Deep Learning Over Spatial Data: From a 3D Reconstructive Perspective

ZiChuan Zhao

A thesis submitted for the Degree of

Doctor of Philosophy in Computer Science

School of Computer Science and Electronic Engineering

University of Essex

June, 2025

Abstract

The ability to create and interact with high-fidelity digital representations of the physical world is crucial for applications ranging from digital twins to autonomous navigation. Yet, the robust interpretation and reconstruction of 3D environments from sparse or noisy observations remains a significant challenge for artificial intelligence. This thesis addresses the fundamental problems in 3D reconstruction by exploring the evolving landscape of deep learning over spatial data, from structured parametric models to flexible implicit representations. Our research navigates the trade-offs between these paradigms to develop novel methods that enhance reconstruction fidelity, efficiency, and user control.

The investigation begins by confronting the limitations of explicit parametric methods in handling complex topologies from unstructured point clouds. This analysis motivates a pivot towards implicit neural representations. Our work introduces key innovations in this area, including Seed-Net, an interactive framework that differentiably incorporates sparse user guidance to refine local geometric details in neural fields, and NeuLap, a geometry-aware training scheme that leverages a learned Laplacian prior to refine the convergence process and improve the reconstruction of sharp features from limited data. Building on these insights, the research path leads to the development of a general-purpose backbone for large-scale 3D learning: a Hierarchical Attention OctTree. This architecture introduces a novel attention propagation mechanism that efficiently captures multi-scale spatial context, demonstrating competitive performance and memory efficiency.

Collectively, these contributions offer a suite of methods and a conceptual roadmap for 3D spatial learning. Through advancements in interactive reconstruction, prior-guided optimization, and scalable deep learning architectures, this research contributes to the field of representing and reconstructing high-fidelity 3D spatial data, providing technical insights and solutions that are valuable for various future applications in fields such as digital twins, robotics, and creative design.

Contents

Acknowledgements

I would like to express my sincere gratitude to a number of people who have helped me throughout this doctoral journey.

My deepest gratitude goes to my supervisor, Professor Adrian Clark. His insightful guidance, unwavering encouragement, and rigorous academic standards were indispensable. His mentorship not only shaped the direction of this thesis but also my personal growth as a researcher.

I also wish to thank my former supervisor, Michael Gardner. Despite his retirement before the project's completion, his initial guidance was crucial in laying its foundations. I appreciate his early contributions and wish him a happy retirement.

In addition, very importantly, this research was made possible by the generous financial support and industrial collaboration of BT Group Inc. I am particularly grateful for the guidance from my industrial supervisor, Anasol Peña-Rios. Her perspectives, creative ideas on research methods, and continuous support provided a vital link between academic inquiry and industry practice, greatly enriching this work.

My heartfelt thanks also go to my colleagues and friends for fostering a stimulating and supportive environment.

CHAPTER 1

Introduction and Research Overview

In this chapter, we provide an overview of the research problem, motivations, and objectives in the field of 3D spatial learning and reconstruction. We begin by examining the fundamental rationale behind reconstruction models and their role in spatial learning (1.1), followed by a systematic analysis of key challenges in 3D reconstruction and spatial learning (2.2). These challenges span across multiple aspects including representation, computational efficiency, geometric fidelity, interactive control, and learning generalization. Building upon these challenges, we present our research objectives (1.3) that aim to advance the state-of-the-art in 3D reconstruction through innovative approaches in parametric reconstruction, user-controllable refinement, computational efficiency, and scalable learning frameworks. Finally, we outline our main contributions (1.4) that address these objectives through novel methodologies and architectural innovations in 3D spatial learning.

1.1 Rationale: From Reconstruction Models to 3D Spatial Learning

In the realm of machine learning and artificial intelligence, the ability to understand and represent data is fundamental to various applications. Two prominent approaches have emerged for this purpose: generative models and reconstruction models. This section systematically analyzes these two paradigms and their respective advantages in learning

data representations, with a particular focus on 3D spatial data and reconstruction tasks.

Generative models aim to learn either the joint probability distribution $P(X, Y)$ of the data and labels, or the data distribution $P(X)$ itself, where X represents the input space and Y the label space. Unlike discriminative models that directly learn the conditional distribution $P(Y|X)$, generative models can derive this conditional probability when modeling $P(X, Y)$, or focus solely on modeling the underlying data distribution $P(X)$ as seen in modern deep generative models. This fundamental approach enables generative models to:

- Learn the complete data manifold
- Generate novel samples that follow the learned distribution
- Perform tasks such as image synthesis and style transfer

Reconstruction models, in contrast, focus on learning a mapping from corrupted or incomplete data to the original data space. This task requires the model to:

- Learn the prior distribution of the data
- Model the relationship between corrupted and original data
- Preserve essential information during the reconstruction process

The challenge lies in accurately recovering the full data from partial information, which necessitates a deep understanding of the data's structure and semantics. In the context of 3D reconstruction, these challenges are further amplified:

- Handling noisy and sparse inputs
- Recovering fine geometric details while maintaining global structure
- Ensuring computational efficiency for real-time applications

The key distinction between these two paradigms lies in their learning objectives. While generative models focus on capturing the complete data distribution, reconstruction models must additionally learn the mapping from corrupted to original data. This additional requirement often leads to more robust feature representations, as the model must preserve essential information while filtering out noise. However, generative models excel in tasks requiring novel sample generation

In the context of 3D spatial learning, both paradigms face unique challenges that require specialized solutions:

- **Representation Efficiency**: Developing compact yet expressive representations for complex 3D geometric structures
- **Computational Scalability**: Optimizing computational resources for processing large-scale 3D spatial data
- **Geometric Fidelity**: Preserving intrinsic geometric properties throughout the reconstruction pipeline
- **Interactive Control**: Integrating user-guided feedback mechanisms for iterative refinement

These challenges motivate our research in developing novel approaches that combine the strengths of both paradigms while addressing the specific requirements of 3D spatial learning.

Building upon the discussed challenges and methodologies, our research aims to advance the field of 3D reconstruction. We delve into the intricacies of this domain to deepen the understanding of deep learning over spatial data. Our work confronts the key challenges of representation efficiency, computational scalability, geometric fidelity, and interactive control. We focus on developing innovative approaches that harness the strengths of reconstruction models to create robust feature representations and accurate reconstructions. Specifically, we aim to develop models capable of both learning the complete data distribution and accurately restoring original data from corrupted inputs. By contributing novel solutions that meet these demanding requirements, this thesis seeks to push the boundaries of 3D reconstruction and spatial data representation.

1.2 Challenges in 3D Reconstruction and 3D Spatial Learning

This section analyzes the key challenges in 3D reconstruction and spatial learning, focusing on five main aspects: representation, computational efficiency, geometric fidelity, interactive control, and learning generalization. These challenges form the

foundation for our research objectives and drive the development of novel solutions presented in this thesis.

1.2.1 Representation Challenges

One of the fundamental challenges in 3D reconstruction lies in developing appropriate representations for complex spatial data. This includes learning data representations for the input data and choosing the appropriate representations for the reconstruction output.

Sparse and Noisy Data Handling

Effectively processing and utilizing sparse and noisy input data is a primary challenge in 3D spatial learning. This involves developing robust algorithms that can handle incomplete information and measurement uncertainties while still producing reliable reconstructions.

These input data may include point clouds, single-view, multi-view, and RGB-D images, videos, and other modalities. In our corresponding research, we focus on point cloud data and multi-view images to leverage learned priors for reconstructing from those input data with different approaches.

Representation Trade-offs

Finding the optimal balance between compactness and expressiveness in representations remains a significant challenge. Compact representations are computationally efficient but may lose important details, while highly expressive representations can capture fine details but may be impractical for real-world applications.

For learning representations for 3D input data specifically, a model that generalizes well may tend to oversmooth the input data, leading to a loss of important local geometric structures during the reconstruction process.

Explicit vs. Implicit Representations

The choice between explicit and implicit representations presents distinct advantages and limitations. Explicit representations offer direct control and interpretability but

may struggle with complex topologies, while implicit representations handle topology changes naturally but may lack precise control over surface properties.

In our research, we investigated both explicit and implicit representations. For explicit representations, we narrow down our research to parametric explicit representations to produce an interpretable, interactable, and generalisable 3D reconstruction model in a reversed engineering manner. For implicit representations, we aim to develop generalisable 3D reconstruction models based on learned priors.

Semantic Information Preservation

Maintaining semantic information during the reconstruction process is crucial for many applications. This involves preserving object identities, relationships, and contextual information while performing geometric reconstruction. In our research, we focus on the semantic information on the surfaces, including the preservation of information of local geometric structures and their corresponding semantic meanings.

1.2.2 Computational Efficiency Challenges

As 3D reconstruction applications scale up, computational efficiency becomes crucial.

Scalability for Large-Scale Data

Processing large-scale 3D spatial data requires efficient algorithms and data structures that can handle massive point clouds, high-resolution meshes, and complex scene representations without compromising accuracy or performance.

Algorithmic Efficiency Optimization

Optimizing algorithmic efficiency and reducing data requirements at either the learning or inference stage is crucial for practical applications. This includes developing methods that can learn effectively from limited training data and implementing efficient inference strategies for deep learning models.

In our research, we try to enhance the training efficiency of the proposed models from these three aspects:

- Integrating and transferring priors that guide the learning process

- Enabling human users to selectively provide guidance during the inference process
- Designing efficient algorithms for multi-scale spatial data that pass information efficiently across local and global features.

Multi-scale Information Propagation

Efficient information propagation across multi-scale spatial data requires sophisticated algorithms that can effectively handle different levels of detail while maintaining consistency across scales.

1.2.3 Geometric Fidelity Challenges

Preserving geometric properties during reconstruction is a crucial aspect of 3D reconstruction. Current neural network-based implicit 3D reconstruction models are able to produce high-quality view synthesis results. However, the iso-surface of the implicit field carries little information about the underlying geometric properties of the reconstructed surface, thus leading to the loss of geometric details and geometrical semantics such as edges and corners.

Global-Local Structure Balance

Balancing global structure with local detail preservation requires sophisticated algorithms that can maintain the overall shape while accurately capturing fine-scale features. This involves developing multi-scale approaches that can effectively handle both aspects simultaneously.

Furthermore, when dealing with noisy or sparse input data as discussed above, the reconstruction process faces a significant challenge in distinguishing signal from noise and subtle geometric details.

Surface Detail Reconstruction

Accurate reconstruction of both smooth surfaces and sharp features presents unique challenges. Different surface types require different handling strategies, and maintaining both types of features in a single reconstruction is particularly challenging.

Geometric Constraint Preservation

Maintaining geometric constraints and priors throughout the reconstruction process ensures the physical validity and usability of the results. This includes preserving symmetries, parallelism, and other important geometric relationships.

1.2.4 Interactive Control and Refinement Challenges

Enabling effective user interaction presents a challenge during the 3D reconstruction process, as it requires the system to balance the automated processing with the manual adjustments, and the system must interpret and apply user intentions while maintaining reconstruction quality.

This is essential for the industry applications, as the 3D reconstruction models may not always produce the desired results in an end-to-end manner, and the user may need to provide additional information to guide the reconstruction process.

Interface Design

Designing intuitive interfaces for human-in-the-loop reconstruction requires careful consideration of user experience and workflow efficiency. The interface must provide powerful controls while remaining accessible to users with varying levels of expertise.

Real-time Feedback

Implementing real-time feedback mechanisms is essential for interactive systems. Users need immediate visual feedback to understand the impact of their actions and make informed decisions during the reconstruction process.

Progressive Refinement

Developing progressive refinement strategies allows users to iteratively improve results. This involves creating algorithms that can incrementally update reconstructions based on new input while maintaining stability and consistency.

User Input Integration

Effectively incorporating user input and guidance requires robust algorithms that can balance automated processing with manual adjustments. The system must interpret and apply user intentions while maintaining reconstruction quality.

1.2.5 Learning and Generalization Challenges

Limited Data Learning

Learning complete 3D representations from limited data requires sophisticated approaches that can effectively leverage prior knowledge and geometric constraints to fill in missing information.

This challenge is significant in learning over 3D spatial data, as the amount of 3D datasets for supervising the learning process is far less than the 2D images in computer vision.

Feature Learning

Effective feature extraction and representation learning are fundamental to successful 3D reconstruction. This involves developing methods that can capture meaningful geometric and semantic features across multiple scales.

Especially for indoor/outdoor scenes, the scene-level semantic information may further guide the reconstruction model to infer object-level geometric and semantic features. This dependency may propagate across multiple scales. For example, for a point cloud representing a large outdoor scene, a line of repetitive curved sticks alongside a road may be inferred as lamp posts, and the details of the lamp posts may further be inferred from other clones of this repetitive pattern.

These challenges motivate our research objectives and guide our proposed solutions, as we seek to advance the state-of-the-art in 3D reconstruction and spatial learning. In Chapter 2, we will delve into the existing works related to these challenges, and in later chapters, we will present our proposed solutions to them.

1.3 Research Objectives

The primary objective of this research is to advance 3D spatial learning by addressing critical challenges in reconstruction and representation. This includes generating sparse representations for surface reconstruction from noisy inputs, enhancing the controllability of implicit methods, and improving information propagation and aggregation across multi-scale spatial data.

By systematically tackling these challenges, our research not only aims to improve reconstruction quality but also to provide architectural and methodological innovations in 3D reconstruction and representation. The specific objectives are as follows:

1. **To Enhance 3D Parametric Reconstruction from Noisy Inputs** Develop machine-learning-based methods for parametric surface fitting frameworks to robustly reconstruct 3D shapes from noisy point clouds while preserving structural integrity.

2. **To Enable User-Controllable Refinement in Implicit 3D Reconstruction** Develop interactive pipelines that integrate human feedback to enhance local detail precision in neural field-based reconstructions.

3. **To Improve the Computational Efficiency of Implicit Neural Representations** Optimize neural field training efficiency by integrating geometric priors as guiding signals, reducing computational costs without compromising reconstruction quality.

4. **To Propose a General-Purpose 3D Learning Framework that is Scalable and Adaptive for Complex Scenes** Propose hierarchical architectures based on the attention mechanism for scalable and adaptive 3D deep learning on large-scale or multi-resolution data.

1.4 Contribution

In this research, our objective is to investigate various machine learning-based 3D reconstruction methods, state their advantages and disadvantages, and propose novel

methods that improve existing approaches. We also derive new techniques in the more general area of learning an expressive representation of spatial data.

In detail, our research proposes the following innovations with advancements to 3D spatial learning and reconstruction:

1. **Seed-Net**: Differentiable Interactive Refinement for Implicit Reconstruction

    - **Innovation**: A human-in-the-loop implicit reconstruction pipeline where user-provided positions (e.g., sparse clicks) guide the neural field optimisation via a differentiable feedback layer.
    - **Advancement**: Achieves progressively better reconstruction results with an experiment that mimics user interaction during the reconstruction.
    - **Impact**: Potential in the reconstruction of objects in digital twin systems where user interest may focus on a certain local area of an object.

2. **NeuLap**: Laplacian Priors for Accelerated Neural Field Training

    - **Innovation**: A geometry-aware training framework that uses a pre-trained denoising network to inject a Laplacian-based geometric prior into the error-propagation process of an implicit neural field, guiding optimisation towards more plausible surfaces.
    - **Advancement**: Achieves state-of-the-art reconstruction quality, improving the F-score of the baseline NeuRIS from 0.691 to 0.759 on ScanNet. It enhances performance with limited data and demonstrates superior preservation of sharp geometric features like edges and corners.
    - **Impact**: Enables high-fidelity 3D reconstruction from limited views, particularly for generating watertight surfaces with sharp details, which is critical for digital twin and reverse engineering applications.

3. Partitioned Deforming Boxes for Shape Reconstruction

    - **Innovation**: A dynamic grid-based deformation framework that combines adaptive boxes and deforms them into the desired shape, allowing localised shape adjustments without global re-optimisation.

- **Advancement**: Represents the reconstructed surfaces into a sparse, uniform representation that is suitable for both smooth surface curvature and sharp edges and corners.
- **Impact**: Enables reconstruction for applications requiring real-time shape editing (e.g., digital twin prototyping).

4. **Attention Propagation in OctTrees for General-Purpose 3D Learning**

- **Innovation**: A novel hierarchical attention mechanism for OctTrees, featuring iterative bottom-up and top-down passes that efficiently propagate contextual information across the entire tree structure.
- **Advancement**: Giving a competitive performance to strong grid-based baselines like ConvONet on reconstruction benchmarks while using less GPU memory. This demonstrates superior efficiency and scalability for processing large-scale 3D scenes.
- **Impact**: Provides a foundational, general-purpose architecture for a wide range of 3D deep learning tasks, including semantic segmentation and object detection, by enabling powerful multi-scale feature learning on sparse 3D data.

CHAPTER 2

Literature Review and Technical Foundations

The field of 3D reconstruction and spatial learning has witnessed significant advancements with the emergence of deep learning techniques. This chapter provides a comprehensive review of the fundamental concepts, technical challenges, and the corresponding attempts and solutions in this domain. We begin with an overview of the fundamental concepts and the corresponding technical challenges in Section 2.1 and 2.2, which sets the stage for our detailed examination of various methodological approaches.

The review is structured to progressively build understanding from fundamental concepts to advanced methodologies. We first examine explicit surface reconstruction methods in Section 2.3, which represent the traditional approach to 3D reconstruction. This is followed by an analysis of modern implicit reconstruction techniques in Section 2.4, with particular emphasis on neural representations and their applications in novel view synthesis. Finally, Section 2.5 discusses general-purpose frameworks for deep spatial learning, presenting a unified perspective on architectural solutions and practical considerations.

2.1 Fundamental Concepts and Background

Understanding 3D reconstruction and spatial learning requires a solid foundation in several key technical areas. This section establishes the essential concepts that underpin our subsequent discussions. We begin by examining various data representation schemes (2.1.1) that form the basis for encoding 3D spatial information, followed by fundamen-

tal geometric concepts (2.1.2) that govern spatial relationships and transformations. In (2.1.3), we explore core methodological approaches that have been developed for learning and reconstructing 3D spatial data in various paradigms. Finally, we discuss standard evaluation metrics (2.1.4) that enable quantitative assessment of reconstruction quality and spatial learning performance. These fundamentals provide the necessary theoretical framework for understanding both traditional approaches and modern deep learning-based solutions in 3D reconstruction.

2.1.1 Data Representations

The structure and performance of deep learning algorithms over 3D spatial data for downstream tasks heavily depends on the representation of the input data, since different representations are organised with different information topologies, thus providing different structural features, and may encapsulate different information explicitly or implicitly. By input data here we refer not only to the original input to the system, but also the data that an algorithm directly manipulates, and the desired output of the algorithm.

The representation for spatial data can be classified into two types according to how the data is organised: structured and unstructured representations. A structured representation means that the data units are **organised** in a regular pattern or grid-like structure that maintains consistent relationships between elements, while unstructured representations have no fixed organization pattern between elements, though they still carry information with spatial relationships. For example, a point-cloud is an **unstructured** representation - while each point carries its positional information, the order of the points inside a point cloud is arbitrary and does not follow any regular pattern. The structured data representations are usually more amenable to processing with algorithms transferred from 2D paradigms, but due to their rigid organization, they may not be suitable for non-rigid objects with deformations since the regular structure may change during the deformation while the topology persists.

Point Clouds and Their Properties

One of the most important representations in the unstructured representation family is the **point-cloud** representation. It is one of the most intuitive ways to represent 3D data

by encoding 3D structures as a set of discrete points in space and is notably information efficient. A point cloud $\mathcal{P} = \{p_i \in \mathbb{R}^3\}_{i=1}^{N}$ consists of N points, where each point p_i typically contains 3D coordinates (x, y, z) and may include additional attributes such as color, normal vectors, or other feature information. It usually serves as either the input or the output of the algorithm, representing a set of sampled points on the surface of the object, and the algorithm is designed to be invariant to the point order.

The major challenge with point clouds is that they are inherently **unordered data**, requiring algorithms to be designed as **permutation invariant** or **symmetric functions** to ensure consistent results regardless of point ordering. Another challenge for point clouds is that while they preserve positional information in 3D space, they lack explicit connectivity information between points, which introduces ambiguity when surface information is required for reconstruction or analysis tasks.

Surface Representations

Surface Representations form a distinct family of 3D representations that explicitly preserve surface topology information in 3D space. A surface can either be explicitly represented as a graph, or implicitly represented as the isocontour of a scalar function (discussed in subsection 2.1.1).

The **mesh representation** is one of the most preferred and mature surface representations in both computer vision and computer graphics, which represents a surface as a graph of connected polygons (typically triangles). Formally, a mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ consists of vertices $\mathcal{V}$, edges $\mathcal{E}$, and faces $\mathcal{F}$. This representation is storage-efficient but is sensitive to noise, missing data, and resolution problems. It has historically faced challenges with deep learning methods, which traditionally did not work well on graph-structured data. However, with the emergence of graph neural networks like GATs [143] and Graph-GAN [145], these limitations are being addressed.

Another graph-based surface representation that has become standard in computer-aided design (CAD) is the **parametric surface representation**, including spline-based approaches like Bézier surfaces and NURBS (Non-Uniform Rational B-Splines) [105]. NURBS surfaces are defined by a set of control points, weights, and knot vectors that together define a parametric function $S(u, v) : [0, 1]^2 \to \mathbb{R}^3$ mapping a 2D parameter space to 3D space. This kind of representation offers several advantages over meshes:

- **Compactness**: Requires fewer parameters to represent smooth surfaces
- **Precision**: Represents curved surfaces exactly rather than approximating them
- **Multi-resolution**: Can be evaluated at arbitrary resolution
- **Differentiability**: Provides continuous derivatives important for analysis

Advantages in Surface Representation

Bézier surfaces, which are extensions of Bézier curves, offer several advantages for surface representation in 3D reconstruction and CAD applications:

1. **Local Control**: Changes to the control points affect only the local region of the surface, making it easy to edit and modify specific areas without impacting the entire shape
2. **Smoothness**: Bézier surfaces can achieve high degrees of smoothness (G1 continuity or higher), which is essential for creating aesthetically pleasing and functional designs
3. **Variety of Shapes**: They can represent a wide range of shapes from simple planes to complex curved surfaces, making them versatile for diverse modeling tasks.
4. **Efficient Computation**: The parametric nature of Bézier surfaces provides a sparse representation that allows for efficient computation of surface properties and manipulations, which is crucial for real-time applications and interactive design environments.
5. **Industrial Standard Alignment**: Bézier surfaces align with industrial standards for 3D modeling, ensuring compatibility with widely used CAD software and facilitating seamless integration into existing workflows. This alignment is crucial for maintaining consistency and interoperability across different stages of the design and manufacturing process.

Implicit Representations

Implicit representations define 3D geometry as the level set (typically the zero-level set) of a continuous function $f : \mathbb{R}^3 \to \mathbb{R}$. Rather than explicitly storing geometry, these

representations encode a shape as:

$$S = \{x \in \mathbb{R}^3 | f(x) = 0\} \tag{2.1}$$

The function $f(x)$ typically represents a signed distance function (SDF) where $|f(x)|$ gives the distance to the nearest surface point, with negative values inside the object and positive values outside. Alternatively, occupancy functions map points to binary values indicating whether a point lies inside (1) or outside (0) the shape, or a continuous value representing the probability of occupancy as a real number in the range of $[0, 1]$.

Traditional implicit representations include:

- **Signed Distance Fields (SDFs)**: Store distance to the nearest surface with sign
- **Occupancy Fields**: Binary fields indicating inside/outside status
- **Radial Basis Functions (RBFs)**: Represent shapes as weighted sums of radial functions

More recently, **Neural Implicit Representations** have gained significant attention in 3D reconstruction. These approaches represent 3D shapes as the level sets of neural networks that map spatial coordinates to signed distances or occupancy values. Notable examples include:

- **DeepSDF** [99]: Represents shapes as learned continuous SDFs
- **Neural Radiance Fields (NeRF)** [90]: Represents both geometry and appearance
- **Occupancy Networks** [87]: Learn continuous occupancy functions

Key advantages of implicit representations include:

- **Continuous**: Represent surfaces at arbitrary resolution
- **Topologically flexible**: Can represent shapes of arbitrary genus
- **Watertight**: Naturally produce closed surfaces
- **Differentiable**: Enable gradient-based optimization
- **Compositional**: Support Boolean operations (union, intersection)

The major limitations are:

- **Extraction cost**: Converting to explicit representations (e.g., via Marching Cubes [82, 73] and Dual Contouring [52, 12]) is computationally expensive
- **Query efficiency**: Evaluating points requires forward passes through neural networks
- **Feature correspondence**: Difficult to establish point correspondences across shapes

Multi-view and RGB-D Images

The structured representation family has been extensively studied since the early days of 2D computer vision. **Multi-view images** represent a popular structured input data format used across various tasks, as they provide relatively rich data for the restoration of 3D information while being suitable for direct processing by existing image-based 2D algorithms, with no depth sensors required for data acquisition. Formally, a multi-view dataset consists of a set of images $\{I_i\}_{i=1}^N$ captured from different viewpoints, along with camera parameters $\{K_i, R_i, T_i\}_{i=1}^N$ where K_i represents intrinsic parameters, and R_i, T_i represent rotation and translation respectively.

However, determining the optimal number of views for specific tasks remains challenging, potentially leading to either information loss or redundancy. Another limitation is the loss of intrinsic geometric properties in the input data.

The **RGB-D data** acquired by sensors (e.g., structured light, time-of-flight, stereo) and the derived **RGB-D images** are another widely applied structured data representation. An RGB-D image pairs a standard RGB image with a corresponding depth map D, where each pixel (u, v) has both color $I(u, v) \in \mathbb{R}^3$ and depth information $D(u, v) \in \mathbb{R}$.

Algorithms working with such representations benefit from computational efficiency as they don't require extra data conversion and aggregation in the system pipeline, making them suitable for real-time applications such as autonomous driving, on-the-fly scanning, and monitoring. These algorithms also leverage existing deep-learning approaches for images since RGB-D data can be treated as either an image with an extra channel or two separate images (RGB and depth). While a single RGB-D frame lacks complete geometry information, rich geometric data can be aggregated by combining

RGB-D information from multiple views, making it more information-rich than multi-view images while sharing similar limitations.

Volumetric Representations

The **Volumetric** representation family provides structured 3D representations that directly encode spatial information while maintaining a regular lattice structure suitable for grid-based algorithms.

Voxels extend the concept of "pixels" into 3D space, discretizing it into unit cubes with associated occupation states to describe the 3D structure. Formally, a voxel grid can be defined as a 3D tensor $V \in \mathbb{R}^{W \times H \times D \times C}$ where W, H, D represent width, height, and depth dimensions, while C represents the feature channels at each voxel location.

This representation facilitates the transfer of pixel-based algorithms to 3D, and the voxel occupation state naturally maps to probability distributions in 3D space, making it suitable for generative models to learn structural priors. However, the cubic space complexity limits resolution, making it suboptimal for detailed structures.

The **octree** representation offers an alternative with logarithmic growth in resolution through its tree structure. Rather than using a uniform grid, an octree recursively subdivides space into eight equal octants, allocating higher resolution only to regions containing geometry. This approach achieves significant memory savings, especially for sparse scenes. However, its irregular memory access patterns and complexity in implementation with current deep-learning approaches has limited its widespread adoption.

A common limitation of volumetric representations is that despite storing original 3D information, they may not preserve intrinsic surface topology, potentially creating ambiguity in surface-related tasks (e.g., visualization, physical simulation).

Structured Hyper-representations

Structured hyper-representations extend traditional volumetric approaches by organizing spatial data in more sophisticated hierarchical or hybrid structures. These approaches aim to balance the advantages of different representation types while mitigating their individual limitations.

Key examples include:

- **Hierarchical Grid Structures**: Extensions of octrees that organize voxels in multi-resolution grids. Examples include Sparse Voxel Octrees (SVOs) [64] and Hierarchical Surface Prediction (HSP) [40].

- **Hybrid Representations**: Combinations of multiple representation types. For instance, Atlas-Net [35] represents 3D shapes as collections of parametric surface patches.

- **Multi-scale Grid Representations**: Approaches that maintain information at multiple resolutions simultaneously, such as Feature Grids and Hierarchical Feature Learning [116].

These hyper-representations offer several advantages:

- **Adaptive Resolution**: Can allocate higher detail to complex regions

- **Memory Efficiency**: Reduce storage requirements compared to uniform grids

- **Multi-scale Features**: Capture both fine details and global structure

- **Hierarchical Processing**: Enable coarse-to-fine processing strategies

For a comprehensive review of data representations in deep learning over 3D spatial data, we refer the reader to [3].

The choice of representation fundamentally impacts algorithm design, performance, and the types of geometric or topological properties that can be effectively captured and manipulated. Understanding these trade-offs is essential for developing effective 3D learning systems.

2.1.2 Geometric Concepts

Geometric concepts form the mathematical language used to describe and analyze 3D shapes and spaces. A thorough understanding of these principles is crucial for developing and evaluating 3D reconstruction and spatial learning algorithms. This subsection delves into key areas including differential geometry, surface properties, topology, and implicit representations like level sets.

Differential Geometry Basics

Differential geometry provides tools to analyze the local properties of curves and surfaces. Surfaces can be defined explicitly (parametrically) or implicitly. Parametric surfaces, like Bézier or NURBS patches, are defined by functions mapping a 2D domain to 3D space, $\mathbf{s}(u, v)$. Implicit surfaces are defined as level sets (see 2.1.2). For both, concepts such as tangent spaces, curvature (e.g., Gaussian and mean curvature), and the first and second fundamental forms are fundamental. Tangent spaces define the local linear approximation of a surface at a point. Curvature measures how much a surface deviates from being flat at a point, capturing intrinsic geometric information. These concepts are essential for understanding shape characteristics, surface smoothness, and for developing algorithms related to shape matching, registration, and surface fairing. Understanding local geometry is critical for methods that operate directly on surface meshes or point clouds.

Surface Properties and Normals

Surface properties describe the characteristics of a 3D object's boundary. Key properties include continuity (C0, C1, C2), which defines the smoothness of the surface and its derivatives. Surface normals, vectors perpendicular to the tangent plane at each point, are particularly important. Normals are crucial for rendering (calculating lighting and shading), defining orientation, and are often estimated or predicted in reconstruction tasks. Accurate estimation and representation of surface normals are vital for achieving visually plausible and geometrically correct reconstructions. Other properties like surface area and volume are also fundamental geometric quantities.

Topology and Manifolds

Topology studies the properties of shapes that are invariant under continuous deformations, such as stretching or bending, but not tearing or gluing. Concepts like connectivity, genus (number of "handles"), and the Euler characteristic describe the global structure of a shape. For instance, topology helps determine if a surface is closed (watertight) or has boundaries. Many surfaces encountered in 3D modeling and reconstruction can be modeled as 2-manifolds. A 2-manifold is a topological space where every point has a neighborhood that is topologically equivalent to an open disk in the

Euclidean plane ($\mathbb{R}^2$). Understanding manifold properties is crucial for mesh processing, parameterization, and ensuring the topological correctness of reconstructed surfaces.

Level Sets and Implicit Fields

Implicit representations define surfaces indirectly, typically as the zero level set of a function $f : \mathbb{R}^3 \to \mathbb{R}$. The surface S is defined as $S = \{\mathbf{x} \in \mathbb{R}^3 | f(\mathbf{x}) = 0\}$. A prominent example is the Signed Distance Function (SDF), where $f(\mathbf{x})$ represents the shortest distance from point $\mathbf{x}$ to the surface, with the sign indicating whether $\mathbf{x}$ is inside or outside the object. The Laplacian of the SDF, Δf, evaluated at the surface approximates the mean curvature H of the surface S. Specifically, for a true SDF f, the relationship on the surface is:

$$\Delta f(\mathbf{x}) = -2H(\mathbf{x}) \quad \text{for } \mathbf{x} \text{ where } f(\mathbf{x}) = 0$$

This property links the volumetric field f to the intrinsic geometry of the surface it represents. For SDFs represented by neural networks, the Laplacian can often be computed efficiently using automatic differentiation. This allows its use as a geometric regularizer during network training, promoting smoother surfaces or enforcing specific curvature properties. Level set methods, pioneered by Osher and Sethian, use these implicit functions to track moving interfaces and represent complex shapes. Implicit fields, particularly SDFs learned by neural networks (e.g., DeepSDF [99]), have become central to modern deep learning approaches for 3D reconstruction. They offer advantages in representing arbitrary topologies, handling noisy data, and enabling continuity and differentiability, which are beneficial for gradient-based optimization in deep learning. These representations decouple the surface geometry from a specific discretization like a mesh.

Geometric Operators and Priors

Beyond describing geometry, mathematical operators can analyze and enforce desirable shape properties. The Laplace-Beltrami operator (Δ), a generalization of the standard Laplacian to surfaces (manifolds), is particularly significant. In Cartesian coordinates, it is defined as:

$$\Delta f = \nabla \cdot \nabla f = \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

It measures the local variation or curvature of functions defined on the surface. Minimizing quantities related to the Laplacian often promotes surface smoothness or fairness, reducing high-frequency details or noise. Consequently, Laplacian-based terms are frequently employed as regularization priors in geometric optimization problems and deep learning frameworks. These priors encode assumptions about expected surface characteristics, guiding the reconstruction or learning process towards geometrically plausible results, as seen in methods aiming to preserve sharp features or ensure smoothness. Similar differential analysis can also be applied to explicit geometric data, such as depth maps or normal fields, to estimate intrinsic, viewpoint-invariant surface properties like curvature.

2.1.3 Methods for 3D Reconstruction and Deep Spatial Learning

Advancements in deep learning have profoundly reshaped the landscape of 3D reconstruction and spatial learning. Traditional methods often relied on geometric constraints and optimization techniques, while modern approaches leverage neural networks to learn complex spatial relationships directly from data. This subsection outlines fundamental methodologies, focusing first on how spatial data is encoded and features are learned, and subsequently detailing deep learning approaches tailored for dense 3D reconstruction from various input modalities.

Data Representations and Feature Encoding

Effective feature encoding is paramount for applying deep learning to 3D data. The choice of representation and encoding strategy depends heavily on the input data modality and the specific task.

Processing Point Clouds Point clouds, as raw, unordered sets of points in 3D space (typically $\mathbf{p}_i = (x_i, y_i, z_i)$), pose unique challenges for standard deep learning architectures like CNNs that assume structured input. Early pioneering work addressed the unordered nature using symmetric functions. PointNet [108] introduced a shared Multi-Layer Perceptron (MLP) applied to each point independently, followed by a max-pooling operation to achieve permutation invariance and aggregate a global feature vector. PointNet++ [109] improved upon this by introducing hierarchical feature

learning. It recursively partitions the point set and applies PointNet locally, capturing finer-grained geometric structures at multiple scales. Dynamic Graph CNN (DGCNN) [156] proposed constructing local graphs in feature space dynamically at each layer, allowing the network to learn edge features and aggregate information adaptively. These encoders form the backbone for many point cloud analysis tasks, including classification, segmentation, and as input stages for reconstruction methods.

Leveraging Multiview Images Reconstructing 3D geometry from multiple 2D images is a long-standing vision problem. Deep learning methods build upon principles from traditional Multi-View Stereo (MVS). Feature extraction typically starts with applying 2D Convolutional Neural Networks (CNNs) as backbones to each input view, producing per-pixel feature maps. A core challenge is aggregating these 2D features into a coherent 3D representation, respecting geometric consistency across views. Deep MVS methods like MVSNet [169] explicitly leverage epipolar geometry. They construct 3D cost volumes by warping features from source views onto reference camera frustums along hypothesised depth planes. A 3D CNN then regularizes this cost volume to predict a depth map for the reference view. Subsequent works refined cost volume construction, aggregation techniques, and efficiency. Other approaches learn implicit correspondences or use attention mechanisms to fuse information across views, preparing features for volumetric or implicit reconstruction.

Volumetric and Implicit Representations Beyond input encoding, the target representation for reconstruction also influences methods. Early deep learning methods often used voxel grids, dividing space into uniform cells and predicting occupancy or SDF values [16]. While direct, voxels suffer from high memory consumption and limited resolution. Implicit functions, as discussed in 2.1.2, offered a memory-efficient alternative. Networks like Occupancy Networks [87] and DeepSDF [99] learn functions that map 3D coordinates (and potentially a shape code) to occupancy probability or signed distance, enabling continuous representations and arbitrary resolution querying.

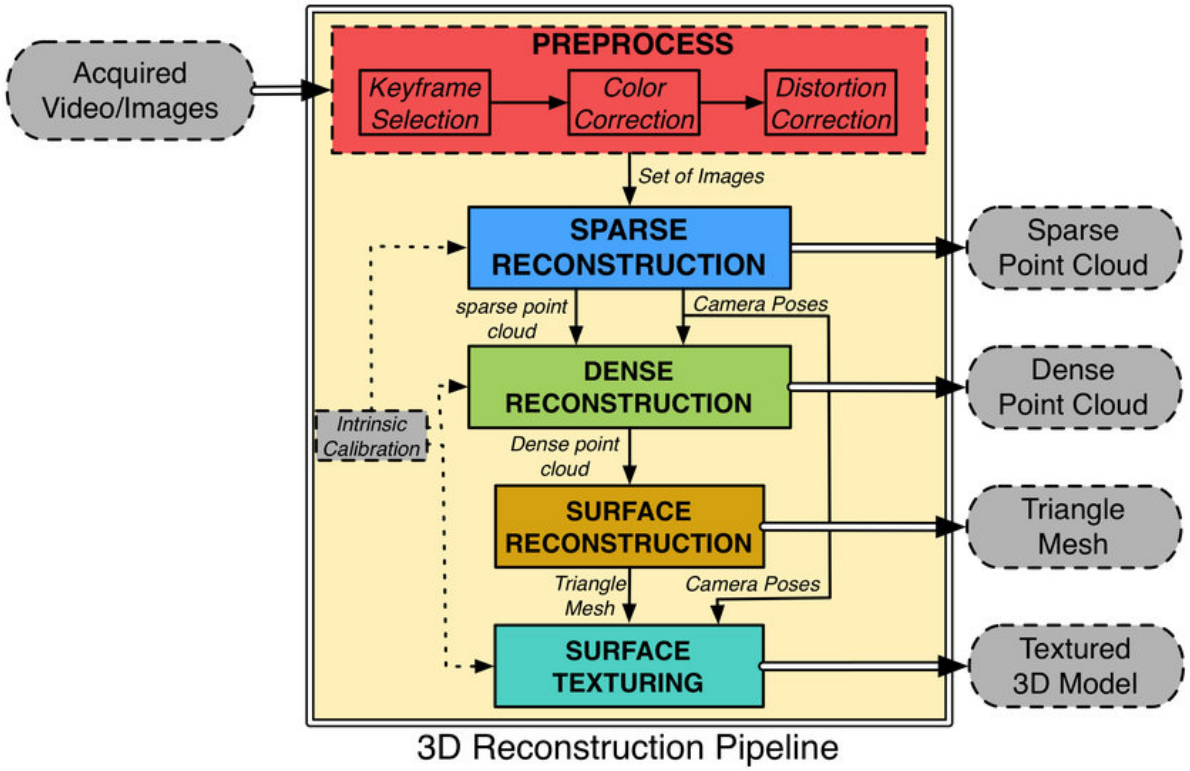
Deep Learning Approaches for Dense Reconstruction

The goal of dense reconstruction is to capture detailed geometry and, often, appearance from input data. Deep learning has yielded powerful methods operating on various

inputs.

The 3D Reconstruction Pipeline Context The overall 3D reconstruction pipeline often involves stages like data acquisition, preprocessing, sparse reconstruction (e.g., Structure-from-Motion for camera poses), dense reconstruction, surface generation, and texturing, as illustrated in Figure 2.1. While pipelines vary, this thesis focuses specifically on the **dense reconstruction** stage, where detailed spatial information (geometry and/or appearance) is inferred, typically from sparse initial estimates or directly from input data like images or point clouds. This stage is critical for high-fidelity results and challenges networks to learn robust and generalizable spatial representations. We believe advancing dense reconstruction methods directly contributes to improved generalised spatial learning capabilities.

Figure 2.1: A typical pipeline for 3D reconstruction [44]. This thesis focuses on the Dense Reconstruction stage.



Reconstruction from Point Clouds Given an input point cloud (often sparse or incomplete), deep learning methods aim to generate a dense and complete surface representation. Some approaches generate explicit surface elements. AtlasNet [35] learns

to deform a collection of 2D patches (parameterized squares) onto the target surface. FoldingNet [167] learns a mapping from a fixed 2D grid to the 3D surface, effectively "folding" the grid into the target shape. Other methods leverage implicit representations learned from point cloud features. Occupancy Networks [87] can be conditioned on point cloud features (e.g., derived from PointNet) to predict occupancy for any query point in space. Similarly, DeepSDF [99] concepts can be adapted to infer continuous SDFs from input points, allowing for surface extraction via methods like Marching Cubes [82].

Reconstruction from Multiview Images Dense reconstruction from multiple images has seen dramatic progress with neural rendering techniques that implicitly or explicitly model geometry alongside appearance.

Neural Radiance Fields (NeRF) NeRF [90] revolutionized novel view synthesis and implicitly captures geometry. It represents a scene using a fully-connected neural network (MLP) that maps a 5D coordinate (3D location $\mathbf{x}$ and 2D viewing direction $\mathbf{d}$) to a volume density σ and view-dependent emitted color $\mathbf{c}$. Images are rendered by sampling points along camera rays and integrating color weighted by density using principles of volume rendering:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \quad \text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

While producing highly photorealistic renderings, vanilla NeRF suffers from slow training and rendering times. Numerous follow-up works have addressed these limitations through efficient representations like explicit voxel grids [173], hash grids [93], or improved sampling strategies. Extracting explicit surfaces from NeRF often involves querying density σ and applying methods like Marching Cubes, although the resulting geometry may lack fine detail. Some variants explicitly incorporate surface constraints or representations (e.g., predicting an SDF alongside density) to improve geometric quality [150].

3D Gaussian Splatting (3DGS) 3D Gaussian Splatting [57] offers an alternative approach achieving state-of-the-art rendering quality with real-time performance. Instead of an implicit MLP, 3DGS represents the scene explicitly using a large number of

3D Gaussians. Each Gaussian is defined by its position (mean μ), shape (covariance $\mathbf{\Sigma}$), color (represented by Spherical Harmonics coefficients), and opacity (α). These parameters are optimized directly to minimize the rendering loss against input views. Rendering involves projecting the 3D Gaussians onto the 2D image plane and "splatting" their contribution onto pixels using a differentiable tile-based rasterizer. While highly efficient for rendering, 3DGS does not inherently represent a continuous surface. The geometry is captured by the distribution of Gaussian centroids, which resembles a point cloud. Its explicit nature makes editing potentially easier than NeRF, but sophisticated surface reconstruction from the Gaussians remains an area of research. Its success highlights the power of explicit, point-based representations when combined with efficient differentiable rendering.

2.1.4 Evaluation Metrics for 3D Reconstruction

Evaluating the quality of 3D reconstructions requires quantitative metrics that capture different aspects of geometric fidelity and, where applicable, appearance or complexity. Various metrics exist, often tailored to specific data representations like point clouds, meshes, or volumes.

Chamfer Distance (CD) For comparing two point sets, S_1 (prediction) and S_2 (ground truth), the Chamfer Distance is widely used [22, 2]. It measures the average squared distance from each point in one set to its nearest neighbor in the other set:

$$d_{\text{CD}}(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2 \tag{2.2}$$

Sometimes, only one direction is used, or the two terms are combined using max instead of sum. It provides a general measure of point set similarity.

Earth Mover's Distance (EMD) Also known as the Wasserstein-1 distance, EMD is another metric for point sets, particularly effective when $|S_1| = |S_2|$:

$$d_{\text{EMD}}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \frac{1}{|S_1|} \sum_{x \in S_1} ||x - \phi(x)||_2 \tag{2.3}$$

Here, $\phi : S_1 \to S_2$ is a bijection (one-to-one mapping). EMD finds the minimum average distance required to "move" points from S_1 to match S_2. It often captures structural differences better than CD but is more computationally expensive [2].

Hausdorff Distance For comparing meshes or point clouds, the Hausdorff Distance measures the maximum nearest-neighbor distance between two sets S_1 and S_2:

$$d_{\mathrm{H}}(S_1, S_2) = \max\left(\sup_{x \in S_1} \inf_{y \in S_2} ||x - y||_2, \sup_{y \in S_2} \inf_{x \in S_1} ||x - y||_2\right) \tag{2.4}$$

It represents the greatest distance from a point in one set to the closest point in the other set, making it sensitive to outliers and capturing worst-case errors.

Intersection over Union (IoU) For volumetric representations (e.g., voxel grids or implicit functions evaluated on a grid), IoU measures the overlap between the predicted volume V_{pred} and the ground truth volume V_{gt} [87, 16]:

$$\mathrm{IoU}(V_{pred}, V_{gt}) = \frac{|V_{pred} \cap V_{gt}|}{|V_{pred} \cup V_{gt}|} \tag{2.5}$$

Where $|\cdot|$ denotes volume (or count for voxel grids). IoU ranges from 0 (no overlap) to 1 (perfect overlap) and is standard for evaluating occupancy predictions.

Quadric Error Metric (related to Quadric Loss) While often used for mesh simplification, the concept behind the Quadric Loss provides a point-to-surface distance useful for evaluation. Given an output point s (in homogeneous coordinates $[x, y, z, 1]^T$) and its corresponding nearest vertex t on an input mesh $\mathcal{M}$, let $\mathcal{P}_t$ be the set of plane equations $p_i = [a_i, b_i, c_i, d_i]^T$ for the faces incident on vertex t. The squared distance of s to a plane p_i is $(p_i^T s)^2 = s^T(p_i p_i^T)s$. The sum of squared distances to these incident planes defines the quadric error at t for point s:

$$E_{\text{quad}}(s, t) = \sum_{p_i \in \mathcal{P}_t} s^T(p_i p_i^T)s \tag{2.6}$$

Averaging this error over all output points s (using their nearest neighbors t) gives a metric sensitive to surface deviations, particularly good at penalizing displacements that violate sharp edges defined by the incident planes.

Rendering Quality Metrics For methods that reconstruct appearance alongside geometry (like NeRF or 3DGS), evaluation often relies on comparing rendered images to ground truth views using standard image quality metrics:

- **Peak Signal-to-Noise Ratio (PSNR):** Measures pixel-wise error, higher is better.

- **Structural Similarity Index (SSIM):** Measures perceptual similarity based on luminance, contrast, and structure, range [-1, 1], higher is better.
- **Learned Perceptual Image Patch Similarity (LPIPS):** Uses deep features to measure perceptual distance, lower is better [90].

Complexity Metrics Beyond geometric accuracy, the complexity of the reconstructed representation itself can be important. Metrics like the **Akaike Information Criterion (AIC)** and **Bayesian Information Criterion (BIC)** can be adapted to balance reconstruction error (e.g., squared error term) against the number of parameters k used in the representation (e.g., number of polygons, Gaussians, or network parameters):

$$\text{AIC} = N \ln(\text{MSE}) + 2k \tag{2.7}$$

$$\text{BIC} = N \ln(\text{MSE}) + k \ln N \tag{2.8}$$

Where N is the number of data points and MSE is the mean squared error. Lower values indicate a better trade-off between accuracy and complexity.

2.2 Overview of Technical Challenges

Developing robust and effective methods for 3D reconstruction and spatial learning involves addressing a complex set of interdependent challenges. These range from fundamental issues related to data representation and computational demands to achieving high geometric fidelity and ensuring models can learn effectively and generalize to new data. Understanding these challenges is crucial for contextualizing the specific approaches detailed later in this review and motivates the contributions of this thesis. This section provides a high-level overview of these key technical hurdles.

2.2.1 Fundamental Challenges

Data Representation and Acquisition

A primary challenge lies in representing 3D spatial information effectively. Real-world data acquisition, often via sensors like LiDAR or RGB-D cameras, typically yields unstructured, sparse, noisy, and incomplete point clouds [39]. Handling such imperfections is critical for reliable reconstruction. Furthermore, 3D data can be represented in

diverse formats, including explicit representations like polygon meshes and parametric surfaces (e.g., B-splines, NURBS) [105], or implicit representations like voxel grids, occupancy fields [87], Signed Distance Functions (SDFs) [99], and Neural Radiance Fields (NeRF) [90].

Each representation involves trade-offs: explicit surfaces offer direct manipulation but can struggle with complex or changing topologies; implicit functions handle arbitrary topologies naturally and can be memory-efficient but may require complex extraction procedures (e.g., Marching Cubes [82]) and can be difficult to edit directly [165]. Point clouds are raw but lack explicit connectivity, while voxel grids suffer from cubic scaling in memory and computation [3]. Choosing an appropriate representation and designing networks that effectively process its specific structure (e.g., permutation invariance for point clouds [108]) remains a key design consideration. Handling varying densities, irregular structures, and integrating multi-modal data (e.g., images and point clouds) add further complexity.

Computational Considerations

Processing 3D data is computationally intensive due to its high dimensionality. Large point clouds, high-resolution voxel grids, or dense queries required by some implicit methods (like original NeRF [90]) demand significant memory and processing power. This poses scalability challenges for large-scale scenes or environments. Achieving real-time performance, crucial for interactive applications or robotics, often requires substantial optimization.

Strategies to mitigate these costs include developing efficient algorithms (e.g., Poisson Surface Reconstruction [55], Ball-Pivoting [5]), using specialized data structures like Octrees [116] or sparse voxel grids with sparse convolutions [33], and designing efficient neural network architectures or query strategies (e.g., multi-resolution hash grids in Instant-NGP [93], tensor factorizations [10], or explicit structures like 3D Gaussian Splatting [56]). Optimizing algorithms for specific hardware and developing adaptive or incremental methods remain active research areas.

Quality, Fidelity, and Detail

Achieving high-quality 3D reconstructions that accurately capture both the overall structure and fine geometric details of objects and scenes is paramount. Challenges include ensuring geometric fidelity (accuracy relative to the ground truth), preserving sharp features and high-frequency surface details (textures, small structures), and avoiding visual artefacts like noise, holes, or inaccurate topology [39]. Handling complex topologies (e.g., thin structures, self-intersections, objects with holes or handles) is particularly challenging for many methods, especially traditional parametric approaches [172].

Implicit methods, while flexible topologically, can face difficulties in accurately representing sharp edges or fine details due to inherent biases in network architectures (e.g., spectral bias of MLPs) [124]. Techniques like periodic activations (SIRENs [124]) or Fourier features [132] help mitigate this but introduce their own complexities. Extracting high-quality surfaces from implicit fields, particularly NeRF's density field, requires careful formulation, as demonstrated by methods like NeuS [150] which bridges SDFs and volume rendering. There is often a trade-off between representation robustness (e.g., using strong priors) and fidelity, where priors can lead to overly smooth results, losing important details. Ensuring global consistency across large reconstructions also remains difficult.

2.2.2 Learning and Generalization Challenges

Beyond fundamental representation and computational issues, learning-based approaches introduce their own set of challenges.

Learning from Limited or Imperfect Data

Deep learning models typically require large amounts of training data. Acquiring and annotating large-scale, high-quality 3D datasets is expensive and time-consuming. Therefore, a significant challenge is developing methods that can learn effectively from limited, sparse, or noisy data (few-shot learning). Implicit methods optimized solely via photometric loss (like NeRF) are particularly susceptible to geometric artefacts when input views are sparse [96]. Incorporating geometric priors (e.g., depth, normals, symmetry) or regularization techniques is often necessary to improve robustness in

such scenarios [19, 146].

Generalization and Adaptability

Ensuring that learned models generalize well to unseen objects, scenes, or data distributions is crucial for practical deployment. Many early implicit methods, including NeRF, required per-scene optimization, limiting their ability to generalize [90]. Developing architectures and training strategies that promote generalization across diverse inputs is an active area of research. This involves techniques like conditioning models on input features [175], using robust regularization [96], employing meta-learning to learn adaptable priors [123], or designing architectures with appropriate inductive biases.

Inductive Biases and Feature Learning

Designing deep learning models that effectively capture the underlying structure of 3D data requires incorporating appropriate inductive biases [92]. Key considerations for 3D include locality (nearby points/voxels are related), transformation equivariance or invariance (robustness to rotation, translation), and hierarchical structure (part-whole relationships) [109, 116]. Architectures must be designed to leverage these biases, for example, through local aggregation operations (spatial convolutions, neighbourhood processing in PointNet++ [109]), specialized layers for rotation handling [18], or multi-scale processing. Learning robust and discriminative features across different scales and representations remains a core challenge in 3D deep learning.

2.3 Explicit Surface Fitting and Inference

Explicit surface reconstruction methods directly model the geometry of 3D objects by fitting graph representations (e.g. meshes, Bézier patches) to input data. Unlike implicit approaches that represent surfaces indirectly, explicit methods generate surface elements that can be directly manipulated, offering advantages in applications requiring precise control over surface properties. This section explores various explicit reconstruction techniques, their mathematical foundations, and domain-specific applications.

2.3.1 Parametric Reconstruction

Parametric reconstruction approaches represent 3D surfaces using mathematical functions with finite sets of parameters. These methods are particularly valuable in computer-aided design, reverse engineering, and applications requiring compact, editable surface representations. By controlling a limited set of parameters, these techniques can efficiently represent complex geometries while providing intuitive handles for subsequent modification.

Mesh Representation and Parametric Surfaces

Mesh Representation Polygon meshes represent the most common explicit surface representation in computer graphics and 3D reconstruction. A mesh consists of vertices, edges, and faces that collectively approximate a surface. Mathematically, a mesh M can be defined as $M = (V, E, F)$, where V is a set of vertices with positions in 3D space, E is a set of edges connecting vertices, and F is a set of faces defined by sequences of edges.

While meshes offer flexibility in representing surfaces of arbitrary topology, they present challenges in ensuring smoothness and continuity. High-quality mesh reconstruction typically requires operations such as mesh simplification, re-meshing, and surface fairing to balance between geometric fidelity and representation efficiency. Algorithms such as Marching Cubes [82] and Poisson Surface Reconstruction [55] generate meshes from implicit functions or point clouds, but often require post-processing to achieve desired surface properties.

Bézier B-Spline Surface Representations Parametric surface representations form the foundation of many computer-aided design systems and explicit reconstruction approaches. Among these, Bézier and B-spline surfaces are particularly important due to their mathematical elegance and intuitive control mechanisms.

Bézier curves and patches are parametric representations extensively used in CAD due to their desirable properties, such as flexibility in shape control and ease of blending with other curves or surfaces. The mathematical definition of a Bézier curve is grounded in the Bernstein polynomial basis, providing a smooth and continuous representation of the curve's points:

$$B(t) = \sum_{i=0}^{n} B_{i,n} P_i \tag{2.9}$$

$$= \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \tag{2.10}$$

where P_i are the control points, $B_{i,n}$ are the Bernstein basis functions, and t is the parameter that varies within the range $[0, 1]$. The properties of Bernstein polynomials ensure that the curve will pass through the first and last control points, with the curve being influenced by the other control points in a weighted manner to provide smooth transitions.

While Bézier representations offer elegant formulations, they have limitations for complex modeling tasks due to their global influence property—modifying one control point affects the entire curve or surface. B-splines address this limitation by extending the capabilities of Bézier representations, offering local control and higher-order continuity across segments. A B-spline surface of degree (p, q) is defined as:

$$S(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) P_{i,j} \tag{2.11}$$

where $N_{i,p}$ and $N_{j,q}$ are the B-spline basis functions, and $P_{i,j}$ are the control points. The basis functions are defined recursively and provide C^{p-1} continuity across knot spans, enabling flexible modeling of complex geometries while maintaining smoothness properties.

NURBS (Non-Uniform Rational B-Splines) further extend this flexibility by introducing rational weights to control points, enabling the exact representation of conic sections and offering enhanced modeling capabilities. Piegl and Tiller [105] provide a comprehensive survey on NURBS, highlighting their mathematical foundations and geometric properties that make them the industry standard for CAD applications.

In 3D reconstruction, these parametric representations offer compact and editable models that can efficiently represent complex geometries while providing intuitive handles for subsequent modification. For instance, a bicubic Bézier patch is controlled by a 4×4 grid of control points, while B-spline and NURBS surfaces offer more flexibility through their local control properties, making them particularly valuable for reverse engineering and interactive modeling applications.

Learning-Based Mesh and Patch Generation

Modern approaches to explicit reconstruction increasingly leverage deep learning to generate geometric representations directly from various inputs. These methods often focus on creating patch-based or full mesh representations, aiming to improve generalizability and handle complex topologies.

Patch-based methods in 3D shape representation aim to leverage local similarities across object categories to enhance generalizability and control over shape deformations. These approaches face challenges in handling complex topologies and maintaining high-fidelity details. Tretschk et al. [140] introduce PatchNets, which utilize patch-level similarities to improve generalizability. In the context of textureless deformable 3D surfaces, Tsoli and Argyros [141] extend these concepts with their Patch-Based Reconstruction framework for texture-less deformable 3D surfaces, achieving increased accuracy even with limited training data. Meanwhile, Groueix et al. [35] further explore this domain with AtlasNet, which learns 3D surface generation using parametric surface elements, offering improved precision and generalization capabilities, though it may struggle with complex topologies.

Recent deep learning approaches have also focused on directly generating explicit mesh representations from inputs like single images or point clouds, offering an alternative to patch-based or implicit methods. A notable example is BSP-Net [13], which learns to represent a 3D shape as a Binary Space Partitioning (BSP) tree. The network produces a set of convex polytopes whose union forms the final, watertight mesh. This method is end-to-end trainable and generates a compact, editable representation reminiscent of constructive solid geometry. Another line of work focuses on deforming a template mesh. For example, Mesh R-CNN [32] extends Mask R-CNN for 3D object detection and mesh reconstruction from single images. It predicts a coarse voxel representation which is converted to a mesh, and then refines the vertex positions of the mesh with a graph convolutional network.

Reverse Engineering and Parametric Fitting

A key application of explicit reconstruction is reverse engineering, which focuses on converting unstructured geometric data, such as point clouds or meshes, into parametric CAD models suitable for engineering workflows. This often involves fitting B-spline or

NURBS surfaces to the data.

These methods address a fundamental challenge in computational design: converting unstructured geometric data into parametric representations suitable for engineering workflows. To address the issue of complex initial structures, Yin et al. [170] propose NURBS-based Automated Model Generation, a robust CAD model generation method that automates this conversion process. Buonamici et al. [7] explore Reverse Engineering Modeling strategies, identifying limitations in current frameworks while introducing strong assumptions on output structure. PIE-NET, introduced by Wang et al. [154], is a technique for parametric inference of point cloud edges, which improves edge detection accuracy but relies on assumptions about basic geometry types in CAD models. These approaches collectively advance the state of the art in generating editable CAD representations from raw 3D data, though they continue to face challenges with complex geometries and maintaining feature fidelity.

B-Spline and NURBS methods are central to explicit reconstruction, offering superior flexibility and local control compared to Bézier representations. However, fitting these surfaces to raw data like point clouds presents challenges in feature preservation, topological correctness, and computational efficiency. A primary concern is the preservation of sharp features during the fitting process. Liew et al. [74] investigated edge-aware B-spline surface fitting to maintain sharpness, while Kawasaki et al. [54] proposed a fairing technique that explicitly balances surface smoothing with feature preservation through careful parameter tuning.

Yoshihara et al. [172] present a method for topologically robust B-spline surface reconstruction from point clouds, effective for complex models but computationally intensive. Meanwhile, Galvez et al. [25] use Particle Swarm Optimization for NURBS surface reconstruction, offering high accuracy but facing challenges with complex topologies. Abdul-Hossen et al. [1] discuss Bi-cubic B-spline Mathematical Modeling for 3D surface reconstruction, providing surface control without altering control points. Leal et al. [66] present a method for constructing NURBS surfaces from unorganized points, achieving low fitting error while maintaining surface continuity.

Recent advances in B-spline surface reconstruction include neural approaches. Iglesias and Galvez [47] introduce functional networks for B-spline surface reconstruction, demonstrating how neural-inspired computational models can efficiently fit B-spline

surfaces to scattered data points. Their approach shows significant advantages in handling noisy data compared to traditional methods. Similarly, Zheng and Wang [180] propose a fast B-spline curve fitting method using the L-BFGS optimization algorithm, significantly reducing computational complexity by optimizing control points and foot points simultaneously.

A significant challenge in B-spline surface fitting is knot placement optimization. Zhao et al. [179] address this with an adaptive knot placement method using a Gaussian Mixture Model (GMM)-based continuous optimization algorithm. Their approach treats knot positions as variables optimized through evolutionary strategies, demonstrating improved fitting accuracy compared to uniform knot placement methods. For complex topologies, NorouzzadehRavari and Hariri [97] propose a reconstruction method for B-spline curves and surfaces using adaptive group testing, which efficiently identifies optimal knot positions for complex geometries. This approach complements the topologically robust methods proposed by Yoshihara et al. [172], providing comprehensive solutions for handling arbitrary topological structures.

Constructive Solid Geometry (CSG)

CSG is a traditional solid modeling technique that defines complex shapes by applying Boolean operations (union, intersection, difference) to simpler primitive shapes like spheres, cubes, and cylinders. The result is a binary tree where leaf nodes are primitives and internal nodes are operations. While powerful for parametric and editable modeling, inferring a CSG representation from raw 3D data like point clouds is a challenging inverse problem. Recent works have started to bridge this gap. UCSG-Net [53] is an attempt at unsupervised discovery of CSG trees from 3D data, showcasing the potential of integrating traditional explicit modeling with deep learning. Neural Parts [100] uses invertible neural networks to learn expressive 3D shape abstractions, decomposing objects into constituent parts that can be manipulated.

2.3.2 Challenges Specific to Explicit Reconstruction

While sharing fundamental hurdles with other paradigms (see Section 2.2), the implementation of explicit surface reconstruction methods presents several technical challenges specific to their nature. These primarily relate to handling input data im-

perfections when fitting structured representations like meshes or parametric surfaces, managing computational costs associated with these structures, and correctly capturing complex surface topology. This section explores these specific challenges and discusses solution approaches tailored to explicit representations.

Data Sparsity and Noise in Explicit Fitting

Fitting explicit models like meshes or B-splines directly to sparse, noisy, and incomplete point clouds, typical of real-world scans, is inherently challenging. Traditional explicit methods often struggle with non-uniform sampling densities and measurement noise, which can lead to poorly fitted surfaces or topological artefacts [39].

Recent learning-based approaches attempt to address these issues by leveraging data-driven priors. For instance, Yuan et al. [178] introduce PCN (Point Completion Network), a probabilistic framework that models the uncertainty in point cloud data, enabling more robust surface reconstruction in the presence of noise and outliers by completing partial point clouds. Another pioneering work in this area is PointCleanNet [111], which directly operates on point clouds to remove noise and outliers. It employs a spatial transformer network to orient local patches into a canonical representation before a PointNet-based architecture classifies points as either inliers or outliers.

To better preserve sharp features, which are often smoothed out during denoising, EC-Net [177] was proposed as an edge-aware consolidation network. It uses a regression-based approach to recover point positions relative to detected edges, which is particularly important for explicit models where feature definition is critical. Other methods have focused on unsupervised learning to avoid the need for paired clean and noisy training data. Total Denoising [43] operates on the principle that denser point clusters are more likely to represent the true underlying surface, though it can struggle to retain fine geometric details. More recent gradient-based approaches, such as Score-Denoise [83], frame the problem as learning the gradient of the log-probability of the clean data distribution, allowing noisy points to be moved towards higher probability regions.

The research gap in this area remains significant, particularly for applications requiring high-fidelity reconstruction from limited data, such as in autonomous vehicles, medical imaging, and mobile AR/VR systems. Novel approaches that integrate statisti-

cal priors with geometric constraints show promise in addressing these challenges.

Computational Efficiency for Explicit Models

Explicit surface reconstruction methods, particularly those involving complex mesh operations or iterative fitting of parametric surfaces, can be computationally intensive. Processing large-scale point clouds or refining high-resolution meshes using traditional approaches like Poisson Surface Reconstruction [55] or the Ball-Pivoting Algorithm [5] can become prohibitively expensive. Optimizing knot placement or control point positions for B-spline and NURBS surfaces over large datasets also presents significant computational demands [179, 180].

Several strategies have emerged to improve computational efficiency. Hierarchical data structures, such as octrees and kd-trees, enable efficient spatial queries and localized processing. Wu et al. [163] propose Adaptive Multi-Resolution Reconstruction, a multi-resolution approach that adaptively refines the reconstruction based on local geometric complexity, reducing computational overhead in regions with simple geometry.

Parallelization and GPU acceleration have also proven effective in scaling reconstruction algorithms. In recent researches, learning-based methods exploit GPU acceleration to achieve real-time performance, with Liu et al. [80] reporting reconstruction speeds orders of magnitude faster than traditional methods with their GPU-Accelerated Reconstruction.

Neural network-based optimizations have also emerged as powerful tools for improving computational efficiency. Thomas et al. [139] introduce KPConv (Kernel Point Convolution), providing flexible and deformable convolution operations for point clouds that significantly reduce computational overhead while preserving geometric details. Similarly, Wang et al. [156] propose Dynamic Graph CNN with optimized graph construction algorithms that achieve state-of-the-art performance on 3D shape analysis tasks while reducing computational complexity compared to traditional point processing methods.

For B-spline specific optimizations, Wang and Zheng [147] present a parallel and adaptive surface reconstruction method based on implicit PHT-splines (Polynomial splines over Hierarchical T-meshes). Their approach leverages parallel computing

architectures to efficiently handle large-scale point cloud data, demonstrating significant speed improvements over sequential implementations while maintaining surface quality.

The integration of machine learning with traditional geometric processing has also yielded promising results. Li et al. [68] propose SO-Net (Self-Organizing Network), a permutation-invariant architecture for point cloud processing that hierarchically extracts features from point clouds using self-organizing maps. This approach achieves competitive performance while requiring significantly less computation time than previous methods, particularly for large point sets.

Despite these advancements, computational efficiency remains a critical challenge, particularly for resource-constrained devices and real-time applications. The optimization of algorithms for specific hardware architectures and the development of adaptive, incremental reconstruction methods represent promising research directions.

Topology Handling in Explicit Representations

Accurately capturing and representing complex topologies is a fundamental challenge for explicit methods. Objects with intricate features, thin structures, holes, handles, or self-intersections can be difficult to represent faithfully with standard mesh structures or single parametric patches. Traditional parametric approaches often assume simple, disk-like topology, limiting their direct applicability to complex real-world objects without sophisticated patch decomposition or trimming strategies. Ensuring mesh connectivity is correct, manifold, and free of defects requires careful algorithmic design and often post-processing.

Several approaches address topology handling in explicit reconstruction. Podolak and Rusinkiewicz [106] introduce Atomic Volumes, a topology-invariant representation that can recover complex structures.

The integration of topological data analysis with deep learning provides powerful tools for handling complex geometric structures. For instance, [8] introduced TopologyNet, which leverages persistent homology to extract topological features from 3D biomolecular structures. These features are then used by a convolutional neural network to predict molecular properties, such as protein-ligand binding affinities. While not a surface reconstruction method, TopologyNet demonstrates how topological priors

can create robust representations for geometric deep learning, a concept relevant to topology-aware reconstruction.

Recent advancements leverage graph-based representations for topology inference. Wang et al. [149] introduce Pixel2Mesh, generating 3D mesh models from single RGB images using graph convolutional networks that maintain topological consistency during mesh generation. This approach has been extended in Pixel2Mesh++ [148], which improves topological accuracy for complex objects.

For CAD-specific topology handling, Yin et al. [170] present a NURBS-based automated model generation approach that preserves topological features essential for engineering applications. Their method adaptively handles complex topological constraints while ensuring manufacturability of the reconstructed models.

Learning-based approaches have shown particular promise for complex topology inference. Tang et al. [134] propose a skeleton-bridged deep learning method for generating meshes with complex topologies from single RGB images, effectively handling objects with holes, handles, and non-manifold structures that challenge traditional approaches.

For point cloud-based reconstruction with challenging topologies, Qin et al. [110] develop a mass-driven topology-aware curve skeleton extraction method for incomplete point clouds. Their approach robustly infers topological structure even from partial observations, facilitating complete surface reconstruction from incomplete data.

Neural implicit functions have also demonstrated effectiveness in handling complex topologies. Williams et al. [160] present a deep geometric prior for surface reconstruction that naturally accommodates arbitrary topological structures without explicit topology specification. Similarly, Michalkiewicz et al. [89] introduce implicit surface representations as layers in neural networks, enabling end-to-end learning of complex topological structures directly from data.

For Bézier patch-based reconstruction, a critical challenge involves determining the appropriate patch layout and connectivity to accurately represent the underlying topology. This challenge becomes particularly evident when reconstructing objects with sharp features or varying curvatures, where a single patch representation may be insufficient.

2.3.3 Applications and Domain-Specific Solutions

Explicit surface reconstruction methods find applications across diverse domains, each with specific requirements and constraints. This section explores domain-specific applications and tailored solutions that leverage the strengths of explicit representation.

CAD/CAM Applications

Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) represent primary application domains for explicit surface reconstruction, particularly for reverse engineering and quality control. These applications demand high precision, editability, and compatibility with industry-standard workflows.

For quality control applications, Jiang et al. [51] demonstrate PointGroup, an automated inspection system that reconstructs manufactured parts as parametric surfaces, enabling precise comparison with design specifications. The system leverages Bézier patches to represent complex surface features, providing intuitive visualization of deviations.

Recent advancements in machine learning have enhanced CAD/CAM applications by automating feature recognition and parameter estimation. Sharma et al. [121] introduce CSGNet, a deep learning framework that identifies CAD features from raw point clouds, facilitating automatic generation of parametric models. These approaches significantly reduce the manual effort required in traditional reverse engineering workflows.

Interactive Modeling

Interactive modeling applications leverage the intuitive controllability of explicit surface representations, enabling users to refine and manipulate reconstructed geometries. These applications span various domains, from digital content creation to medical image analysis.

In the context of digital content creation, Takayama et al. [128] introduce GeoBrush, a system for interactive patch-based modeling, allowing artists to create and edit complex surfaces through intuitive manipulation of control points. This approach demonstrates the potential of parametric representations in creative workflows, where artistic control and expressiveness are paramount.

Recent developments in mixed reality have expanded the scope of interactive modeling. Newcombe et al. [95] demonstrate KinectFusion, a real-time surface reconstruction and editing in immersive environments, enabling intuitive spatial interaction with digital models. These advancements highlight the potential of explicit representations in human-computer interaction, particularly for spatial computing applications.

Interactive systems for parametric surface editing have seen significant advancements in usability. Duncan et al. [21] present Interchangeable Components for Hands-On Assembly Based Modelling, a system that enables intuitive physical interaction with digital B-spline surfaces through tangible interfaces. This approach bridges the gap between physical and digital modeling paradigms, making parametric surface manipulation more accessible to non-expert users.

For collaborative environments, Pena-Rios et al. [103] introduce a framework for mixed agents in virtual observation lenses for immersive learning, enabling multiple users to collaboratively edit and refine parametric surface models in shared virtual spaces. Their approach integrates fuzzy logic systems [102] to handle ambiguity in user interactions, improving the robustness of collaborative editing sessions.

Learning-based approaches have also enhanced interactive modeling capabilities. Liu et al. [77] present an Interactive 3D Modeling system with a Generative Adversarial Network that learns from user interactions to suggest plausible completions and modifications to partial models. This approach reduces the manual effort required for complex modeling tasks while preserving user control over the final result.

For architectural applications, interactive furniture layout systems by Merrell et al. [176] and Yu et al. [86] demonstrate how constraint-based interactive systems can incorporate design guidelines into the modeling process, ensuring that user-created models satisfy domain-specific requirements while maintaining creative freedom.

The research challenges in interactive modeling include balancing automation with user control, ensuring real-time performance, and developing intuitive interaction metaphors. Bézier patch-based approaches offer promising solutions due to their compact representation and intuitive control mechanisms, though challenges remain in handling complex topologies and ensuring smooth transitions between user edits.

2.4 Implicit Reconstruction and View Synthesis

The faithful representation and rendering of 3D shapes and scenes is a fundamental goal in computer vision and graphics. Traditional methods often rely on discrete representations like meshes, voxels, or point clouds. Recently, representing scenes using continuous functions learned implicitly from data by neural networks has emerged as a powerful alternative paradigm [165]. An implicit neural representation typically defines a function $f_\theta : \mathbb{R}^3 \to \mathbb{R}^k$, parameterized by network weights θ, that maps a 3D coordinate $p \in \mathbb{R}^3$ to a property value (e.g., occupancy, signed distance, color, density). These representations offer advantages in memory efficiency, the ability to represent arbitrary topologies without fixed resolution limits, and the potential to capture fine details. This section reviews key developments in this area, focusing on methods relevant to 3D reconstruction, charting the progression from foundational geometric representations to the influential Neural Radiance Field (NeRF) framework, and discussing challenges and solutions involving priors, efficiency, and fidelity, which motivate the contributions presented later in this thesis.

2.4.1 Foundational Neural Implicit Representations

Early works focused primarily on representing 3D geometry implicitly. Occupancy Networks [87] pioneered learning a continuous function $f_\theta(p) \in [0, 1]$ representing the occupancy probability at point p, with the surface implicitly defined as the 0.5-level set $\{p \mid f_\theta(p) = 0.5\}$. While versatile, extracting features from raw point clouds for occupancy prediction proved challenging. Convolutional Occupancy Networks (ConvONet) [104] addressed this by integrating convolutional encoders, leveraging their ability to extract robust local features from structured inputs like projected point clouds, leading to improved reconstruction, particularly from noisy data. Subsequent refinements like Dynamic Plane ConvONets [75] and Sign-Agnostic CONet [135] further enhanced feature extraction and handled ambiguities in local predictions, respectively.

Representing geometry using Signed Distance Functions (SDFs) offered an alternative with useful properties, such as providing surface normals via the gradient. Here, $f_\theta(p)$ predicts the shortest distance to the surface, with the sign indicating inside/outside. DeepSDF [99] demonstrated learning continuous SDFs conditioned on

latent shape codes, enabling shape completion and representation by leveraging strong learned shape priors. To improve SDF learning directly from point clouds without requiring pre-trained shape spaces, methods like Neural-Pull [84], which pulls query points towards the surface, and Neural-IMLS [157] / Deep Implicit MLS [78], which adapt moving least-squares principles, were developed to handle noisy and unoriented point clouds. MeshSDF [114] explicitly connected implicit SDFs back to meshes via a differentiable iso-surface extraction layer, facilitating integration with mesh-based pipelines.

Other important developments include architectural innovations and alternative formulations. Implicit Geometric Regularization [34] used the Eikonal loss ($||\nabla f|| = 1$) to encourage learned functions to behave like true SDFs. SIRENs [124] employed periodic activations ($\sin(x)$) enabling MLPs to represent fine details more effectively than standard ReLU networks, addressing the spectral bias problem but requiring careful initialization. Local Deep Implicit Functions (LDIF) [31] offered accuracy and compactness by decomposing space into locally learned functions. Neural Kernel Fields [159, 46] achieved scalable, state-of-the-art reconstruction by framing the problem through learned kernel ridge regression.

Concurrent to these reconstruction-focused methods, early generative models like IM-NET [14] also explored learning implicit fields, typically by training autoencoders or GANs on shape collections to learn a latent space from which new implicit shapes could be generated.

2.4.2 Neural Radiance Fields (NeRF) for Geometry and Appearance

A significant breakthrough unifying geometry and appearance representation for photo-realistic rendering was Neural Radiance Fields (NeRF) [90]. NeRF models a scene using a continuous 5D function $f_\theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, parameterized by an MLP, that maps a 3D coordinate $\mathbf{x}$ and viewing direction $\mathbf{d}$ to an emitted RGB color $\mathbf{c}$ and volume density σ. Rendering involves querying f_θ along camera rays and compositing the outputs using differentiable volume rendering:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \quad \text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Optimization minimizes the photometric error between rendered and ground truth pixels. While primarily demonstrated for view synthesis, the learned density field σ implicitly encodes the scene geometry. However, the original NeRF faced significant challenges: extremely slow training and rendering due to dense MLP queries, difficulty in extracting high-quality geometry from the density field, and a per-scene optimization process limiting generalization. Subsequent work has largely focused on addressing these limitations.

Extensions for Efficiency and Scalability

Numerous works targeted NeRF's efficiency. Some focused on faster network inference (FastNeRF [30]) or replaced the MLP with explicit data structures. These explicit structures include sparse voxel grids (Plenoxels [173]), which can be directly optimized for super-fast convergence [126], or point-based features (Point-NeRF [166]). Mip-NeRF [4] addressed aliasing and scale ambiguity issues by rendering anti-aliased conical frustums instead of rays, improving detail preservation. Instant-NGP [93] introduced multi-resolution hash grids, enabling dramatically faster training by allowing a smaller MLP to query spatially hashed features. This technique, along with tensor factorizations like TensoRF [10], proved highly influential in reducing memory and computation. Other strategies involve distributing the network, such as KiloNeRF [112], which uses thousands of small MLPs for faster rendering.

Hierarchical representations have also been crucial for scalability and managing levels of detail. Early hybrid methods like OctNetFusion [115] fused depth data into an octree-guided implicit representation. More recent works like OctField [133], PlenOctrees [174], and Neural Geometric Level of Detail [129] use octrees to efficiently manage large scenes. For very large-scale environments, decomposition approaches like Block-NeRF [130] and CityNeRF [164] model scenes as collections of individual NeRFs. Fourier features [132] remained important for detail across many of these methods. Relatedly, 3D Gaussian Splatting (3DGS) [56] emerged, using an explicit set of optimizable 3D Gaussians to achieve state-of-the-art real-time rendering, representing a shift towards explicit-hybrid approaches.

Improving Surface Reconstruction from Radiance Fields

Directly extracting accurate surfaces from NeRF's density σ proved difficult. To bridge NeRF's rendering quality with the geometric definition of SDFs, NeuS [150] reformulated volume rendering using an underlying SDF network $f_\theta(\mathbf{x})$. By deriving density from the SDF such that it concentrates near the zero-level set, NeuS produces weights peaked at the surface, enabling high-quality surface extraction via Marching Cubes without explicit mask supervision. This combination proved powerful for reconstruction. NeuS2 [155] integrated hash encodings for significant speedups, making SDF-based neural rendering more practical.

2.4.3 Incorporating Priors and Handling Sparsity

Implicit methods, especially NeRF-based ones optimized via photometric loss, often struggle with sparse input views or noisy data, leading to geometric artefacts or incomplete reconstructions. Incorporating prior knowledge can regularize the optimization and improve robustness. Such priors can be integrated explicitly via encoders, which carries a trade-off between feature compression and information loss, or implicitly through architectural choices and regularization losses.

Depth and Geometric Priors

Explicit depth or geometric cues provide strong supervisory signals. MINE [69] used MPI concepts for depth-aware NeRF, while DS-NeRF [19] directly used sparse SfM depth as supervision. RegNeRF [96] regularized the learned geometry and appearance to improve interpolation from sparse views. Priors on geometric properties like symmetry can also be incorporated; SNeS [48] leverages symmetry for neural surface reconstruction from incomplete data. For surface methods, SparseNeuS [81] used auxiliary geometry encoding volumes. Depth-NeuS [50] incorporated depth consistency losses. NeuRIS [146] leveraged estimated surface normals within the loss to better reconstruct poorly textured regions where photometric loss is ambiguous. Ray-ONet [6] used ray intersection features. Intrinsic Neural Fields [60] applied geometric regularization (Laplace-Beltrami) for better texture. Furthermore, leveraging strong structural priors, such as Manhattan-world assumptions, can significantly improve scene representation

and completion from sparse inputs [37].

Object-Centric, Patch-Based, and Generalization Priors

Other priors operate at the object level or aim for better generalization across scenes. ShaRF [113] and AutoRF [94] used object-centric formulations for single-view NeRF. Kulkarni et al. [61] predicted ray distances. Patch-based rendering [125] and view interpolation (IBRNet [153]) improved generalization by reasoning locally or from nearby views. Methods like PixelNeRF [175] enabled generalization by conditioning the implicit network on image features extracted from the input views. CompNVS [72] introduced a composable approach for novel view synthesis to improve generalization across scenes, supervising scene codes with RGB-D data.

Semantic Priors

Beyond purely geometric priors, semantic understanding of the scene can also guide reconstruction. Semantic NeRF [181] incorporates semantic segmentation labels into the radiance field, allowing rendering of semantic maps alongside color and geometry. Panoptic Neural Fields [62] further extend this to instance-level understanding, representing each object instance with its own latent code, enabling instance segmentation and manipulation within the neural field representation. Integrating such semantic knowledge can help resolve ambiguities and improve the coherence of complex scene reconstructions.

Meta-Learning and Learned Initializations

Meta-learning offers ways to learn priors over function spaces or optimization processes themselves, often improving convergence or few-shot adaptation. MetaSDF [123] learned SDF priors for reconstruction from partial data. Learned initializations [131] significantly speed up optimization by providing better starting weights. Higher-Order Function Networks [91, 158] and Deep Meta Functionals [76] learned mappings from inputs to implicit network weights. Curriculum DeepSDF [20] used curriculum learning. These highlight learning priors *about* the models or learning process.

2.4.4 Challenges Specific to Implicit Representations

While implicit neural representations have shown remarkable progress, several key challenges specifically related to their formulation and optimization persist, motivating ongoing research and the work presented in this thesis. These challenges, distinct from but related to the general issues outlined in Section 2.2, primarily concern achieving high fidelity, managing computational costs associated with neural networks, and ensuring effective learning and generalization.

Representation Fidelity and Detail in Implicit Fields

Capturing fine geometric details and high-frequency appearance accurately within continuous implicit functions remains a central challenge. Standard MLP architectures often exhibit a spectral bias, favouring lower frequencies and struggling to represent sharp edges or intricate textures. While techniques like Fourier features [132], periodic activations (SIRENs [124]), and style integration [98] help, there is often a trade-off between representing fine details and ensuring global coherence or avoiding overfitting to noise. Strong priors, while helpful for robustness, can lead to over-smoothed results, losing important high-frequency surface components. To address this, hybrid representations aim to combine the advantages of implicit functions with explicit surface meshes. A prominent example, Deep Marching Tetrahedra (DMTet)[122], leverages a deformable tetrahedral grid to define an implicit SDF, which is then converted to an explicit mesh via a differentiable marching tetrahedra layer. This allows for direct optimization of surface geometry, enabling the model to synthesize fine details with fewer artifacts than methods that only regress an implicit function [122]. Other hybrid approaches [27] and techniques for improved mesh extraction [12, 114] also offer partial solutions. Addressing this balance motivates exploring alternative paradigms, such as incorporating user guidance for selective refinement, as investigated in the SeedNet framework (Section 3.1 in Chapter 3).

Computational Efficiency of Neural Fields

Implicit methods rely on neural networks, whose training and inference can be computationally demanding. Original NeRF required extensive training time and slow

rendering due to dense queries per ray [90]. While the approaches detailed in Section 2.4.2, such as hash grids (Instant-NGP [93]) and explicit data structures (Plenoxels [173], 3DGS [56]), have dramatically reduced training and rendering times, the computational demands (memory footprint, training cost, inference latency for network-based methods) remain significant, especially for large scenes or on resource-constrained devices. Research continues into sparse representations, efficient sampling strategies, network pruning/distillation, and hardware acceleration [63]. Integrating priors strategically, as explored in NeuLap (Chapter 3), can also improve convergence efficiency.

Generalization and Few-Shot Learning for Implicit Models

Many implicit methods, particularly early NeRF variants, are optimized per-scene, lacking the ability to generalize to new scenes without retraining. Achieving effective generalization, especially when learning from only a few input views (few-shot reconstruction), is a major challenge. As discussed in Section 2.4.3, progress relies on developing architectures that can condition effectively on input views or features (e.g., PixelNeRF [175]), using robust regularization techniques specifically designed for few-shot scenarios (e.g., RegNeRF [96]), or leveraging meta-learning to learn adaptable priors over shapes or scenes (e.g., MetaSDF [123]). While powerful, meta-learning is often constrained by the network's capacity to learn a sufficiently general yet strong prior, and the bi-level optimization can be demanding and sensitive to the task distribution. Balancing prior strength with fidelity to observed data is critical. A fundamental challenge in prior integration is preventing the prior from over-constraining the model, which can suppress fine details and lead to results that are plausible but not faithful to the input data. This motivates research into dynamic strategies where the influence of a prior guiding signal changes or decays during the inference process. Such an approach could provide strong regularization initially and then progressively allow the network to fit the fine-grained details, a concept explored later in this thesis (Chapter 3).

2.4.5 Generative and Interactive Implicit Models

Beyond reconstructing existing scenes, implicit representations facilitate generative modeling [14, 88, 70]. Recent advances, particularly using diffusion models [107, 168], enable compelling text-to-3D generation. Furthermore, the inherent limitations of

purely automated reconstruction, particularly in ambiguous or data-sparse scenarios, motivate research into interactive models. While most research pursues full automation, enabling a human-in-the-loop (explored in Section 3.1 of Chapter 3) allows for targeted refinement and control, addressing ambiguities or enforcing specific constraints not easily captured by automated priors alone.

This review highlights the rapid evolution of implicit neural representations, from foundational geometric methods to sophisticated radiance fields capable of photorealistic rendering and detailed reconstruction. Key challenges remain in fidelity, efficiency, and generalization, driving research towards better architectures, prior integration, and interactive paradigms, setting the stage for the specific contributions of this thesis. For comprehensive surveys, readers are referred to [137, 138], [165], [28], and [101].

2.5 General-Purpose Frameworks for Deep Spatial Learning

While the previous sections focused on explicit and implicit representations primarily for reconstruction and view synthesis, deep learning offers a broader range of frameworks applicable to diverse 3D spatial data types and tasks. Processing 3D data, whether unstructured point clouds, regular voxel grids, or structured meshes, presents unique challenges related to representation choices, computational scale, efficiency, and the need for appropriate spatial inductive biases (discussed generally in Section 2.2). This section reviews fundamental architectural paradigms developed to address these challenges for tasks like classification, segmentation, detection, and completion on various 3D data formats [3]. Understanding these general-purpose frameworks provides context for many methods in the field and informs the design of novel approaches.

This section is intentionally selective, focusing on the general-purpose frameworks most relevant to the research presented in this thesis, rather than providing an exhaustive survey of the entire field.

2.5.1 Architectural Paradigms for 3D Deep Learning

Several dominant architectural paradigms have emerged, each tailored to specific strengths and weaknesses of different 3D representations and incorporating different

inductive biases.

Point Cloud Networks and Convolutional Approaches

Point clouds, as raw, unstructured sets of points $P \subset \mathbb{R}^3$, lack explicit topology and are unordered. A key challenge is designing networks that are permutation-invariant, meaning the output is independent of the order of points in the input set. PointNet [108] pioneered this using shared Multi-Layer Perceptrons (MLPs) applied independently to each point, followed by a symmetric aggregation function (e.g., max-pooling) to produce a global feature vector, effectively achieving permutation invariance. PointNet++ [109] addressed PointNet's limitation in capturing local structures by introducing a hierarchical architecture. It recursively applies PointNet-like modules to local neighbourhoods sampled at different scales (using techniques like Farthest Point Sampling and ball queries), enabling the learning of features that capture both local geometric details and global context, reflecting a hierarchical inductive bias.

Many subsequent works focused on defining convolution-like operations directly on point clouds. Examples include PointCNN [71], Pointwise CNNs [45], Relation-Shape CNN [79], and Geo-CNN [65]. PointConv [161] used MLPs to learn weighting and density functions for aggregation. KPConv [139] proposed deformable kernel points adaptable to local geometry. DGCNN [156] dynamically constructed local graphs. Other approaches like SO-Net [68] used self-organizing maps, while PCPNet [36] focused on learning local shape properties. FoldingNet [167] proposed a point cloud auto-encoder using a graph-based deformation approach. These methods aim to leverage the success of 2D CNNs by adapting convolutional principles to the irregular structure of point clouds.

Voxel-Based and Hierarchical Methods

Representing 3D data on regular volumetric grids (voxels) allows the direct application of standard 3D Convolutional Neural Networks (CNNs), leveraging their strong performance in image analysis. Early works like 3DShapeNets [162] and VoxNet [85] demonstrated this for shape classification and retrieval. Methods like 3D-R2N2 [16] used recurrent 3D CNNs to integrate information from multiple views for reconstruction. However, the primary challenge of voxel representations is their cubic complexity

in memory and computation with respect to resolution, limiting their practical application to relatively low-resolution grids. To address this scalability issue, sparse voxel representations and associated sparse convolution operations [33, 15] were developed, focusing computation only on non-empty voxels, making deeper networks and higher effective resolutions feasible.

Hierarchical data structures like octrees provide an alternative, adaptive approach to partitioning 3D space, offering better memory efficiency for many geometries than dense grids. OctNet [116], O-CNN [152], and Octree Generating Networks [136] introduced methods for applying convolutions or generating data directly on octree structures. These methods naturally incorporate a hierarchical inductive bias, enabling deeper networks on higher-resolution data compared to dense voxel approaches.

Attention Mechanisms and Transformers in 3D

The success of Transformers [142] in NLP, driven by the self-attention mechanism, spurred interest in applying them to 3D data. Self-attention allows models to capture long-range dependencies by computing pairwise interactions between all elements in a set, offering adaptive receptive fields unlike fixed-kernel convolutions. This is appealing for 3D data where global context and relationships between distant parts can be important. Furthermore, attention mechanisms are inherently permutation equivariant, making them suitable for point clouds, as demonstrated by Set Transformer [67] and Point Cloud Transformer (PCT) [38].

However, the standard self-attention mechanism has quadratic complexity $O(n^2)$ with respect to the number of input elements n, making it computationally expensive for large point clouds or high-resolution voxel grids. This motivates research into efficient attention variants or applying attention within more structured representations.

As will be explored in Chapter 3, applying attention mechanisms within hierarchical structures like octrees offers a promising direction. An octree provides an efficient, adaptive representation, and its structure allows for:

- **Multi-scale Attention:** Applying attention both within levels (capturing local context) and across levels (propagating global information).
- **Implicit Positional Encoding:** Leveraging the tree structure's inherent spatial information.

- **Efficiency:** Potential for sparse attention mechanisms that exploit the hierarchy, mitigating the quadratic complexity of dense attention.

This combination aims to leverage the expressive power of attention for capturing complex dependencies while retaining the efficiency and multi-scale benefits of hierarchical representations, motivating the development of frameworks like the Attention Propagation method proposed in this thesis.

Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) provide another general framework, applicable when 3D data can be represented as a graph (e.g., meshes [41, 23], or point clouds where neighbourhood graphs are constructed). GNNs operate by passing messages between connected nodes, iteratively updating node features based on their local neighbourhood [59]. Different GNN variants use different message passing and aggregation functions, such as graph attention (GAT [143]), relational convolutions (R-GCN [119]), continuous B-Spline kernels (SplineCNN [24]), hierarchical pooling (DiffPool [171]), or U-Net like structures (Graph U-Nets [26]) [163, 182]. GNNs offer flexibility in handling irregular structures but rely on the meaningful definition of graph connectivity. Unsupervised GNNs like Variational Graph Auto-Encoders [58], Deep Graph Infomax [144], and GraphTER [29] also contribute to representation learning.

2.5.2 Learning Strategies and Outlook

Beyond core architectures, various learning strategies enhance the capabilities of these frameworks. Self-supervised learning aims to learn meaningful representations from unlabeled 3D data [118, 42, 11], reducing reliance on large annotated datasets. Few-shot learning methods tackle scenarios with very limited labeled data. Knowledge distillation can compress large models for efficient deployment. These strategies, combined with the architectural paradigms discussed, continue to push the boundaries of what is possible with deep learning on 3D spatial data, enabling applications from autonomous driving and robotics to virtual reality and digital twins.

2.6 Summary and Research Gaps

This chapter has presented a comprehensive review of the literature in 3D reconstruction and spatial learning. We began by establishing fundamental concepts in Section 2.1, including diverse data representations (point clouds, meshes, implicit fields, etc.) and the challenges inherent in processing 3D data (Section 2.2), such as handling sparsity, noise, computational scale, and ensuring geometric fidelity.

We then surveyed explicit reconstruction methods (Section 2.3), particularly parametric approaches like B-spline and NURBS surfaces, which offer compact and editable representations crucial for CAD but face challenges in topology handling and fitting to imperfect data. Next, we explored the rapidly evolving field of implicit neural representations (Section 2.4), from foundational methods like DeepSDF and Occupancy Networks to the influential NeRF framework and its derivatives (e.g., NeuS) that unify geometry and appearance modeling. We highlighted their strengths in topology flexibility and detail capture but also noted challenges in surface extraction quality, computational cost, and robustness to sparse inputs. Finally, we reviewed general-purpose deep learning frameworks (Section 2.5) applicable across 3D tasks, including point cloud networks (PointNet++), voxel-based CNNs, hierarchical methods (OctNet), and the growing use of attention mechanisms and GNNs.

This review reveals several persistent challenges and research opportunities:

- **Efficiency and Scalability:** Processing large-scale 3D data and training complex neural fields remain computationally expensive, despite advances like hash grids and sparse structures. There is a need for more efficient architectures and training strategies, particularly for real-time and interactive applications.

- **Geometric Fidelity and Detail Preservation:** Achieving high-fidelity reconstructions that accurately capture sharp features and fine details, especially with implicit methods prone to over-smoothing or requiring dense data, remains difficult. Balancing fidelity with robustness is key.

- **Robustness to Imperfect Data:** Handling sparse, noisy, and incomplete input data, common in real-world scenarios, continues to be a major hurdle for both explicit fitting and implicit optimization methods.

- **Integration of Priors and User Control:** Effectively incorporating geometric priors (e.g., smoothness, planarity) or semantic knowledge to guide reconstruction is crucial, especially with limited data. Furthermore, enabling intuitive user interaction for refinement and control offers a way to overcome limitations of fully automated methods.

- **Generalization and Adaptability:** Many state-of-the-art methods, particularly neural fields, require per-scene optimization. Developing frameworks that generalize well to unseen data or adapt quickly with limited examples is essential for broader applicability.

- **Robustness and Quality of Parametric Reconstruction:** Enhancing the robustness and topological handling of explicit parametric methods, like Bézier patch reconstruction, especially from noisy inputs, remains an important goal for applications demanding editable, structured outputs.

- **Hierarchical and Multi-Scale Reasoning:** Effectively capturing and propagating information across different scales within 3D data is crucial for understanding both local details and global structure, motivating research into architectures like hierarchical attention mechanisms.

These identified gaps directly motivate the research objectives of this thesis. Objective 1, enhancing parametric reconstruction quality, addresses the limitations discussed in Section 2.3 and is realized in the research presented in Section 3.3 in Chapter 3. Objective 2, addressing the need for user control, motivates the interactive SeedNet framework presented in Section 3.1 in Chapter 3, building on insights from Sections 2.3 and 2.4. Objective 3, improving the efficiency and fidelity of neural fields, tackles challenges from Section 2.4 and inspires the NeuLap approach presented in Section 3.2 in Chapter 3. Finally, Objective 4, developing scalable and adaptive frameworks, connects to challenges in Sections 2.2 and 2.5 and motivates the exploration of hierarchical attention mechanisms presented in Section 3.4 in Chapter 3.

CHAPTER 3

Proposed Methods for 3D Reconstruction

This chapter presents the core methodological contributions of this thesis, detailing four novel deep learning approaches for 3D reconstruction. These methods addresses the challenges discussed in Chapter 2, including representation, user interaction, and computational efficiency.

3.1 Seed-Net: Interactive Refinement with Human Feedback

SeedNet addresses the challenge of balancing fidelity to input data with the generalization capability of learned priors in 3D reconstruction. The method aims to prevent over-interpolation from averaged training data, thereby preserving details while inferring global structures. This is achieved through a combination of interactive human guidance and attention mechanisms. A human-in-the-loop (HITL) approach is motivated by the limitations of fully automated, prior-guided reconstruction methods. These methods can be over-constrained by learned priors, leading to results that are overly influenced by the global features of the training data or that suppress unique local details. This can manifest as over-smoothed regions or the loss of specific, high-frequency features that are critical for a particular application. By introducing human guidance, the reconstruction process can be locally steered, allowing users to inspect the initial output and selectively refine areas where the automated reconstruction is insufficient.

To validate the demand for such interactive control, a preliminary user study was conducted, targeting professionals in relevant fields. The study involved 40 participants from the game development industry and 7 from the XR (Extended Reality) industry. Within the game industry cohort, a significant portion (17 out of 40) felt that current implicit reconstruction methods did not align well with their established modeling workflows and thus did not provide further feedback. However, of the remaining 23 interested participants, a strong majority (17 out of 23) expressed a clear desire for manual control over the generation process to refine the results. In the XR industry, 4 out of 7 participants also indicated a positive inclination towards having such interactive refinement capabilities. While the scale of this user study is limited, the feedback strongly suggests a practical need for user-controllable tools that bridge the gap between automated reconstruction and manual artistic control, empowering users to preserve essential details and correct prior-induced artifacts.

SeedNet offers two main advantages:

1. Independence from pre-trained networks for initial OctTree construction from input, enhancing generalizability, especially with noisy or out-of-distribution data that might disrupt OctTree generation.

2. Introduction of human interaction to guide reconstruction, focusing efforts on parts critical for downstream tasks.

SeedNet utilizes a sparse grid representation for efficient 3D scene encoding and manipulation. Human guidance provides cues for selective refinement in specific regions, allowing desired details to be represented within their local coordinate systems. Attention mechanisms applied over human guidance and 3D space, enable heuristic refinement across the entire scene. This approach preserves fine-grained details and considers local structures and symmetry priors, enhancing the accuracy and realism of the implicit reconstruction.

The core insight is that the surfaces of most man-made 3D objects consist of predominantly low-frequency signals with scattered high-frequency details, such as edges and corners. Interactive guidance enables users to specify these high-frequency details, which are then used to heuristically infer improvements for the overall reconstruction. Users can iteratively refine reconstructions by selecting "seeds" (local areas for refine-

ment). The model re-samples these areas at higher resolution and encodes the result using an improved projection method. During decoding, each re-sampled encoding combines with feature codes from the sparse grid to predict values for query points. These predictions are merged by calculating attention weights relative to the seed's encoding and the feature code at the query point.

3.1.1 Human-in-the-loop Pipeline Design

The general pipeline for deep learning-based 3D reconstruction involves an encoder network (Enc_g) that maps input data (e.g., a point cloud) into a latent code C. For local encoders (e.g., grid-based), this results in multiple latent codes c_i for each region i, possibly with a global feature code c_g, denoted as $C = \{c_1, ..., c_n, c_g\}$. Surface reconstruction from a Deep Implicit Function (DIF) typically involves sampling query points $Q \subset \mathbb{R}^3$. For each query point $q \in Q$, a latent code c_q is derived from C and q's position. A decoder then evaluates the scalar field value y_q:

$$y_q = f(c_q, q) \tag{3.1}$$

SeedNet introduces human interaction by allowing users to iteratively select positions, termed **seeds**, for refinement. The system re-samples data around each seed to refine the reconstruction based on the encoded local information, then an attention-based mechanism, informed by learned priors of 3D scenes, enables each seed to contribute globally. For example, in reconstructing a meeting room, a seed placed on a chair should ideally contribute to refining all chairs in the scene.

Let S be the set of seeds, with each seed $s \in S$ defined by parameters like position, re-sampling radius, method, and pose. An initial latent code c_s for the seed is derived from the original encoded global input. A seed encoder, Enc_l, then processes each $s \in S$ with its corresponding c_s to produce the final seed code $z_s = \text{Enc}_l(c_s, s) \in Z$.

With global encoding C from Enc_g and seed encodings Z from Enc_l, the field value y_s^q for a query point q and a seed s is:

$$y_s^q = F(c_q, z_s, q) \tag{3.2}$$

This can be viewed as $c_q \mapsto (z_s \mapsto (q \mapsto y))$, where c_q generates a function mapping seed information to a querier for each seed.

To enable global contributions, learned priors are essential. An affine transformation matrix T_s^q (output by an MLP receiving (z_s, c_q, q)) projects the query point q to the seed's local space:

$$y_s^q = F(c_q, z_s, T_s^q \times q) \tag{3.3}$$

The affine transformation matrix T_s^q, learned by an MLP, is the core mechanism designed to address coordinate ambiguity. By projecting the query point q into the seed's local coordinate system, the decoder F can operate in a canonical space. This helps decouple the local refinement from its global position and orientation. This process allows the network to implicitly learn local symmetries, contributing to translation and rotation invariance to a large extent. However, full equivariance is not formally guaranteed by this design alone. Its effectiveness depends on the diversity of the training data. The model's robustness is therefore enhanced by standard data augmentation, such as random transformations applied during the base network's training.

Alternatively, for SDF-based models, a deformation approach might be suitable:

$$y_s^q = F(c_q, G(z_s, c_q, q)) \tag{3.4}$$

where G is a learnable function deforming the prediction from Equation 3.1 based on seed s. Equation 3.3 is often more suitable for occupancy fields.

3.1.2 Differentiable Feedback Layer

To combine predictions from multiple seeds into a unified outcome, a contribution value (attention weight) λ_s^q is computed for each query point q and seed s by a learnable function $A(c_q, z_s, q) \mapsto \lambda_s^q$. These weights are normalized using a softmax operation across all seeds for a given query point, yielding $\dot{\lambda}_s^q$. The combined prediction for the approach in Equation 3.3 is:

$$y^q = \sum_{s \in S} \dot{\lambda}_s^q F(c_q, z_s, T_s^q \times q) \tag{3.5}$$

And for the deformation approach (Equation 3.4):

$$y^q = F(c_q, \sum_{s \in S} \dot{\lambda}_s^q G(z_s, c_q, q)) \tag{3.6}$$

While not explicitly formulated as a standard Transformer attention, the mechanism here shares the core concept. Given a query point q, we can conceptualize:

- The **Query** Q^q derived from the query point position q and its global context feature c_q.
- The **Keys** K^s derived from each seed's latent code z_s (and potentially its position/-context).
- The **Values** V^s as the individual implicit field predictions $y_s^q = F(c_q, z_s, T_s^q \times q)$ generated by each seed-specific path.

The learnable function A computes compatibility scores between the query Q^q and keys K^s to produce the weights λ_s^q. The subsequent softmax normalization and weighted sum (Equation 3.5) effectively perform the attention mechanism, combining the values V^s based on query-key compatibility.

A key advantage is that inference for each seed is individual. The computational cost for the main inference grows linearly with the number of seeds. During interaction, previously computed results for existing seeds do not need re-inference; only the combination and new seed outputs are computed.

3.1.3 Training Process

The training process for SeedNet, particularly the Seed-ConvONet implementation, involves two main stages:

1. **Global Network Pre-training/Loading:** First, a base global reconstruction network (e.g., ConvONet) is trained on the target dataset or a pre-trained version is loaded. This network provides the initial global scene encoding (C) and a baseline reconstruction ($\hat{y}_g^q$).

2. **Seed Network Training:** Second, the seed-specific components (Enc$_l$, F or G, transformation MLP, attention function A) are trained. During this stage, for each training sample, K points are randomly selected from the input point cloud to act as seeds. The seed encoder Enc$_l$ and the seed decoder (F or G) are shared across all seeds. The network is trained to minimize a combined loss function:

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{global} + \beta \mathcal{L}_{local}$$

where $\mathcal{L}_{global}$ is the reconstruction error calculated after combining the predictions from all seeds and the global network (using Equation 3.7), and $\mathcal{L}_{local}$ is the sum or

average of reconstruction errors for each seed's individual prediction ($\hat{y}_s^q$) before combination. In our experiments, both α and β were set to 1.

It is important to note that the human interaction aspect of SeedNet, specifically the selection of seeds for refinement, occurs *after* the network has been fully trained. No further network training or fine-tuning is performed during the interactive refinement phase.

3.1.4 Progressive Refinement Strategy

This section demonstrates an implementation adapting SeedNet to the Convolutional Occupancy Network (ConvONet) [104] for interactive 3D scene reconstruction from point clouds. ConvONet encodes input point clouds into feature grids using a U-Net architecture to capture multi-scale features. For a query point q, its feature vector c_q is interpolated from the grid, and a decoder outputs an occupancy value $y_q = \text{Dec}(c_q, q)$. ConvONet's detail reconstruction ability is tied to grid resolution (voxel or tri-planar), with computational complexity scaling as $O(N^3)$ or $O(N^2)$ respectively, potentially limiting performance on large scenes with fine details.

Encoding Seeds in Seed-ConvONet

Seeds are defined as spherical regions of fixed radius r. Local point clouds within a seed are encoded by:

1. Performing PCA on points within the seed to estimate a normal $\mathbf{n}_s$ for the local patch. PCA is used here to estimate the dominant orientation (normal) of the local surface patch within the seed's radius. This assumes that at the scale of a seed, the local geometry is often convex and as a part of a larger surface that can be reasonably projected onto a plane. The estimated normal $\mathbf{n}_s$ is then used to orient the 2D projection grid for the seed encoder, allowing the CNN to process the local geometry in a consistent, view-independent manner relative to the local surface.
2. Creating a planar grid aligned with $\mathbf{n}_s$.
3. Projecting points onto this grid, encoding their signed distances.
4. Applying a small CNN to the projected grid to produce seed code z_s.

This method is motivated by the observation that fine-grained details are often convex and aligned with local parent structures, allowing for precise yet computationally efficient 2D convolution-based encoding. A seed is described by $(x_s, \mathbf{n}_s)$, where x_s includes global position and local grid position. Fourier Positional Encoding [90] is applied to seed and query point positions. A three-layer MLP generates the seed descriptor s from $(x_s, \mathbf{n}_s)$. Figure 3.2 illustrates this re-sampling process.

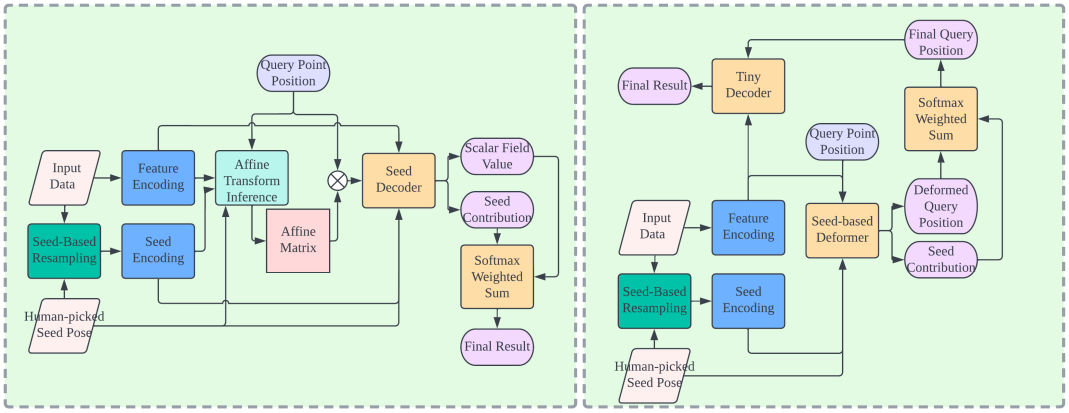


Figure 3.1: Conceptual illustration of the SeedNet pipeline options. Left: Scalar value prediction per seed, combined via weighted average. Right: Deformation prediction per seed, with a single decoder for the final result.

Seed-wise Reconstruction and Combination

An MLP generates the transformation ($\mathbf{T}_s^q$ from (s, c_s) and (q, c_q), where c_q is the ConvONet feature for query q. A decoder network then learns both the occupancy $y_s^q = f(c_q, z_s, \phi(\mathbf{T}_s^q \times q))$ (where ϕ is Fourier Positional Encoding) and the attention weight $\lambda_s^q = a(c_q, z_s, \phi(\mathbf{T}_s^q \times q))$. The final reconstruction combines these predictions with the original ConvONet output ($\hat{y}_g^q$) using a constant weight λ_g^q (before softmax) for the base reconstruction:

$$y^q = \frac{\hat{y}_g^q e^{\lambda_g^q} + \sum_{s \in S} \hat{y}_s^q e^{\lambda_s^q}}{e^{\lambda_g^q} + \sum_{s \in S} e^{\lambda_s^q}} \tag{3.7}$$

This iterative approach benefits from caching. When new seeds are added at step t, the computationally expensive per-seed outputs $\hat{y}_s^q$ and attention logits λ_s^q for the existing seeds from step $t-1$ are be cached. Only the outputs for the new seeds need to be computed. The final combination (Equation 3.7) involves recalculating the softmax

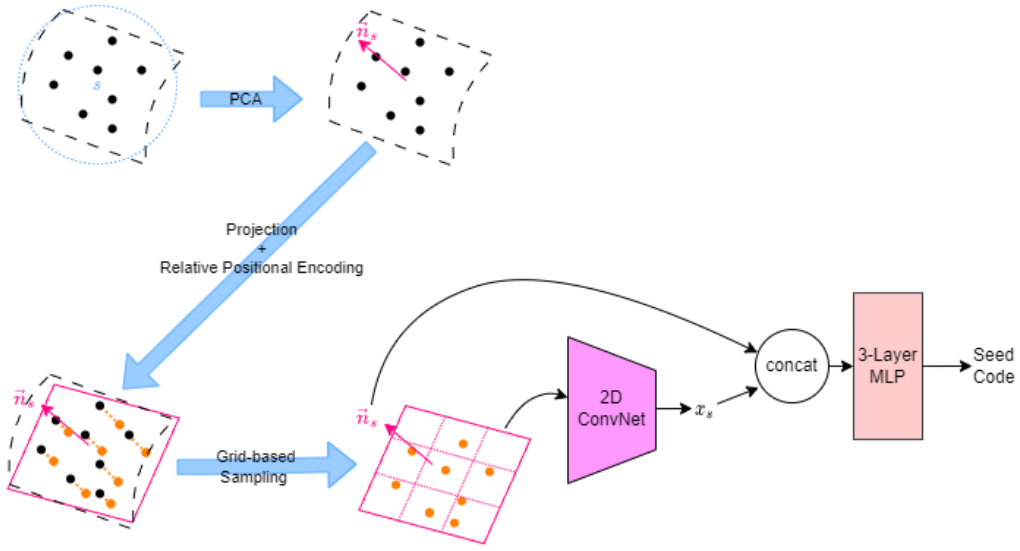


Figure 3.2: Seed-based re-sampling and encoding process for Seed-ConvONet. Points near the seed are projected onto a PCA-derived planar grid. A CNN encodes point distances and plane pose into a seed code.

denominator with the new λ_s^q values and performing the weighted sum, which is significantly faster than recomputing all individual seed predictions.

3.1.5 Experimental Validation

Experiments were conducted to evaluate Seed-ConvONet against a mixed dataset comprising synthetic data [104] and real-world scans from the ScanNet dataset [17]. The datasets were split into 70% for training, 10% for validation, and 20% for testing.

Key aspects analyzed:

1. **Reconstruction Accuracy**: Metrics included Volumetric Intersection-over-Union (IoU), Chamfer-L_1 distance, and normal consistency, following [87]. Two seed sampling strategies were evaluated:

   - *Random Sampling*: K seeds were placed at locations chosen uniformly at random from the input point cloud.
   - *Heuristic Sampling* (results marked with * in Table 3.1): This strategy aimed to mimic informed user interaction by placing seeds in potentially high-detail regions. The algorithm involved:

(a) Calculating local position variation for points in the input cloud (e.g., variance of neighbour positions within a small radius).

(b) Identifying candidate seed locations corresponding to points with high local variation.

(c) Selecting the top K candidate locations while enforcing a minimum distance constraint between selected seeds to ensure diversity and avoid redundant sampling on the same local feature.

2. **GPU Memory Cost**: Compared memory efficiency for different configurations (number of seeds, seed code length).

3. **Ablation Study**: Investigated the contributions of query point transformation ($\mathbf{T}_s^q$) and Fourier positional encoding.

Seed-ConvONet with a 16^3 3D grid was tested with 4, 8, 16, and 32 seeds (seed code lengths 16, 32, 64), compared against ConvONet (16^3 baseline and 32^3 high-res reference). Results indicated that Seed-ConvONet (16^3) with ≥ 8 seeds (code length 64, heuristic sampling) could outperform ConvONet (32^3) in terms of IoU and normal consistency, while using less GPU memory.

Table 3.1 summarizes these findings.

The benchmarking focuses on ConvONet because SeedNet is designed as a pluggable module that can be integrated with similar occupancy-based networks. The primary goal is not to outperform all state-of-the-art automatic methods, but rather to demonstrate the value of incorporating human interaction. The comparison is not entirely direct, as SeedNet's performance is augmented by user guidance. However, the results clearly show that by mimicking human interaction through heuristic seed placement, the corresponding model (16^3 Seed-ConvONet) achieves significantly better outcomes than its automatic baseline and can surpass a higher-resolution version (32^3 ConvONet). This highlights the substantial impact of targeted user refinement and demonstrates the value of incorporating user priors for detail enhancement.

Qualitative Results

Figure 3.3 provides a visual comparison of the reconstruction quality. The images demonstrate that Seed-ConvONet successfully leverages sparse user-provided seeds

to reconstruct finer details and correct geometric inaccuracies present in the baseline model. For the qualitative results, the seeds are manually picked based on the initial reconstruction result for each scene, with up to 3 seeds selected per scene to mimic the user interaction. The position of each seed is visually indicated by a cross of the straight lines of the same color in the 'Ours' (with seed) row of the figure, illustrating the area of local refinement.

Specifically, the seed placement and its effects for each scene are as follows:

- **Scene 1:** The seeds are placed at the thin cylinder of the lamp and on the surface of the wardrobe. This configuration enables the model to restore the thin cylindrical structure of the lamp and to fill the hole on the wardrobe's surface, demonstrating the ability of Seed-ConvONet to correct both fine and coarse geometric errors through targeted refinement.

- **Scene 2:** A seed is placed on the chair to smoothen its flat surfaces. Notably, the influence of this seed extends beyond the immediate vicinity, contributing to the refinement of the entire scene, which highlights the global effect of local user guidance in the SeedNet framework.

- **Scene 3:** Three seeds are placed at the back of the chair, the top of the lamp, and one of the legs of the table. This arrangement results in higher surface accuracy across these features, underscoring the effectiveness of multi-seed guidance for challenging regions.

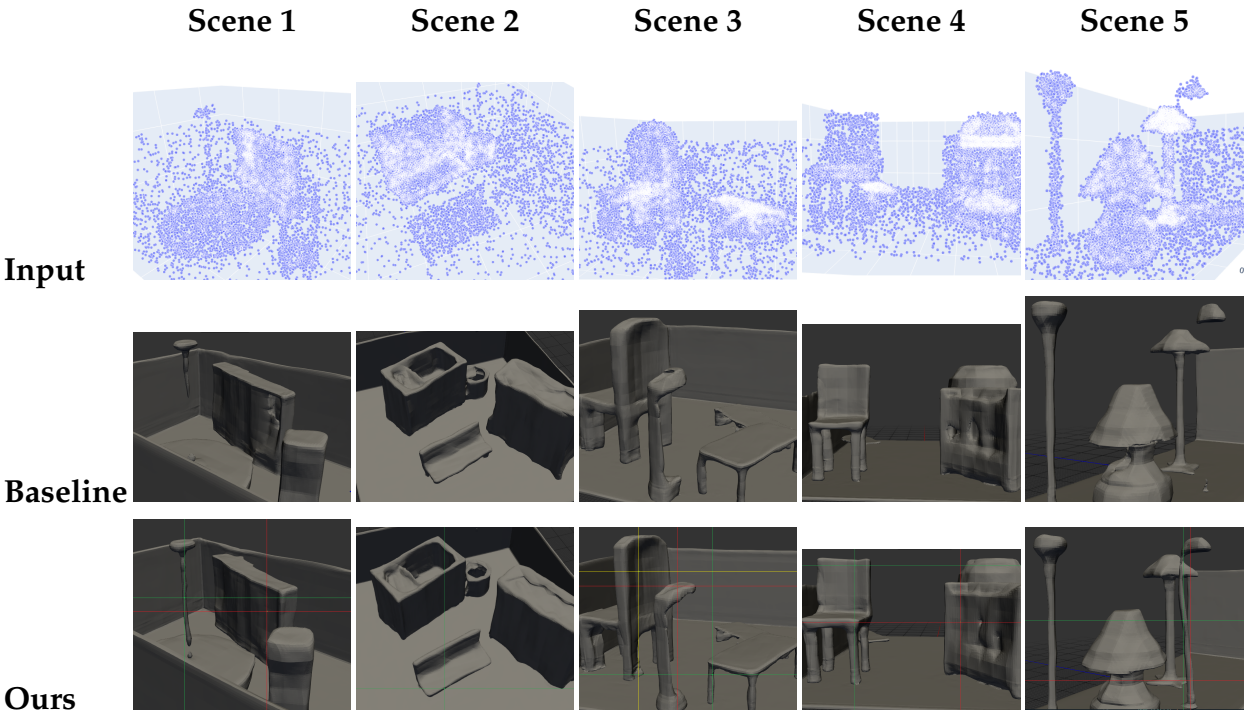
- **Scene 4:** One seed is placed on the chair and another on the wardrobe, aiming to fix the hole on the wardrobe's surface and to refine the edges of the chair. However, the refinement does not completely resolve the surface defect, and the chair is not significantly improved. This case illustrates a limitation of the current approach, where seed placement does not always guarantee successful correction, further research may be needed to stabilize the outcome.

- **Scene 5:** Two seeds are placed on the lamp, successfully restoring the thin cylindrical structure. This demonstrates the model's capacity to recover delicate geometric details when provided with appropriate user guidance.

For instance, across the various scenes, details like the legs and backrests of chairs and the structure of lamps are visibly improved. This highlights the effectiveness of the interactive refinement approach in enhancing local geometric fidelity.

The mean inference time for ConvONet is 7574 ms on a single NVIDIA RTX 3090 GPU, while SeedNet requires 7632 ms for the initial reconstruction. Each additional seed update incurs a mean update time of 1132 ms, demonstrating the efficiency of interactive refinement in SeedNe.

Figure 3.3: Qualitative results of Seed-ConvONet on five scenes. Each column represents a synthetic scene. For each, we show: input point cloud (top), baseline ConvONet reconstruction (middle), and our refined result (bottom).



Ablation Study and Complexity Analysis

To further elucidate the contributions of key architectural components in Seed-ConvONet, we conducted an ablation study focusing on two mechanisms: Fourier positional encoding and the query point transformation (PT). Figure 3.4 summarizes the results of this analysis.

The results demonstrate that both positional encoding and point transformation significantly enhance reconstruction quality. Specifically, the inclusion of point trans-

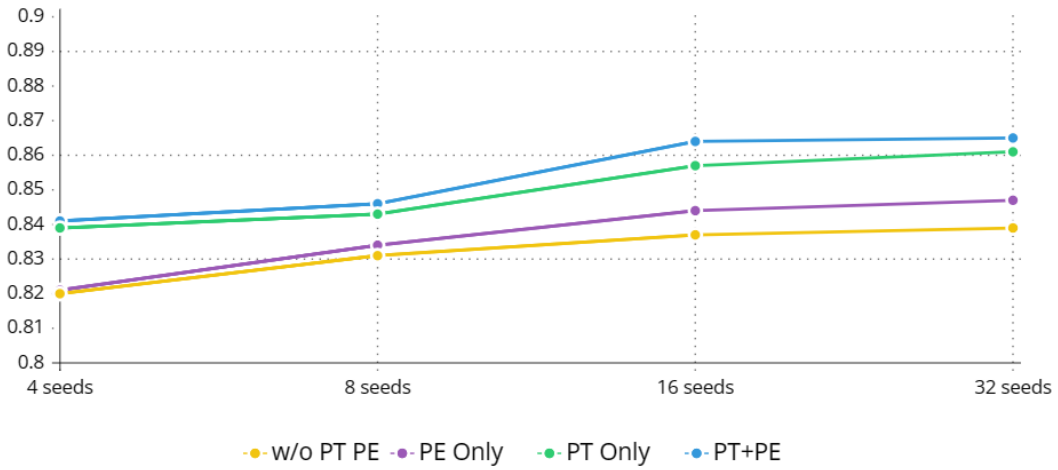


Figure 3.4: Ablation study of Seed-ConvONet: Impact of positional encoding (PE) and point transformation (PT) on reconstruction accuracy.

formation yields a notable improvement in Intersection-over-Union (IoU) and normal consistency, while positional encoding further refines the model's ability to capture fine geometric details. The combination of both mechanisms achieves the best overall performance, underscoring their complementary roles in the SeedNet architecture.

3.1.6 Conclusion and Future Work

SeedNet introduces a paradigm for 3D reconstruction by incorporating human interaction into the inference process. It allows users to iteratively refine reconstructions by selecting "seeds", offering potential for adaptability to diverse tasks. However, key limitations were identified:

- **Interference with Global Output**: Local seed reconstructions can interfere with the global output, potentially causing inconsistencies.
- **Competitiveness**: While innovative, the reconstruction quality might not always match state-of-the-art fully automatic methods without careful seed placement.

Potential future improvements:

- **Enhanced Integration Mechanisms**: More sophisticated attention or adaptive weighting to balance local/global contributions.
- **Optimized Seed Placement**: Algorithms to suggest optimal seed locations.

- **Decoupled Seed Reconstructor Training**: Training global and local (seed-specific) reconstructors separately and then combining them, potentially using transfer learning and distillation. This could address training instability where global encoder error signals dominate local ones.

- **Over-fitting Seed Reconstructors for Detail**: Intentionally over-fitting local seed reconstructors during inference time to input details in the seed's vicinity to better preserve high-frequency signals, which are often filtered out by networks aiming for generalization.

- **Simplified Seed Encoders**: Moving away from complex projection-based seed encoders (like PCA-derived planes) as they introduce restrictive local assumptions.

Further research is needed to validate these potential improvements and benchmark against newer, rapidly evolving reconstruction paradigms.

3.2 NeuLap: Enhancing Neural Fields via Laplacian Priors

Implicit neural representations have shown remarkable results in 3D reconstruction and view synthesis. However, they often require extensive training time and substantial input data; a traditional NeRF model is trained from scratch for each new scene and requires hundreds of images from different views. This section introduces NeuLap, a novel approach to address these limitations by integrating geometric priors into the training process of neural implicit fields, specifically targeting SDF-based representations like NeuS. The goal is to guide the corresponding network to converge with fewer views and training steps while preserving the quality of the reconstructed surfaces. NeuLap aims to improve computational efficiency by finding a novel way of integrating geometric priors into the training process. The core idea is to leverage the Laplacian of the signed-distance field (SDF) around the surface as a supervisory signal. This signal guides the neural network to learn geometrically plausible surfaces more rapidly and with fewer views. We demonstrate NeuLap's effectiveness by integrating it into the NeuS framework [150], showing accelerated convergence and improved reconstruction quality. The proposed method involves rendering an additional channel representing the Laplacian of the implicit surface. A denoising network then refines this rendered

Laplacian. The difference between the rendered and denoised Laplacian provides an error signal. This error signal, carrying the learnt geometric prior from the denoising network, is combined with the primary task's loss function. This approach avoids direct encoding of low-dimensional priors into high-dimensional signals. The priors are termed "low-dimensional" because geometric features like smoothness or curvature, when represented by a global or patch-wise regularizer, can be conceptualized as a more compressed form of information compared to the per-point detail captured by high-dimensional neural network weights or a full point cloud. Instead, it integrates prior information into the error propagation process. The Laplacian was chosen due to its properties: it is numerically sparse near surfaces, zero on flat areas, and its extrema highlight edges and corners. Furthermore, the projected Laplacian of an implicit field is view-independent, providing a stable geometric signature.

3.2.1 Geometry-aware Training Framework

The NeuLap framework augments an existing implicit neural representation based on SDF, such as NeuS, to incorporate geometric priors effectively. The overall pipeline is illustrated in Figure 3.5. It consists of two main stages. First, a denoising network, denoted as Ψ, is pre-trained to refine noisy Laplacian images. During the reconstruction process, the augmented implicit neural representation, Φ, is trained for the given scene. During the training of Φ, it renders both the standard output (e.g., RGB image) and an additional Laplacian image $\hat{\mathcal{L}}$. This rendered Laplacian image $\hat{\mathcal{L}}$ is then processed by the pre-trained and frozen denoising network Ψ to produce a refined version $\dot{\mathcal{L}}$. The discrepancy between $\hat{\mathcal{L}}$ and $\dot{\mathcal{L}}$ forms an additional loss term. This loss term guides Φ to learn surfaces consistent with the geometric priors encoded in Ψ.

We formalize images of size $w \times h$ as either data $\mathcal{A}$ or functions $\mathcal{A}(x, y)$ mapping 2D positions to channel values. $\mathcal{I}$ denotes a ground-truth RGB image. $\mathcal{N}$ represents a ground-truth normal image, $\mathcal{L}$ is the corresponding ground-truth Laplacian image. Rendered images are denoted with a hat, e.g., $\hat{\mathcal{I}}$, $\hat{\mathcal{N}}$, and $\hat{\mathcal{L}}$. Images processed by the denoising network are marked with a dot, e.g., $\dot{\mathcal{L}}$. The implicit neural representation network is denoted as function Φ and the denoising network as function Ψ.

The NeuS framework, chosen as the testbed, represents surfaces as the zero-level set

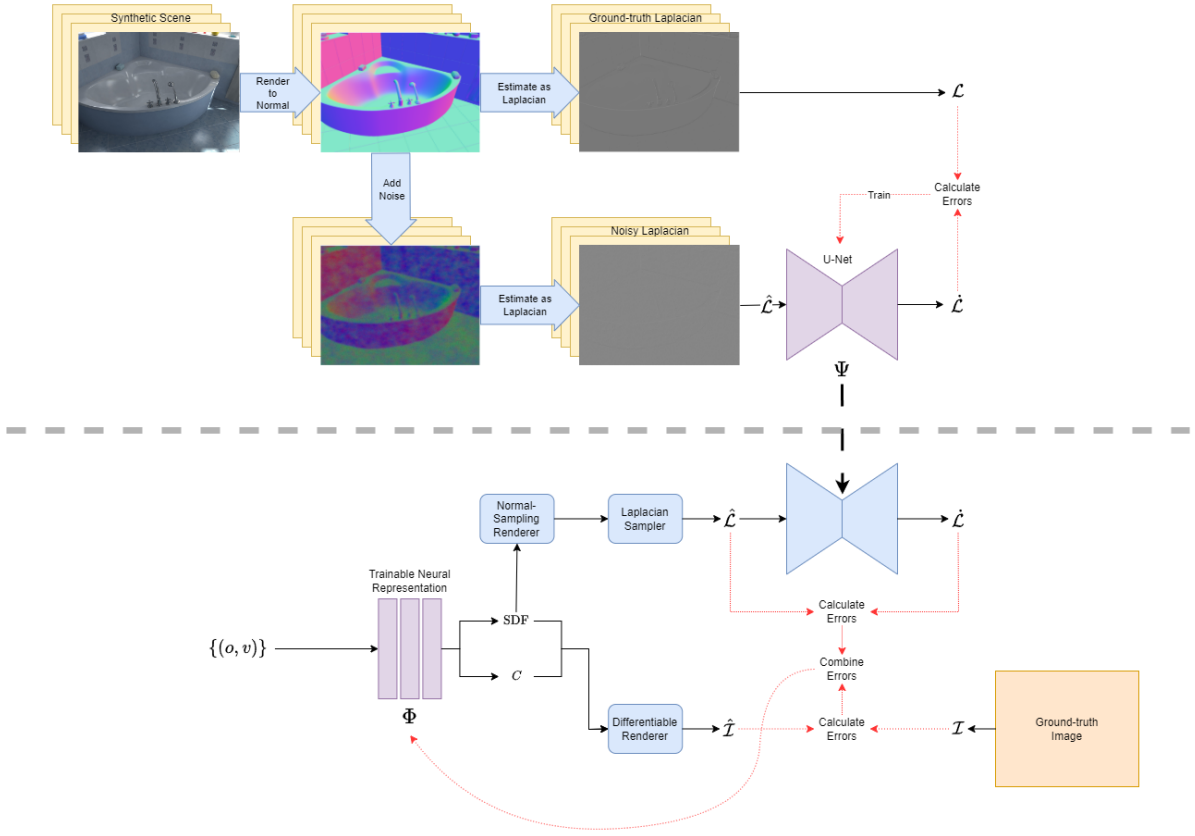


Figure 3.5: The pipeline of NeuLap. Orange boxes denote data, purple boxes denote trainable modules, and blue boxes denote non-trainable modules. The process involves: I. Training a denoising network Ψ using rendered Laplacian images with and without noise. II. Training an augmented implicit neural representation Φ that renders a Laplacian image $\hat{\mathcal{L}}$. This $\hat{\mathcal{L}}$ is denoised by Ψ into $\dot{\mathcal{L}}$, and the difference guides the training of Φ alongside the primary task loss.

of a Signed Distance Function (SDF), $f(\mathbf{p})$. Its rendering equation is:

$$C(o,v) = \int_0^{+infty} w(t)c(\mathbf{p}(t),v)dt \tag{3.8}$$

where $C(o,v)$ is the color for a ray with origin o and direction v, $w(t)$ is a weighting function, and $c(\mathbf{p}(t),v)$ is the color at point $\mathbf{p}(t)$. To obtain surface normals, the gradient of the SDF, $\nabla f(\mathbf{p})$, is required. This can be approximated using finite differences:

$$\nabla f(p) \approx \left(\frac{f(p+\Delta x) - f(p-\Delta x)}{2\Delta x}, \frac{f(p+\Delta y) - f(p-\Delta y)}{2\Delta y}, \frac{f(p+\Delta z) - f(p-\Delta z)}{2\Delta z} \right) \tag{3.9}$$

The augmented NeuS network is modified to render a normal image $\hat{\mathcal{N}}$ by integrating estimated normals near the surface. The normal rendering equation can be approximated as:

$$\hat{\mathcal{N}}(o,v) = \int_0^{+infty} W_\epsilon(f(\mathbf{p}(t)))\nabla f(\mathbf{p}(t))dt \tag{3.10}$$

Here, $W_\epsilon(s)$ is a weighting function that peaks at $s = 0$ (i.e., on the surface $f(\mathbf{p}(t)) = 0$). It is defined as a Gaussian probability density function with zero mean and a small standard deviation ϵ:

$$W_\epsilon(s) = \frac{1}{\epsilon\sqrt{2\pi}} \exp\left(-\frac{s^2}{2\epsilon^2}\right) \tag{3.11}$$

By decreasing ϵ during training, the rendered normal image becomes progressively sharper. The rationale for decreasing ϵ is that as training progresses, the underlying SDF representation $f(\mathbf{p})$ becomes more accurate and its zero-level set becomes more concentrated around the true surface. Initially, a larger ϵ allows the weighting function W_ϵ to integrate information from a wider, less certain region around the estimated surface. As the SDF refines, a smaller ϵ focuses the integration onto this more accurately localized surface, leading to sharper details in the rendered normal map when the SDF itself is reliable.

3.2.2 Laplacian Constraint Integration

The core of NeuLap is the integration of a Laplacian-based geometric constraint. This involves generating Laplacian images, training a network to denoise them, and incorporating this into the main network's loss.

Laplacian Image Generation from Normals

Given an SDF query function $F : \mathbb{R}^3 \to \mathbb{R}$, its Laplacian is:

$$\nabla^2 F(p) = \frac{\partial^2 F(p)}{\partial x^2} + \frac{\partial^2 F(p)}{\partial y^2} + \frac{\partial^2 F(p)}{\partial z^2} \tag{3.12}$$

In our framework, the rendered normal image $\hat{\mathcal{N}}(x, y)$ (either in world or camera space) is used to approximate the Laplacian image of the SDF projected onto the camera plane. The Laplacian is approximated by convolving the rendered normal image. This approach is chosen over direct computation from the SDF (e.g., using second-order finite differences) primarily for practical compatibility with the data generation process for the denoising network Ψ. General-purpose renderers for creating synthetic training scenes are often mesh-based and can readily provide ground-truth normal images. However, they typically do not directly output Laplacian images of an underlying implicit field. Thus, synthesizing noisy and clean (ground-truth) normal images and then converting both to Laplacian images via a consistent convolution process (Equation 3.13) ensures that the denoising network is trained on data that aligns with what the main reconstruction network Φ will produce during its own training. The conversion from normal image to Laplacian image is achieved by convolving each channel of the normal image with the following kernel:

$$K_L = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & -0.5 \\ 0 & -0.5 & 0 \end{pmatrix} \tag{3.13}$$

The results from the three channels (assuming normals are 3D vectors) are summed and normalized (e.g., divided by 3) to obtain the final 2D Laplacian image $\hat{\mathcal{L}}$. This projected Laplacian is view-independent and sparse, capturing essential surface characteristics. The process is visualized in Figure 3.6.

Denoising Network Training

A denoising network Ψ is crucial for generating the supervisory signal. In our implementation, we use a U-Net architecture for Ψ. To train Ψ, a dataset of noisy Laplacian images ($\hat{\mathcal{L}}_{noisy}$) and their corresponding clean ground-truth Laplacian images ($\mathcal{L}_{gt}$) is required. This dataset is synthesized by:

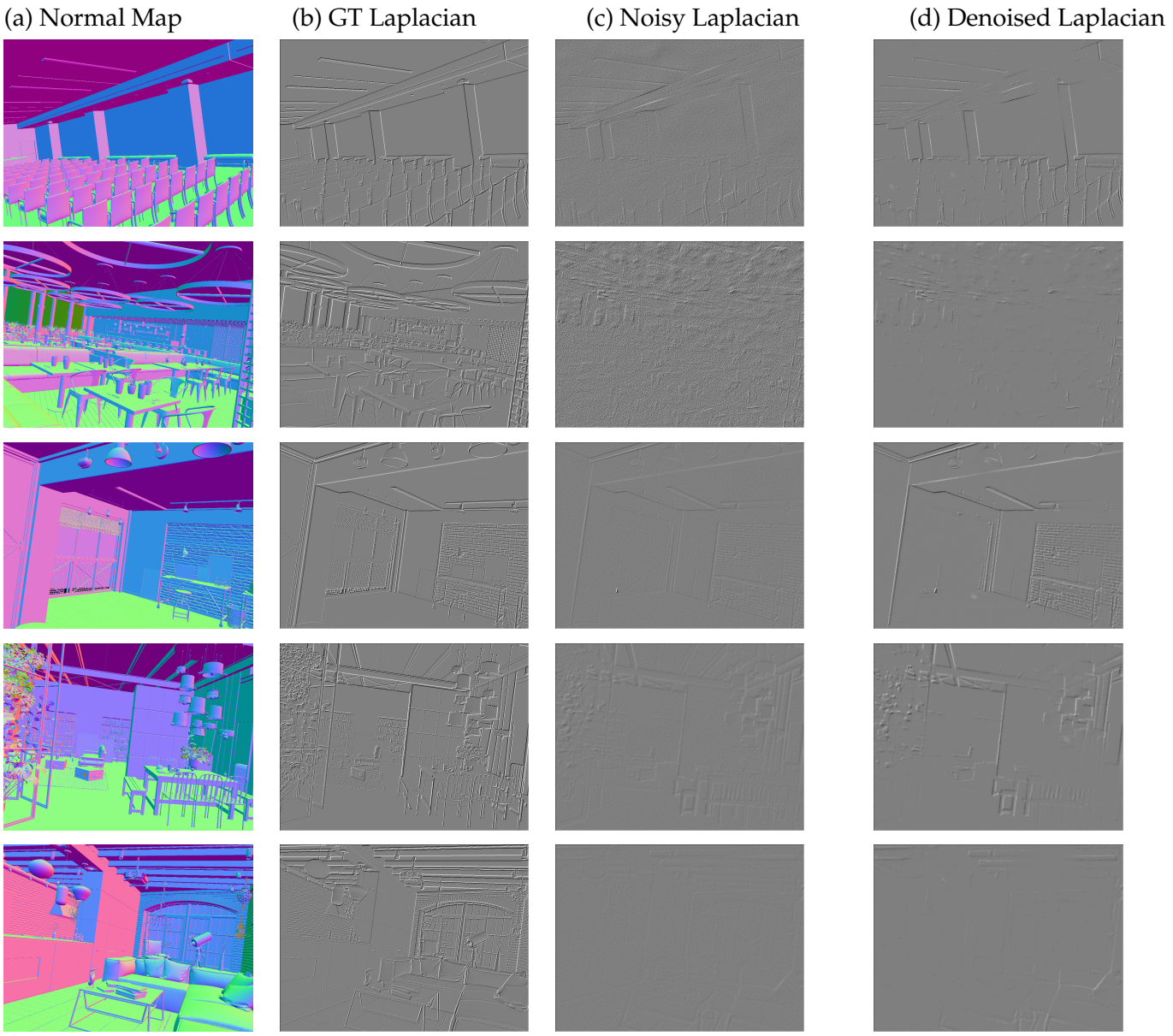


Figure 3.6: Visualization of the Laplacian conversion and denoising process across multiple examples. For each row, from a source **(a) normal map**, we can derive a **(b) ground truth (GT) Laplacian**. During training, the reconstruction network produces a noisy normal map, which results in a **(c) noisy Laplacian**. The denoising network Ψ takes this as input and produces a clean **(d) denoised Laplacian**, which provides the supervisory signal.

1. Rendering ground-truth normal images $\mathcal{N}_{gt}$ from procedurally generated 3D scenes.

2. Adding synthetic noise to $\mathcal{N}_{gt}$ to create noisy normal images $\hat{\mathcal{N}}_{noisy}$. This mimics the output of a partially trained Φ.

3. Converting both $\mathcal{N}_{gt}$ and $\hat{\mathcal{N}}_{noisy}$ to their respective Laplacian images, $\mathcal{L}_{gt}$ and $\hat{\mathcal{L}}_{noisy}$, using the convolution method described above (Equation 3.13).

The U-Net Ψ is then trained to take $\hat{\mathcal{L}}_{noisy}$ as input and output a denoised version $\dot{\mathcal{L}}$ that tries to match $\mathcal{L}_{\text{gt}}$.

The input to the U-Net is structured to leverage contextual information from temporally adjacent frames. Specifically, for a target Laplacian image $L_t \in \mathbb{R}^{W\times H\times C_L}$ (where C_L is the number of channels for the Laplacian, typically $C_L = 1$), two neighboring Laplacian images, L_{n1} and L_{n2}, from adjacent camera viewpoints or time steps are also provided, along with their respective camera poses P_t, P_{n1}, P_{n2}. The processing at the initial stage of the U-Net encoder is as follows:

1. **Pose Augmentation**: Each Laplacian image L_i (for $i \in \{t, n1, n2\}$) is augmented with its corresponding camera pose P_i. The pose is transformed into a per-pixel feature representation $P'_i \in \mathbb{R}^{W\times H\times C_P}$, where C_P is the dimensionality of the pose features. The global camera pose P_i for each image is represented as a 6D vector, comprising the 3D camera position and a 3D representation of its orientation. This vector is broadcast across the spatial dimensions of the image to create a per-pixel feature map $P'_i \in \mathbb{R}^{W\times H\times 6}$, making the pose feature dimensionality $C_P = 6$.

2. **Target Image Feature Extraction**: The augmented target image L''_t is processed by a convolutional layer, $Conv_A$, to produce initial feature maps $F_t = Conv_A(L''_t) \in \mathbb{R}^{W'\times H'\times C_F}$.

3. **Neighboring Images Feature Extraction and Aggregation**:

   (a) The augmented neighboring images L''_{n1} and L''_{n2} are processed by another convolutional layer, $Conv_B$. This layer is designed to generate richer feature representations, for instance, by having its output possess C_N channels, where C_N might be configured to be larger than C_F, in our case, we chose $C_N =$

$2 \cdot C_F$. We denote $F_{n1} = Conv_B(L''_{n1}) \in \mathbb{R}^{W' \times H' \times C_N}$ and $F_{n2} = Conv_B(L''_{n2}) \in \mathbb{R}^{W' \times H' \times C_N}$.

(b) To ensure invariance to the order of the two neighboring images, their feature maps are aggregated using an element-wise max-pooling operation, inspired by PointNet [108]:

$$F_n = \max(F_{n1}, F_{n2}) \in \mathbb{R}^{W' \times H' \times C_N}$$

4. **Feature Concatenation for U-Net Input**: The feature maps from the target image path, F_t, and the aggregated neighboring image path, F_n, are concatenated along their channel dimension:

$$F_{concat} = \text{concat}(F_t, F_n) \in \mathbb{R}^{W' \times H' \times (C_F + C_N)}$$

This concatenated feature map F_{concat} then serves as the input to the subsequent layers of the U-Net's encoder-decoder structure.

Integration with NeuS Training during Reconstruction

Once Ψ is trained, its weights are frozen. During the training of the augmented NeuS network (Φ) in the reconstruction step, in each step:

1. Φ outputs an RGB image $\hat{\mathcal{I}}$ and a normal image $\hat{\mathcal{N}}$.
2. $\hat{\mathcal{N}}$ is converted to a Laplacian image $\hat{\mathcal{L}}$.
3. $\hat{\mathcal{L}}$ is fed into Ψ, which outputs the denoised Laplacian image $\dot{\mathcal{L}}$.

The Laplacian loss is then defined as the difference between the rendered and denoised Laplacians:

$$L_{\text{Lap}} = \frac{1}{m} \sum_k \mathcal{R}(\hat{\mathcal{L}}_k, \dot{\mathcal{L}}_k) \tag{3.14}$$

where m is the number of samples/pixels, and $\mathcal{R}$ is a difference metric, typically L1 loss, to align with NeuS. This Laplacian loss is combined with the original NeuS loss:

$$L = L_{\text{NeuS}} + \eta L_{\text{Lap}} \tag{3.15}$$

The weight η for the Laplacian loss is annealed during training, for example, using an exponential decay: $\eta_t = \gamma \eta_{t-1}$, where $\gamma \in (0, 1)$.

The rationale for annealing the Laplacian loss weight η is to manage the influence of the geometric prior throughout the training process. Initially, when the main reconstruction network Φ is poorly initialized and its output is very noisy, the denoised Laplacian $\dot{\mathcal{L}}$ from Ψ provides a strong and valuable guiding signal towards geometrically plausible surfaces. As training progresses and Φ begins to learn the scene structure more accurately from the input views, its own rendered Laplacian $\hat{\mathcal{L}}$ becomes more reliable. Decreasing η over time reduces the reliance on the external prior. This allows the network to fit the observed data more closely and avoids potential over-smoothing or being overly constrained by a prior that, while generally beneficial, might not perfectly capture all the fine details of the specific scene being reconstructed. It represents a trade-off between adherence to the learned geometric regularities and fidelity to the input images.

3.2.3 Sharp Feature Preservation

A key advantage of using the Laplacian of the implicit field is its inherent ability to represent and thus help preserve sharp geometric features. The Laplacian operator responds strongly to rapid changes in the field, which correspond to high-curvature regions like edges and corners. On flat or smoothly curving surfaces, the Laplacian tends to be small or zero. Specifically, for an SDF, the Laplacian relates to the sum of principal curvatures (mean curvature, up to a factor).

By guiding the reconstruction with a denoised Laplacian target, the network is encouraged to form well-defined structures where the Laplacian is significant (edges, corners) and smooth regions where it is minimal. This property makes the Laplacian prior particularly effective for reconstructing objects with both planar surfaces and sharp details, without explicit feature detection mechanisms.

3.2.4 Experiment Results and Analysis

The efficacy of NeuLap is evaluated through a series of experiments on various datasets and compared against baseline methods. The evaluation focuses on three key aspects: reconstruction and view synthesis quality, robustness under challenging conditions, and computational efficiency.

Experimental Setup

Datasets The Laplacian denoising network Ψ was trained on the Hypersim dataset [117], a large-scale synthetic dataset of indoor scenes rendered as multi-view images including both RGB and normal images. Scanned datasets were avoided for training Ψ to prevent learning scanner-specific noise patterns. For evaluating the augmented NeuS (Φ with NeuLap), we used a separate split of Hypersim for view synthesis tasks to prevent data leakage. Reconstruction accuracy was evaluated on ScanNet [17]. The Synthesis-NeRF dataset [90] was used to test out-of-distribution (OOD) performance.

Baselines Baselines for comparison include the original NeuS [150] and NeuRIS [146]. Results are presented for these methods with and without the NeuLap augmentation. Table 3.2 also includes a "NeRF" column for context, representing a foundational baseline in neural rendering, though direct comparison requires care as discussed in Section 3.2.4.

Noise Simulation for Denoiser Training The noise generation process for training the denoising network Ψ is designed to mimic artifacts from partially trained implicit networks:

1. Apply Gaussian blur with random radius to the ground-truth normal image to mimic the noise from an non-converged implicit network.

2. Add 3D fractal Perlin noise of random strength/frequency, projected onto the camera plane to mimic the distortion during convergence and high-frequency positional encoding effects.

3. Add white noise of random strength.

This creates a dataset of noisy normal images $\hat{\mathcal{N}}_{noisy}$ which are then converted to noisy Laplacian images.

Evaluation Metrics The quality of the 3D reconstruction is evaluated using several standard metrics, based on the reconstructed point cloud C and the ground-truth point cloud C^*. These metrics are defined as follows:

- **Accuracy (Acc.)**: The average distance from each reconstructed point to its nearest neighbor in the ground-truth cloud, defined as $\text{mean}_{c \in C}(\min_{c^* \in C^*} \|c - c^*\|)$. Lower values indicate that the reconstruction is closer to the true surface.
- **Completeness (Comp.)**: The average distance from each ground-truth point to its nearest neighbor in the reconstructed cloud, defined as $\text{mean}_{c^* \in C^*}(\min_{c \in C} \|c^* - c\|)$. Lower values indicate that the reconstruction covers the ground-truth object more thoroughly.
- **Precision (Prec.)**: The fraction of reconstructed points for which the distance to the nearest ground-truth point is less than a threshold τ (e.g., 5 cm). It measures the fidelity of the reconstruction. Higher is better.
- **Recall**: The fraction of ground-truth points for which the distance to the nearest reconstructed point is less than the threshold τ. It measures how much of the ground truth is captured. Higher is better.
- **F-score**: The harmonic mean of Precision and Recall, calculated as $2 \cdot \frac{\text{Prec.} \cdot \text{Recall}}{\text{Prec.} + \text{Recall}}$. It provides a single score that balances precision and recall. Higher values are better.

Quantitative Evaluation

The quantitative impact of NeuLap was assessed across several benchmarks.

Table 3.2 presents a comprehensive comparison of our method against several state-of-the-art 3D reconstruction techniques on the ScanNet dataset. The evaluation is based on multiple standard metrics: Accuracy (Acc.), Completeness (Comp.), Precision (Prec.), Recall, and F-score. Our method (referred to as "Ours") consistently outperforms other baselines, including COLMAP [120], NeuralRecon [127], NeRF [90], NeuS [150], and NeuRIS [146], across all reported metrics, demonstrating its superior reconstruction quality.

Robustness to OOD and Limited Data Table 3.3 presents results on the Synthesis-NeRF dataset (OOD) and on ScanNet with 50% of the training data, now evaluated using F-score. These experiments demonstrate NeuLap's ability to stabilize performance under challenging conditions. With only half the data, the F-score for both NeuS and NeuRIS is improved with the NeuLap prior, suggesting it is particularly helpful when

data is scarce. In the OOD scenario, NeuLap brings notable improvements for NeuRIS, although it slightly degrades performance for NeuS, suggesting the learned prior may not generalize perfectly across all architectures.

Ablation Study

To validate the contributions of the key components within the NeuLap framework, we conduct an ablation study using NeuRIS as the baseline. We analyze the impact of the denoising network's architecture, specifically its use of neighboring frames and pose information. We compare our full model against several variants:

- **Baseline**: The original NeuRIS model without any Laplacian-based guidance.
- **Full NeuLap**: The proposed method with all components, applied to NeuRIS. This includes the U-Net denoiser that uses the target frame, two neighboring frames, and their corresponding camera poses.
- **w/o Neighboring Frames**: The denoising network Ψ is trained and evaluated using only the single target Laplacian image, without temporal context from adjacent views.
- **w/o Pose Information**: The denoising network Ψ operates on Laplacian images alone, without the camera pose features concatenated to the input.
- **Laplacian Regularization**: Instead of using a learned denoising network, we apply a simple L1 regularization directly on the rendered Laplacian image ($\lambda||\hat{\mathcal{L}}||_1$), encouraging sparsity. This tests the benefit of a learned prior versus a generic one.

The results, presented in Table 3.4, are evaluated on the ScanNet dataset using the F-score metric.

The results from this study are expected to show that each component contributes positively to the final performance, with the full model achieving the best reconstruction quality. This would validate our design choices for the geometry-aware training framework.

Qualitative Evaluation

In addition to quantitative metrics, visual inspection of the reconstructed geometry provides critical insight into the benefits of NeuLap. Qualitative comparisons on a simple and a complex scene from ScanNet are shown in Figure 3.7 and Figure 3.8, respectively. For each scene, we compare the reconstruction from the baseline NeuRIS model against our NeuLap-augmented version and the ground truth.

The visual results highlight the contributions of the Laplacian prior. When used together with the NeuRIS model, the NeuLap augmentation provides key refinements that bring the results closer to the ground truth. For instance, in the simpler scene (Figure 3.7), the prior visibly improves the reconstruction by sharpening edges and flattening large planar surfaces like the ground and walls (View 1). In areas with more complex geometry (Views 2 and 3), the prior enhances the definition and correctness of the structure. This targeted improvement is also evident in the complex scene (Figure 3.8), where NeuLap helps produce more coherent and detailed surfaces. The prior effectively guides the network to refine the geometry, reducing subtle surface noise and preventing the over-smoothing of sharp features, leading to a more faithful representation of the underlying geometry.

Discussion of Results

The experimental results demonstrate the general effectiveness of the NeuLap prior. However, some specific outcomes warrant further discussion. One notable observation from Table 3.2 is that the baseline NeRF model achieves a higher F-score than the baseline NeuS model (0.455 vs. 0.412) on the ScanNet dataset. While NeuS produces reconstructions with higher accuracy (0.107 vs. 0.162), indicating that the extracted surfaces are closer to the ground truth where they exist, it struggles with completeness (0.122 vs. 0.066). This discrepancy may be attributed to the fundamental differences in their underlying representations when applied to complex indoor environments. NeuS, being based on a Signed Distance Function (SDF), is optimized to learn continuous, watertight surfaces. This property is advantageous for clean object-centric reconstructions but can be a limitation in cluttered and geometrically complex indoor scenes like those in ScanNet, which often feature thin structures, disconnected elements, and background clutter. The strict SDF assumption can lead NeuS to produce overly smoothed

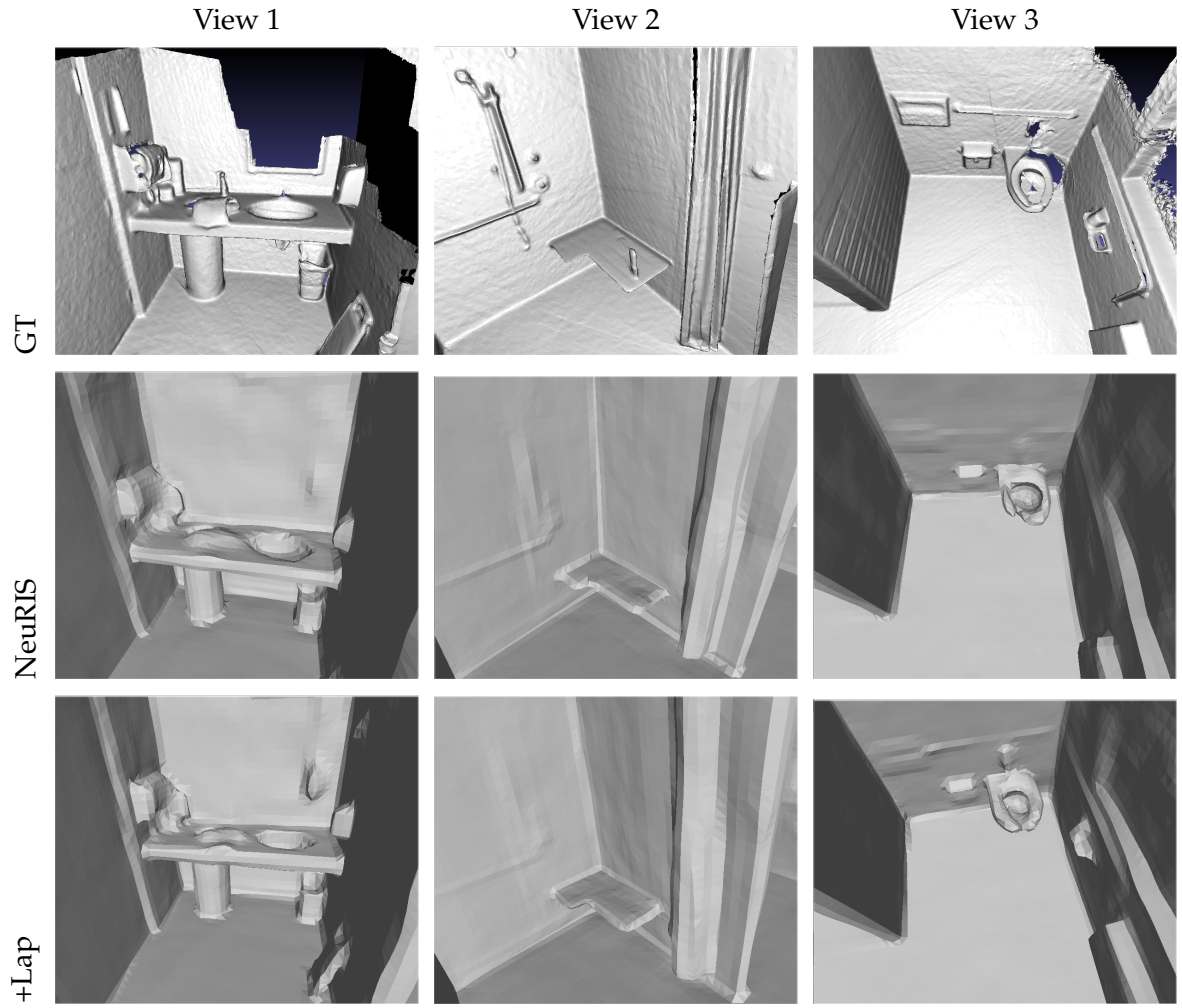


Figure 3.7: Qualitative comparison of reconstruction results on a simple scene. We show the **Ground Truth**, the reconstruction from baseline **NeuRIS**, and the result from our **NeuRIS+Lap**. NeuLap yields smoother surfaces and better preservation of sharp geometric features.

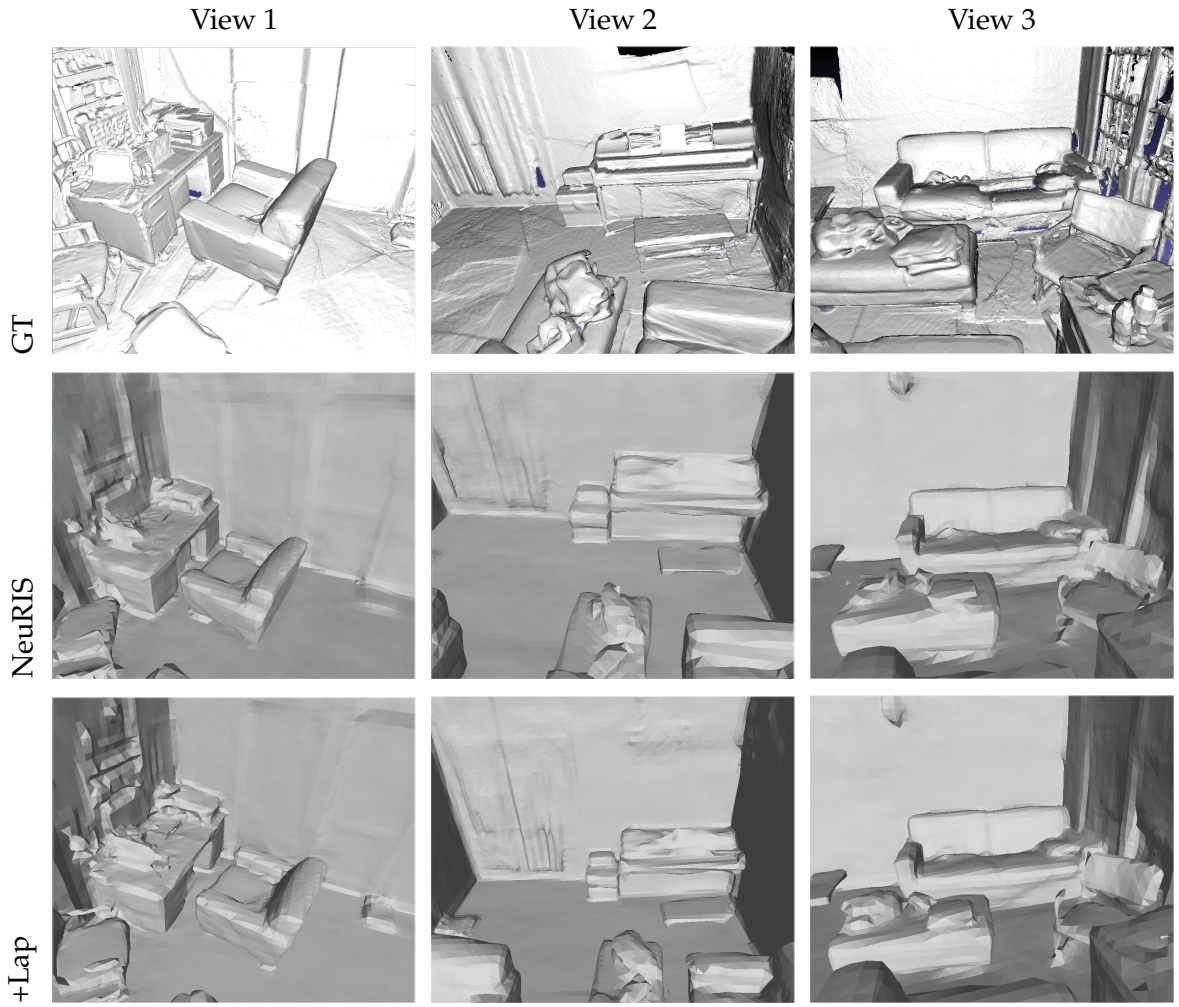


Figure 3.8: Qualitative comparison of reconstruction results on a complex scene. We show the **Ground Truth**, the reconstruction from baseline **NeuRIS**, and the result from our **NeuRIS+Lap**. NeuLap helps produce more coherent and detailed surfaces, leading to more complete reconstructions.

or incomplete surfaces in these challenging regions. In contrast, NeRF's volumetric representation is more flexible and does not impose a strict surface constraint. It can more effectively capture volumetric details and non-watertight geometry, leading to better completeness and recall, which in turn results in a higher overall F-score. This highlights a key trade-off: NeuS provides higher fidelity on well-defined surfaces, while NeRF offers more robust coverage of complex scene content. It is important to note that subsequent SDF-based methods like NeuRIS have already addressed this limitation, achieving a significantly higher F-score than NeRF (0.691 vs. 0.455) by better handling indoor scene complexities. Our NeuLap augmentation pushes this further, improving both NeuS and NeuRIS to achieve state-of-the-art results that combine high accuracy and completeness.

3.2.5 Conclusion and Future Work

This work proposed NeuLap, a novel method for integrating structural priors into implicit neural field training. By training a denoising network for Laplacian images and using its output as a supervisory signal, surface information is encoded as a geometric prior. This prior guides the training of the implicit reconstruction network via an auxiliary loss term. NeuLap effectively addresses the challenge of coupling reconstruction with priors by integrating information into the error function itself. Experimental results, while needing further quantitative data on acceleration, suggest that NeuLap can improve reconstruction quality and stabilize performance, particularly with limited data or in OOD scenarios.

Several avenues for future research emerge from this work:

1. Investigating more advanced models than a simple U-Net for denoising Laplacian images, potentially leveraging the sparse nature of these images.

2. Developing more sophisticated algorithms to mimic or directly sample the noise characteristics of partially trained implicit reconstruction networks for better denoiser training.

3. Extending the NeuLap methodology to other implicit reconstruction and view synthesis frameworks beyond NeuS.

These questions offer promising directions for further advancing efficient and robust 3D neural reconstruction.

3.3 Partitioned Grid Representation for Parametric Surfaces

3.3.1 Problem Statement

Reconstructing 3D shapes from point clouds into parametric surfaces is crucial for CAD applications. However, existing methods often struggle with noisy input data or produce representations that lack topological guarantees or are difficult to edit. This section introduces a method aimed at generating a structured, topologically informed intermediate representation based on a partitioned grid, which can facilitate the subsequent fitting of parametric surface patches. The goal of this research is to create a framework that is robust to noise and can capture the coarse topology of the input shape and make it easier for generating parametric surfaces suitable for CAD applications.

3.3.2 Proposed Method: Grid Partitioning Network

The proposed method utilizes a deep neural network to infer a partitioned 3D grid structure that approximates the input point cloud's topology. This grid serves as an intermediate representation for subsequent parametric surface fitting.

The process involves distinct stages for training and inference.

Training Stage

Given an input training pair (X, Y), where $X \in \mathbb{R}^{N\times 3}$ is a noisy point set and $Y \in \mathbb{R}^{M\times 6}$ is a corresponding dense, denoised point set with normals (p_i, n_i), the network is trained as follows:

1. **Orientation Normalization:** Both X and Y are rotated using a Spatial Transformer Network [49] to align them with a canonical orientation. This minimizes the projection error of points in Y onto the principal axes defined by $\mathcal{T} = \{\pm\mathbf{e}_1, \pm\mathbf{e}_2, \pm\mathbf{e}_3\}$. Let the transformed sets be X' and Y'.

2. **Supervisory Signal Generation (Target Grid Construction):** The supervisory signal Y' is used to define a target grid structure (P, O).

   - Points in Y' are grouped based on their normal vector's alignment with the axes in $\mathcal{T} = \{\pm\mathbf{e}_1, \pm\mathbf{e}_2, \pm\mathbf{e}_3\}$. The group index k for a point with normal $\mathbf{n}$ is determined using Equation 3.16.

$$k = \underset{j\in\{1..6\}}{\arg\max}(\mathcal{T}_j \cdot \mathbf{n}) \tag{3.16}$$

   - For each group G_k, points are projected onto the corresponding axis direction. Kernel Density Estimation (KDE) with Gaussian kernels is applied to the projected positions. Local maxima of the estimated density define the positions of orthogonal planes along that axis. This approach is preferred over K-Means clustering, which struggled with uneven plane distributions (e.g., closely spaced planes for detailed areas vs. distant planes for smooth regions). The set of all inferred planes defines the target grid boundaries P.
   - These planes P partition the 3D space into a set of grid cells $\mathbb{V}$. An occupancy vector $O \in \{0, 1\}^{|\mathbb{V}|}$ is determined by checking which grid cells contain points from Y'. Soft occupancy $O^{\text{soft}} \in [0, 1]^{|\mathbb{V}|}$ can also be used.

3. **Network Architecture and Loss:** The input X' is processed by an encoder network (e.g., PointNet++ [109] or PointConv [161]) to produce a latent code Z. An MLP decoder with two heads takes Z as input:

   - Head 1: Predicts the plane parameters $\hat{P}$ (linear output).
   - Head 2: Predicts the cell occupancy $\hat{O}$ (sigmoid output).

   The network is trained by minimizing a combined loss:

$$L = \lambda_p L_p + \lambda_o L_o$$

   where $L_p = MSE(P, \hat{P})$ is the plane position loss and $L_o = CrossEntropy(O, \hat{O})$ is the occupancy loss. λ_p and λ_o are weighting factors.

Inference Stage

During inference, given a new input point cloud X_{test}:

1. X_{test} is normalized using the learned Spatial Transformer Network.
2. The normalized point cloud X'_{test} is fed through the trained encoder-decoder network.
3. The network outputs the predicted grid structure $(\hat{P}, \hat{O})$. This structure represents the inferred topology and coarse geometry of the input shape.

Reconstruction into Parametric Surfaces

The inferred grid structure $(\hat{P}, \hat{O})$ serves as a scaffold for generating the final parametric surface representation. This reconstruction is a post-processing step performed during inference. The process consists of initializing and refining a network of Bèzier patches:

1. **Cubic Bèzier Patch Initialization:** Each external facet of an occupied grid cell (where $\hat{O} >$ threshold) is converted into a cubic Bèzier patch. The initial control points are derived from the grid vertices. Crucially, the connectivity of the patches is inherited from the grid, meaning adjacent patches share control points along their common edges. This ensures an initial C^0 continuity across the surface.
2. **Iterative Fitting to Point Cloud:** The control points of the patch network are then iteratively optimized to fit the input point cloud X_{test}. The objective is to minimize the Chamfer-L1 distance between points sampled from the Bèzier surfaces and the input cloud. During each optimization step:
    - A dense set of points is sampled from the current Bèzier surfaces.
    - The Chamfer-L1 distance to X_{test} is computed.
    - The gradient of the distance with respect to the control point positions is calculated. This gradient indicates the direction to move each control point to reduce the fitting error.
    - Control points are updated according to the gradient. Shared control points receive a combined update, maintaining surface continuity.

This iterative process typically converges within 1.2 seconds, resulting in a fitted parametric surface that captures the geometry of the input data while respecting the topology defined by the grid.

3.3.3 Experiments and Results

Experimental Setup

We evaluated the Grid Partitioning Network on the ShapeNetCore dataset [9] . We used the official train-test split of ShapeNetCore, and reserved 10% of the training set for validation. The primary metric used for evaluating the final fitted surface quality is the Chamfer Distance (CD) between the reconstructed surface points and the ground truth point cloud Y. We compare against AtlasNet [35] as a baseline for patch-based reconstruction.

Intermediate Results: Grid Topology

Figures 3.9 to 3.14 show examples of the intermediate grid structure $(\hat{P}, \hat{O})$ generated by the network during inference. The input point cloud (X_{test}) is shown alongside the visualized occupied grid cells. These results demonstrate the network's ability to capture the coarse topology of various shapes, such as chair structures (Figure 3.9) and tables (Figure 3.11). However, challenges remain for highly complex topologies (e.g., Figure 3.10). Note that the visualized planes $\hat{P}$ may not perfectly align with the surface; their role is primarily to define the grid structure. The occupancy $\hat{O}$ determines the shape's structure within that grid.

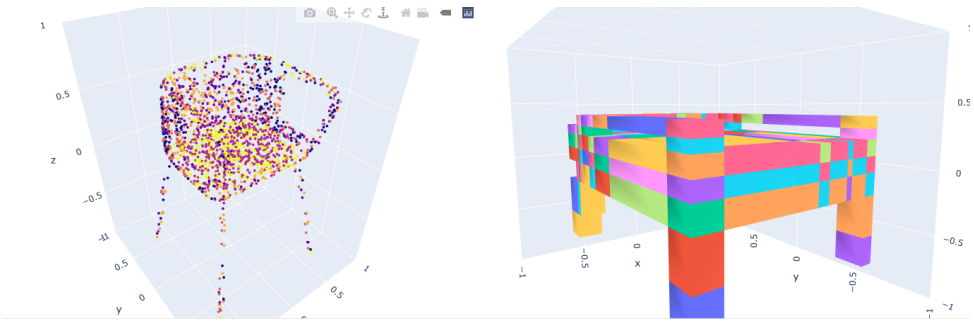


Figure 3.9: Example of the inferred grid topology for a chair. Left: Input point cloud (X_{test}). Right: Visualized occupied grid cells based on $(\hat{P}, \hat{O})$. The grid captures the overall structure, including the ring shape of the arms.

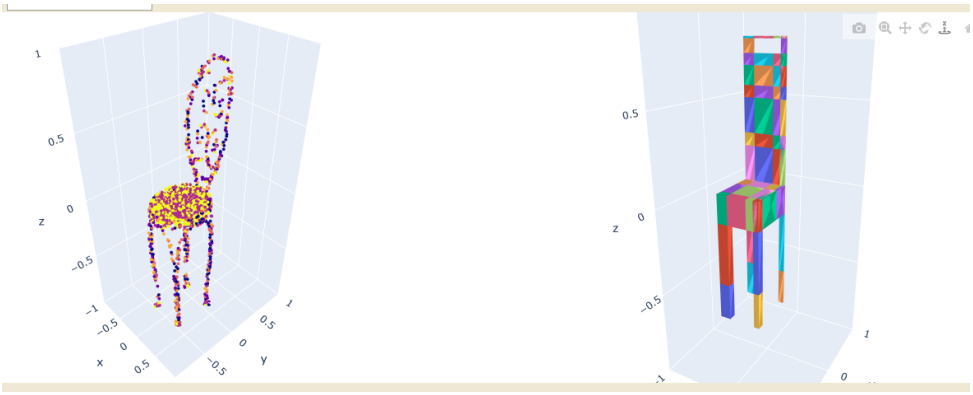


Figure 3.10: Example of a more complex chair structure where the inferred grid topology struggles to capture all details accurately.

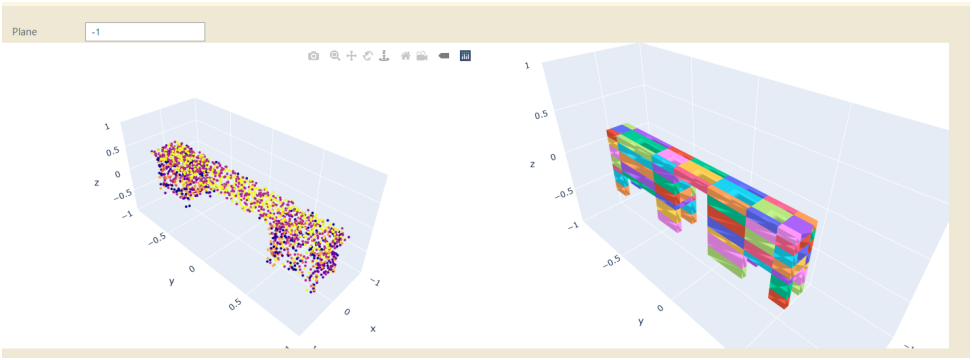


Figure 3.11: Example of the inferred grid topology for a table.

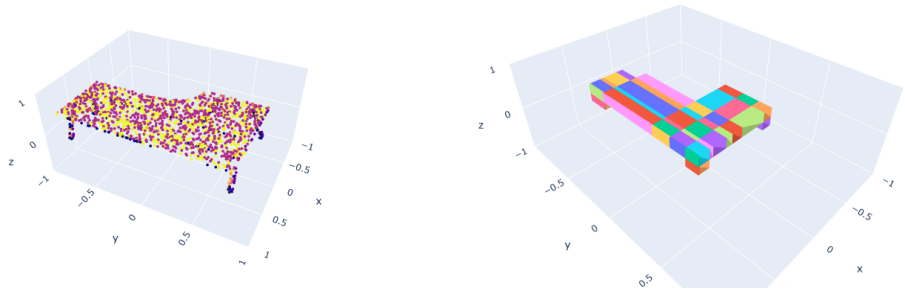


Figure 3.12: Another example of the inferred grid topology for a table.

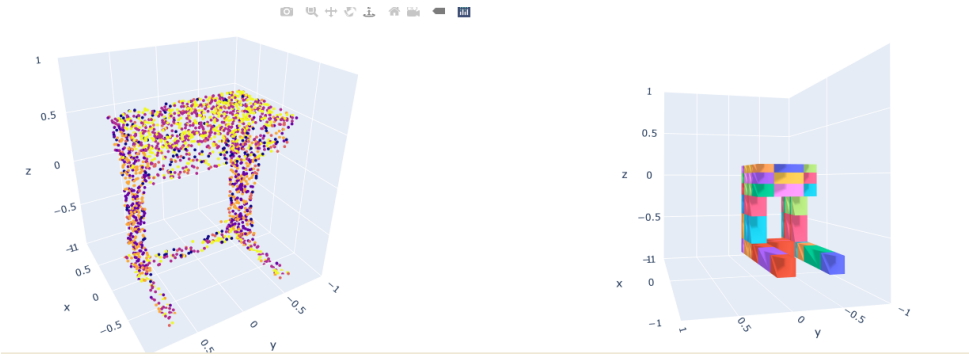


Figure 3.13: Example of the inferred grid topology for a desk.

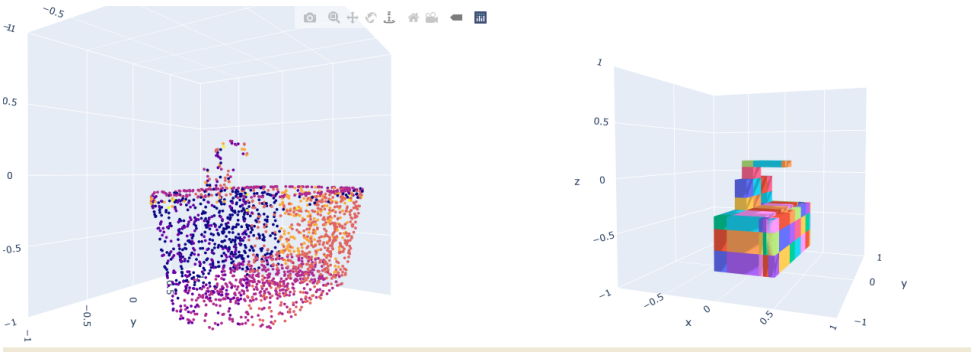


Figure 3.14: Example of the inferred grid topology for a basin.

Experiment Results: Surface Reconstruction

Table 3.5 presents the Chamfer Distance results after fitting cubic Bèzier patches (see Section 3.3.2) and compares them to AtlasNet. The "Grid-NxNxN" notation indicates the resolution of the partitioning grid inferred by the network. The number of resulting surface patches varies depending on the inferred occupied grid cells but is implicitly controlled by the grid resolution N. For instance, Grid-7x7x7 implies a maximum potential of $6 \times 7^2 = 294$ faces per axis pair, though the actual number is determined by $\hat{O}$.

3.3.4 Analysis and Discussion

The proposed Grid Partitioning Network offers a method for inferring a topologically aware structure from point clouds, which serves as an intermediate step towards parametric surface reconstruction. The desired key advantage lies in generating a structured representation that implicitly defines patch connectivity, The intermediate results (Figures 3.9-3.14) show promise in capturing coarse topology. Quantitative

results (Table 3.5) indicate that higher grid resolutions (e.g., 7x7x7) achieve competitive Chamfer Distance compared to the AtlasNet baseline, suggesting the derived cubic Bèzier surfaces can effectively approximate the target shapes.

However, limitations identified during development:

- **Detail Representation:** The current approach of deriving patches from grid facets can lead to blocky intermediate representations and potentially oversmoothed final surfaces after Bèzier fitting, especially when compared to methods directly optimizing patch layouts. Fine details might be lost.
- **Grid Resolution Sensitivity:** Performance is sensitive to the chosen grid resolution (N). A low N might miss details, while a high N increases complexity and does not guarantee better topological accuracy, as seen in failure cases (Figure 3.10).
- **Limited OOD Generalization:** The performance on OOD categories (*cel., *wat.) degrades compared to in-distribution categories (Table 3.5). This suggests the fixed orthogonal partitioning struggles with significantly different shape structures not seen during training. The reliance on axis-aligned partitioning might be too restrictive.
- **Intermediate Representation Quality:** While the topology is often captured, the exact placement of the inferred planes $\hat{P}$ might not be optimal, potentially requiring significant refinement during the final Bèzier fitting stage. Comparison with alternative intermediate representations, such as learned cuboid abstractions, is warranted in future work.

Future work should focus on adaptive grid structures, investigating alternative post-processing methods for surface generation (e.g., differentiable marching cubes), and incorporating geometric priors or regularizers during network training or patch fitting to improve the quality and CAD-suitability of the final parametric surfaces.

3.4 Hierarchical Attention Propagation in Octrees

This section introduces a novel hierarchical attention mechanism tailored for OctTrees. It is designed to generate powerful multi-scale features for 3D deep learning tasks.

While the method is general-purpose, this chapter focuses on its application to implicit 3D reconstruction, where high-quality feature representations are crucial for recovering detailed geometry from sparse or noisy input. Our approach facilitates efficient feature learning and propagation. It addresses key challenges in processing extensive and complex 3D spatial data. We begin by discussing the motivation for this architecture within the context of reconstruction. We then define the problem and detail the proposed method.

3.4.1 Motivation and Background

Implicit 3D reconstruction methods have shown great success in representing complex topologies. However, their effectiveness heavily relies on the quality of the underlying feature representations. These features must capture both fine-grained local geometry and broader global context from sparse inputs. The processing of 3D spatial data to extract such features presents unique challenges, including managing computational complexity and capturing multi-scale geometric details. Traditional deep learning models often struggle with the irregular structure of 3D data. Attention mechanisms, particularly transformers, have shown immense success in other domains. They can model long-range dependencies and adaptively focus on relevant information. Applying attention to hierarchical 3D structures like OctTrees is a promising direction. OctTrees offer an efficient and adaptive representation of 3D space. They concentrate computation where detail is present. However, naively applying dense attention to all nodes or voxels in 3D is often intractable. This is due to the cubic growth in data points and quadratic complexity of attention. Our work explores how to effectively integrate attention mechanisms within the OctTree framework. This allows for scalable and powerful feature learning in 3D. While attention on OctTrees is not entirely new, the specific implementation details and architectural goals are crucial differentiators. For instance, models like OctFormer [151] also leverage Octrees, but their primary goal is to solve an efficiency challenge for tasks like segmentation and detection. OctFormer partitions point clouds into local windows that contain a fixed number of points, enabling efficient batch processing for localized attention. Its attention mechanism is therefore primarily constrained within these local windows.

In contrast, our method is designed for implicit 3D reconstruction and focuses on

building a global understanding of the geometry through explicit cross-scale information flow. Instead of localized window-based attention, we use iterative bottom-up aggregation and top-down refinement passes that operate over the entire OctTree structure. This message-passing strategy is designed to build a deep contextual understanding by propagating information across the full hierarchy, which fundamentally differs from OctFormer's localized attention approach.

The subsequent paragraphs briefly review fundamental concepts. These concepts underpin our approach to attention in OctTrees.

Knowledge Representation and Message Propagation

Neural networks learn by transforming input data through layers of neurons. This process can be viewed as extracting, filtering, and reorganizing components of information. Deep learning models use cascaded structures for knowledge representation, with each layer extracting features and passing them to the next. Traditional architectures often have limited information aggregation scopes. They also tightly couple data representation with specific neuron indices, while vector-based knowledge representation and message passing offer more flexibility. Concepts are encoded as vectors, allowing richer representations. Message passing enables dynamic information flow between neuronal units. While CNNs and GCNs pass information along explicit local links, offering efficiency, they make strong local assumptions. Transformers use vector representation and self-attention for message passing. They implicitly generate a soft graph, allowing long-range information flow.

Rationale for OctTree-based Transformers

OctTrees provide an efficient, adaptive, and hierarchical 3D data representation. This structure aligns well with multi-scale processing capabilities of transformers. The hierarchy can be leveraged for sparse attention, reducing the computational load. This combination promises scalability for large and complex 3D scenes. OctTrees inherently encode positional information, potentially simplifying or enhancing positional encoding, and their adaptive resolution matches the transformer's ability to focus attention. This enables contextual understanding at multiple scales simultaneously. Principles developed for OctTrees can generalize to other tree-based data structures, providing a

novel approach to attention in processing tree-structured data.

3.4.2 Problem Statement

The primary task addressed in this section is high-fidelity 3D shape reconstruction from point cloud. We frame this as learning an implicit function, such as an occupancy field, which requires rich feature descriptors at any queried 3D location. The input to our system is a point cloud of a 3D scene, which is first converted into an OctTree structure. The core problem is to develop an efficient and powerful mechanism to compute and refine feature representations for each node in this OctTree. These features must capture both local geometric details and global contextual information from the input data. The output is a hierarchical feature grid, embodied by the OctTree, where each node stores a rich feature vector. For the task of reconstruction, these features can be interpolated to provide a feature vector for any continuous 3D coordinate, which is then decoded into an occupancy value. While our experimental validation focuses on reconstruction, the resulting feature hierarchy is general-purpose and can be readily applied to other downstream tasks like semantic segmentation or object detection. It may also be generalized to other tree-based data structures. The overall goal is to achieve state-of-the-art representation quality while maintaining computational tractability for large-scale scenes.

3.4.3 Proposed Method: Hierarchical Attention Propagation

We propose a hierarchical attention mechanism operating on OctTrees. This method integrates the strengths of OctTree data structures with attention-based message passing. The core idea is a two-pass process: bottom-up aggregation and top-down refinement. This facilitates comprehensive information flow across different scales of the OctTree.

Multi-scale Feature Learning

Initial OctTree Construction and Leaf Node Encoding The process begins by constructing an OctTree from the input 3D scene data. For very large scenes, the overall space is first divided into a grid, with each cell of this grid then represented by an individual OctTree. This partitioning facilitates manageable processing and allows for

interactions at the root levels of these Octrees, contributing to a global scene understanding. Within each OctTree, each node represents a spatial region, and leaf nodes represent the finest level of detail. To capture local geometric features at the leaf level, we employ Point Convolution. Specifically, we can use operations like PointConv [161]. The features from raw point cloud data within each leaf node are aggregated. This aggregation, typically mean pooling, produces an initial latent code for each leaf node.

Fourier Position Encoding To enhance the model's understanding of spatial relationships, we use Fourier position encoding. The spatial coordinates (x, y, z) of each OctTree cell (or node center) are mapped to a higher-dimensional space. This encoding is expressed as:

$$\gamma(x,y,z) = [\sin(2^0\pi x), \cos(2^0\pi x), ..., \sin(2^{N-1}\pi x), \cos(2^{N-1}\pi x), \quad \sin(2^0\pi y), \cos(2^0\pi y), ..., \sin(2^{N-1}\pi y), \tag{3.17}$$

Here, N is the number of frequency bands. This encoding provides a rich, continuous representation of position. It is concatenated with the cell's feature vector. This provides explicit spatial context to the model.

Cross-scale Information Flow

Bottom-Up Attention Propagation Information propagates from leaf nodes towards the root in a bottom-up manner. For each parent node at depth d, its latent code $z^{(d)}_{(...,i)}$ is computed by an attention-weighted aggregation of its children's latent codes $z^{(d+1)}_{(...,i,j)}$, stated in equation 3.18.

$$z^{(d)}_{(...,i)} = \sum_{j=0}^{7} \alpha_{ij} z^{(d+1)}_{(...,i,j)} \tag{3.18}$$

The attention weights α_{ij} are computed using scaled dot-product attention. Query $q_i^{(\alpha)}$ and key $k_j^{(\alpha)}$ vectors are derived from node features:

$$\begin{aligned} q_i^{(\alpha)} &= W_q z^{(d)}_{(...,i)} \\ k_j^{(\alpha)} &= W_k z^{(d+1)}_{(...,i,j)} \end{aligned} \tag{3.19}$$

W_q and W_k are learnable weight matrices. Notably, the value vector V (typically $W_v z$) is omitted in this formulation. Instead, the children's latent codes $z^{(d+1)}_{(...,i,j)}$ are used directly as the values. This design choice means the attention weights α_{ij} are applied directly to

the input features from the layer below. The purpose is to maintain a more direct linear combination of the source features, weighted by their relevance, rather than combining transformed value vectors. The attention weights are computed as:

$$\alpha_{ij} = \frac{\exp({q_i^{(\alpha)}}^T k_j^{(\alpha)}/\sqrt{d_k})}{\sum_{j'=0}^{7} \exp({q_i^{(\alpha)}}^T k_{j'}^{(\alpha)}/\sqrt{d_k})} \tag{3.20}$$

where d_k is the key vector dimensionality. This pass aggregates short-range spatial information up the tree. Each node thus obtains a latent code as a weighted combination of its children's codes, where the weights are learned through the attention mechanism.

Top-Down Refinement A top-down pass refines node representations to incorporate broader context (mimicking long-range attention). Starting at the root, a new latent code $\zeta^{(0)}$ is inferred for the root node using a similar attention mechanism with different projection matrices $W_q^{(\zeta)}, W_k^{(\zeta)}$. The difference $\Delta z^{(0)} = z^{(0)} - \zeta^{(0)}$ is calculated. This difference is propagated down the tree. For a parent node at level d with difference $\Delta z_{(\ldots,i)}^{(d)}$, the difference propagated to its child j is:

$$\Delta z_{(\ldots,i,j)}^{(d+1)} = \alpha_{ij} \Delta z_{(\ldots,i)}^{(d)} \tag{3.21}$$

Here, α_{ij} are the attention weights from the bottom-up pass. Child node latent codes are updated:

$$z_{(\ldots,i,j)}^{(d+1)} \leftarrow z_{(\ldots,i,j)}^{(d+1)} + \Delta z_{(\ldots,i,j)}^{(d+1)} \tag{3.22}$$

This recursive process allows global refinements to propagate throughout the OctTree.

Iterative Refinement The bottom-up and top-down passes can be iterated multiple times. This allows for more complex information flow and potentially more robust representations. Let Z_k denote the set of all node features at the beginning of iteration k. One full iteration (bottom-up pass followed by a top-down pass) constitutes a transformation $\mathcal{F}$, such that $Z_{k+1} = \mathcal{F}(Z_k)$. The function $\mathcal{F}$ is deterministic for fixed model weights (e.g., W_q, W_k).

During the bottom-up pass, parent features $z_p^{(k+1),BU}$ are computed based on attention over child features $z_{c_j}^{(k)}$. The queries $q_p = W_q z_p^{(k)}$ and keys $k_{c_j} = W_k z_{c_j}^{(k)}$ make the attention weights $\alpha_{p,c_j}^{(k)}$ dependent on Z_k. The update is $z_p^{(k+1),BU} = \sum_j \alpha_{p,c_j}^{(k)} z_{c_j}^{(k)}$. The top-down pass then uses these fresh $z^{(k+1),BU}$ features and $\alpha^{(k)}$ weights to compute

corrections and update nodes to Z_{k+1}. This entire transformation $\mathcal{F}$ is continuous but highly non-linear.

A formal proof of convergence of the sequence Z_k to a unique fixed point Z^* (where $Z^* = \mathcal{F}(Z^*)$) is mathematically challenging for such a general non-linear iterative scheme. It would typically require showing that $\mathcal{F}$ is a contraction mapping in a suitable metric space, which is not straightforward without specific constraints on the learned weights or feature distributions.

However, several aspects are noteworthy:

- The softmax function ensures attention weights sum to one for each parent over its children, making the bottom-up feature aggregation a weighted average, which can have stabilizing effects.

- In practice, similar iterative processes in graph neural networks are often run for a fixed, small number of iterations. The model weights are trained end-to-end through the unrolled iterations, learning to make this fixed number of steps effective for the downstream task.

- Alternatively, iteration continues until an empirical convergence criterion is met, such as the change $\|Z_{k+1} - Z_k\|$ falling below a threshold, or a maximum number of iterations is reached. This is to balance refinement quality with computational cost and to mitigate potential over-smoothing, where features might become too similar with excessive iterations.

The focus is on achieving useful representations within a practical number of steps, as shaped by the learning process of the overall model.

Algorithmic Summary The complete process is summarized in Algorithms 1, 2, and 3. The main algorithm outlines the overall iterative refinement, while the subsequent algorithms detail the bottom-up aggregation and top-down refinement passes.

Algorithm 1: Overall OctTree Attention Propagation

Input : 3D scene data (e.g., point cloud)

Output: OctTree with refined node representations

```
octree ← ConstructInitialOctTree(scene);
foreach leafNode in octree.leaves do
    leafNode.latentCode ← PointConvolutionFeatures(leafNode.points);
    leafNode.latentCode ← Concatenate(leafNode.latentCode,
     FourierEncode(leafNode.position));
end foreach
for iteration from 1 to maxIterations do
    octree ← BottomUpPass (octree);
    octree ← TopDownPass (octree);
    if ConvergenceCriterionMet() and iteration > 1 then
        break ;
    end if
end for
```

Algorithm 2: Bottom-Up Attention Pass

Input : OctTree with node representations

Output: OctTree with updated parent representations and attention weights

```
Function BottomUpPass(octree)
    for level from octree.depth - 1 down to 0 do
        foreach parentNode in octree.nodesAtLevel(level) do
            children ← parentNode.children;
            q_parent ← W_q · parentNode.latentCode;
            k_children ← W_k · childrenLatentCodes;
            α ← Softmax(q_parent^T · k_children / √d_k);
            parentNode.attentionWeights ← α;
            parentNode.latentCode ← Σ_j (α[j] · children[j].latentCode);
    return octree;
```

Algorithm 3: Top-Down Refinement Pass

Input : OctTree from bottom-up pass

Output: OctTree with refined node representations

Function `TopDownPass(octree)`

  for `level` *from 0 to* `octree.depth` *- 1* **do**

    foreach `parentNode` *in* `octree.nodesAtLevel(level)` **do**

      if `level` *= 0* **then**

        $q_{\text{root}} \leftarrow W_q^{(\zeta)} z_{\text{root}}$;

        $k_{\text{root}} \leftarrow W_k^{(\zeta)} z_{\text{root}}$;

        $\text{attention}_{\text{root}} \leftarrow \text{Softmax}(q_{\text{root}} \cdot k_{\text{root}}^T / \sqrt{d_k})$;

        $\zeta_{\text{root}} \leftarrow \text{attention}_{\text{root}} \cdot z_{\text{root}}$;

        `parentNode.deltaZ` $\leftarrow$ `parentNode.latentCode` - ζ_{root} ;

      `children` $\leftarrow$ `parentNode.children` ;

      $\alpha_{\text{parent}} \leftarrow$ `parentNode.attentionWeights`;

      for *i from 0 to* `children.length` *- 1* **do**

        `children[i].deltaZ` $\leftarrow \alpha_{\text{parent}}[i] \cdot$ `parentNode.deltaZ` ;

        `children[i].latentCode` $\leftarrow$ `children[i].latentCode` + `children[i].deltaZ` ;

  return `octree`;

Attention Mechanism Design

Core Tree-based Attention Principles Tree-based structures offer a middle ground between dense global attention and strictly local operations. Message passing occurs both within levels (implicitly) and across levels (explicitly). The linearity of the attention mechanism is key. It allows messages (feature transformations or refinements) to be propagated across levels. This effectively covers node-node relationships spanning different parts of the tree. This recursive message passing can approximate full attention with significantly reduced complexity. For an OctTree of k levels with m children per node (typically $m = 8$), the total number of nodes is $N = O(m^k)$ in the worst case. Dense attention would require $O(N^2) = O(m^{2k})$ pairwise computations. Our approach processes each parent-children group independently, requiring $O(m^2)$ attention computations per parent node. With approximately N/m parent nodes across all levels, the

total complexity becomes $O(N \cdot m) = O(m^{k+1})$, which is significantly more tractable than the quadratic scaling of dense attention.

Multi-head Attention Support The proposed attention mechanism can be extended to multi-head attention. This enhances the model's capacity to capture diverse relationships from different perspectives. In this extension, each head h would have its own learnable projection matrices for query, key, and value. Specifically, for each head h, input features z are projected to $q^h = W_q^h z$, $k^h = W_k^h z$, and importantly, $v^h = W_v^h z$. The use of per-head value projections (W_v^h) is a standard component of multi-head attention. It allows each head to transform the input features into a value representation tailored to its specific focus. This is a generalization from the simplified description in the bottom-up pass (Section 3.4.3), where raw child features were used directly as values to illustrate a direct linear combination.

Attention is then computed in parallel for each head h, typically as:

$$\text{output}_i^h = \sum_j \alpha_{ij}^h v_j^h$$

where α_{ij}^h are the attention weights computed by head h. The outputs from all N_H heads, $\{\text{output}_i^h\}_{h=1}^{N_H}$, are usually concatenated and then linearly projected by a matrix W_O to produce the final combined output for the node:

$$\text{output}_i = W_O(\text{concat}(\text{output}_i^1, ..., \text{output}_i^{N_H}))$$

This process effectively combines information from different learned representation subspaces. It results in multiple distinct attention weight sets (one α^h per head for each parent-child group) and potentially allows for more complex and nuanced refined representations (ζ) during the top-down pass if multi-head is applied there too. This combines information from different representation subspaces.

3.4.4 Experimental Evaluation and Results

Our method is evaluated on the task of 3D shape reconstruction from sparse point clouds. We compare our approach against established baselines on a standard benchmark dataset. The evaluation focuses on assessing the quality of the reconstructed geometry, both quantitatively and qualitatively.

Experimental Setup

Dataset We conduct our experiments on the ScanNet v1 and v2 datasets [17]. ScanNet is a large-scale dataset of 3D scans of indoor scenes, providing a standard benchmark for reconstruction tasks. For each scene, we use an input point cloud of 16,384 points sampled from the ground truth surface.

Metrics To quantitatively evaluate reconstruction quality and efficiency, we employ several standard metrics:

- **Efficiency Metrics:** We measure the peak GPU Memory usage during training, the number of Training Iterations required to reach an F-Score of 0.7 on the validation set, and the per-scene Inference Time.
- **Geometric Metrics:** We compute Intersection over Union (IoU) on a 256^3 voxel grid, Chamfer-L_1 distance, Normal Consistency (the cosine similarity between surface normals of the reconstruction and ground truth), and F-Score (at a threshold of 5cm).

Implementation Details Our model is implemented in PyTorch and trained on a single NVIDIA RTX 3090 GPU. The OctTree structure is built up to a maximum depth of 8 for our model variations. The feature dimension d for the latent codes in our hierarchical attention model is set to 128. For our ablation studies, we evaluate models with the number of attention heads $N_H \in \{1, 2, 4, 8\}$ and the number of refinement iterations $N_{iter} \in \{1, 2, 4, 8\}$. Our main reported model for ablations uses $N_H = 4$ heads and $N_{iter} = 4$ iterations.

Attention Block Implementation Our hierarchical attention mechanism is built upon the multi-head attention block described above. For a model with latent feature dimension $d = 128$ and N_H heads, the dimension for each head is $d_h = d/N_H$. Each attention block, whether for bottom-up aggregation or top-down refinement, consists of three linear projection matrices: W_q, W_k, and W_O, all of dimension $d \times d$. The input features are projected to generate queries and keys. As noted in Section 3.4.3, the value vectors are the children's latent codes themselves, omitting a projection matrix W_v to maintain a direct combination of source features. After the scaled dot-product attention,

the outputs of all heads are concatenated and passed through the final projection W_O. The weights of the attention block are shared across all nodes within a single pass (i.e., one set of weights for the bottom-up pass, and another for the top-down pass). This weight sharing makes the model parameter-efficient, regardless of the OctTree's depth or complexity.

Comparison with Baselines

Baseline Method We compare our method primarily against **Convolutional Occupancy Networks (ConvONet)** [104]. ConvONet is a powerful and widely-used method for 3D reconstruction that leverages convolutional features on a 3D grid to predict implicit occupancy fields. We use the 3D grid-based version of ConvONet with a feature grid resolution of 64^3 as a strong baseline. A higher resolution version of ConvONet could not be trained on our hardware due to GPU memory constraints. This comparison serves to demonstrate the advantage of our approach in achieving competitive or superior reconstruction quality with greater resource efficiency.

Quantitative Results Table 3.6 presents the quantitative comparison of our method against the ConvONet baseline. The results highlight a crucial trade-off between computational efficiency and geometric accuracy.

Our model with a shallow OctTree (Depth 3) achieves geometric quality on par with ConvONet-Grid64^3. It produces a better F-Score and a lower Chamfer-L_1 distance with a comparable IoU, while consuming only about half the GPU memory (11.8G vs 21.7G). This demonstrates that our method can match the performance of a standard grid-based approach with significantly higher memory efficiency, albeit with slightly slower training convergence (260k vs 230k iterations).

When using a deeper OctTree (Depth 5), our model substantially outperforms the baseline across all geometric metrics, achieving an IoU of 0.817 (a 13% relative improvement) and an F-Score of 0.912 (an 11% relative improvement). This superior performance is achieved while still using less GPU memory than the 64^3 ConvONet baseline (19.2G vs 21.7G). However, this highlights an important trade-off between computational space and time. The improved accuracy and memory efficiency come at the cost of a significantly longer training period and a moderately increased inference

time.

The Depth 5 model requires more than double the training iterations of ConvONet (510k vs 230k) to reach the baseline F-Score, which suggests that our model is more challenging to optimize and converges more slowly. This slower convergence may stem from several factors. First, the iterative bottom-up and top-down refinement process creates a deep computational graph, making gradient propagation difficult, similar to training very deep recurrent networks. Second, the extensive weight sharing of the attention blocks, while parameter-efficient, requires the model to learn a single, highly generalized set of parameters that must function across diverse scales and geometries, which can complicate the optimization landscape.

Despite the longer training time, the memory savings are critical for scalability. The baseline ConvONet could not be trained at a higher resolution (e.g., 128^3) on our hardware due to its prohibitive memory requirements, which scale cubically. This underscores the key advantage of our OctTree-based attention mechanism: it provides a scalable path to higher resolutions and more detailed reconstructions that are infeasible for dense grid-based methods under typical hardware constraints. Our approach effectively focuses computation only where detail exists, creating a more favorable trade-off between final performance and resource consumption, particularly when memory is the primary bottleneck.

Qualitative Results Figure 3.15 shows a qualitative comparison of the reconstructions on two scenes from the ScanNet dataset. The visual results corroborate our quantitative findings. Our method, produces reconstructions that is qualitatively competitive to the baseline. This demonstrates the effectiveness of our hierarchical attention mechanism in building rich feature representations that translate to higher-quality geometry.

Ablation Studies

To validate the design choices of our proposed method, we conduct several ablation studies on both the shallow (Depth 3) and deep (Depth 5) model configurations.

Ablation on Core Mechanisms To validate the key components of our architecture, we perform an ablation study that builds up our model step-by-step. Table 3.7 shows the impact of the aggregation mechanism, the top-down propagation pass, and iterative

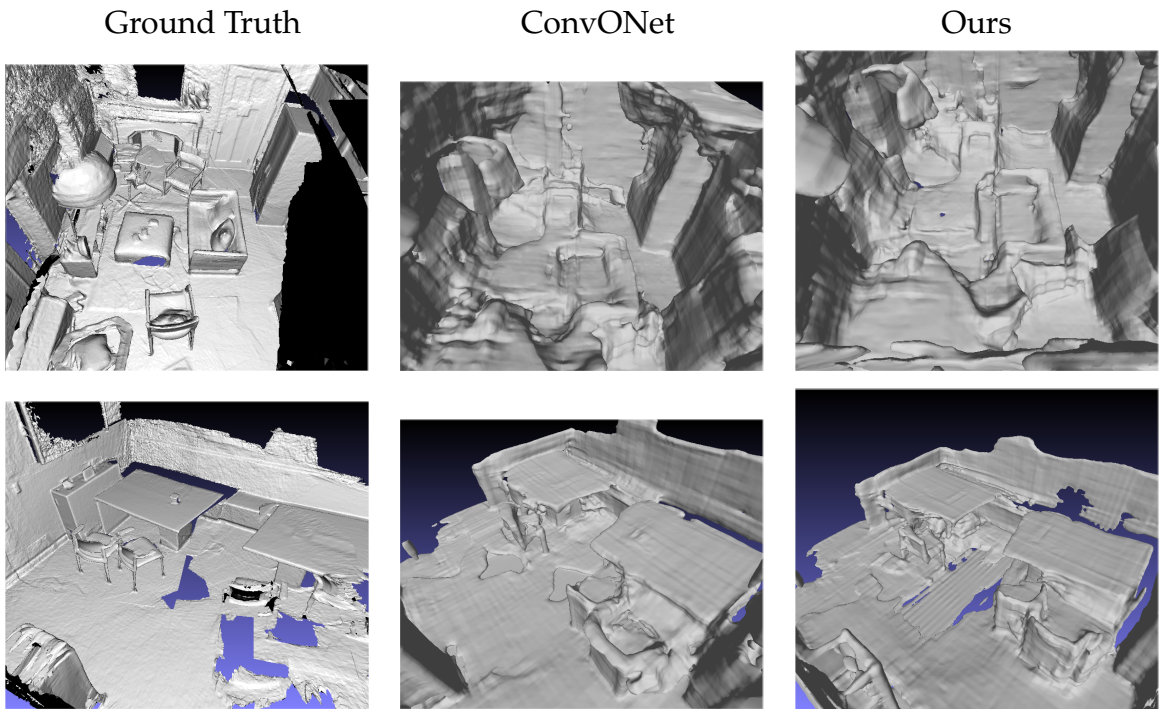


Figure 3.15: Qualitative comparison of 3D reconstructions on two ScanNet scenes (rows). Columns show the Ground Truth mesh, the result from the ConvONet baseline, and our method's reconstruction.

refinement. We start with two simplified baselines. The "No Attention" model replaces our attention-based aggregation with a simple mean pooling in the bottom-up pass. The "Attention (Bottom-up only)" model uses our attention mechanism for aggregation but omits the top-down propagation pass. This can be conceptualized as a model before any full refinement iterations are performed. The results clearly show that each component provides a significant performance boost. Moving from a simple pooling to our attention mechanism provides a notable gain (e.g., F-Score from 0.520 to 0.690 for Depth 5), demonstrating the benefit of learned, adaptive aggregation. Adding the top-down pass ("1 Iteration") further improves the score (to 0.724), confirming the value of propagating global context back down the tree. Finally, increasing to four iterations yields the largest performance leap (to 0.912), highlighting that repeated cross-scale information flow is critical for building high-quality feature representations.

Effect of Multi-Head Attention We also study the impact of the number of attention heads (N_H) on performance. Table 3.8 shows that across both depths, increasing the number of heads from 1 to 4 yields a consistent improvement. This suggests that multi-

ple heads help the model to jointly attend to information from different representation subspaces, as intended. Using 8 heads provides no further benefit and degrades performance in both configurations. This is likely because the feature dimension per head becomes too small to capture complex relationships effectively. Based on this, we use $N_H = 4$ for our final model configuration.

3.4.5 Further Enhancements and Considerations

Theoretical Considerations

Several theoretical aspects warrant further investigation:

Expressiveness Analysis The expressiveness of hierarchical attention compared to full dense attention needs rigorous analysis. While our method reduces computational complexity, it's important to characterize what types of spatial relationships can be captured through the tree-structured message passing versus direct pairwise attention.

Positional Encoding Requirements The choice of positional encoding (Fourier vs. learned vs. tree-structure-aware) significantly impacts the model's ability to distinguish spatial relationships. The current Fourier encoding may not fully capture the hierarchical nature of the OctTree structure.

Gradient Flow Properties The iterative nature of the algorithm raises questions about gradient flow during back-propagation. Deep iteration might lead to vanishing or exploding gradients, similar to issues encountered in very deep recurrent networks.

Predicting Extra Leaf Nodes

To handle incomplete or corrupted input data, a mechanism can predict and add leaf nodes. A small neural network operates on the refined latent codes of existing leaf nodes. It outputs a probability p for subdivision and eight vectors $\{\mathbf{v}_i\}_{i=1}^{8}$. If $p > 0.5$, the node is divided. The new children's latent codes $\mathbf{z}_i$ are derived from the parent's code $\mathbf{z}_{\text{parent}}$ and the predicted vectors: $\mathbf{z}_i = \mathbf{z}_{\text{parent}} \odot \mathbf{v}_i$. This adaptively refines the OctTree structure.

Integration with Vector-Neurons for SO(3) Equivariance

For rotation equivariance, integration with Vector-Neurons [18] is possible. Latent codes become sets of 3D vectors (e.g., $k/3$ vectors for a k-dim code). Standard dot products in attention are replaced with Vector-Neuron dot products. Linear transformations use Vector-Neuron equivariant layers. The refinement network for predicting nodes would also use Vector-Neurons.

Future Work and Broader Applications

Looking ahead, several avenues for future work are promising. The immediate next step is a more extensive empirical validation for the reconstruction task, benchmarking against a wider array of state-of-the-art implicit reconstruction methods beyond ConvONet. This will provide a more comprehensive understanding of the model's performance and limitations in this context.

A significant future direction is to leverage the general-purpose nature of the learned feature representations. As designed, the hierarchical attention mechanism is not task-specific. Applying the model as a feature-encoding backbone for tasks like 3D semantic segmentation and object detection would be a crucial step. Success in these areas would validate the approach not just as an effective component for implicit reconstruction, but as a foundational and versatile architecture for a wide range of 3D understanding problems.

3.4.6 Conclusion

This section detailed a hierarchical attention propagation mechanism for OctTrees. The method combines the structural efficiency of OctTrees with the expressive power of attention. While demonstrated on the task of implicit 3D reconstruction, where it achieves competitive results with high parameter efficiency, the architecture itself is designed to be a general-purpose feature learner. We have presented a framework for multi-scale feature learning and cross-scale information flow, discussed potential enhancements, and validated its effectiveness. The proposed approach stands as a promising direction for robust, efficient, and scalable 3D spatial data processing, with potential applications spanning from reconstruction to semantic understanding.

3.5 Chapter Summary

Where the previous chapter focused on methods producing explicit, parametric surface representations, such as the grid-based partitioning approach for generating Bézier patches (Section 3.3), this chapter delves into three distinct methodologies for enhancing **implicit** 3D reconstruction. By incorporating various forms of prior knowledge and structural biases, each approach addresses fundamental challenges in fidelity, computational efficiency, and scalability. These contributions offer unique solutions that advance the capabilities of neural implicit representations.

In Section 3.1, we introduced **Seed-Net**, a framework that integrates human intelligence directly into the reconstruction pipeline. By allowing users to interactively place "seeds" in regions requiring refinement, Seed-Net leverages explicit human priors to correct artifacts and recover fine-grained details that are often lost or over-smoothed by fully automated methods. The core mechanism uses an attention-based propagation scheme, enabling local corrections to inform and improve related structures throughout the entire scene. This human-in-the-loop approach demonstrates the value of combining machine-level generalization with human-level domain expertise, bridging the gap between automated reconstruction and manual artistic control.

Then we presented **NeuLap**, which shifts from explicit human guidance to an implicit, learned geometric prior. NeuLap introduces a novel training paradigm that uses the Laplacian of the implicit surface as a supervisory signal. A pre-trained denoising network refines the rendered Laplacian map, and the difference between the raw and denoised maps provides an auxiliary loss. This geometric prior encourages the network to learn smoother surfaces and preserve sharp features, significantly accelerating convergence and improving reconstruction quality, particularly under data-scarce conditions. This method showcases how priors can be effectively integrated into the optimization process itself, guiding the network towards more plausible geometric outcomes.

In Section 3.4, a more foundational architectural approach with the **Hierarchical Attention Propagation mechanism for OctTrees** was explored. Rather than injecting priors through interaction or loss functions, this method provides a powerful, multi-scale feature representation backbone. By employing iterative bottom-up aggregation and top-down refinement passes with a sparse attention mechanism, the model effi-

ciently captures both local geometric details and long-range contextual dependencies within the data. This architecture provides a scalable and parameter-efficient way to build rich feature hierarchies, forming a robust foundation upon which high-fidelity implicit reconstruction and other 3D understanding tasks can be built.

These three contributions illustrate a spectrum of strategies for improving 3D reconstruction. From the direct intervention of Seed-Net, to the learned geometric constraints of NeuLap, and finally to the inherent structural advantages of the hierarchical attention architecture, this chapter highlights the critical role of priors in pushing the boundaries of what is achievable with deep learning over spatial data. Each method offers a different trade-off between user effort, data requirements, and computational structure, providing a versatile toolkit for tackling diverse reconstruction challenges.

Model	IoU	Chamfer-L1	Normal	IoU*	Chamfer-L1*	Normal*
Baseline						
ConvONet-16^3	0.845	0.046	0.872	0.845	0.046	0.872
With seed-code length 16						
4 Seeds-16^3	0.841	0.046	0.840	0.853	0.044	0.875
8 Seeds-16^3	0.846	0.045	0.852	0.861	0.044	0.884
16 Seeds-16^3	0.864	0.044	0.887	0.868	0.043	0.886
32 Seeds-16^3	0.865	0.044	0.885	0.867	0.043	0.883
With seed-code length 32						
4 Seeds-16^3	0.851	0.045	0.877	0.872	0.044	0.887
8 Seeds-16^3	0.874	0.044	0.897	0.890	0.043	0.893
16 Seeds-16^3	0.883	0.043	0.924	**0.893**	**0.042**	0.936
32 Seeds-16^3	0.886	0.043	0.922	**0.893**	**0.042**	**0.937**
With seed-code length 64						
4 Seeds-16^3	0.872	0.044	0.884	0.891	**0.042**	**0.939**
8 Seeds-16^3	**0.894**	**0.042**	**0.940**	**0.896**	**0.042**	**0.939**
16 Seeds-16^3	**0.895**	**0.042**	**0.941**	**0.896**	**0.041**	**0.941**
32 Seeds-16^3	**0.895**	**0.042**	**0.941**	**0.896**	**0.042**	**0.941**
High resolution reference						
ConvONet-32^3	0.891	0.044	0.936	0.891	0.044	0.936

Table 3.1: Experiment results on integrating SeedNet into Convolutional Occupancy Network, with Uniform Seed Sampling and Heuristic Seed Sampling. The metrics column headings with * state for the results of heuristic seed sampling. The original baseline

Table 3.2: Quantitative comparison of 3D reconstruction quality on the ScanNet dataset. Metrics include Accuracy (Acc.), Completeness (Comp.), Precision (Prec.), Recall, and F-score. Acc./Comp. are measured in cm. Best results are highlighted.

Method	Acc.(↓)	Comp.(↓)	Prec.(↑)	Recall(↑)	F-score(↑)
COLMAP [120]	0.061	0.089	0.642	0.571	0.604
NeuralRecon [127]	0.041	0.091	0.753	0.575	0.652
NeRF [90]	0.162	0.066	0.375	0.579	0.455
NeuS [150]	0.107	0.122	0.450	0.380	0.412
NeuS+Lap	**0.093**	**0.102**	**0.521**	**0.440**	**0.477**
NeuRIS [146]	0.054	0.052	0.720	0.665	0.691
NeuRIS+Lap	**0.040**	**0.046**	**0.775**	**0.745**	**0.759**

Table 3.3: Reconstruction F-score Comparison (higher is better) on OOD (Synthesis-NeRF) and Lack-of-Data (50% ScanNet) Scenarios. Best results per model pair highlighted.

Model	Nerf-Synthetic (OOD)	50% ScanNet Data (Lack-of-Data)
NeuS	**0.551**	0.351
NeuS+NeuLap	0.513	**0.423**
NeuRIS	0.410	0.612
NeuRIS+NeuLap	**0.490**	**0.633**

Table 3.4: Ablation study of NeuLap components on the ScanNet dataset, using NeuRIS as the baseline. Performance is measured by mean F-score (higher is better).

Method	**F-score**
Baseline (NeuRIS)	0.691
Laplacian Regularization (L_1)	0.693
NeuLap (w/o Pose Info)	0.720
NeuLap (w/o Neighboring Frames)	0.745
Full NeuLap (on NeuRIS)	**0.759**

Table 3.5: Chamfer Distance ($\times 10^3$, lower is better) comparison on ShapeNet categories. Results are shown for surfaces derived from the inferred partitioned grid by fitting cubic Bèzier patches (see Section 3.3.2), compared to AtlasNet. OOD categories marked with *.

Method	pla.	ben.	cab.	car	cha.	mon.	lam.	spe.	fir.	cou.	tab.	mean	*cel.	*wat.
AtlasNet (25 patches)	0.97	1.33	1.66	1.94	1.58	1.12	1.75	2.01	1.01	1.25	1.21	1.44	1.42	1.33
AtlasNet (125 patches)	1.03	1.32	1.54	1.92	1.46	1.19	1.78	2.23	0.84	1.32	1.44	1.46	0.89	1.34
Grid-3x3x3	2.31	2.20	2.19	2.08	1.76	1.99	2.15	2.29	1.85	1.58	1.74	2.01	2.43	2.21
Grid-5x5x5	2.10	2.05	2.00	1.95	1.65	1.85	2.00	2.10	1.70	1.50	1.65	1.81	1.92	1.81
Grid-7x7x7	1.50	1.45	1.25	1.75	1.25	1.30	1.58	1.95	1.20	1.20	1.15	1.42	1.87	1.98

Table 3.6: Quantitative comparison for 3D shape reconstruction on the ScanNet dataset. Our method is compared against a strong grid-based baseline. We report efficiency metrics (GPU Memory, Training Iterations, Inference Time) and geometric quality metrics. ↑ indicates higher is better, ↓ indicates lower is better. Best results on geometric metrics are in bold.

Method	GPU Mem (G) ↓	Train Iters (K) ↓	Infer Time (s) ↓	IoU ↑	Chamfer-L_1 ↓	Normal Cons. ↑	F-Score ↑
ConvONet-Grid64^3 [104]	21.7	**230**	**8.16**	0.722	0.075	0.877	0.820
Ours (Depth 3)	11.8	260	9.13	0.721	**0.069**	0.804	**0.832**
Ours (Depth 5)	19.2	510	12.2	**0.817**	**0.042**	**0.903**	**0.912**

Table 3.7: Ablation study on the core mechanisms of our method. We report F-Score on the ScanNet validation set with a fixed 4 attention heads. The "No Attention" uses mean pooling in the bottom-up pass instead of our attention mechanism. The 0 iter performs the bottom-up pass only, without the top-down propagation pass, and n-iters performs n iterations of the bottom-up and top-down passes.

Method	F-Score (Depth 3) ↑	F-Score (Depth 5) ↑
No Attention	0.450	0.520
0 iter	0.510	0.690
1 iter	0.542	0.724
2 iter	0.600	0.780
4 iter (Ours)	**0.832**	**0.912**
8 iter	0.831	0.821

Table 3.8: Ablation study on the number of attention heads (N_H). We report F-Score on the ScanNet validation set with a fixed 4 iterations.

N_H	F-Score (Depth 3) ↑	F-Score (Depth 5) ↑
1	0.821	0.901
2	0.829	0.908
4 (Ours)	**0.832**	**0.912**
8	0.817	0.889

CHAPTER 4

Conclusion and Future Directions

This chapter summarizes the key contributions and findings of the thesis. It provides a brief overview of the research objectives, the proposed solutions, and the results obtained. The research journey began with an exploration of explicit, parametric reconstruction methods, encountered fundamental challenges that led to a strategic pivot towards implicit representations, and culminated in the development of a novel, general-purpose backbone for 3D spatial learning.

4.1 Summary of Contributions and Findings

This thesis presents a trajectory of research that navigates the complex landscape of 3D reconstruction. Our investigation commenced with explicit, parametric methods and progressively transitioned towards more robust and scalable implicit and general-purpose learning frameworks.

The initial exploration, proposed in Section 3.3, focused on **explicit, parametric surface-based reconstruction**. We investigated methods to generate partitioned deforming boxes directly from point clouds, aiming for structured, CAD-like outputs. However, this path revealed obstacles in robustly inferring topology and connectivity from unstructured data. Upon this critical finding, we shifted our focus to implicit representations.

The research then pivoted to **prior-guided implicit 3D reconstruction**, as presented in Chapter 3. This chapter introduced two novel methods that leverage different forms

of prior knowledge. **Seed-Net** introduced a human-in-the-loop pipeline, enabling user-controllable refinement through interactive "seed" points. This work demonstrated how explicit, sparse user feedback can be differentiably integrated to guide the optimization of a neural field, achieving significant improvements in local detail. **NeuLap** incorporated a learned geometric prior by using the surface Laplacian as a supervisory signal during training. By penalizing deviations from a denoised Laplacian map, NeuLap accelerates convergence and enhances the preservation of sharp geometric features, improving reconstruction quality from limited data. Finally, In Section 3.4 of Chapter 3, we addressed the need for a more foundational and scalable architecture for 3D deep learning. We introduced a **Hierarchical Attention Propagation mechanism for OctTrees**, a general-purpose backbone for learning over large-scale spatial data. By combining bottom-up feature aggregation and top-down information propagation with a sparse attention mechanism, this architecture efficiently captures both local details and global context. Its strong performance and memory efficiency provide a powerful foundation for a wide range of 3D tasks, including high-fidelity reconstruction.

4.2 Revisiting Research Objectives

The contributions of this thesis directly address the research objectives set out in Chapter 1.

1. **To Enhance 3D Parametric Reconstruction from Noisy Inputs**: This objective was the focus of our initial research (Section 3.3 in Chapter 3). While a robust, end-to-end solution was not achieved, the investigation partially fulfilled this objective by identifying the theoretical and practical barriers of explicit parametric methods, thereby providing more insights that guided the subsequent, more successful research directions.

2. **To Enable User-Controllable Refinement in Implicit 3D Reconstruction**: This objective was met by the development of **Seed-Net** (Section 3.1 in Chapter 3). Its interactive refinement mechanism provides a direct and intuitive way for users to guide the reconstruction process, effectively integrating human intelligence to enhance geometric precision where it is most needed.

3. **To Improve the Computational Efficiency of Implicit Neural Representations**: **NeuLap** (Chapter 3) has addressed this objective. By integrating a geometric prior, it accelerates the training of neural fields, leading to faster convergence and state-of-the-art results without requiring architectural changes to the baseline implicit model.

4. **To Propose a General-Purpose 3D Learning Framework that is Scalable and Adaptive for Complex Scenes**: The **Hierarchical Attention OctTree** (Chapter 3) satisfies this objective. It provides a scalable and efficient backbone that excels at processing large-scale 3D data, outperforming strong baselines and demonstrating its capability as a foundational architecture for various 3D understanding tasks.

4.3 Limitations and Future Directions

While this thesis advances the field of 3D spatial learning, it also opens up new avenues for future research. We identify the following limitations and corresponding directions for future work:

- **End-to-End Differentiable Parametric Reconstruction**: The challenges encountered in differentiable topological inference in explicit reconstruction remain a significant open problem. Future work could explore novel representations or learning paradigms, such as reinforcement learning or graph neural networks, to tackle the challenge of joint topology and geometry optimization in an end-to-end differentiable manner.

- **Generality of Learned Priors**: The geometric prior in NeuLap is learned from a dataset of synthetic scenes, which may not generalize perfectly to all types of shapes or artifacts. Future work could explore online or self-supervised methods for learning priors directly from the input data, potentially leading to more adaptive and robust guidance signals.

- **Broader Applications of the Attention-in-OctTree Backbone**: The Hierarchical Attention OctTree was primarily evaluated on reconstruction tasks. Its powerful multi-scale feature learning capabilities make it a strong candidate for a wide array of other 3D tasks, such as semantic and instance segmentation, object detection,

and scene flow estimation. Systematically adapting and evaluating the backbone for these applications is a promising research direction.

4.4 Concluding Remarks

The research presented in this thesis has navigated the evolving landscape of deep learning for 3D reconstruction. From the structured ambition of parametric modeling to the flexible power of implicit fields and the scalable architecture of attention-based OctTrees, this work has contributed novel methods, critical insights, and a powerful general-purpose framework. The journey underscores a key lesson: that progress in 3D spatial learning requires not only algorithmic innovation but also a deep understanding of the fundamental trade-offs between explicit structure and implicit flexibility. The contributions herein provide both practical tools and conceptual frameworks that will, we hope, inspire future explorations in the quest to teach machines to see and understand our three-dimensional world.

Bibliography

[1] Abdul-Mohssen J Abdul-Hossen and Baghdad Email. 3D Surface Reconstruction of Mathematical Modelling Used for Controlling the Generation of Different Bicubic B-Spline in Matrix Form Without Changing the Control Points. *Eng. \&Tech. Journal*, 34(1):136–152, 2016.

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *35th International Conference on Machine Learning, ICML 2018*, 1:67–85, 2018.

[3] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Bjorn Ottersten. A survey on Deep Learning Advances on Different 3D Data Representations. 1(1):1–35, aug 2018.

[4] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. *arXiv:2103.13415 [cs]*, August 2021. arXiv: 2103.13415.

[5] Fausto Bernardini, Chandrajit L. Bajaj, Jindong Chen, and Daniel R. Schikore. Automatic reconstruction of 3D CAD models from digital scans. *International Journal of Computational Geometry and Applications*, 9(4-5):327–369, 1999.

[6] Wenjing Bian, Zirui Wang, Kejie Li, and Victor Adrian Prisacariu. Ray-ONet: Efficient 3D Reconstruction From A Single RGB Image. July 2021. arXiv: 2107.01899.

[7] Francesco Buonamici, Monica Carfagni, Rocco Furferi, Lapo Governi, and Yary Volpe. *CAD Reconstruction: A Study on Reverse Modelling Strategies*. Number January. Springer International Publishing, 2020.

[8] Zixuan Cang and Guo-Wei Wei. Topologynet: Topology based deep convolutional and multi-task neural networks for biomolecular property predictions. *PLOS Computational Biology*, 13(7):e1005690, July 2017.

[9] Angel X Chang, Thomas A Funkhouser, Leonidas J Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *CoRR*, abs/1512.0, 2015.

[10] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields, March 2022. arXiv:2203.09517 [cs].

[11] Siheng Chen, Chaojing Duan, Yaoqing Yang, Duanshun Li, Chen Feng, and Dong Tian. Deep Unsupervised Learning of 3D Point Clouds via Graph Topology Inference and Filtering. *IEEE Transactions on Image Processing*, 29:3183–3198, 2020.

[12] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural Dual Contouring. *arXiv:2202.01999 [cs]*, February 2022. arXiv: 2202.01999.

[13] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning, 2020.

[14] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5932–5941, 2019.

[15] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. *CoRR*, abs/1904.08755, 2019.

[16] Christopher B. Choy, Danfei Xu, Jun Young Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9912 LNCS:628–644, 2016.

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

[18] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas Guibas. Vector Neurons: A General Framework for SO(3)-Equivariant Networks. April 2021. arXiv: 2104.12229.

[19] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer Views and Faster Training for Free, April 2022. arXiv:2107.02791 [cs].

[20] Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J Guibas. Curriculum DeepSDF. *European Conference on Computer Vision*, pages 51–67, March 2020. arXiv: 2003.08593 Publisher: Springer.

[21] Noah Duncan and Sai-kit Yeung. Interchangeable Components for Hands-On Assembly Based Modelling. 2016.

[22] Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3D object reconstruction from a single image. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:2463–2471, 2017.

[23] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. MeshNet: Mesh neural network for 3D shape representation. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 8279–8286, 2019.

[24] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Muller. SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[25] Akemi Gálvez and Andrés Iglesias. Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points. *Information Sciences*, 192:174–192, 2012.

[26] Hongyang Gao and Shuiwang Ji. Graph U-nets. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:3651–3660, 2019.

[27] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning Deformable Tetrahedral Meshes for 3D Reconstruction, November 2020. arXiv:2011.01437 [cs].

[28] Kyle Gao, Yina Gao, Hongjie He, Denning Lu, Linlin Xu, and Jonathan Li. NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review, November 2022. arXiv:2210.00379 [cs].

[29] Xiang Gao, Wei Hu, and Guo-Jun Qi. GraphTER: Unsupervised Learning of Graph Transformation Equivariant Representations via Auto-Encoding Node-Wise Transformations. pages 7161–7170, 2020.

[30] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-Fidelity Neural Rendering at 200FPS, April 2021. arXiv:2103.10380 [cs].

[31] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local Deep Implicit Functions for 3D Shape. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4856–4865, December 2019. arXiv: 1912.06126 Publisher: IEEE Computer Society ISBN: 978-1-7281-7168-5.

[32] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. jun 2019.

[33] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018.

[34] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, February 2020. arXiv: 2002.10099.

[35] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. AtlasNet: A Papier-M\^ach\'e Approach to Learning 3D Surface Generation. February 2018. arXiv: 1802.05384.

[36] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. PCPNet learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85, 2018.

[37] Haoyu Guo, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. Neural 3D Scene Reconstruction with the Manhattan-world Assumption, May 2022. arXiv:2205.02836 [cs].

[38] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, June 2021. arXiv: 2012.09688.

[39] Timo Hackel, Nikolay Savinov, Lubor Ladicky, Jan Dirk Wegner, Konrad Schindler, and Marc Pollefeys. Semantic3d.net: A new large-scale point cloud classification benchmark. *CoRR*, abs/1704.03847, 2017.

[40] Christian Hane, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3D object reconstruction. *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, pages 412–420, 2018.

[41] Rana Hanocka. MeshCNN: A network with an edge. *ACM Transactions on Graphics*, 38(4), 2019.

[42] Kaveh Hassani and Mike Haley. Unsupervised Multi-Task Feature Learning on Point Clouds. pages 8160–8171, oct 2019.

[43] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total denoising: Unsupervised learning of 3d point cloud cleaning, 2019.

[44] Juan Hernández, Klemen Istenič, Nuno Gracias, Narcís Palomeras, Ricard Campos, Eduard Vidal, Rafael García, and Marc Carreras. Autonomous Underwater Navigation and Optical Mapping in Unknown Natural Environments. *Sensors*, 16(8):1174, jul 2016.

[45] Binh Son Hua, Minh Khoi Tran, and Sai Kit Yeung. Pointwise Convolutional Neural Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[46] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural Kernel Surface Reconstruction, June 2023. arXiv:2305.19590 [cs].

[47] Andrés Iglesias, Gonzalo Echevarría, and Akemi Gálvez. Functional networks for B-spline surface reconstruction. *Future Generation Computer Systems*, 20(8):1337–1353, 2004.

[48] Eldar Insafutdinov, Dylan Campbell, João F. Henriques, and Andrea Vedaldi. SNeS: Learning Probably Symmetric Neural Surfaces from Incomplete Data, June 2022. arXiv:2206.06340 [cs].

[49] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. jun 2015.

[50] Hanqi Jiang, Cheng Zeng, Runnan Chen, Shuai Liang, Yinhe Han, Yichao Gao, and Conglin Wang. Depth-NeuS: Neural Implicit Surfaces Learning for Multi-view Reconstruction Based on Depth Information Optimization, March 2023. arXiv:2303.17088 [cs] version: 1.

[51] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. pages 4866–4875, 2020.

[52] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002.

[53] Kacper Kania, Maciej Zięba, and Tomasz Kajdanowicz. UCSG-Net – Unsupervised Discovering of Constructive Solid Geometry Tree. *Advances in Neural Information Processing Systems*, 2020-Decem, June 2020. arXiv: 2006.09102 Publisher: Neural information processing systems foundation.

[54] Taro Kawasaki, Pradeep Kumar Jayaraman, Kentaro Shida, Jianmin Zheng, and Takashi Maekawa. An image processing approach to feature-preserving B-spline surface fairing. *CAD Computer Aided Design*, 99:1–10, 2018.

[55] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, page 61–70, Goslar, DEU, 2006. Eurographics Association.

[56] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[57] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering, 2023.

[58] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. (2):1–3, 2016.

[59] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–14, 2017.

[60] Lukas Koestler, Daniel Grittner, Michael Moeller, Daniel Cremers, and Zorah Lähner. Intrinsic Neural Fields: Learning Functions on Manifolds, March 2022. arXiv:2203.07967 [cs].

[61] Nilesh Kulkarni, Justin Johnson, and David F. Fouhey. What's Behind the Couch? Directed Ray Distance Functions (DRDF) for 3D Scene Reconstruction, April 2022. arXiv:2112.04481 [cs].

[62] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation, May 2022. arXiv:2205.04334 [cs].

[63] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular Primitives for High-Performance Differentiable Rendering, November 2020. arXiv:2011.03277 [cs].

[64] Samuli Laine and Tero Karras. Efficient sparse voxel octrees. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, page 55–63, New York, NY, USA, 2010. Association for Computing Machinery.

[65] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S. Davis. Modeling local geometric structure of 3D point clouds using geo-cnn. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:998–1008, 2019.

[66] Nallig Leal, Esmeide Leal, and John William Branch. Simple Method for Constructing NURBS Surfaces from Unorganized Points A-PDF Split DEMO : Purchase from www.A-PDF.com to remove the watermark. 2010.

[67] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, October 2019. arXiv: 1810.00825.

[68] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. SO-Net: Self-Organizing Network for Point Cloud Analysis. mar 2018.

[69] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. MINE: Towards Continuous Depth MPI with NeRF for Novel View Synthesis, July 2021. arXiv:2103.14910 [cs].

[70] Manyi Li and Hao Zhang. D^{2}IM-Net: Learning Detail Disentangled Implicit Fields from Single Images. December 2020. arXiv: 2012.06650.

[71] 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*, 2018-Decem:820–830, 2018.

[72] Zuoyue Li, Tianxing Fan, Zhenqiang Li, Zhaopeng Cui, Yoichi Sato, Marc Pollefeys, and Martin R. Oswald. CompNVS: Novel View Synthesis with Scene Completion, July 2022. arXiv:2207.11467 [cs].

[73] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep Marching Cubes: Learning Explicit Surface Representations. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.

[74] Khang Jie Liew, Ahmad Ramli, Nur Nadiah Abd Hamid, and Ahmad Abd Majid. Sharp edge preservation using bicubic B-spline surfaces. *ScienceAsia*, 43:20–26, 2017.

[75] Stefan Lionar, Daniil Emtsev, Dusan Svilarkovic, and Songyou Peng. Dynamic Plane Convolutional Occupancy Networks. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1828–1837. IEEE, January 2021. arXiv: 2011.05813.

[76] Gidi Littwin and Lior Wolf. Deep Meta Functionals for Shape Representation. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:1824–1833, August 2019. arXiv: 1908.06277 Publisher: Institute of Electrical and Electronics Engineers Inc.

[77] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3D Modeling with a Generative Adversarial Network. jun 2017.

[78] 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. pages 1788–1797, 2021.

[79] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:8887–8896, 2019.

[80] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A Closer Look at Local Aggregation Operators in Point Cloud Analysis. 2020.

[81] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. SparseNeuS: Fast Generalizable Neural Surface Reconstruction from Sparse Views, August 2022. arXiv:2206.05737 [cs].

[82] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[83] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, page 4563–4572. IEEE, October 2021.

[84] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces, May 2021. arXiv:2011.13495 [cs].

[85] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. Technical report, 2015.

[86] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. Association for Computing Machinery.

[87] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4455–4465, 2019.

[88] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep Level Sets: Implicit Surface Representations for 3D Shape Inference. January 2019. arXiv: 1901.06802.

[89] Mateusz Michalkiewicz, Jhony Kaesemodel Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:4742–4751, 2019.

[90] 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 *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12346 LNCS, pages 405–421. Springer Science and Business Media Deutschland GmbH, March 2020. arXiv: 2003.08934 ISSN: 16113349.

[91] Eric Mitchell, Selim Engin, Volkan Isler, and Daniel D Lee. Higher-Order Function Networks for Learning Composable 3D Object Representations. July 2019. arXiv: 1907.10388.

[92] Tom M Mitchell. The Need for Biases in Learning Generalizations, 1980.

[93] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[94] Norman Müller, Andrea Simonelli, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. AutoRF: Learning 3D Object Radiance Fields from Single View Observations, April 2022. arXiv:2204.03593 [cs].

[95] Richard A Newcombe, Andrew Fitzgibbon, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, and Steve Hodges. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, oct 2011.

[96] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing Neural Radiance Fields for View Synthesis from Sparse Inputs, December 2021. arXiv:2112.00724 [cs].

[97] Alireza Norouzzadeh Ravari and Hamid D. Taghirad. Reconstruction of B-spline curves and surfaces by adaptive group testing. *CAD Computer Aided Design*, 74:32–44, 2016.

[98] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation, March 2022. arXiv:2112.11427 [cs].

[99] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:165–174, 2019.

[100] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, March 2021. arXiv: 2103.10429.

[101] Giuseppe Patané, 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. 2022.

[102] Anasol Peña-Rios, Hani Hagras, Michael Gardner, and Gilbert Owusu. A type-2 fuzzy logic based system for augmented reality visualisation of georeferenced data. *IEEE International Conference on Fuzzy Systems*, 2018-July, 2018.

[103] Anasol Pena-Rios, Hani Hagras, Gilbert Owusu, and Michael Gardner. Furthering Service 4.0: Harnessing Intelligent Immersive Environments and Systems. *IEEE Systems, Man, and Cybernetics Magazine*, 4(1):20–31, 2018.

[104] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer vision – ECCV 2020*, pages 523–540, Cham, March 2020. Springer International Publishing.

[105] L. Piegl. On NURBS: a survey. *IEEE Computer Graphics and Applications*, 11(1):55–71, jan 1991.

[106] Joshua Podolak and Szymon Rusinkiewicz. Atomic volumes for mesh completion. In *Symposium on Geometry Processing*, July 2005.

[107] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion, September 2022. arXiv:2209.14988 [cs, stat].

[108] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. dec 2016.

[109] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. jun 2017.

[110] Hongxing Qin, Jia Han, Ning Li, Hui Huang, and Baoquan Chen. Mass-Driven Topology-Aware Curve Skeleton Extraction from Incomplete Point Clouds. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2805–2817, 2020.

[111] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39, pages 185–203. Wiley Online Library, 2020.

[112] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. *arXiv:2103.13744 [cs]*, August 2021. arXiv: 2103.13744.

[113] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. ShaRF: Shape-conditioned Radiance Fields from a Single View, June 2021. arXiv:2102.08860 [cs].

[114] Edoardo Remelli, Artem Lukoianov, Stephan R Richter, Benoît Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. MeshSDF: Differentiable Iso-Surface Extraction. *arXiv preprint arXiv:2006.03997*, June 2020. arXiv: 2006.03997.

[115] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctNetFusion: Learning Depth Fusion from Data. *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, pages 57–66, 2018.

[116] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:6620–6629, 2017.

[117] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV) 2021*, 2021.

[118] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *Advances in Neural Information Processing Systems*, 32(NeurIPS), 2019.

[119] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10843 LNCS(1):593–607, 2018.

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

[121] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural Shape Parser for Constructive Solid Geometry. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018.

[122] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis, 2021.

[123] Vincent Sitzmann, Eric R Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning Signed Distance Functions. *arXiv preprint arXiv:2006.09662*, June 2020. arXiv: 2006.09662 ISBN: 2006.09662v1.

[124] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. *Advances in Neural Information Processing Systems*, 2020-Decem, June 2020. arXiv: 2006.09661 Publisher: Neural information processing systems foundation.

[125] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable Patch-Based Neural Rendering, July 2022. arXiv:2207.10662 [cs].

[126] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. *arXiv:2111.11215 [cs]*, November 2021. arXiv: 2111.11215.

[127] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *CVPR*, 2021.

[128] Kenshi Takayama, Ryan Schmidt, Karan Singh, Takeo Igarashi, Tamy Boubekeur, and Olga Sorkine. Geobrush: Interactive mesh geometry cloning. *Computer Graphics Forum (proceedings of EUROGRAPHICS)*, 30(2):613–622, 2011.

[129] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. *arXiv:2101.10994 [cs]*, January 2021. arXiv: 2101.10994.

[130] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable Large Scene Neural View Synthesis. *arXiv:2202.05263 [cs]*, February 2022. arXiv: 2202.05263.

[131] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned Initializations for Optimizing Coordinate-Based Neural Representations, March 2021. arXiv:2012.02189 [cs].

[132] 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, June 2020. arXiv:2006.10739 [cs].

[133] Jia-Heng Tang, Weikai Chen, Jie Yang, Bo Wang, Songrun Liu, Bo Yang, and Lin Gao. OctField: Hierarchical Implicit Functions for 3D Modeling. *arXiv:2111.01067 [cs]*, November 2021. arXiv: 2111.01067.

[134] Jiapeng Tang, Xiaoguang Han, Junyi Pan, Kui Jia, and Xin Tong. A skeleton-bridged deep learning approach for generating meshes of complex topologies from single rgb images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4536–4545, 2019.

[135] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. Sign-Agnostic CONet: Learning Implicit Surface Reconstructions by Sign-Agnostic Optimization of Convolutional Occupancy Networks. *arXiv preprint arXiv:2105.03582*, May 2021. arXiv: 2105.03582.

[136] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2107–2115, 2017.

[137] Anju Tewari, Otto Fried, Justus Thies, Vincent Sitzmann, S. Lombardi, Z Xu, Tanaba Simon, Matthias Nießner, Edgar Tretschk, L. Liu, Ben Mildenhall, Pranatharthi Srinivasan, R. Pandey, Sergio Orts-Escolano, S. Fanello, M. Guang Guo, Gordon Wetzstein, J-y Zhu, Christian Theobalt, Manju Agrawala, Donald B. Goldman, and Michael Zollhöfer. Advances in neural rendering. *Computer Graphics Forum*, 41, 2021.

[138] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Niessner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhoefer, and Vladislav Golyanik. Advances in Neural Rendering, March 2022. arXiv:2111.05849 [cs].

[139] Hugues Thomas, Charles R. Qi, Jean Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas Guibas. KPConv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:6410–6419, 2019.

[140] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12361 LNCS:293–309, August 2020. arXiv: 2008.01639 Publisher: Springer Science and Business Media Deutschland GmbH.

[141] Aggeliki Tsoli and Antonis A Argyros. Patch-Based Reconstruction of a Textureless Deformable 3D Surface from a Single RGB Image. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4034–4043. IEEE, October 2019.

[142] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. jun 2017.

[143] Petar Veličković, Arantxa Casanova, Pietro Liò, Guillem Cucurull, Adriana Romero, and Yoshua Bengio. Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–12, 2018.

[144] Petar Veličković, William Fedus, William L. Hamilton, Yoshua Bengio, Pietro Liò, and R. Devon Hjelm. Deep graph infomax. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–17, 2019.

[145] Hongwei Wang, Jialin Wang, Jia Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Wenjie Li, Xing Xie, and Minyi Guo. GraphGAN. *IEEE Transactions on Knowledge and Data Engineering*, pages 2508–2515, 2019.

[146] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. NeuRIS: Neural Reconstruction of Indoor Scenes Using Normal Priors. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 139–155, Berlin, Heidelberg, October 2022. Springer-Verlag.

[147] Jun Wang, Zhouwang Yang, Liangbing Jin, Jiansong Deng, and Falai Chen. Parallel and adaptive surface reconstruction based on implicit PHT-splines. *Computer Aided Geometric Design*, 28(8):463–474, 2011.

[148] Nanyang Wang, Yinda Zhang, and Zhuwen Li. Pixel2Mesh - Generating Meshes from Single RGB Images. *Eccv*, pages 52–67, 2018.

[149] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. *Lecture*

Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11215 LNCS:55–71, 2018.

[150] 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, February 2023. arXiv:2106.10689 [cs].

[151] Peng-Shuai Wang. Octformer: Octree-based transformers for 3D point clouds. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4), 2023.

[152] 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 (SIGGRAPH)*, 36(4), 2017.

[153] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. IBRNet: Learning Multi-View Image-Based Rendering, April 2021. arXiv:2102.13090 [cs].

[154] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. PIE-NET: Parametric Inference of Point Cloud Edges. (NeurIPS):1–12, jul 2020.

[155] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction, November 2023. arXiv:2212.05231 [cs].

[156] 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*, 38(5), 2019.

[157] Zixiong Wang, Pengfei Wang, Qiujie Dong, Junjie Gao, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. Neural-IMLS: Learning Implicit Moving Least-Squares for Surface Reconstruction from Unoriented Point Clouds. *arXiv:2109.04398 [cs]*, November 2021. arXiv: 2109.04398.

[158] Ziyun Wang, Volkan Isler, and Daniel D. Lee. Surface HOF: Surface Reconstruction from a Single Image Using Higher Order Function Networks. *Proceedings -*

International Conference on Image Processing, ICIP, 2020-Octob:2666–2670, December 2019. arXiv: 1912.08852 Publisher: IEEE Computer Society.

[159] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural Fields as Learnable Kernels for 3D Reconstruction, November 2021. arXiv:2111.13674 [cs].

[160] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10122–10131, 2019.

[161] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. nov 2018.

[162] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1912–1920, 2015.

[163] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, XX(Xx):1–21, 2020.

[164] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. CityNeRF: Building NeRF at City Scale. *arXiv:2112.05504 [cs]*, December 2021. arXiv: 2112.05504.

[165] 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, April 2022. arXiv:2111.11426 [cs].

[166] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based Neural Radiance Fields, March 2022. arXiv:2201.08845 [cs].

[167] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[168] Guangming Yao, Hongzhi Wu, Yi Yuan, Lincheng Li, Kun Zhou, and Xin Yu. Learning Implicit Body Representations from Double Diffusion Based Neural Radiance Fields, January 2022. arXiv:2112.12390 [cs].

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

[170] Ge Yin, Xiao Xiao, and Fehmi Cirak. Topologically robust CAD model generation for structural optimisation. *Computer Methods in Applied Mechanics and Engineering*, 369:1–26, 2020.

[171] Rex Ying, Christopher Morris, William L. Hamilton, Jiaxuan You, Xiang Ren, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems*, 2018-Decem:4800–4810, 2018.

[172] Hiroki Yoshihara, Tatsuya Yoshii, Tadahiro Shibutani, and Takashi Maekawa. Topologically robust B-spline surface reconstruction from point clouds using level set methods and iterative geometric fitting algorithms. *Computer Aided Geometric Design*, 29(7):422–434, 2012.

[173] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. *arXiv:2112.05131 [cs]*, December 2021. arXiv: 2112.05131.

[174] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for Real-time Rendering of Neural Radiance Fields. *arXiv:2103.14024 [cs]*, August 2021. arXiv: 2103.14024.

[175] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, pages 4578–4587, December 2021. arXiv: 2012.02190.

[176] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4), July 2011.

[177] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *ECCV*, 2018.

[178] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network, 2019.

[179] Xiuyang Zhao, Caiming Zhang, Bo Yang, and Pingping Li. Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation. *CAD Computer Aided Design*, 43(6):598–604, 2011.

[180] Wenni Zheng, Pengbo Bo, Yang Liu, and Wenping Wang. Fast B-spline curve fitting by L-BFGS. *Computer Aided Geometric Design*, 29(7):448–462, 2012.

[181] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021.

[182] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. pages 1–22, 2018.