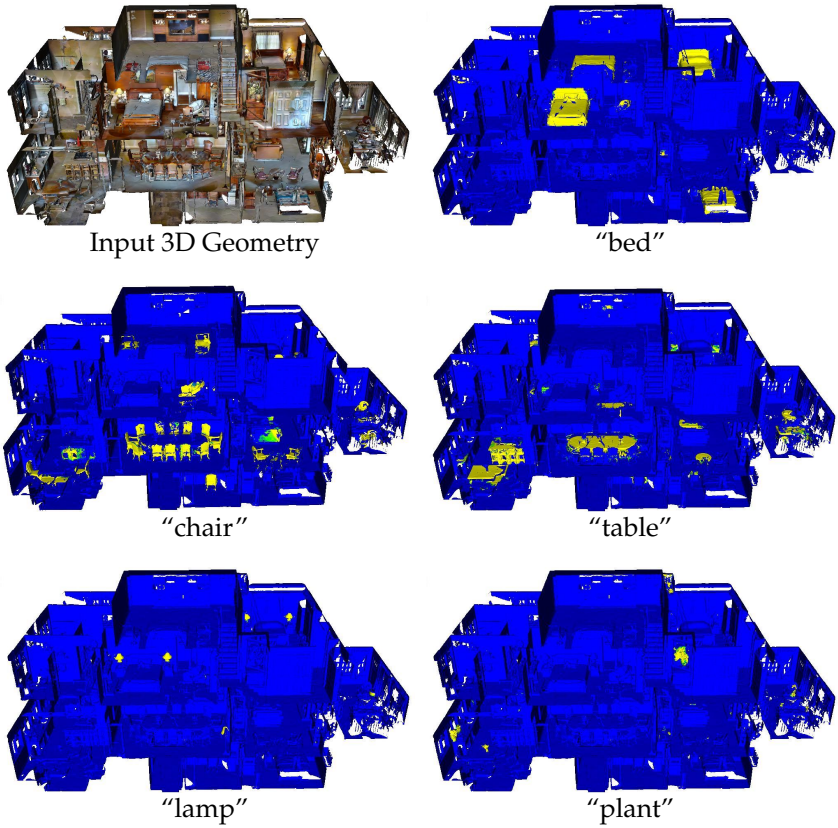


Figure 6.13: *Open-Vocabulary Queries for Common Object Types.*

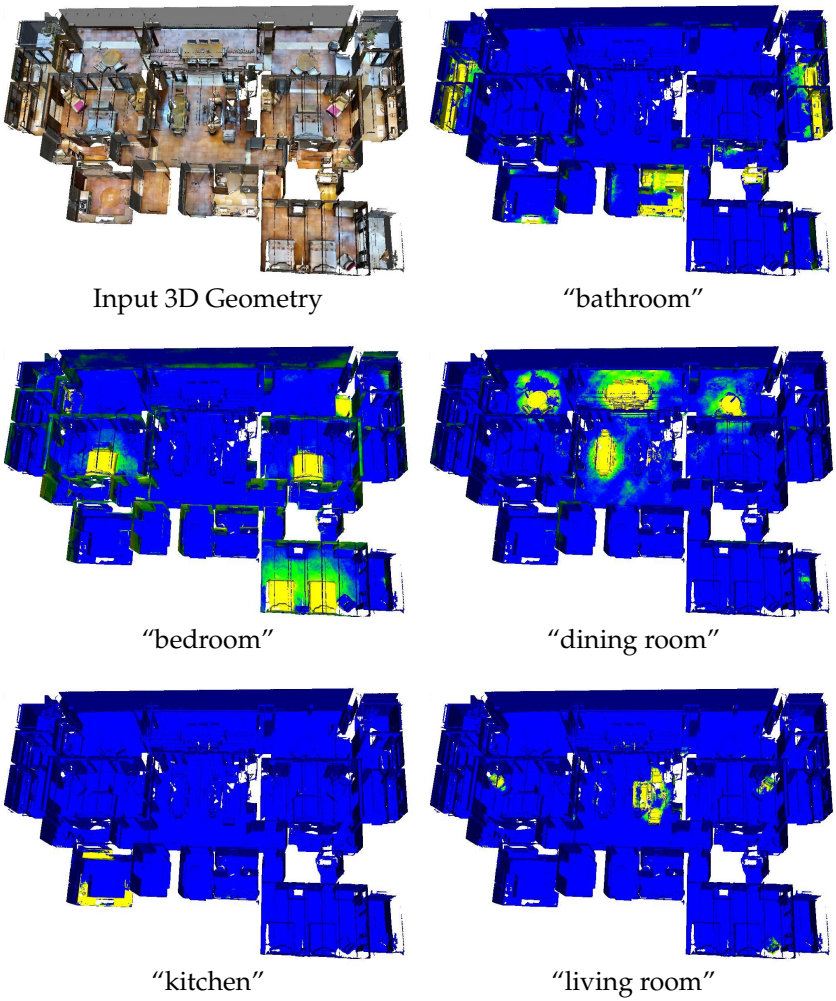


Figure 6.14: *Open-Vocabulary Queries for Room Types.*

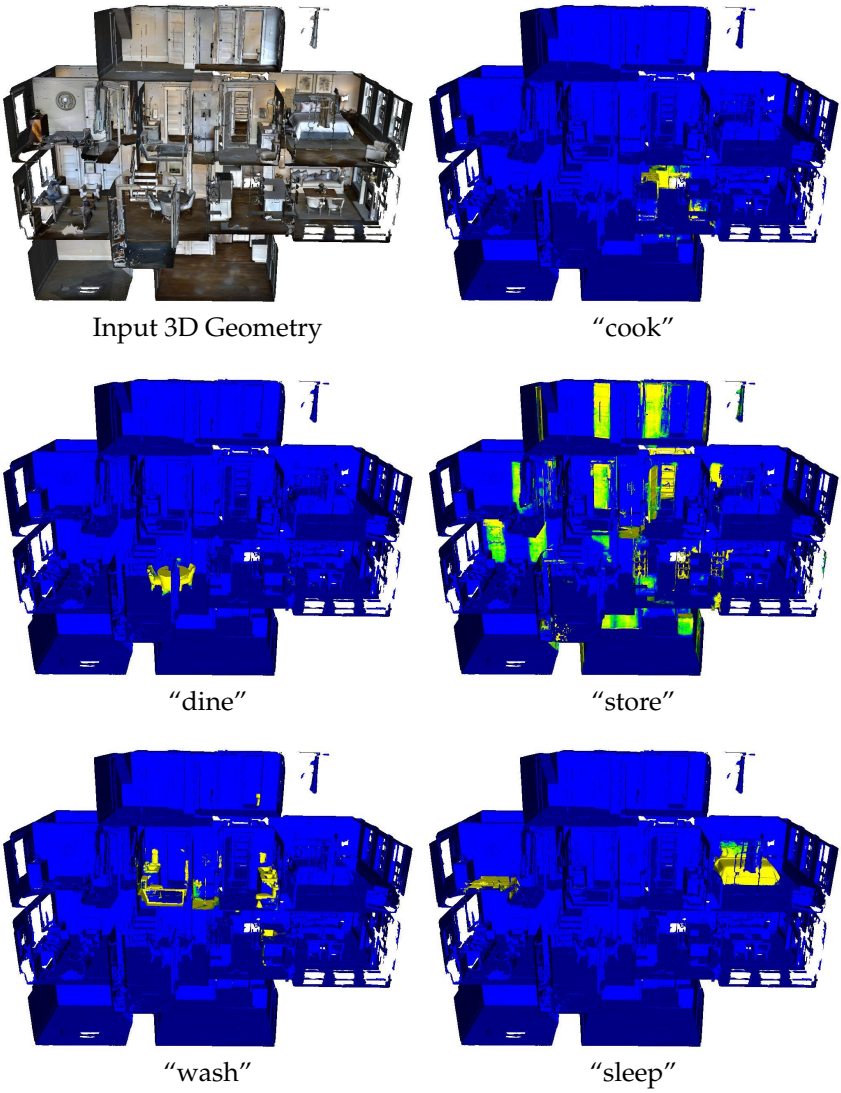


Figure 6.15: *Open-Vocabulary Queries for Activity Sites.*

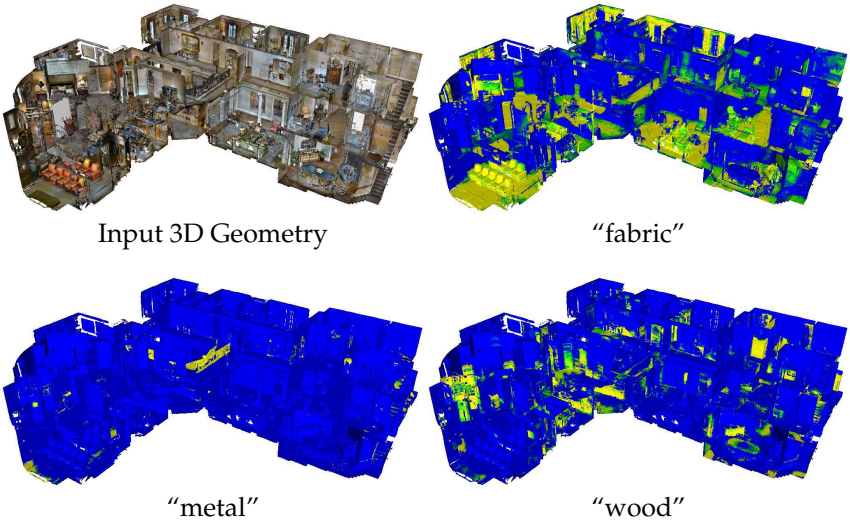


Figure 6.16: *Open-Vocabulary Queries for Materials.*

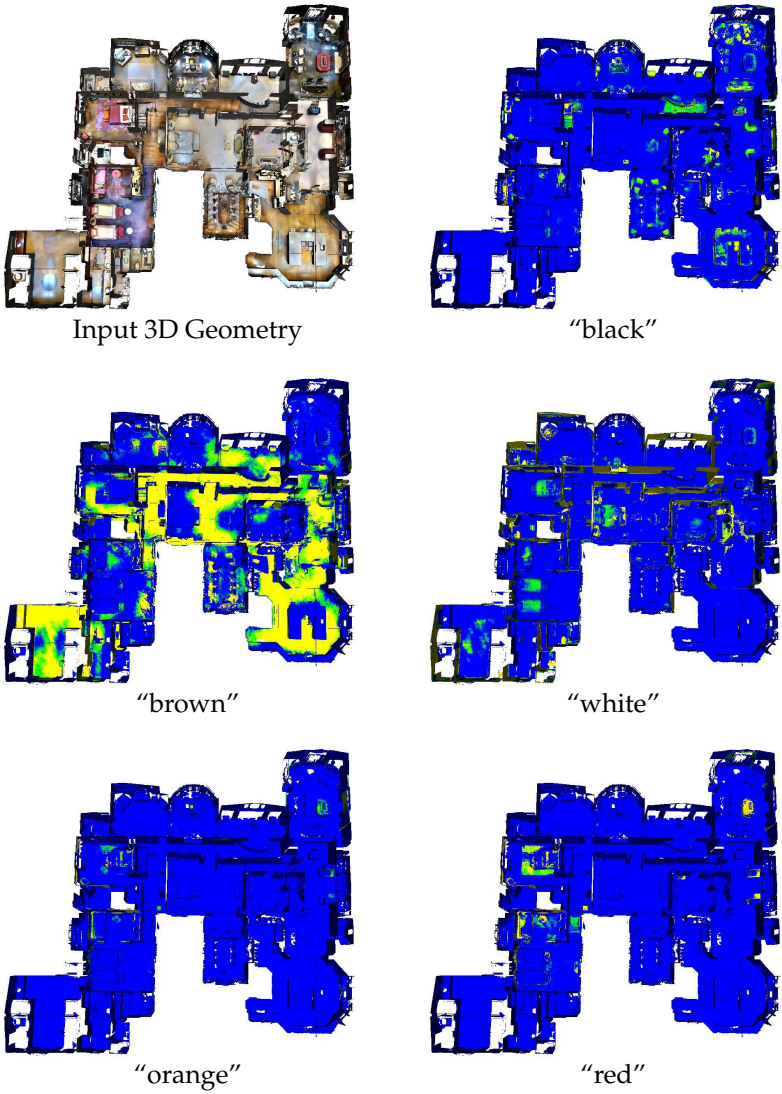


Figure 6.17: *Open-Vocabulary Queries for Colors.*

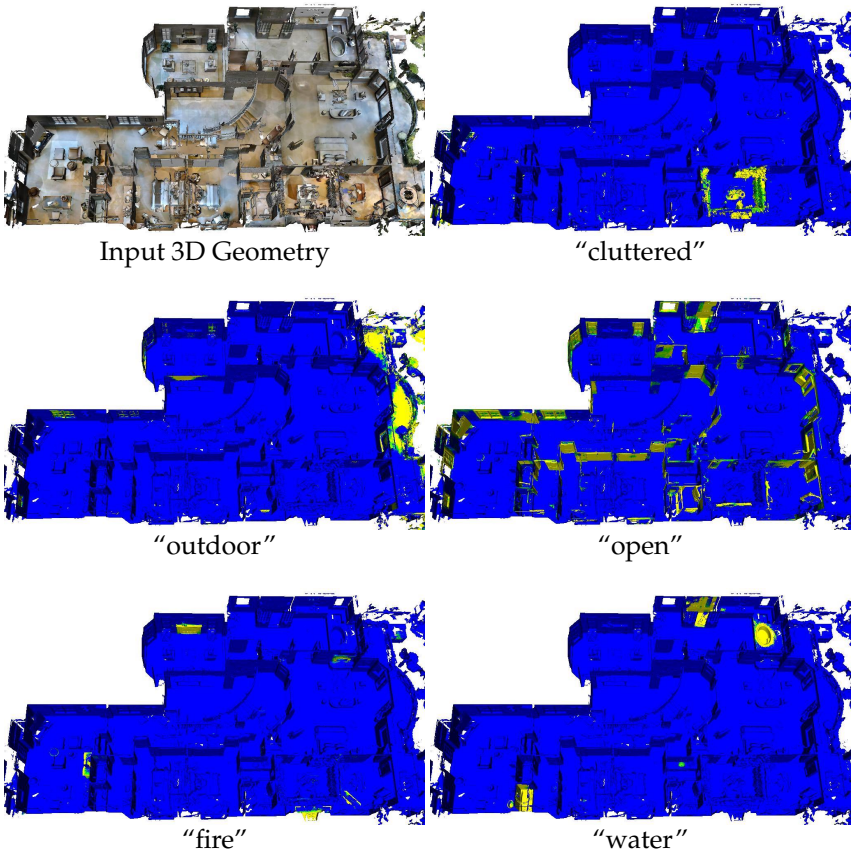


Figure 6.18: *Open-Vocabulary Queries for Abstract Concepts.*

CHAPTER 7

Conclusion

The realm of artificial intelligence is continuously evolving, with scene representation playing a pivotal role in its advancement. As AI predominantly relies on neural networks, the significance of *neural* scene representation has grown exponentially. This thesis has delved deep into this area, introducing a set of novel neural scene representations that have not only advanced the current state-of-the-art in 3D reconstruction and scene understanding but also paved the way for groundbreaking applications. In particular, the contributions made in this thesis belong to the pioneering effort to revolutionize the fields of 3D reconstruction from point clouds, dense visual SLAM, and 3D scene understandings with open vocabularies.

7.1 Core Contributions & Applications

This thesis has made seminal contributions across various facets of neural scene representation. Here's a brief overview:

- Development of a neural implicit-based 3D representation that has found applications in 3D reconstruction, generative modeling, and robot grasping.
- Addressing the challenge of slow inference speed by introducing a differentiable Poisson solver, which has been adopted for detailed mesh generation and real-time rendering.
- Proposing a hierarchical neural representation for online 3D reconstruction from unposed RGB-D sequences, which has been widely adopted in novel view synthesis and 3D surface reconstruction.
- Introducing a zero-shot approach for high-level scene understanding, inspiring research in open-set 3D maps and 3D-assisted dialog.

In detail, in Chapter 3, we introduced a neural implicit-based 3D representation that enables the fine-grained implicit 3D reconstruction of single objects, scales to large indoor scenes, and generalizes well from synthetic to real data. Noteworthy follow-up works, such as NKF [235] and NKSR [81], have leveraged our 3D feature grid encoder, achieving enhanced point cloud reconstruction quality. Additionally, distinguished works like EG3D [21] and TensoRF [25] have revolutionized the field of controllable 3D generative modelling and novel view synthesis, primarily by incorporating our efficient tri-plane representations. In a significant advancement, HexPlane [19] and *K*-Planes [54] further extend the 3 feature plane by decomposing a 4D space-time dynamic scene into 6 feature planes, leading to $100\times$ speed up in training. Furthermore, our tri-plane models have demonstrated their efficacy, particularly in enhancing the precision of robotic grasping techniques [92, 193].

In Chapter 4, we have observed that the slow inference speed of neural implicit representations is a key limitation for real-world ap-

plications. To overcome this limitation, we turned our attension to the classic yet ubiquitous point cloud representation, introducing a differentiable variant of the Poisson solver. This seamlessly bridges the explicit 3D point representation with the 3D mesh through the implicit indicator field, boosting inference speeds by one order of magnitude. Due to the unique advantages of the proposed differentiable Poisson solver, researchers have harnessed its capabilities to obtain detailed meshes from point clouds, especially those stemming from latent point diffusion models [137, 263]. Additionally, some works also directly apply SAP to multi-view reconstruction [125, 126], paving the way for fast optimization speed and real-time rendering of high-resolution images.

In Chapter 5, we delved into a more practical challenge: the online 3D reconstruction from unposed RGB-D sequences, commonly referred to as dense RGB-D SLAM. Addressing this, we introduced a hierarchical neural representation that incorporates multi-level local information, aiming not just at capturing precise 3D geometry but also at obtaining camera poses within expansive indoor environments. Compared to recent neural implicit SLAM systems, our approach stands out for its enhanced scalability, robustness, and efficiency due to the proposed hierarchical design. Notably, our proposed representation – hierarchical feature grids together with tiny MLPs – was also introduced in a seminar work Instant-NGP [150] one month after our paper appeared on arXiv. It has since gained traction, finding applications in rapid novel view synthesis [111,206], 3D surface modeling [88, 118, 260], and in the realm of dense visual RGB(D) SLAM [93, 113, 222, 280].

In Chapter 6, our focus shifted to the intricate task of high-level scene understanding within a reconstructed 3D scene. Unlike traditional approaches, where they train from labeled 3D datasets and can handle only one single task, we pioneered a zero-shot approach that eliminates the need for any labeled data and enables open-vocabulary queries. Our method has unlocked a plethora of novel 3D scene understanding tasks like rare object identification, material estimation, affordance prediction, activity estimation, room type

predictions, etc. Inspired by our work, several studies have emerged, exploring open-set 3D maps similar to ours but using SLAM [85] or NeRF [100, 215]. Furthermore, 3D-LLM [74] has expanded our foundation to new tasks such as 3D question answering, 3D-assisted dialog, and navigation.

All in all, while there is much more work to be done, we believe that the research presented in this thesis has laid a strong foundation for the future of learning-based 3D reconstruction and scene understanding.

7.2 Future Work

The rapid progress over the last years and recent breakthroughs in neural implicit representations and large language models have demonstrated a promising future in 3D reconstruction and scene understanding. In this section, we provide some interesting future directions in this domain.

Reconstructing Static Scenes in Dynamic Environments. Most methods for modeling *static* scenes often operate under the assumption of a distraction-free environment, meaning the absence of any non-persistent elements during the capture session. However, real-world scenarios are full of such dynamic elements or *distractors*. These can range from the pronounced shadows cast by operators navigating the scene to pedestrians casually walking within the camera's field of view. Overlooking these distractors during reconstruction can significantly harm the quality of the reconstruction scene. Enhancing 3D reconstruction to account for these dynamic elements can ease both the capture and post-processing phases. A recent work RobustNeRF [184] touched upon this topic but its efficacy is primarily limited to synthetic or relatively constrained scenes. Adopting recent advances in foundation models for semantic segmentation like SAM [103] for automatic distractor removal is an extremely stimulating direction.

3D Reconstruction/NeRF Meets Continual Learning. 3D reconstruction/NeRF normally only captures the scene at a certain timestamp. However, real-world environments are evolving continuously. Consider an apartment: throughout the day, the captured images might reflect shifts in object placement, the introduction of new items, or variations in lighting and weather conditions, etc. The challenge lies in enabling models to continually learn from such evolving data streams. Simply retraining models on all revealed data is resource-intensive, while updates based solely on new data lead to catastrophic forgetting, i.e. where previously learned scene geometries and appearances are lost. Moreover, Given that most scene geometries remain consistent over short time intervals, there is a pressing need for strategies that can update reconstructions/NeRFs both efficiently and locally. The recent work CLNeRF [17] integrates continual learning with NeRF, offering an initial solution to the catastrophic forgetting problem. However, its application is limited to small scenes and still struggles to efficiently address local changes. Injecting the power of foundation models into the continuous 3D reconstruction pipeline might offer a promising avenue to overcome these challenges.

Finetuning Large Vision-Language Models. As discussed in Chapter 6, our open-vocabulary model demonstrates commendable versatility across a variety of 3D scene understanding tasks. However, when it comes to specific tasks, such as 3D semantic segmentation on a fixed class set, there is a noticeable performance disparity compared to models trained exclusively on labeled data. This raises a compelling query: How can we adeptly fine-tune our model using limited data to excel in a specialized task, without compromising its inherent capabilities? This challenge not only underscores the intricacies of model adaptability but also holds significant implications for the broader realm of task-specific 3D scene understanding. One promising idea is to adapt recent works of adapting foundation models, like OFT [174] or LoRA [76].

APPENDIX A

Appendix

A.1 Derivations for Differentiable Poisson Solver

A.1.1 Point Rasterization

Given the origin of the voxel grid $\mathbf{c}_0 = (x_0, y_0, z_0)$, and the size of each voxel $\mathbf{s} = (s_x, s_y, s_z)$, we scatter the point normal values to the voxel grid vertices, weighted by the trilinear interpolation weights. For a given point $\mathbf{p}_i := (\mathbf{c}_i, \mathbf{n}_i) \in \{\mathbf{p}_i, i = 1, 2, \cdots, N\}$, with point location $\mathbf{c}_i = (x_i, y_i, z_i)$ and point normal $\mathbf{n}_i = (\hat{x}_i, \hat{y}_i, \hat{z}_i)$, we can compute the neighbor indices as $\{\mathbf{j}\}$, where $\mathbf{j} = (j_x, j_y, j_z) \in (\left\lfloor \frac{x_i - x_0}{s_x} \right\rfloor, \left\lceil \frac{x_i - x_0}{s_x} \right\rceil) \times (\left\lfloor \frac{y_i - y_0}{s_y} \right\rfloor, \left\lceil \frac{y_i - y_0}{s_y} \right\rceil) \times (\left\lfloor \frac{z_i - z_0}{s_z} \right\rfloor, \left\lceil \frac{z_i - z_0}{s_z} \right\rceil)$. Here $\lfloor \rfloor$ and $\lceil \rceil$ denote the floor and ceil operators for rounding integers. We denote the trilinear sampling weight function as $\mathcal{T}(\mathbf{c}_p, \mathbf{c}_v, \mathbf{s})$, where $\mathbf{c}_p$ and $\mathbf{c}_v$ denote the location of the point and the grid vertex. The

contribution from point $\mathbf{p}_i$ to voxel grid vertex $\mathbf{j}$ can be computed as:

$$\mathbf{v}_{j \leftarrow i} = \mathcal{T}(\mathbf{c}_i, \mathbf{s} \odot \mathbf{j} + \mathbf{c}_0, \mathbf{s})\mathbf{n}_i \tag{A.1}$$

Hence the value at grid index $\mathbf{j} \in r \times r \times r$ can be computed via summing over all neighborhood points:

$$\mathbf{v}_j = \sum_{i \in \mathcal{N}_j} \mathcal{T}(\mathbf{c}_i, \mathbf{s} \odot \mathbf{j} + \mathbf{c}_0, \mathbf{s})\mathbf{n}_i \tag{A.2}$$

where $\mathcal{N}_j$ denotes the set of point indices in the neighborhood of vertex j.

A.1.2 Spectral Methods for Solving PSR

We solve the PDEs using spectral methods [18]. In three dimensions, the multidimensional Fourier Transform and Inverse Fourier Transform are defined as:

$$\tilde{f}(\mathbf{u}) := \mathrm{FFT}(f(\mathbf{x})) = \iiint_{\infty}^{\infty} f(\mathbf{x}) e^{-2\pi i \mathbf{x} \cdot \mathbf{u}} d\mathbf{x} \tag{A.3}$$

$$f(\mathbf{x}) := \mathrm{IFFT}(\tilde{f}(\mathbf{u})) = \iiint_{\infty}^{\infty} \tilde{f}(\mathbf{u}) e^{2\pi i \mathbf{x} \cdot \mathbf{u}} d\mathbf{u} \tag{A.4}$$

where $\mathbf{x} := (x, y, z)$ are the spatial coordinates, and $\mathbf{u} := (u, v, w)$ represent the frequencies corresponding to x, y and z. Derivatives in the spectral space can be analytically computed:

$$\frac{\partial}{\partial x_j} f(\mathbf{x}) = \iiint_{\infty}^{\infty} 2\pi i x_j \tilde{f}(\mathbf{u}) e^{2\pi i \mathbf{x} \cdot \mathbf{u}} d\mathbf{u} = \mathrm{IFFT}(2\pi i x_j \tilde{f}(\mathbf{u}))$$

In discrete form, we have the rasterized point normals $\mathbf{v} := (v_x, v_y, v_z)$, where $v_x, v_y, v_z \in \mathbb{R}^n$. Hence in spectral domain, the divergence of the rasterized point normals can be written as:

$$\mathrm{FFT}(\nabla \cdot \mathbf{v}) = 2\pi i (\mathbf{u} \cdot \tilde{\mathbf{v}}) \tag{A.5}$$

The Laplacian operator can be simply written as:

$$\text{FFT}(\nabla^2) = -4\pi^2||\mathbf{u}||^2 \tag{A.6}$$

Therefore, the unnormalized solution to the Poisson Equations $\tilde{\chi}$, not accounting for boundary conditions, can be written as:

$$\tilde{\chi} = \tilde{g}_{\sigma,r}(\mathbf{u})\frac{i\mathbf{u} \odot \tilde{\mathbf{v}}}{-2\pi||\mathbf{u}||^2} \qquad \tilde{g}_{\sigma,r}(\mathbf{u}) = \exp\Big(-2\frac{\sigma^2||\mathbf{u}||^2}{r^2}\Big) \tag{A.7}$$

Where $\tilde{g}_{\sigma,r}(\mathbf{u})$ is a Gaussian smoothing kernel of bandwidth σ for grid resolution of r in the spectral domain to mitigate the ringing effects as a result of the Gibbs phenomenon from rasterizing the point normals.

The unnormalized indicator function in the physical domain χ' can be obtained via inverse Fourier Transform:

$$\chi' = \text{IFFT}(\tilde{\chi}) \tag{A.8}$$

We further normalize the indicator field to incorporate the boundary condition that the indicator field is valued at zero at point locations and valued ± 0.5 inside and outside the shapes.

$$\chi = \underbrace{\frac{m}{\text{abs}(\chi'|_{\mathbf{x}=0})}}_{\text{scale}} \underbrace{\Big(\chi' - \frac{1}{|\{\mathbf{c}\}|}\sum_{\mathbf{c}\in\{\mathbf{c}\}}\chi'|_{\mathbf{x}=\mathbf{c}}\Big)}_{\text{subtract by mean}} \tag{A.9}$$

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] Matan Atzmon and Yaron Lipman. Sald: Sign agnostic learning with derivatives. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021.

[4] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *ACM Trans. on Graphics (TOG)*, 2018.

[5] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] Sai Praveen Bangaru, Michaël Gharbi, Fujun Luan, Tzu-Mao Li, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. Differentiable rendering of neural sdfs through reparameterization. In *SIGGRAPH Asia Conference Papers*, 2022.

[7] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[8] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[9] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: registering human bodies in motion. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[13] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and

Michael Cohen. Unstructured lumigraph rendering. *ACM Trans. on Graphics (TOG)*, 2001.

[14] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Proc. Robotics: Science and Systems (RSS)*, 2013.

[15] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] Shengqu Cai, Eric Ryan Chan, Songyou Peng, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, and Gordon Wetzstein. DiffDreamer: Towards consistent unsupervised single-view scene extrapolation with conditional diffusion models. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2023.

[17] Zhipeng Cai and Matthias Mueller. Clnerf: Continual learning meets nerf. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2023.

[18] Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, and Thomas A Zang. *Spectral methods: fundamentals in single domains*. Springer Science & Business Media, 2007.

[19] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[20] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[21] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan

Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[22] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[23] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[24] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 2015.

[25] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[26] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[27] Zhiqin Chen. *Neural Mesh Reconstruction*. PhD thesis, Simon Fraser University, 2023.

[28] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Trans. on Graphics (TOG)*, 2022.

[29] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Trans. on Graphics (TOG)*, 2021.

[31] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Mitigating the hubness problem for zero-shot learning of 3d objects. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2019.

[32] Ali Cheraghian, Shafin Rahman, Dylan Campbell, and Lars Petersson. Transductive zero-shot learning for 3d point cloud classification. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[33] Ali Cheraghian, Shafin Rahman, Townim F Chowdhury, Dylan Campbell, and Lars Petersson. Zero-shot learning on 3d point cloud objects and beyond. *International Journal of Computer Vision (IJCV)*, 2022.

[34] Ali Cheraghian, Shafin Rahman, and Lars Petersson. Zero-shot learning of 3d point cloud objects. In *MVA*, 2019.

[35] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[36] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] Jaesung Choe, Sunghoon Im, François Rameau, Minjun Kang, and In So Kweon. Volumefusion: Deep depth fusion for 3d scene reconstruction. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[38] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[39] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In

Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR), 2019.

[40] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[41] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2016.

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

[43] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

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

[45] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. on Graphics (TOG)*, 2017.

[46] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] François Darmon, Bénédicte Bascle, Jean-Clément Devaux, Pascal Monasse, and Mathieu Aubry. Improving neural implicit surfaces

geometry with patch warping. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[48] Shengheng Deng, Xun Xu, Chaozheng Wu, Ke Chen, and Kui Jia. 3d affordancenet: A benchmark for visual object affordance understanding. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[49] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[50] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[51] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. Points2surf learning implicit surfaces from point clouds. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[52] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[53] Hehe Fan, Xiaojun Chang, Wanyue Zhang, Yi Cheng, Ying Sun, and Mohan Kankanhalli. Self-supervised global-local structure modeling for point cloud domain adaptation with reliable voted pseudo labels. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[54] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[55] Qiancheng Fu, Qingshan Xu, Yew Soon Ong, and Wenbing Tao. Geo-neus: Geometry-consistent neural implicit surfaces learning for

multi-view reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[56] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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

[58] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas A. Funkhouser. Local deep implicit functions for 3d shape. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[60] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. Learning 3d semantic segmentation with only 2d image supervision. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2021.

[61] Golnaz Ghiasi, Xiuye Gu, Yin Cui, and Tsung-Yi Lin. Open-vocabulary image segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[62] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[63] Shubham Goel, Georgia Gkioxari, and Jitendra Malik. Differentiable stereopsis: Meshes from multiple views using differentiable rendering. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[64] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[65] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[66] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022.

[67] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3d scene reconstruction with the manhattan-world assumption. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[68] Huy Ha and Shuran Song. Semantic abstraction: Open-world 3d scene understanding from 2d vision-language models. In *Proc. Conf. on Robot Learning (CoRL)*, 2022.

[69] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[70] Christian Häne, Sohubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2019.

[71] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: a self-prior for deformable meshes. In *ACM Trans. on Graphics (TOG)*, 2020.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[73] Yining Hong, Chunru Lin, Yilun Du, Zhenfang Chen, Joshua B Tenenbaum, and Chuang Gan. 3d concept learning and reasoning from multi-view images. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[74] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[75] Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. Iterative poisson surface reconstruction (iPSR) for unoriented points. *ACM Trans. on Graphics (TOG)*, 2022.

[76] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022.

[77] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[78] Zeyu Hu, Xuyang Bai, Jiaxiang Shang, Runze Zhang, Jiayu Dong, Xin Wang, Guangyuan Sun, Hongbo Fu, and Chiew-Lan Tai. Vmnet: Voxel-mesh network for geodesic-aware 3d semantic segmentation. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[79] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2016.

[80] Jackson Huang. U-net implementation in pytorch. https://github.com/jaxony/unet-pytorch, 2017.

[81] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler,

and Francis Williams. Neural kernel surface reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[82] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[83] Jingwei Huang, Hao Su, and Leonidas J. Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv.org*, 1802.01698, 2018.

[84] Jingwei Huang, Haotian Zhang, Li Yi, Thomas Funkhouser, Matthias Nießner, and Leonidas J Guibas. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[85] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, et al. Conceptfusion: Open-set multimodal 3d mapping. In *Proc. Robotics: Science and Systems (RSS)*, 2023.

[86] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2017.

[87] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proc. of the International Conf. on Machine learning (ICML)*, 2021.

[88] Chenxing Jiang, Hanwen Zhang, Peize Liu, Zehuan Yu, Hui Cheng, Boyu Zhou, and Shaojie Shen. H2-mapping: Real-time dense mapping using hierarchical hybrid representation. *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[89] Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas J

Guibas. Shapeflow: Learnable deformation flows among 3d shapes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[90] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[91] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[92] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. In *Proc. Robotics: Science and Systems (RSS)*, 2021.

[93] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[94] Olaf Kähler, Victor A Prisacariu, and David W Murray. Real-time large-scale dense 3d reconstruction with loop closure. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[95] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[96] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[97] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics (TOG)*, 2013.

[98] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, June 26-28, 2006*, volume 256, pages 61–70, 2006.

[99] Petr Kellnhofer, Lars C Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[100] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2023.

[101] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[102] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Machine learning (ICML)*, 2015.

[103] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2023.

[104] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *Proc. of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009.

[105] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[106] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *Proc. Conf. on Robot Learning (CoRL)*, 2021.

[107] Ravikrishna Kolluri. Provably good moving least squares. *ACM Transactions on Algorithms (TALG)*, 2008.

[108] Chen Kong, Chen-Hsuan Lin, and Simon Lucey. Using locally corresponding cad models for dense 3d reconstructions from a single image. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[109] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[110] Weicheng Kuo, Yin Cui, Xiuye Gu, AJ Piergiovanni, and Anelia Angelova. F-vlm: Open-vocabulary object detection upon frozen vision and language models. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2023.

[111] Andreas Kurz, Thomas Neff, Zhaoyang Lv, Michael Zollhöfer, and Markus Steinberger. Adanerf: Adaptive sampling for real-time rendering of neural radiance fields. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[112] John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. Mseg: A composite dataset for multi-domain semantic segmentation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[113] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2022.

[114] Jia Li and Alfred O Hero. A spectral method for solving elliptic equations for surface reconstruction and 3d active contours. In *Proc. IEEE International Conf. on Image Processing (ICIP)*, 2001.

[115] Jinke Li, Xiao He, Yang Wen, Yuan Gao, Xiaoqiang Cheng, and Dan Zhang. Panoptic-phnet: Towards real-time and high-precision lidar

panoptic segmentation via clustering pseudo heatmap. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[116] Xueting Li, Sifei Liu, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Putting humans in a scene: Learning affordance in 3d indoor environments. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[117] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[118] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[119] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[120] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[121] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[122] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[123] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey.

Barf: Bundle-adjusting neural radiance fields. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[124] Chen-Hsuan Lin, Oliver Wang, Bryan C. Russell, Eli Shechtman, Vladimir G. Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[125] Lixiang Lin, Songyou Peng, Qijun Gan, and Jianke Zhu. FastHuman: Reconstructing high-quality clothed human in minutes. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2024.

[126] Lixiang Lin, Jianke Zhu, and Yisu Zhang. Multiview textured mesh recovery by differentiable rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[127] Stefan Lionar, Daniil Emtsev, Dusan Svilarkovic, and Songyou Peng. Dynamic plane convolutional occupancy networks. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[128] Bo Liu, Shuang Deng, Qiulei Dong, and Zhanyi Hu. Language-level semantics conditioned 3d point cloud segmentation. *arXiv preprint arXiv:2107.00430*, 2021.

[129] Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulmotaleb El Saddik, Christian Theobalt, Eric Xing, and Shijian Lu. 3d open-vocabulary segmentation with foundation models. *arXiv preprint arXiv:2305.14093*, 2023.

[130] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[131] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: rendering deep implicit signed distance

function with differentiable sphere tracing. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[132] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[133] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[134] Yunze Liu, Qingnan Fan, Shanghang Zhang, Hao Dong, Thomas Funkhouser, and Li Yi. Contrastive multimodal fusion with tupleinfonce. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[135] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[136] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Trans. on Graphics (TOG)*, 1987.

[137] Zhaoyang Lyu, Jinyi Wang, Yuwei An, Ya Zhang, Dahua Lin, and Bo Dai. Controllable mesh generation through sparse latent point diffusion models. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[138] Chaofan Ma, Yuhuan Yang, Yanfeng Wang, Ya Zhang, and Weidi Xie. Open-vocabulary semantic segmentation with frozen vision-language models. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2022.

[139] Qianli Ma, Shunsuke Saito, Jinlong Yang, Siyu Tang, and Michael J Black. Scale: Modeling clothed humans with a surface codec of ar-

ticulated local elements. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[140] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[141] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[142] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[143] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017.

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

[145] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[146] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[147] Björn Michele, Alexandre Boulch, Gilles Puy, Maxime Bucher, and Renaud Marlet. Generative zero-shot learning for semantic segmentation of 3d point clouds. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2021.

[148] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[149] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[150] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 2022.

[151] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[152] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 2017.

[153] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[154] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context data augmentation for 3d scenes. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2021.

[155] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie

Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[156] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2011.

[157] Michael Niemeyer. *Neural Scene Representations for 3D Reconstruction and Generative Modeling*. PhD thesis, Eberhard Karls University of Tübingen, 2023.

[158] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2021.

[159] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[160] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

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

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

[163] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. on Graphics (TOG)*, 2013.

[164] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[165] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[166] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[167] Despoina Paschalidou, Osman Ulusoy, Carolin Schmitt, Luc Van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[168] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[169] Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser. OpenScene: 3d scene understanding with open vocabularies. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[170] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[171] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[172] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[173] Charles R. Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[174] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[175] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proc. of the International Conf. on Machine learning (ICML)*, 2021.

[176] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu. Denseclip: Language-guided dense prediction with context-aware prompting. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[177] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny mlps. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[178] Edoardo Remelli, Artem Lukoianov, Stephan R Richter, Benoît Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[179] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctNetFusion: Learning depth fusion from data. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[180] Damien Robert, Bruno Vallet, and Loic Landrieu. Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[181] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[182] David Rozenberszki, Or Litany, and Angela Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[183] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017.

[184] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[185] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[186] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[187] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-lidar self-supervised dis-

tillation for autonomous driving data. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[188] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: bundle adjusted direct RGB-D SLAM. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[189] Jonas Schult, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dualconvmesh-net: Joint geodesic and euclidean convolutions on 3d meshes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[190] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[191] Silvia Sellán and Alec Jacobson. Stochastic poisson surface reconstruction. *ACM Trans. on Graphics (TOG)*, 2022.

[192] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022.

[193] Bokui Shen, Zhenyu Jiang, Christopher Choy, Leonidas J Guibas, Silvio Savarese, Anima Anandkumar, and Yuke Zhu. Acid: Action-conditional implicit visual dynamics for deformable object manipulation. In *Proc. Robotics: Science and Systems (RSS)*, 2022.

[194] Vincent Sitzmann. *Self-supervised Scene Representation Learning*. PhD thesis, Stanford University, 2020.

[195] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[196] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persis-

tent 3d feature embeddings. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[197] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[198] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The Replica dataset: A digital replica of indoor spaces. *arXiv*, 2019.

[199] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[200] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[201] Edgar Sucar, Kentaro Wada, and Andrew Davison. Nodeslam: Neural object descriptors for multi-view shape reconstruction. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2020.

[202] Jiaming Sun, Xi Chen, Qianqian Wang, Zhengqi Li, Hadar Averbuch-Elor, Xiaowei Zhou, and Noah Snavely. Neural 3d reconstruction in the wild. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022.

[203] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[204] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving:

Waymo open dataset. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[205] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[206] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023.

[207] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018.

[208] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. Sa-convonet: Sign-agnostic optimization of convolutional occupancy networks. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[209] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2017.

[210] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[211] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[212] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[213] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[214] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[215] Nikolaos Tsagkas, Oisin Mac Aodha, and Chris Xiaoxuan Lu. Vl-fields: Towards language-grounded neural implicit spatial representations. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023.

[216] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[217] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[218] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2014.

[219] Vibhav Vineet, Ondrej Miksik, Morten Lidegaard, Matthias Nießner, Stuart Golodetz, Victor A Prisacariu, Olaf Kähler, David W Murray, Shahram Izadi, Patrick Pérez, et al. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2015.

[220] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions.

In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[221] Dan Wang, Xinrui Cui, Xun Chen, Zhengxia Zou, Tianyang Shi, Septimiu Salcudean, Z Jane Wang, and Rabab Ward. Multi-view 3d reconstruction with transformers. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[222] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[223] Jiashun Wang, Huazhe Xu, Jingwei Xu, Sifei Liu, and Xiaolong Wang. Synthesizing long-term 3d human motion and interaction in 3d scenes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[224] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[225] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[226] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics*, 2017.

[227] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[228] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection

from multi-view images via 3d-to-2d queries. In *Proc. Conf. on Robot Learning (CoRL)*, 2022.

[229] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 2019.

[230] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf–: Neural radiance fields without known camera parameters. *arXiv*, 2021.

[231] Silvan Weder, Johannes L Schonberger, Marc Pollefeys, and Martin R Oswald. Neuralfusion: Online depth fusion in latent space. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[232] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[233] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *RSS '12 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[234] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Proc. Robotics: Science and Systems (RSS)*, 2015.

[235] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[236] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[237] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks.

In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[238] Adrian Wolny. 3d u-net model for volumetric semantic segmentation written in pytorch. https://github.com/wolny/pytorch-3dunet, 2020.

[239] Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. Multi-view mesh reconstruction with neural deferred shading. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[240] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[241] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman, and Joshua B Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[242] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[243] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[244] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2019.

[245] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann,

and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, 2022.

[246] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[247] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Open-vocabulary panoptic segmentation with text-to-image diffusion models. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[248] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[249] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[250] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[251] Xu Yan. Pytorch implementation of pointnet and pointnet++. https://github.com/yanx27/Pointnet_Pointnet2_pytorch, 2020.

[252] Zike Yan, Yuxin Tian, Xuesong Shi, Ping Guo, Peng Wang, and Hongbin Zha. Continual neural mapping: Learning an implicit scene representation from sequential observations. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[253] Jihan Yang, Runyu Ding, Zhe Wang, and Xiaojuan Qi. Regionplc: Regional point-language contrastive learning for open-world 3d scene understanding. *arXiv preprint arXiv:2304.00962*, 2023.

[254] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[255] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[256] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[257] Dongqiangzi Ye, Zixiang Zhou, Weijia Chen, Yufei Xie, Yu Wang, Panqu Wang, and Hassan Foroosh. Lidarmultinet: Towards a unified multi-task network for lidar perception. *arXiv preprint arXiv:2209.09385*, 2022.

[258] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. Gifs: Neural implicit function for general shape representation. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[259] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[260] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[261] Nir Zabari and Yedid Hoshen. Open-vocabulary semantic segmentation using test-time distillation. In *ECCVW*, 2022.

[262] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014.

[263] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[264] Yihan Zeng, Chenhan Jiang, Jiageng Mao, Jianhua Han, Chaoqiang Ye, Qingqiu Huang, Dit-Yan Yeung, Zhen Yang, Xiaodan Liang, and Hang Xu. Clip2: Contrastive language-image-point pretraining from real-world point cloud data. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[265] Jason Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-view 3d reconstruction in the wild. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[266] Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Critical regularizations for neural surface reconstruction in the wild. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[267] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2021.

[268] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv*, 2020.

[269] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[270] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[271] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[272] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[273] Wenbin Zhao, Jiabao Lei, Yuxin Wen, Jianguo Zhang, and Kui Jia. Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[274] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[275] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[276] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022.

[277] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[278] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[279] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

[280] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys. NICER-SLAM: Neural

implicit scene encoding for rgb slam. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2024.

[281] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural implicit scalable encoding for slam. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.