University of Cyprus
Department of Computer Science

SITUATION-AWARE MULTI-MODAL UNDERSTANDING IN 3D SCENES

Panayiotis Demetriou

Thesis Dissertation

Department of Computer Science
May 2026

UNIVERSITY OF CYPRUS

Faculty of Pure and Applied Sciences

Department of Computer Science

SITUATION-AWARE MULTI-MODAL UNDERSTANDING IN 3D SCENES

Panayiotis Demetriou

Supervisor:

Prof. Yiorgos Chrysanthous

Co-Supervisors:

Dr. Melinos Averkiou

Dr. Marios Loizou

The Thesis was submitted in partial fulfillment of the requirements for obtaining the Computer Science degree of the Department of Computer Science of the University of Cyprus

May 2026

Acknowledgements

I would like to express my sincere gratitude to the CYENS Center of Excellence for giving me the opportunity to work with them during the course of this thesis. I am especially grateful for their support and for providing access to the GPU clusters, which were essential for running the experiments conducted as part of this work.

I would also like to thank my supervisors for their continuous guidance, support, and valuable feedback throughout the development of this thesis. I am particularly grateful to them for proposing this subject and for helping me shape and refine the work at each stage.

Declaration of Originality

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research, and with full accountability on my part for the accuracy, integrity, and validity of all content.

ABSTRACT

Understanding and reasoning about 3D scenes is a fundamental task with applications across multiple domains, including robotics, autonomous systems, augmented reality, embodied artificial intelligence, and human-computer interaction. In contrast to conventional visual understanding, 3D scene understanding requires the interpretation of spatial structure, geometric relationships, object semantics, and scene-level context within a physically grounded environment. Achieving comprehensive scene understanding therefore depends not only on recognizing individual objects, but also on modeling their spatial configurations, contextual associations, and semantic relationships. This makes the task particularly challenging, as effective reasoning in 3D environments requires the integration of perception, geometry, and textual interpretation to support accurate responses to queries involving both spatial awareness and contextual understanding.

Recent advances in 3D scene understanding increasingly rely on multimodal representations that combine textual information with visual and geometric features extracted from scene objects. Such approaches aim to bridge the gap between natural language and structured 3D representations, enabling models to align linguistic queries with spatial and semantic properties of the environment. One such framework is MSR3D, which serves as the baseline architecture for this work and provides a foundation for multimodal reasoning over 3D scenes. In parallel, recent developments in transformer architectures for point cloud processing have shown strong potential for capturing long-range dependencies, contextual interactions, and spatial relationships within complex 3D environments. Building on these developments, this work investigates the integration of transformer architecture for point cloud representation to enhance multimodal 3D scene understanding beyond the baseline MSR3D framework.

Motivated by these developments, this thesis investigates the effectiveness of Point Transformer V3 as a 3D backbone for multimodal scene understanding tasks. Specifically, the study evaluates the integration of PTv3 into the MSR3D framework by replacing the original PointNet++ backbone. Therefore,this work explores how different feature extraction strategies influence representation quality and multimodal reasoning performance in indoor 3D scenes. Experimental results show that the PTv3-based model can improve performance in a matched-domain setting, particularly when evaluated on ScanNet, suggesting that utilizing transformer architectures for scene encoding can enhance spatial and contextual representation learning. However, results in the full multi-dataset setting indicate that this advantage does not consistently transfer across heterogeneous indoor datasets, where PointNet++ demonstrates more stable generalization. These findings highlight both the potential of PTv3 for multimodal 3D scene reasoning and the need for improved domain adaptation or multi-dataset training strategies.

The conclusions of this thesis indicate that the choice of 3D backbone has a significant impact on the quality of geometric feature extraction and, consequently, on multimodal reasoning performance in 3D scenes. The findings suggest that PTv3 offers a promising direction for context-aware scene encoding, particularly through its ability to capture spatial relationships across the scene. At the same time, the results highlight practical limitations related to computational complexity, memory consumption, and scalability. Thus, future work is directed toward the exploration of alternative 3D backbone architectures, improved multimodal fusion strategies, greater computational efficiency, extension to outdoor and dynamic environments, and integration with embodied AI systems.

Keywords: Point Clouds, 3D Scene Reasoning, MSR3D, PTv3, Multimodal Data

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MSR3D	Multi-Modal Situated Reasoning in 3D Scenes
PTv1	Point Transformer version 1
PTv2	Point Transformer version 2
PTv3	Point Transformer version 3
MSQA	Multi-Modal Situated Question Answering
3D	Three Dimensional
RGB	Red Green Blue
BLIP	Bootstrapped Language-Image Pretraining
VQA	Visual Question Answering
SQA3D	Situated Question Answering in 3D Scenes
UI	User Interface
LLM	Large Language Model
MLP	Multi-Layer Perceptron
FPS	Farthest Point Sampling
BLIP 2	Bootstrapped Language-Image Pretraining version 2
FFNN	Feed-Forward Neural Network
kNN	k-Nearest Neighbors
RBG-D	Red, Green, Blue + Depth
CAD	Computer Aided Design
I/O	Input, and Output
AABB	Axis Aligned Bounding Box
NYU-40	the 40 semantic categories defined by the New York University indoor scene labeling scheme.
GMM	Gaussian Mixture Model
MSG	Multi-scale Grouping
MRG	Multi-resolution Grouping

1 Introduction

Scene understanding and reasoning is the process of perceiving and interpreting 3D scenes by recovering their geometric structure, identifying the objects within them, and reasoning about their spatial arrangement, physical relationships, and interactions in 3D space. Despite recent advances, robust interpreting 3D scene structure and object relationships remains an open problem in modern computer vision and machine learning. As learning based systems are increasingly deployed in real world applications such as robotics [31], autonomous driving [18], and augmented reality [28] their ability to interpret structured spatial information and respond to high-level queries has become critical.

Despite the remarkable success of modern learning based approaches in visual recognition, merely identifying objects within a scene does not constitute true scene understanding [2, 4, 36]. Recognition based systems typically operate by mapping visual inputs to predefined labels, capturing the presence of objects but often neglecting the relationships, context, and structure that give rise to meaningful interpretation. In more complex settings, understanding requires the ability to reason about interactions between entities, infer spatial arrangements, and integrate contextual cues to answer higher level queries. For example, determining what an object is used for, how it relates to surrounding elements, or whether certain conditions are satisfied within a scene cannot be achieved through isolated object predictions alone. Empirical evidence from tasks that require joint reasoning over visual content and semantic queries demonstrates that models with strong recognition performance frequently struggle when deeper inference is required [2, 4, 36]. This highlights a fundamental limitation of current approaches and underscores the need for representations and architectures that move beyond recognition toward structured and context aware reasoning.

Building upon these limitations, the challenge of achieving true scene understanding becomes even more pronounced when moving from 2D image based representations to 3D environments. In real world settings, perception is inherently spatial and embodied, requiring models to reason not only about object identities but also about their geometric arrangement, physical interactions, and consistency across viewpoints. Furthermore, 3D representations provide a richer and more faithful description of the environment, enabling the modeling of depth, scale, and spatial relationships that are often ambiguous or lost in 2D projections. However, this added complexity also introduces new challenges, as systems must integrate information over space, while maintaining coherent representations of the scene. Therefore, tasks such as situational or embodied question answering have emerged [36, 4], where an agent must interpret its environment and respond to queries based on its position, observations, and potential actions within the scene.

These settings further emphasize that understanding extends beyond static recognition, requiring the ability to ground perception in spatial context and reason about the environment in a more holistic and interactive manner.

In this context, 3D scene understanding has emerged as a central research direction, encompassing tasks such as semantic segmentation [42], object detection [13], and relational reasoning over spatial representations of the environment [11, 4]. Operating in three dimensions introduces distinct challenges, as visual information is commonly represented in forms such as point clouds, which are irregular and unordered, or voxel grids, that increase computational and memory complexity due to their sparsity. Unlike dense 2D image grids, these representations lack a consistent structure, making it difficult for models to directly apply standard convolution operations or capture long range dependencies. Consequently, effective 3D understanding requires specialized architectures capable of encoding geometric properties, aggregating local and global context, and maintaining consistency across varying viewpoints. These challenges highlight the need for models that can robustly interpret complex spatial data while supporting higher level reasoning about the scene.

In particular, indoor environments [13, 46, 53, 52] present a particularly compelling setting for understanding 3D scenes, as they are characterized by complex spatial layouts, high object density, and rich semantic structure. Compared to outdoor environments, indoor scenes typically exhibit higher object density and more cluttered spatial configurations, with objects arranged in close proximity and frequent occlusions arising from confined spaces and complex layouts [13, 46, 52]. Large-scale datasets such as ScanNet and ARKitScenes [13, 53] have demonstrated both the feasibility and the challenges of capturing and modeling such environments, highlighting issues related to incomplete observations, sensor noise, and variability in scene composition. Furthermore, indoor understanding requires not only accurate perception of object geometry but also the ability to reason about functional relationships between objects, such as support, containment, and affordances. While methods based on point cloud processing, such as PointNet and its extensions [42, 43], have enabled significant progress in learning from irregular 3D data, they still face limitations in capturing fine grained spatial interactions and integrating contextual information. These challenges make indoor 3D scenes a critical benchmark for advancing models capable of robust, structured, and semantically meaningful understanding.

In addition to geometric complexity, many 3D scene understanding systems must operate on multimodal inputs, combining information from sources such as RGB images, text, and point clouds. While these modalities provide complementary cues, their integration is complicated by differences in structure, scale, and representation space. Aligning heterogeneous data and embedding them into a coherent shared feature space remains a non-trivial problem, particularly when attempting to preserve both spatial and semantic consistency. As a result, designing archi-

tectures that can effectively fuse multimodal inputs while maintaining meaningful relationships across modalities is critical, as deficiencies in this process can directly impact representation quality and downstream reasoning performance

Recent advances in transformer based architectures [64, 58, 57] have demonstrated strong capabilities in modeling long range dependencies and capturing global context across a variety of domains, including natural language processing [50] and, more recently, computer vision [16]. Unlike traditional convolutional [23] or point based networks [42, 43], transformers operate through attention mechanisms that enable flexible interaction between elements, making them well suited for reasoning over structured relationships. These properties are especially appealing in the context of 3D scene understanding, where capturing interactions between spatially distributed entities and integrating information across modalities are fundamental challenges. Consequently, there is growing interest in exploring transformers as a backbone for processing point cloud data, with the potential to overcome limitations of existing architectures. This motivates the investigation of transformer based feature extraction strategies within the scope of this work.

Thus, this thesis focuses on the role of feature representation in multimodal 3D indoor scene understanding [34]. Specifically, it investigates how architectural design choices at the feature extraction stage influence a model's ability to encode spatial structure, capture contextual relationships, and support reasoning over indoor environments. In particular, the study examines how different backbone architectures affect the quality of learned representations, especially in settings that require integration of geometric, visual, and textual information.

To this end, the work explores the trade offs between established point based architectures [42, 43] and more recent transformer based approaches [64, 58, 57] when used as 3D backbones. These paradigms differ significantly in how they model spatial dependencies, aggregate contextual information, and scale to complex scenes. By comparing them in terms of both representation quality and computational efficiency, this study aims to provide a deeper understanding of how architectural choices impact overall system performance, as well as their suitability for supporting multimodal reasoning tasks in realistic 3D environments.

The thesis is organized into five chapters that cover the foundational concepts, the proposed approach, the experimental findings, and the broader implications of this work. Initially, Chapter 2 provides background on the key concepts and related work underlying this study, covering 3D data representations, large-scale 3D datasets, deep learning architectures for point cloud processing, and prior work in visual question answering and situated scene understanding. Building on these concepts, Chapter 3 describes the methodology, detailing the MSR3D baseline framework, the proposed integration of PTv3 as a replacement 3D backbone, the scene-level encoding pipeline, and the interactive multimodal visualization interface developed as part of this work.

Following the methodology, Chapter 4 presents the experimental setup and results, including dataset details, training configurations, evaluation metrics, and a comparative analysis between the PointNet++ baseline encoder and the proposed PTv3 across both single dataset and multi dataset settings. Chapter 5 then concludes the thesis with a summary of the key findings and their implications. Furthermore, it outlines the potential directions for future work, discussing potential improvements to the multimodal fusion strategies, the exploration of alternative 3D encoders, and integration with embodied AI systems.

1.1 Thesis Contributions

This work contributes to the study of multimodal 3D scene reasoning through both architectural investigation and system level development. Initially, a comprehensive analysis and empirical evaluation of the MSR3D framework was conducted on the ScanNet and MSQA datasets. This established a strong baseline for multimodal reasoning in indoor 3D environments and provided insights into the strengths and limitations of object-centric geometric representations when integrated with language and visual modalities.

Moreover, this work investigates the suitability of transformer based 3D backbones by studying PTv3 in depth. The architectural properties of PTv3, including serialized attention and scene-level processing, were analyzed in relation to the requirements of situated reasoning tasks. In contrast to traditional object-centric encoders, PTv3 enables global context modeling prior to object-level aggregation. Additionally, a complete preprocessing and data preparation pipeline was developed to support PTv3 integration. This includes the augmentation of raw point clouds with surface normals, computed via local covariance analysis, as well as normalization and structuring of scene-level inputs. These steps were necessary to align real world 3D scan data with the input assumptions of transformer based point cloud models. PTv3 was successfully integrated into the MSR3D framework as a replacement for the original PointNet++ 3D backbone. This required adapting the scene encoding pipeline from an object-centric to a scene-level paradigm, followed by the design of an object-level aggregation strategy to maintain compatibility with downstream modules. The resulting system preserves the multimodal reasoning capabilities of MSR3D while enhancing geometric representations with global spatial context.

Finally, beyond model development, an interactive multimodal visualization interface was implemented, which enables real time exploration of 3D scenes, and direct interaction with the model through natural language queries. The interface serves as a tool for qualitative analysis and improves interpretability. Overall, this work bridges the gap between local object based representations and global scene understanding, demonstrating how transformer architectures as 3D encoders can be effectively integrated into multimodal reasoning frameworks.

2 Background

Understanding and reasoning about 3D scenes relies on a combination of representations, data, and learning based models that collectively enable the interpretation of complex spatial environments [20, 7]. At the core of this field lie 3D data representations, which define how geometric information is structured and processed. However, effective 3D scene understanding extends beyond representation alone. Large-scale 3D datasets provide the necessary diversity and structure for training and evaluation [13, 53], while specialized deep learning architectures are designed to operate on these representations and extract meaningful geometric and contextual features [42, 43, 20].

As the field progresses, the focus has shifted from low level geometric processing to higher level reasoning tasks, including semantic understanding and visual question answering [20, 2]. More recently, research has moved toward situated and multimodal settings, where models must integrate geometric, visual, and textual information in order to support context aware reasoning in complex environments [34, 27].

This chapter provides an overview of the fundamental components that underpin modern 3D scene understanding. It begins with a discussion of 3D data representations, followed by an examination of commonly used datasets and learning architectures. Finally, it reviews recent advances in visual question answering and situated, multimodal scene understanding, establishing the foundation for the methods explored in this work.

2.1 3D Data Representations

Unlike traditional image based data, 3D representations encode geometric information in forms such as point clouds, meshes, voxel grids, and implicit functions, each offering distinct trade offs in terms of fidelity, efficiency, and computational complexity. Early approaches relied heavily on structured formats like voxels, which enabled the extension of convolutional neural networks to 3D space but suffered from prohibitive memory costs [59, 38, 44]. This limitation motivated the use of more efficient and flexible representations, including unordered point sets and 3D meshes, which better capture surface geometry while avoiding the cubic scaling of volumetric grids.

2.1.1 Point Clouds

A point cloud is a 3D data representation composed of a set of discrete points sampled from the surface or volume of an object or scene.Therefore, they can be formally represented as an

unordered set of points:

$$\mathcal{P} = \{p_i \mid p_i \in \mathbb{R}^3,\ i = 1, \ldots, N\} \tag{2.1}$$

Each point p_i is typically defined by spatial coordinates (x, y, z) and may additionally include attributes such as color, surface normals, or even metadata, like the instance labels, depending on the sensing modality. Point clouds are widely used in computer vision [20], robotics [31], remote sensing [66], and autonomous driving [18, 9, 48] because they are the direct output of many acquisition systems, including LiDAR, depth cameras, and 3D scanners. Compared with voxel grids or 3D meshes, point clouds provide a compact and flexible representation of geometric structure without imposing a regular discretization or requiring explicit surface connectivity. As a result, they are commonly adopted in scene understanding applications, including autonomous driving and robotics.

Despite these advantages, point clouds present several challenges for computational processing. Unlike images, they are unordered and irregular, which means that conventional convolutional operations cannot be applied directly. In addition, point clouds are often sparse, unevenly sampled, noisy, and incomplete, especially when captured in real world environments. Early deep learning approaches addressed these issues by converting point clouds into voxels [59, 38] or multi view images [47], but such transformations often introduced quantization errors or increased computational cost. More recent methods instead operate directly on raw point sets [42, 43, 57, 64].

Such methods include PointNet++ [43] and Point Transformer V3 [57], which constitute the primary focus of this thesis. Despite their differences in input structuring and feature extraction strategies, both approaches are designed to effectively address the challenges associated with processing raw point cloud data. In particular, they enable the direct learning of geometric representations from irregular and unordered point sets, thereby avoiding the need for fully discretized intermediate representations.

2.1.2 Meshes

On the other hand, a mesh is a discrete representation of a geometric domain, in which a continuous surface or volume is approximated by a finite set of elements and their connectivity relationships. More generally, a mesh can be viewed as a partition of space into smaller cells, enabling both geometric modeling and numerical computation [55, 6].

Meshes can be categorized along several dimensions depending on their structure and intended application. A primary distinction is between surface meshes and volumetric meshes. Surface meshes describe only the boundary of an object using polygonal faces, whereas volumetric meshes discretize the interior volume using geometric volumes [55, 5].

Within surface representations, the most common form is the polygon mesh, typically composed of vertices, edges, and faces that define the object's geometry. These meshes may consist of different polygon types, including triangle meshes, quadrilateral meshes, and more general n-gon meshes, each offering trade offs between simplicity, flexibility, and computational efficiency [1].

From a structural perspective, meshes can also be classified as structured, unstructured, or hybrid. Structured meshes exhibit regular connectivity and organization similar to grids, whereas unstructured meshes allow arbitrary connectivity and are better suited for complex geometries. Hybrid meshes combine both approaches to balance efficiency and flexibility [5].

Finally, different data structures may be used to represent mesh connectivity, such as face-vertex, vertex-vertex, or edge based representations, each with distinct computational and memory trade offs [6]. Despite this diversity in representation, all mesh types share a common characteristic: they explicitly encode topological relationships between elements, which distinguishes them from point representations.

2.1.3 Voxel Grids

Voxel grids represent 3D data by discretizing space into a regular grid of volumetric elements, where each voxel encodes occupancy or additional attributes such as density or semantic labels. Formally, a voxel grid can be described as a function over a 3D lattice:

$$V : \mathbb{Z}^3 \rightarrow \mathbb{R}^k \tag{2.2}$$

where each grid cell is associated with a feature vector. This regular structure enables the direct application of 3D convolutional neural networks, making voxel representations an early and widely adopted choice for learning on 3D data [59].

Despite their compatibility with convolutional architectures, voxel grids suffer from several inherent limitations. First and foremost, discretization of continuous space introduces quantization errors [59], which can lead to a loss of geometric detail, particularly at lower resolutions. Furthermore, voxel grids introduce discretization artifacts that can obscure fine geometric details [59], particularly in regions where objects are in close spatial proximity. At limited resolutions, distinct structures may become indistinguishable due to the coarse partitioning of space, leading to a loss of boundary precision. Increasing the resolution alleviates these issues but results in a cubic growth in memory and computational cost, limiting scalability [44] , thus making high resolution voxel representations impractical for large-scale or real time applications. As a result, while voxel based methods provide a structured and convolution friendly representation, their trade off between geometric fidelity and efficiency has motivated the adoption of

alternative representations, such as point clouds [43, 42], which preserve fine grained spatial information without imposing a fixed discretization.

2.2 3D Datasets

Three dimensional datasets capture the geometric structure of scenes or objects in 3D space using the representations described in the last section. In contrast to 2D image datasets, they include explicit depth information, allowing for accurate modeling of spatial relationships within a scene. The primary role of 3D datasets is to enable the development, training, and evaluation of algorithms in fields such as computer vision, robotics, augmented reality, and autonomous navigation [20]. Tasks supported by these datasets include 3D object detection, semantic segmentation, scene reconstruction, and visual question answering. 3D datasets are often captured using sensors such as LiDAR scanners, structured light cameras, or RGB-D sensors, and may include multimodal information such as color imagery, depth maps, etc. Moreover, these datasets are commonly accompanied by annotations such as semantic labels and object instance information, enabling the development of algorithms for a wide range of scene understanding tasks.

Despite sharing common representations and sensing modalities, 3D datasets differ significantly depending on the environments in which they are captured. These differences influence factors such as scale, density, level of detail, and the types of tasks they support. As a result, 3D datasets are typically categorized according to their underlying characteristics.

2.2.1 Single Shape Datasets

In addition to scene-level datasets, which capture complex environments containing multiple objects and contextual relationships, a significant category of 3D datasets focuses on individual objects, commonly referred to as single shape datasets. These datasets consist of isolated 3D models representing standalone objects without surrounding scene context, and are typically used to study object-centric geometric understanding.

Single shape datasets are usually composed of clean and well aligned 3D representations, such as meshes, point clouds, or implicit surfaces. Unlike indoor and outdoor scene datasets, which are often acquired through real world sensing devices, many single shape datasets are curated from synthetic sources or CAD repositories. This allows for precise control over geometry, topology, and labeling, resulting in high quality data with minimal noise and complete surface information.

Prominent examples include ShapeNet [10], a large-scale repository of richly annotated 3D CAD models organized into semantic categories such as chairs, cars, and airplanes, and ModelNet [59], a widely used benchmark for 3D shape classification and retrieval with curated, con-

sistently aligned CAD models. More recent datasets have further expanded the scope and granularity of 3D learning resources. Objaverse [15] provides a large-scale collection of annotated 3D objects, supporting data-hungry tasks such as 3D generation and foundation-model training. ShapeNet-Part [61] extends ShapeNet with part-level annotations for object categories, making it a standard benchmark for semantic part segmentation. PartNet [39] offers fine-grained, hierarchical, and instance-level part annotations, enabling more detailed studies of part-level 3D understanding. In contrast, the ABC dataset [29] focuses on CAD models with explicitly parameterized curves and surfaces, providing geometric ground truth for tasks such as patch segmentation, feature detection, shape reconstruction, and geometric deep learning. Collectively, these datasets have played a foundational role in advancing research in 3D deep learning, particularly in shape classification, retrieval, part segmentation, geometric reasoning, and generative modeling.

One of the defining advantages of single shape datasets is their suitability for learning fine grained geometric features. Since objects are isolated and free from occlusions or background clutter, models can focus on intrinsic shape properties, such as curvature, symmetry, and part structure. This makes them particularly valuable for developing and benchmarking architectures that operate directly on 3D representations, including point based networks, mesh based methods, and implicit neural representations.

However, the absence of environmental context also introduces limitations. Models trained exclusively on single shape datasets may struggle to generalize to real world scenarios, where objects appear in cluttered and dynamic environments with partial observations and sensor noise. Furthermore, the synthetic nature of many datasets can lead to a domain gap when transferring learned representations to real world data.

Despite these limitations, single shape datasets remain an essential component of the 3D vision ecosystem. They provide a controlled setting for studying fundamental geometric learning problems and often serve as a preliminary step before tackling more complex scene-level understanding tasks.

2.2.2 Indoor and Outdoor 3D Scene Datasets

Broadly 3D Scene datasets can be categorized into indoor and outdoor datasets, each serving different research purposes and presenting unique challenges. This distinction arises primarily from differences in the physical scale of the environments, the sensing technologies employed, and the nature of the scenes being captured. Indoor environments are typically constrained, structured, and densely populated with objects, whereas outdoor environments are large-scale, dynamic, and subject to significant environmental variability.

These differences have direct implications on data characteristics such as point cloud density,

level of geometric detail, and annotation complexity. Consequently, the choice between indoor and outdoor datasets is closely associated with the target application domain, with indoor datasets commonly used for tasks such as scene understanding and object interaction, and outdoor datasets being central to applications like autonomous driving and large-scale mapping.

2.2.2.1 Indoor Datasets

Indoor 3D datasets focus on enclosed environments such as residential spaces, offices, and industrial interiors. These datasets are usually captured using RGB-D cameras, structured light sensors, or handheld scanning devices, which operate effectively at close range and produce dense geometric representations of scenes [3, 13, 46].

One of the defining features of indoor datasets is the high density and level of detail in the captured data. The proximity of sensors to surfaces enables the acquisition of fine grained geometric structures, making these datasets particularly suitable for tasks such as instance segmentation, object recognition [46, 13], and high fidelity scene reconstruction [35]. Additionally, indoor environments generally exhibit more stable lighting conditions, which contributes to improved consistency in data acquisition.

However, indoor scenes often present challenges related to clutter, occlusions, and complex object arrangements. Furniture and everyday objects are frequently densely packed, leading to overlapping geometries and partial visibility. Unlike outdoor datasets, indoor datasets typically include rich semantic annotations covering structural elements such as walls [3], floors, and ceilings, as well as a wide variety of object categories.

2.2.2.2 Outdoor Datasets

Outdoor 3D datasets are primarily developed to represent large-scale, open world environments, most commonly urban and road scenes. These datasets are typically collected using LiDAR sensors mounted on mobile platforms such as vehicles, often combined with RGB cameras and localization systems to provide multimodal data. As a result, they are widely used in applications such as autonomous driving [17, 9], large-scale mapping, and navigation, where understanding complex and dynamic environments is essential [48].

A key characteristic of outdoor datasets is their large spatial scale and relatively sparse geometric representation. While LiDAR sensors provide reliable depth measurements across considerable ranges, the resulting point clouds are inherently sparse compared to indoor acquisitions. Moreover, outdoor environments introduce significant variability due to illumination changes, weather conditions [9], seasonal variations, and the presence of dynamic elements such as vehicles and pedestrians. These factors increase the complexity of tasks such as object detection, semantic segmentation, and tracking.

Despite their importance, outdoor datasets present several challenges. Issues such as occlusions, uneven point density, and sensor noise, especially at longer ranges, can degrade data quality. Additionally, the large volume of data makes annotation and processing computationally demanding, requiring efficient algorithms capable of handling scale and sparsity.

2.2.3 3D Scene Question Answering Benchmarks

Beyond geometric and semantic understanding, recent work has introduced benchmarks that evaluate higher level reasoning in 3D environments through question answering tasks. Genrally, a benchmark is a standardized dataset, task, or evaluation framework used to measure and compare the performance of different methods or models. These benchmarks require models to jointly process spatial information and natural language in order to infer relationships, object properties, and contextual information within a scene.

Early efforts such as ScanQA [4] and SQA3D [36] focus on grounding natural language questions in indoor 3D scenes derived from datasets such as ScanNet [13], enabling reasoning over object locations and spatial relationships. More recently, the MSQA [34] benchmark has been proposed to address limitations in scale and modality. MSQA introduces a large-scale dataset containing approximately 251K question and answer pairs across multiple reasoning categories, and incorporates interleaved multimodal inputs, including text, images, and point clouds, to better capture complex real world scenarios. Consequently, MSQA stands as one of the most comprehensive and important benchmarks for indoor 3D scene understanding, which motivates its selection as the primary benchmark in this work.

These benchmarks highlight the growing importance of multimodal reasoning in 3D scene understanding, extending beyond traditional perception tasks toward more comprehensive situation aware intelligence.

2.3 Deep Neural Networks for for 3D Scene Understanding

To enable effective 3D scene understanding, a variety of architectural approaches have been developed to process the different forms of 3D data representations. Rather than relying on a single unified framework, these approaches are typically designed around the underlying representation, such as voxel grids, point clouds, or mesh based structures [59, 42, 21].

Each architectural paradigm defines a distinct strategy for extracting and aggregating geometric information, leading to different compromises in terms of efficiency, scalability, and representational capacity [20, 7]. As a result, the choice of architecture is closely tied to both the data representation and the target application.

Over time, these architectural approaches have evolved alongside advances in both data acquisition and computational methods. Early methods primarily focused on adapting existing techniques from 2D vision to structured 3D representations [38, 59], while more recent approaches have been specifically designed to operate directly on raw geometric data [42, 54, 64]. This evolution reflects a broader shift toward architectures that better preserve spatial information while maintaining computational efficiency.

At the same time, increasing task complexity has driven the development of architectures capable of capturing both local geometric details and global contextual relationships within a scene [43, 64, 58]. Modern approaches extend beyond purely geometric processing, incorporating mechanisms that enable richer feature interactions and improved representation learning [50, 16]. These developments have contributed to significant progress in tasks such as semantic segmentation, object detection, and higher level reasoning in 3D environments [13, 9, 34].

2.3.1 Deep Architectures for 3D Data

Recent advances in deep learning have significantly transformed the field of 3D scene understanding, enabling models to learn rich and task specific representations directly from raw data. Unlike traditional approaches that rely on manually designed features, deep learning methods automatically extract hierarchical features that capture both local geometric structures and global contextual information [20].

Building upon the architectural paradigms discussed previously, deep learning models have been adapted to operate on various 3D data representations, including voxel grids, meshes, and point clouds. Among these, point based methods have gained particular prominence due to their ability to process raw geometric data efficiently while preserving fine grained spatial information.

Modern deep learning approaches for 3D data extend beyond purely geometric processing, incorporating mechanisms such as hierarchical feature learning, attention based modeling, and multimodal fusion. These developments have led to substantial improvements across a wide range of tasks, including object classification, semantic segmentation, and 3D scene reasoning.

In this context, architectures such as PointNet [42] and PointNet++ [43] form the foundation of many contemporary methods, providing a framework for directly learning from point clouds. More recent approaches, including transformer models such as Point Transformer [64], further extend these ideas by incorporating attention mechanisms for improved modeling of global context. In particular, Point Transformer V3 [57] represents a recent advancement in this direction, combining efficiency with expressive feature learning for point cloud understanding.

2.3.2 Voxel Based Architectures

Voxel based architectures represent one of the earliest approaches to 3D scene understanding, extending the principles of grid based processing from 2D images to 3D space. By discretizing a scene into a regular volumetric grid, these methods enable the direct application of 3D convolutional operations, allowing models to learn spatial features in a structured manner.

Early works such as 3D ShapeNets [59] and VoxNet [38] demonstrated the effectiveness of 3D convolutional neural networks for tasks including object classification and detection. These approaches leveraged the regular structure of voxel grids to apply standard convolutional pipelines, making them a natural extension of established 2D vision techniques.

However, the use of voxel representations introduces significant computational challenges. The resolution of a voxel grid grows cubically with respect to the spatial dimensions, leading to high memory consumption and computational cost [59, 44]. As a result, practical implementations are often limited to low resolution grids, which can lead to a loss of fine geometric detail and reduced accuracy in complex scenes. To address these limitations, later works introduced sparse convolutional methods that operate only on occupied voxels, significantly improving efficiency [44].

Overall, voxel based architectures played a crucial role in the development of 3D deep learning by providing a structured framework for processing volumetric data. Their shortcomings have motivated the exploration of more efficient and expressive approaches, particularly those operating directly on point based and sparse representations.

2.3.3 Mesh based Architectures

Mesh based architectures operate on representations that explicitly encode both geometric and topological information through vertices, edges, and faces. Unlike point clouds, which represent geometry as an unordered set of points, meshes provide connectivity information that allows models to capture surface structure more effectively [6].

Early approaches to mesh processing relied on geometric descriptors and handcrafted features defined over mesh elements. More recent methods extend deep learning techniques to non Euclidean domains [7], enabling convolution like operations directly on mesh surfaces. For example, graph based formulations and spectral methods [8, 37] allow the propagation of information across connected vertices, facilitating the learning of local geometric patterns. More recent architectures such as MeshCNN [21] further demonstrate how deep learning models can operate directly on mesh structures by leveraging edge based representations.

Despite their expressive power, mesh based architectures present several challenges. The irregular structure and varying topology of meshes make it difficult to define consistent convolutional

operations, while the complexity of mesh connectivity increases computational overhead [7, 37, 21]. Additionally, large-scale scene understanding tasks often rely on data acquired from sensors, where mesh representations are not directly available and must be reconstructed from raw sensor data [13, 9].

As a result, although mesh based methods offer a detailed representation of surface geometry, they are less commonly used in large-scale 3D scene understanding compared to point based or voxel based approaches. Nevertheless, they remain important in applications that require detailed surface modeling and geometric analysis.

2.3.4 Point based Architectures

Point based architectures operate directly on raw point clouds, avoiding the need for discretization into structured representations such as voxel grids or projections into multiple views [42, 20]. By preserving the original geometric information, these methods provide a flexible and efficient framework for 3D scene understanding.

A key challenge in processing point clouds lies in their unordered nature, which requires models to be invariant to permutations of the input points. Additionally, effective architectures must capture both global structure and local geometric relationships. These challenges have driven the development of specialized neural architectures designed specifically for point set processing, with PointNet and its hierarchical extension PointNet++ representing foundational contributions in this area [42, 43].

2.3.4.1 PointNet

PointNet [42] is a pioneering architecture that directly consumes unordered point sets and learns global features through a simple yet effective design. The model applies shared MLPs independently to each point, transforming them into higher dimensional feature representations. To ensure permutation invariance, a symmetric aggregation function, typically max pooling, is used to combine point-wise features into a global descriptor as shown in Figure 2.1.

This design enables PointNet to process point clouds efficiently while remaining invariant to the ordering of input points. Additionally, the architecture incorporates transformation networks (T-Nets) to learn input and feature space alignments, improving robustness to geometric variations.

Despite its effectiveness, PointNet has a notable limitation: it does not explicitly capture local geometric structures, as each point is processed independently prior to global aggregation. This restricts its ability to model fine grained spatial relationships, which are critical for detailed scene understanding tasks. Nevertheless, PointNet demonstrated that deep neural networks could effectively operate directly on raw point sets without requiring structured spatial representations.

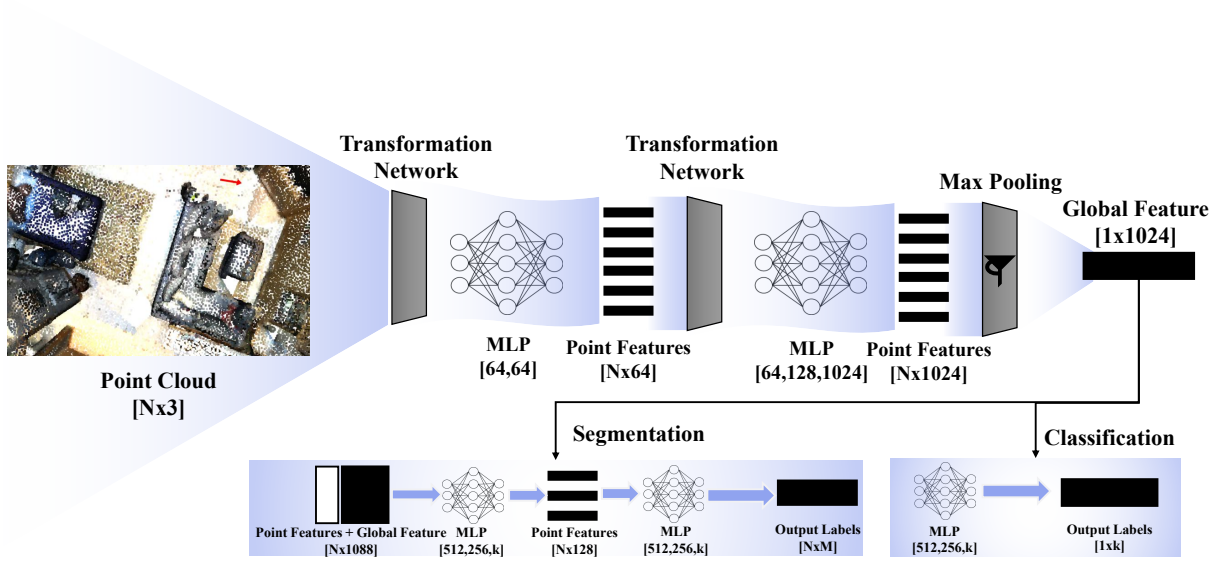

Figure 2.1: Detailed architecture of PointNet for point cloud processing. The network applies shared MLP layers to each point independently, followed by input and feature transformation networks to achieve alignment. A symmetric max pooling function extracts a global feature vector of size 1×1024. This representation is used for object-level classification, while a combination of global and local features is propagated through additional MLP layers for point-wise segmentation.

Its simplicity, computational efficiency, and strong performance on classification and segmentation tasks established it as a foundational framework for subsequent point cloud architectures.

2.3.4.2 PointNet++

PointNet++ [43] extends PointNet by introducing a hierarchical framework that enables the learning of local geometric features. Instead of processing all points independently, PointNet++ groups points into local neighborhoods using spatial sampling and clustering techniques, such as farthest point sampling and kNN grouping.

Within each local region, PointNet is applied recursively to learn features at multiple scales, allowing the network to capture both fine grained local structures and broader contextual information. This hierarchical feature learning significantly improves performance on tasks requiring detailed geometric understanding, such as segmentation and scene parsing.

By combining local feature extraction with global context aggregation, PointNet++ addresses the primary limitation of PointNet and establishes a more powerful and scalable architecture for point cloud processing. As a result, it has become a foundational model in 3D deep learning and serves as a basis for many subsequent architectures.

2.3.5 Point Transformers

While hierarchical architectures such as PointNet++ [43] effectively capture local geometric structures, they rely on predefined neighborhood grouping and fixed aggregation schemes. This can limit their ability to model complex and long range interactions between points, particularly in large-scale and highly irregular scenes. To address these limitations, recent work has introduced transformer based architectures for point cloud processing, leveraging attention mechanisms to learn adaptive relationships directly from the data [64].

The transformer paradigm, originally developed for sequential data, is inherently flexible due to its ability to model pairwise interactions via self attention. However, directly applying standard transformers to point clouds is non trivial due to the unordered and irregular nature of the data. Point Transformer [64] adapts self attention to point clouds by incorporating local neighborhoods and relative positional encoding, enabling the model to capture both spatial relationships and feature similarities in a permutation invariant manner.

2.3.5.1 Point Transformer V1

A key innovation of PTv1 [64] is its vector self-attention mechanism with trainable relative positional encoding. Since point clouds are unordered sets embedded irregularly in 3D space, PTv1 applies self-attention locally over the k-nearest neighbors of each point. Instead of relying on absolute coordinates and applying global attention, the model encodes the relative displacement between neighboring points, allowing the attention operation to adapt to local geometric structure.

The point transformer layer can be written as:

$$\mathbf{y}_i = \sum_{\mathbf{x}_j \in \mathcal{X}(i)} \rho\left(\gamma\left(\varphi(\mathbf{x}_i) - \psi(\mathbf{x}_j) + \delta\right)\right) \odot \left(\alpha(\mathbf{x}_j) + \delta\right) \tag{2.3}$$

where $\mathcal{X}(i)$ denotes the local neighborhood of point $\mathbf{x}_i$, typically defined by its k-nearest neighbors. The functions φ, ψ, and α are learned feature transformations, while γ is an MLP that produces the attention vector. The operator $\odot$ denotes element-wise multiplication.

The function ρ denotes a softmax normalization applied over the local neighborhood $\mathcal{X}(i)$. It converts the attention scores produced by γ into normalized attention weights, so that neighboring points with more relevant geometric and feature relationships contribute more strongly to the output feature $\mathbf{y}_i$. Thus, for each point i, the attention weights are normalized across its neighboring points $j \in \mathcal{X}(i)$.

The relative positional encoding is defined as:

$$\delta = \theta(\mathbf{p}_i - \mathbf{p}_j) \tag{2.4}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the 3D coordinates of points i and j, respectively. The encoding function θ is implemented as an MLP and is trained jointly with the rest of the network. Importantly, this positional encoding is added to both the attention-generation branch and the feature-transformation branch, enabling the network to incorporate local geometric relationships directly into both attention weights and transformed point features.

This formulation enables the model to adaptively weight neighboring points based on both feature similarity and geometric context. As a result, PTv1 achieves strong performance on tasks such as semantic segmentation and classification, outperforming earlier point based architectures.However, PTv1 still relies on explicit neighborhood construction and local attention computation, which can become a bottleneck for large-scale point clouds.

2.3.5.2 Point Transformer V2

Point Transformer V2 (PTv2) [58] builds upon PTv1 by addressing its efficiency limitations and enhancing its ability to model complex geometric structures. One of the primary contributions of PTv2 is the introduction of *Grouped Vector Attention*, which improves both computational efficiency and representational capacity.

In PTv2, feature channels are divided into groups, and attention is computed within each group independently. This reduces computational overhead while allowing the model to capture diverse patterns across different feature subspaces. Additionally, PTv2 introduces more expressive positional encoding schemes that better capture geometric relationships.

Another important advancement is the incorporation of partition based pooling, which improves scalability by organizing points into spatial partitions. This enables more efficient hierarchical processing compared to the purely local neighborhood approach used in PTv1.

Overall, PTv2 demonstrates improved performance and scalability, particularly for large-scale scene understanding tasks. Despite these improvements, the model still depends on explicit spatial partitioning and neighborhood queries, which can limit efficiency when processing extremely dense point clouds.

2.3.5.3 Motivation for Scalable Transformers

Transformer based approaches represent a shift from manually designed local aggregation strategies toward data driven modeling of point interactions. By leveraging attention mechanisms,

these models provide more flexible and context aware feature extraction compared to traditional architectures.

However, a key challenge remains, which is scaling transformer models to large point clouds without incurring prohibitive computational costs. Both PTv1 and PTv2 rely on explicit neighborhood construction or spatial partitioning, which introduces overhead and limits scalability.

Recent work has begun to explore alternative strategies that avoid costly geometric queries and instead impose structure on the point cloud to enable efficient attention computation. These approaches aim to preserve the expressive power of transformers while significantly reducing computational complexity.

In this context, Point Transformer V3 [57] proposes a simplified and highly efficient design based on point cloud serialization and localized attention over structured sequences. This direction highlights a broader trend toward balancing efficiency and performance, enabling transformer based models to scale to large 3D scenes. A detailed discussion of PTv3 is provided in Chapter 3.

2.3.5.4 Hybrid Convolution Transformer Architectures

Although transformer based models [64, 58, 57] have become dominant in recent point cloud understanding, many modern point cloud transformers are not purely attention based architectures. Instead, they combine convolutional operations with attention mechanisms in order to exploit the complementary strengths of both operators [62]. Convolution provides a strong inductive bias for capturing local geometric structures, while attention is more effective for modeling long range dependencies and semantic relationships within a scene.

This hybrid design is particularly relevant for hierarchical point cloud networks, where the role of each operator can vary across different stages of the architecture. In early stages, the point cloud is usually represented at high spatial resolution, meaning that the model must process a large number of point tokens. At this level, local geometric patterns such as edges, surfaces, object boundaries, and small structural details are dominant. Convolutional operations are well suited to this setting because they efficiently aggregate information from local neighborhoods.

In deeper stages, the point cloud is progressively downsampled, reducing the number of tokens while increasing the semantic abstraction of the learned features. At this level, the model benefits more from attention mechanisms, which can capture broader contextual relationships between distant regions of the scene. Therefore, a hybrid architecture can improve efficiency by using convolution where local geometry is most important and attention where high-level contextual reasoning is more useful.

A recent example of this design principle is LitePT [62]. LitePT revisits the structure of point

transformer architectures and proposes a stage aware hybrid design. Instead of applying both convolution and attention uniformly at every level of the network, LitePT uses sparse convolutional blocks in the early encoder stages and attention blocks in the later stages. This allows the model to preserve efficient local geometric processing at high resolution while reserving attention for lower resolution stages where semantic context can be modeled more effectively.

LitePT also introduces PointROPE [62], a parameter free 3D positional encoding adapted from rotary positional embeddings. Since attention does not inherently encode spatial layout, positional encoding is necessary for point cloud transformers. PointROPE applies rotary encoding along the three coordinate axes and is used within the attention module to preserve geometric layout without adding learnable convolutional positional encoding layers.

This architecture demonstrates that the performance of point cloud transformers depends not only on the use of attention, but also on the way different computational operators are distributed across the hierarchy. By allocating convolution and attention according to their respective strengths, LitePT achieves a more efficient and scalable design while maintaining strong performance across semantic segmentation, instance segmentation, and object detection tasks.

2.4 Vision Question Answering

Visual Question Answering is a multimodal task that requires models to answer natural language questions based on visual input. In this setting, a model is given a visual representation along with a question and is expected to produce a corresponding answer. Unlike traditional vision tasks, VQA requires not only visual perception but also language understanding and reasoning capabilities [2].

A key characteristic of VQA is its reliance on multimodal reasoning, where models must integrate information from both visual and textual modalities. Questions can involve a wide range of reasoning types, including object recognition, counting, spatial reasoning, and inference. As a result, VQA is often viewed as a step toward more general and holistic scene understanding [2].

With advances in deep learning, VQA has evolved beyond 2D image based settings to more complex domains, including video and 3D environments. This progression reflects the growing need for models that can reason about richer and more structured representations of the world.

2.4.1 2D Visual Question Answering

The task of 2D VQA is the most established form of the task, where models operate on RGB images. Early approaches typically combined convolutional neural networks for visual feature extraction with recurrent neural networks for encoding the question. These features were then

fused to produce an answer prediction [2].

A defining aspect of 2D VQA is the diversity of question types. Questions may require object recognition, attribute identification, counting, spatial reasoning, or common sense inference. To address these challenges, as mentioned in the previous section later works introduced attention mechanisms that allow models to focus on relevant regions of the image when answering a question.

More recent approaches employ vision transformers architectures, which improve the modeling of global context and enable more effective cross modal interactions [16]. Despite these advancements, 2D VQA systems often suffer from dataset biases, where models may exploit statistical correlations between questions and answers rather than relying on true visual understanding.

2.4.2 3D Visual Question Answering

In contrast, 3D VQA extends the VQA task to 3D environments, where reasoning is performed over 3D representations such as point clouds, meshes, or multi view data. Compared to 2D VQA, 3D VQA enables more explicit modeling of spatial relationships, as geometric information is directly available [33].

This additional spatial information allows models to perform more complex reasoning tasks, such as understanding object positions, distances, and orientations within a scene. For example, questions may involve identifying objects behind others, estimating relative sizes, or reasoning about spatial arrangements.

However, 3D VQA also introduces significant challenges. Models must handle irregular and unstructured data, integrate multiple modalities, and establish alignment between language and 3D geometry. These challenges make the task substantially more complex than its 2D counterpart.

Recent benchmarks such as ScanQA [4] and SQA3D [36] demonstrate the potential of 3D representations for improving reasoning capabilities in VQA tasks. At the same time, they highlight the importance of combining geometric understanding with multimodal learning [33]. As a result, 3D VQA has emerged as an important research direction toward more comprehensive and context aware scene understanding.

2.5 3D Scene Understanding

Scene understanding refers to the broader task of extracting structured information from an environment. In 3D vision, this usually involves identifying objects, assigning semantic labels, separating object instances, and capturing spatial or relational structure within a scene. The

objective is to convert raw geometric observations into representations that are more useful for higher level reasoning and decision making [13, 46, 3].

Recent work has increasingly aimed to unify multiple scene understanding tasks within a single framework. UniSeg3D is an example of this direction, providing a common architecture for semantic segmentation, instance segmentation, panoptic segmentation, referring segmentation, interactive segmentation, and open vocabulary segmentation in 3D environments [60]. This unified perspective is important because these tasks describe complementary aspects of the same scene and can benefit from shared representations.

Even so, scene understanding in this conventional sense remains primarily global in nature. Its purpose is to describe what exists in the environment as a whole. It does not necessarily address how that information should be interpreted relative to a specific observer, viewpoint, or situation. This limitation becomes important in embodied settings, where an agent must reason from a particular position and under partial observations. Consequently, situated scene understanding extends conventional 3D scene understanding by grounding scene interpretation in the observer's current context [60, 34].

2.5.1 Situated Scene Understanding

Situated scene understanding is defined as the task of interpreting a 3D environment from a specific situation, where the situation encodes factors such as the observer's position, orientation, and contextual observations. Unlike general scene understanding, which aims to produce a complete and consistent description of the entire scene, this task focuses on interpreting the environment relative to a localized context [36].

A key property of this setting is that reasoning is constrained by the given situation. Even when a model has access to a large portion of the scene, only a subset of this information is relevant to the current context. As a result, the model must identify and focus on situation dependent evidence, rather than relying on a uniform global representation. This introduces a form of ambiguity, since multiple elements in the scene may be present, but only some are meaningful with respect to the situation.

Another defining characteristic is that spatial relationships are interpreted relative to the observer. Concepts such as distance, visibility, and object relations depend on the specific viewpoint or position from which the scene is considered. Consequently, understanding requires grounding geometric and semantic information within a local frame of reference, rather than relying solely on absolute scene-level coordinates.

In addition, situated scene understanding involves a strong interaction between perception and reasoning. The interpretation of the scene is guided not only by what is present, but also by

what is relevant to a given query or task. This requires models to selectively attend to parts of the environment that are informative for the current context, effectively filtering out irrelevant information.

Overall, situated scene understanding represents a shift from global scene description toward context aware reasoning. It emphasizes the importance of grounding, relevance, and observer dependent interpretation, making it a more realistic formulation for reasoning in complex 3D environments.

2.6 Multimodal Scene Understanding

Multimodal scene understanding studies how different forms of information can be combined to produce a richer interpretation of an environment. In 3D tasks, these forms of information often include geometry, visual appearance, and language. Each modality contributes a distinct type of knowledge: point clouds provide spatial structure, images offer dense visual detail, and text specifies tasks, questions, or contextual information.

Research in multimodal learning has shown that combining modalities effectively requires more than simply merging features. It is also necessary to understand how each modality contributes on its own and how the modalities interact with each other. The ICLR Blogposts review on multimodal learning emphasizes that strong performance depends on modeling both modality specific information and cross modal relationships [27]. This perspective is especially important in 3D scene understanding, where different modalities often provide complementary and unevenly distributed evidence.

An influential example from the broader multimodal literature is Frozen, which demonstrates that a vision encoder can be connected to a frozen language model so that the system can process interleaved image and text inputs [49]. Although this work is not specific to 3D scenes, it is relevant because it shows how multimodal reasoning can emerge from a shared representational framework rather than from narrowly specialized pipelines.

Within the context of situated understanding, MSQA adopts this multimodal perspective directly by using interleaved text, image, and point cloud inputs to represent both the question and the situation [34]. This design reflects the fact that understanding in realistic 3D environments often depends on several sources of information at once. Multimodal scene understanding is therefore not only a matter of combining inputs, but also a matter of grounding reasoning across complementary representations of the same environment [34, 49, 27].

3 Methodology

This chapter presents the methodology followed in this thesis for investigating multimodal reasoning in 3D environments. It first introduces the MSR3D task setting and defines the aims of the study, before describing the scene encoding strategies used to represent 3D environments. Particular emphasis is placed on the comparison between the original PointNet++ object-centric backbone and the proposed PTv3 scene-level encoding pipeline. The chapter then outlines the overall model architecture, the integration of multimodal inputs, the interactive visualization interface, and the main contributions of the implemented system.

3.1 MultiModal 3D Scene Reasoning

The task addressed in this thesis is multimodal scene reasoning in 3D environments, with a particular focus on the MSR3D framework [34]. In this setting, the objective is to answer natural language queries about a scene by jointly processing information from multiple modalities, including 3D point clouds, 2D images, and textual descriptions. Unlike traditional vision tasks that operate on a single modality, this problem requires the model to integrate heterogeneous sources that interleave in a single question.

More specifically, the model is provided with a 3D scene, a situation description that defines the context of an agent within that scene, and a question that must be answered based on both spatial and semantic understanding. The situation may include information about the agent's position, orientation, and observations, while the scene itself is represented through geometric and visual data. As a result, the task involves not only recognizing objects and their properties, but also interpreting relationships such as distance, visibility, and relative positioning from a contextual perspective.

A key challenge of this setting lies in the alignment of modalities. The model must transform point cloud features, image representations, and text embeddings into a shared space in which meaningful interactions can occur. This requires effective feature extraction mechanisms for each modality, as well as a fusion strategy that preserves both geometric structure and semantic content. In addition, the reasoning process must remain sensitive to the given situation, since the relevance of scene elements depends on the context in which the query is posed.

Within this context, the MSR3D model provides a structured approach to multimodal reasoning in 3D scenes. It combines a 3D backbone for geometric feature extraction, a 2D backbone for visual information, and a language model for processing and generating text, all integrated through a unified fusion mechanism. This design enables the model to ground language queries

in the spatial structure of the environment while leveraging complementary visual cues, and was therefore selected as the baseline framework for this study.

3.2 Aims and Objectives

The main objective of this thesis was to investigate the performance of PTv3 [57] as a 3D backbone for the MSR3D model [34]. More specifically, the study aimed to examine whether PTv3 could serve as an effective alternative to the original backbone of the model, while maintaining the ability of MSR3D to support multimodal reasoning over complex 3D indoor scenes. In this sense, the work was not limited to a simple replacement of one module with another, but rather focused on understanding how such a modification would influence the behavior of the overall architecture and its ability to process geometric information within the broader multimodal pipeline.

In pursuit of this objective, as a first step, a detailed study of MSR3D and its components was conducted, with particular emphasis on the 3D backbone module. This initial stage was necessary in order to build a clear understanding of the internal structure of the model, the role of each component, and the way in which the different modalities are combined during the reasoning process. Particular attention was given to the interaction between the 3D backbone and the remaining modules, since this formed the foundation for the integration of PTv3 at a later stage.

In a subsequent step, after grasping the components of the model, the focus shifted to the dataset by reviewing its format and creating various visualizations for the 3D Scene Point Clouds. This stage was important not only for understanding the organization and structure of the data, but also for obtaining a more intuitive view of the geometric content of the scenes and the type of information that the model is required to process. Through this process, the relationship between the raw point cloud data and the corresponding scene-level reasoning tasks became clearer, thus supporting the later experimental stages of the work.

Following the experiments on MSR3D, it was natural to explore the capabilities of PTv3 and then proceed with its integration as a 3D backbone for MSR3D. This phase involved studying the architecture and design principles of PTv3 in order to determine how it could be adapted to the requirements of the existing MSR3D framework. The goal of this integration was to preserve the general functionality of the model while replacing the original geometric feature extractor with a more recent transformer based alternative. In this way, the study moved from understanding the baseline system to actively modifying it in order to assess the impact of the proposed change.

Finally, after the successful integration of PTv3 and the comparison of the performances between the two backbones, and the final step was to create an interactive UI that would allow the user to

not only visualize the 3D Scene and situation, but also interact with the model and ask custom questions about the scene. The purpose of this interface was to complement the experimental results with a more practical and intuitive way of exploring the model's behavior. By allowing direct interaction with the scene and the generated responses, the UI served as an additional means of demonstrating the functionality of the final system and illustrating the potential of the developed approach in a more accessible and applied setting.

3.3 Model Architecture

MSR3D [34] is a multimodal framework designed for situated reasoning in 3D environments. The framework combines geometric scene representations, visual information, and natural language inputs within a unified architecture, enabling models to reason about objects, spatial relationships, and contextual interactions inside complex indoor scenes.

The framework operates on object-centric scene representations extracted from large-scale 3D datasets, including ScanNet [13], ARKitScenes [53], and 3RScan [52]. By integrating multiple datasets with varying scene layouts, sensor characteristics, and annotation styles, MSR3D provides a challenging benchmark for evaluating robustness and generalization in multimodal 3D reasoning tasks.

A central component of MSR3D is its ability to align geometric object representations with language and visual features through a shared multimodal embedding space. This enables the framework to support tasks requiring grounded reasoning, spatial understanding, and context aware interpretation of 3D scenes.

3.3.1 Architecture

The MSR3D model architecture consists of core modules that work together to process multimodal inputs, including PointNet++ [43] as the 3D backbone for point cloud feature extraction, a 2D backbone for image feature extraction and a text tokenizer. Moreover, the model includes a LLM that is responsible to generate the answers based on the embeddings that are produced from the previous modules. The multimodal fusion is achieved with a placeholder token approach where the embeddings from the 3D backbone and 2D backbone substitute specific tokens in the situation and query text.

3.3.2 3D Backbone

Within MSR3D, the role of the 3D backbone is to convert point cloud data into learned object embeddings that can be used by the downstream multimodal reasoning pipeline. These em-

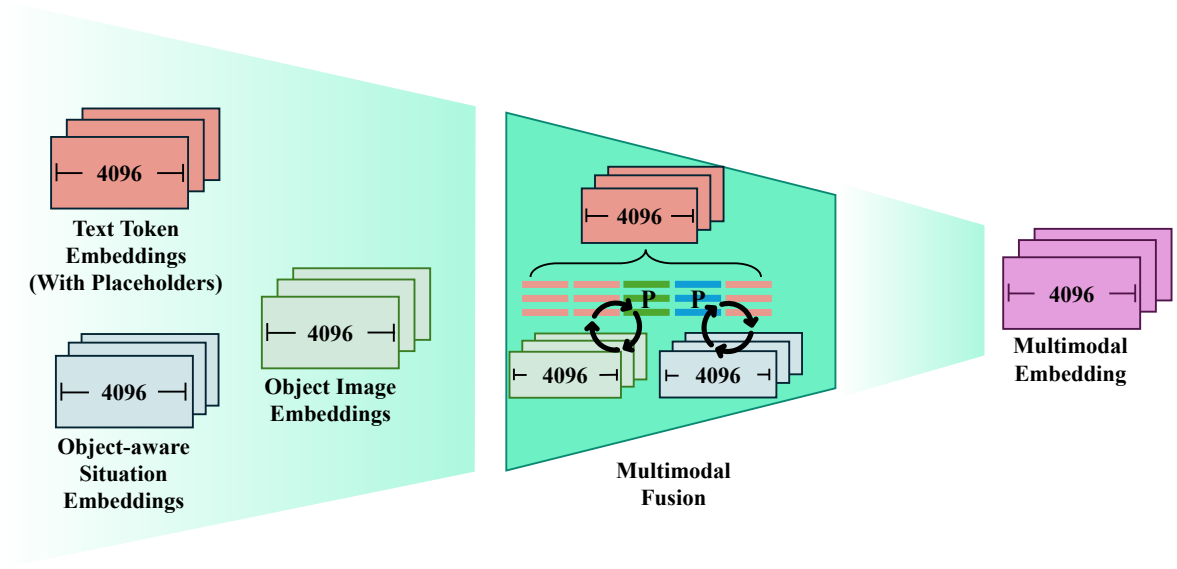

Figure 3.1: Multimodal fusion mechanism that integrates text token embeddings with object-level image and point cloud embeddings by replacing placeholder tokens with aligned modality-specific representations in a shared embedding space (4096 dimensions), producing a unified multimodal embedding of the input prompt.

beddings provide the geometric information required for grounding language queries in the 3D scene. After being extracted by the backbone, the object features are combined with spatial information, such as object position and extent, and are then passed to the situation encoder. In this way, the 3D backbone does not directly generate answers, but provides the geometric foundation on which the situation encoder and language model perform higher level reasoning.

This design is important because MSR3D operates on situated questions, where the answer may depend not only on object identity but also on the relationship between objects, the agent's position, and the surrounding scene context. Therefore, the quality of the 3D backbone directly affects how effectively the model can represent the physical structure of the environment before it is fused with textual and visual modalities.

3.3.3 2D Backbone

The model uses a pretrained BLIP 2 [32] model as the 2D backbone to extract image features from the images included in the situation description.

More concretely, uses a lightweight 12-layer Transformer encoder, between frozen pretrained image encoders and LLMs. The Transformer is used to bridge the gap between the two modalities. To be more precise, the Querying Transformer is pretrained in two stages.

The first stage initializes vision to language representation learning using a frozen image encoder, while the second stage facilitates vision to language generative learning through a frozen

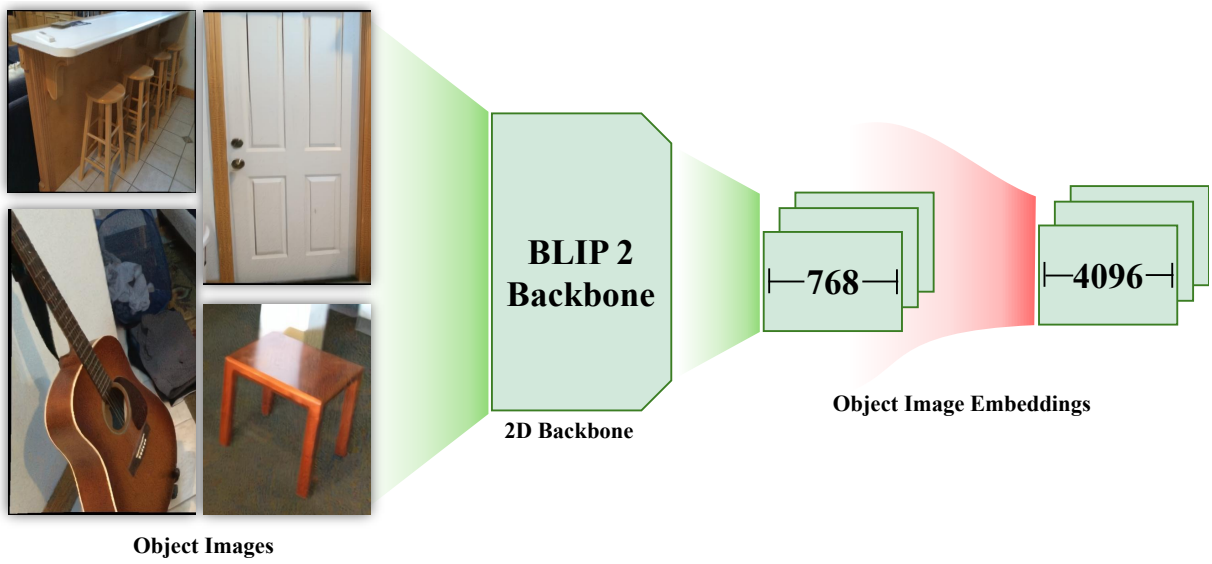

Figure 3.2: Architecture of the 2D backbone, where object images are encoded using a BLIP-2 model to produce visual embeddings, which are then projected from a lower-dimensional feature space into a shared 4096-dimensional representation.

language model. With this approach, BLIP 2 achieves strong performance on various vision-language tasks while keeping the computational cost low.

3.3.4 Text Encoder

The text encoder is responsible for tokenizing the input text, and then embedding the tokens into a high dimensional space. In this context, text refers to the question, situation and role descriptions which are all used as a single input to the model for each sample. The first step in the text encoding pipeline is to tokenize the text using a pretrained tokenizer, Llama Tokenizer [30]. After tokenization, the tokens consist of both text tokens and images as the question and situation descriptions may include images. In MSR3D, this is handled by substituting the image tokens with special placeholder tokens. These placeholder tokens are then replaced with the corresponding image embeddings extracted from the 2D backbone during the multimodal fusion process. Moving forward, the token embeddings are obtained by feeding the tokens into an LLM, and more specifically, Vicuna 7B [65]. The LLM generates high dimensional embeddings for each token, capturing the semantic meaning of the text.

3.3.5 Situation Encoder

The Situation Encoder is designed to generate a compact yet expressive representation of the agent's surrounding environment by integrating multimodal scene information. More precisely, it takes as input scene-level object embeddings, object spatial locations, object masks, as well as

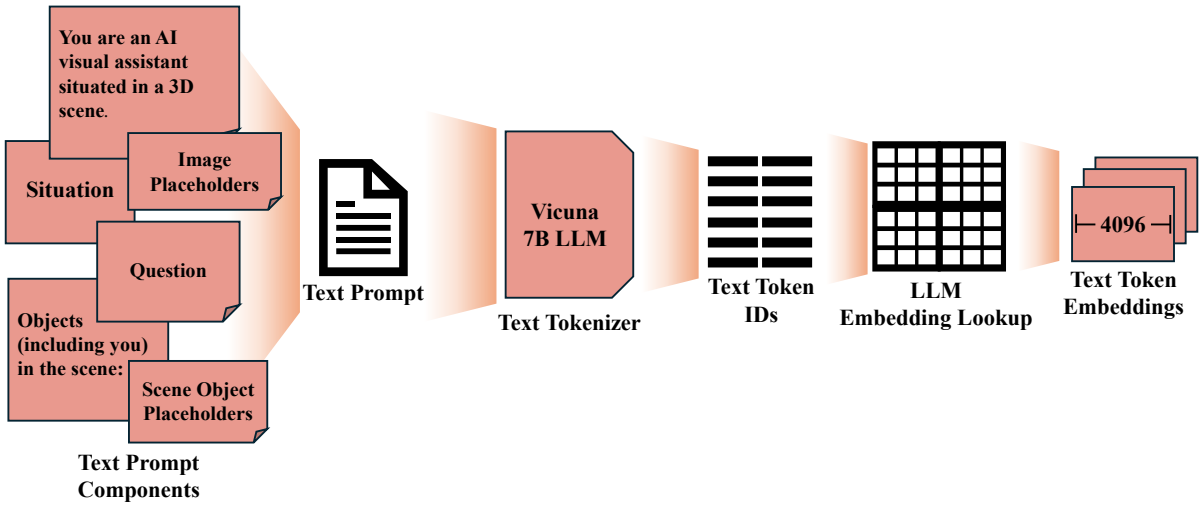

Figure 3.3: Text processing module, where the input prompt is tokenized and encoded using a language model to produce token embeddings in a shared 4096-dimensional space, including special placeholder tokens representing 2D image and 3D object inputs.

the agent's own location and orientation, as shown in Figures 3.5 and 3.8. These heterogeneous inputs are processed by the OSE3D Situation Encoder, which jointly encodes object-centric and agent centric context. The encoder produces object-level situation aware embeddings, capturing both semantic and spatial relationships within the scene. This final embedding serves as a global, situation aware descriptor that encodes the state of the environment and the agent within it.

3.3.5.1 LLM

The LLM module is responsible for generating the final textual output based on the encoded prompt representations from the Text Encoder, 2D backbone and Situation Encoder projected onto a higher dimensionality space by Linear Layers. These representations are first combined into a unified embedding after the multimodal fusion, which serves as the input to the Vicuna-7B language model [65]. The LLM processes the combined embeddings and generates the final answer to the input question. Finally, the output embedding is decoded into text and presented as the model's response as shown in Figure 3.4.

3.4 Scene Encoding

3.4.1 PointNet++

In order to support multimodal reasoning within the MSR3D framework, raw 3D scene data must be transformed into a structured representation that can be consistently processed across

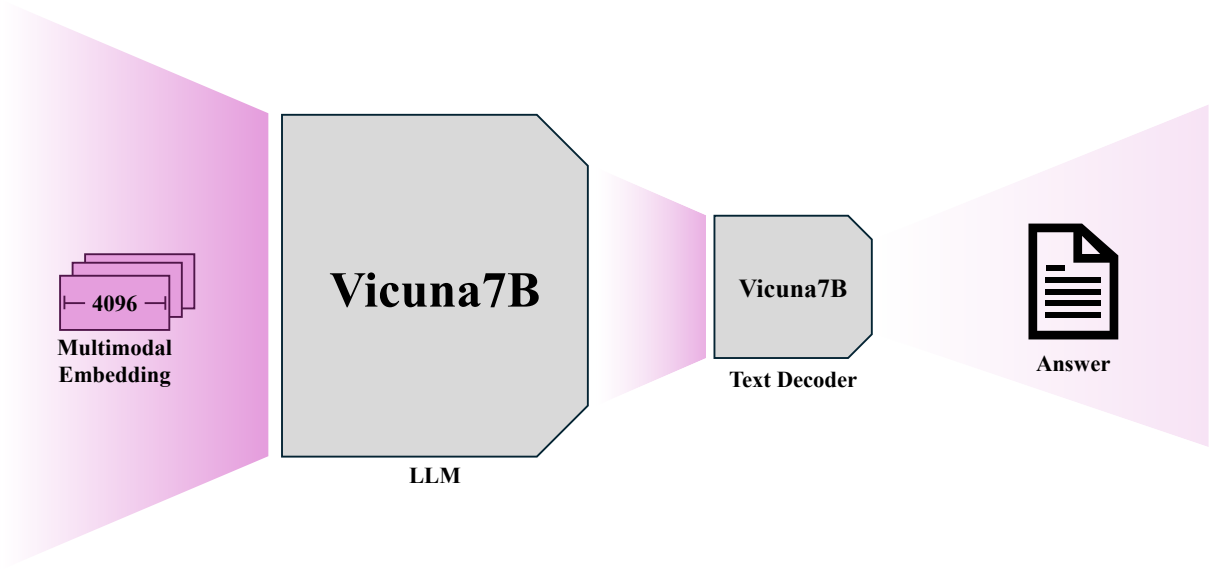

Figure 3.4: The 4096-dimensional multimodal embedding is passed into the Vicuna7B LLM, transformed into text-token representations, and decoded by a text de-tokenizer to generate the final answer.

the different datasets. While the datasets considered in this work vary in format and annotation structure, they are unified through a common preprocessing pipeline that converts each scene into an object-centric representation.

Each input scene is initially defined as a point cloud, where individual points encode spatial coordinates along with associated attributes such as color,instance and semantic labels, and normal. Rather than processing the scene as an unstructured collection of points, the pipeline organizes this information at the object-level as shown in Figure 3.5. Specifically, points are grouped according to instance annotations, allowing each object to be represented as a set of geometric and photometric features. This transformation enables the model to reason over semantically meaningful entities instead of isolated points.

For each object, a compact feature representation is constructed by aggregating point level information, including spatial location, extent, and appearance. In parallel, additional metadata such as instance labels and spatial positioning within the scene are preserved. This object-centric abstraction forms the basis for subsequent processing stages, where both geometric structure and semantic context are required.

To ensure consistency across datasets, a unified preprocessing procedure is applied. This includes normalization of spatial coordinates, alignment of feature representations, and standardization of annotation formats. As a result, scenes originating from different sources can be treated uniformly by the model, facilitating joint training and evaluation.

Given the variability in scene complexity, the number of objects per scene is constrained to a fixed upper limit. Scenes containing more objects are subsampled, while scenes with fewer

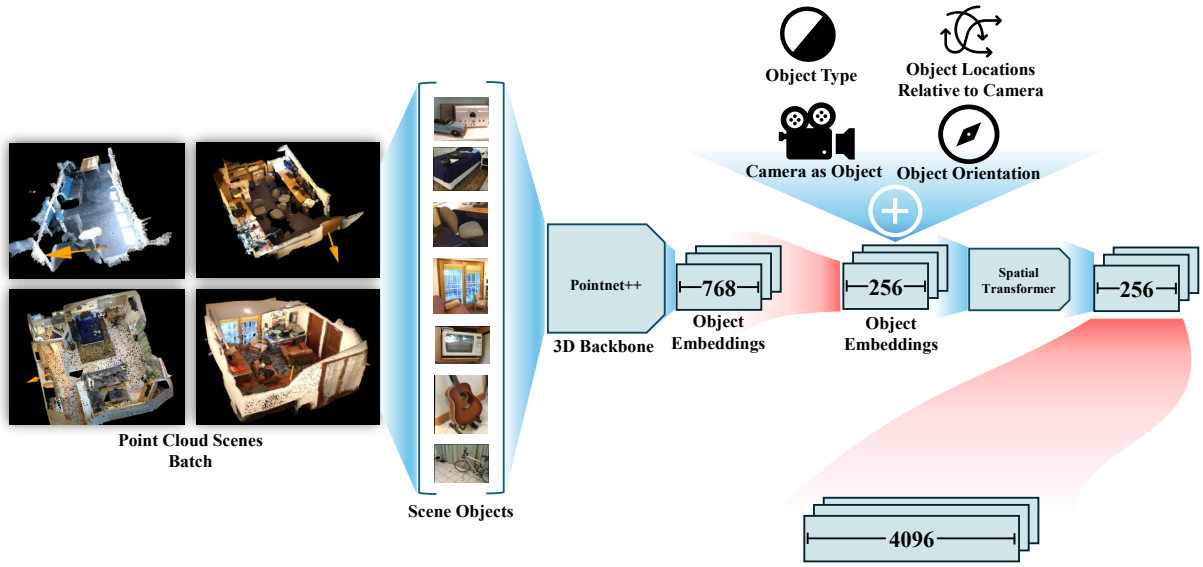

Figure 3.5: 3D backbone for scene encoding, where 3D scene object point clouds, are processed using a PointNet++ architecture to extract object features, which are enriched with spatial information and transformed through a spatial module before being projected into a shared 4096 dimensional embedding space.

objects are padded accordingly. This design ensures stable computational requirements while maintaining a representative subset of the scene.

The final output of this pipeline is a structured scene representation composed of object-level feature embeddings and associated spatial information. This representation serves as the input to the 3D backbone and subsequent multimodal fusion modules, enabling the model to integrate geometric, visual, and textual information within a unified framework.

3.4.1.1 Backbone Pretraining

The object-centric feature extraction in the baseline pipeline was initialized using a PointNet++ backbone [43], following established practices in 3D vision language grounding. Specifically, the implementation was aligned with the design adopted in VIL3DRef [26], where PointNet++ was employed to encode object-level point sets into compact and discriminative feature embeddings.

The motivation for using a pretrained encoder is that the model can benefit from geometric priors learned from previous 3D vision language grounding tasks. Since VIL3DRef also operates on object-level 3D representations, its PointNet++ encoder provides a suitable initialization for MSR3D. The pretrained weights allow the model to begin with features that already capture object shape, local surface structure, and point distribution patterns, instead of learning these representations entirely from scratch. After feature extraction, the object embedding is projected into the dimensionality required by the downstream MSR3D modules. The projected feature is

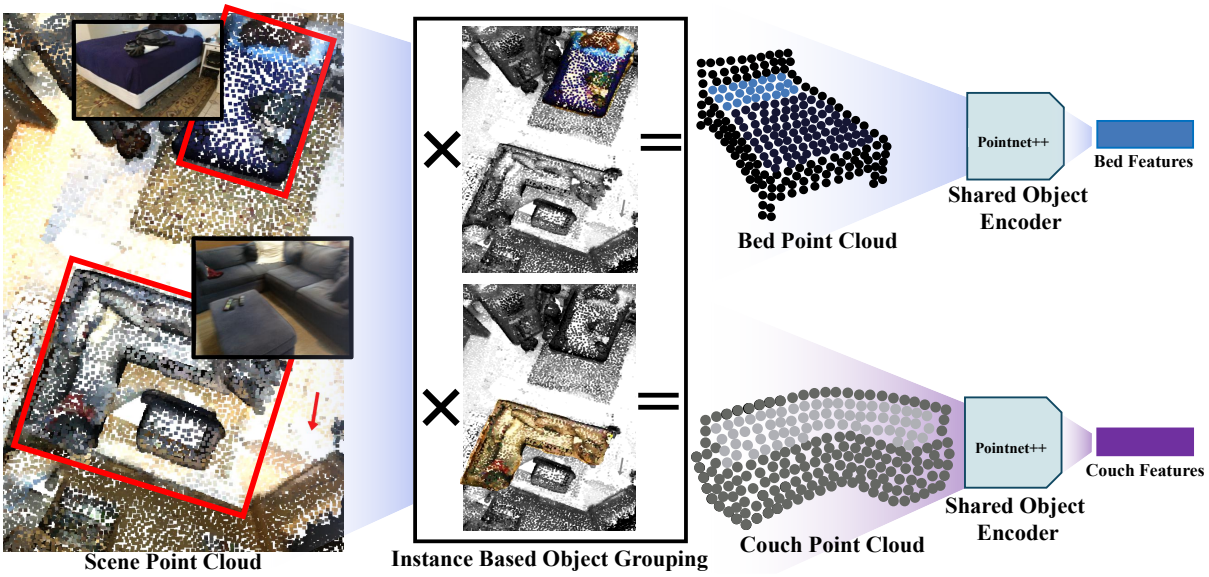

Figure 3.6: Baseline object-centric feature extraction pipeline using PointNet++. The scene point cloud is first decomposed into object-level point clouds through instance grouping. Each object is then independently encoded using a shared PointNet++ encoder to produce object-level feature embeddings. While this approach captures local geometric structure, it does not explicitly model interactions between objects during feature extraction.

then combined with spatial information and passed to the situation encoder. This enables the PointNet++ representation to be aligned with the multimodal embedding space used by the language and image components of MSR3D [34] . Compared to earlier point-based methods such as PointNet [42], the hierarchical design of PointNet++ allows local features to be progressively aggregated across successive abstraction layers.

Although VIL3DRef integrates both visual and textual modalities, only the geometric encoding component was utilized in this work. This ensured that object representations remained purely geometry driven at this stage, forming a stable and modality agnostic foundation for subsequent multimodal fusion.

3.4.1.2 Object-Centric Input Construction

In the original MSR3D scene encoding pipeline, the 3D scene is transformed from a raw point cloud into an object-centric representation before feature extraction as shown in FIgure 3.6. Let a scene point cloud be denoted as

$$\mathcal{P} = \{(p_i, c_i, s_i, m_i)\}_{i=1}^{N} \tag{3.1}$$

where $p_i \in \mathbb{R}^3$ represents the 3D coordinates of point i, $c_i \in \mathbb{R}^3$ denotes its RGB color, s_i is the semantic label, and m_i is the instance label. In the object-centric formulation, all points with

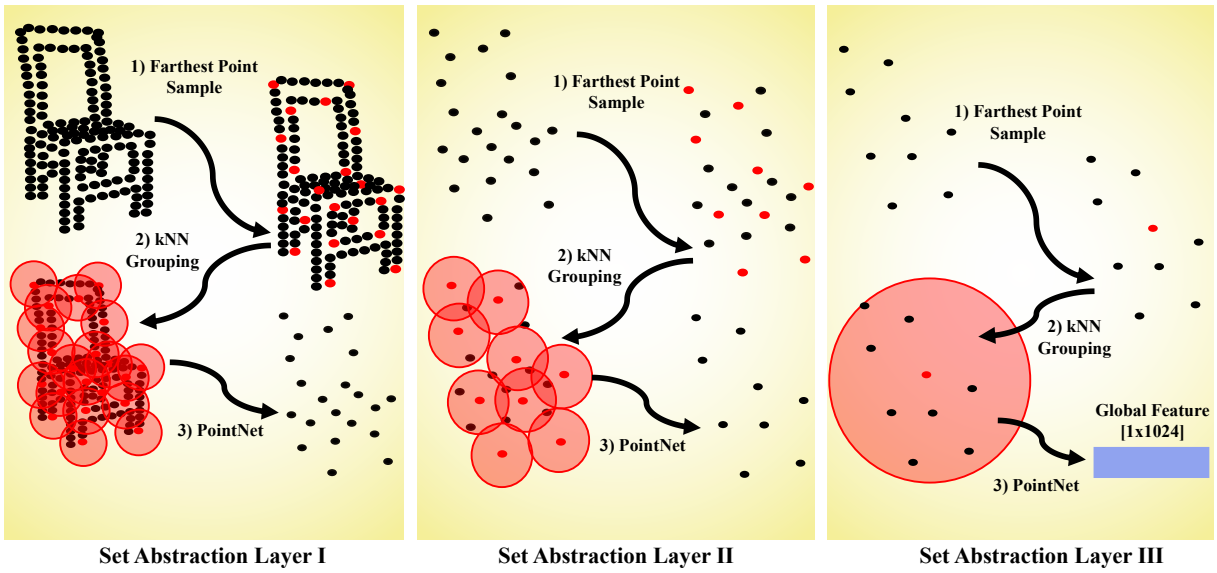

Figure 3.7: Set abstraction in PointNet++. Each layer performs Farthest Point Sampling (FPS), k-NN grouping, and PointNet (MLP + max pooling) to extract local features while reducing the number of points. Stacking layers increases the receptive field, and the final layer aggregates all points into a global feature vector (e.g., 1×1024), which can be used for classification or further processed for segmentation.

the same instance label are grouped into an object-level point set:

$$\mathcal{P}_o = \{(p_i, c_i) \mid m_i = o\} \tag{3.2}$$

Each object point set is then sampled or padded to a fixed number of 1024 points, in order to produce a consistent input tensor for the PointNet++ encoder. This step is necessary because different objects may contain significantly different numbers of points due to variations in object size, distance from the sensor, occlusion, and reconstruction quality. Thus, each scene is represented as a fixed-size tensor of shape $O \times P \times C$, where O is a fixed number of objects, P is the number of points per object and C is the feature vector of each point. This object-centric representation allows MSR3D to process scenes as collections of semantically meaningful entities rather than as unstructured point sets. However, because each object is encoded independently, the contextual relationship between neighboring objects is not directly modeled during the initial 3D feature extraction stage.

3.4.1.3 Set Abstraction Layers

PointNet++ extends PointNet by introducing a hierarchical feature extraction mechanism based on set abstraction layers [43]. Each set abstraction layer consists of three main operations: sampling, grouping, and local feature aggregation as shown in Figure 3.7. First, a subset of representative points is selected from the input point cloud using farthest point sampling. Given

an input set of points $\mathcal{P}$, farthest point sampling selects a set of centroids

$$\mathcal{C} = \{q_1, q_2, \ldots, q_M\} \tag{3.3}$$

where $M < N$. The objective is to choose centroids that cover the spatial extent of the point cloud as uniformly as possible. This reduces the number of points while preserving the overall geometric structure.

Second, for each centroid q_j, a local neighborhood is constructed using either ball query or k-nearest neighbor grouping. In ball query grouping, the neighborhood is defined as:

$$\mathcal{N}(q_j) = \{p_i \mid \|p_i - q_j\|_2 \leq r\} \tag{3.4}$$

where r is the neighborhood radius. This operation defines a local region around each centroid and allows the network to learn local geometric patterns. The k-nearest neighbor (kNN) grouping method computes the Euclidean distance between each point p_i and the centroid q_j, and selects the k points with the smallest distances to form the local neighborhood. The Euclidean distance is defined as:

$$d(p_i, q_j) = \|p_i - q_j\|_2 = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{3.5}$$

Third, a PointNet style function is applied to each local neighborhood. For each point $p_i \in \mathcal{N}(q_j)$, a shared multilayer perceptron is used to compute point-wise features:

$$g_i = \phi([p_i - q_j, f_i]) \tag{3.6}$$

where f_i is the input feature of point i, $p_i - q_j$ represents the relative position of the point with respect to the centroid, and ϕ denotes a shared MLP. The local feature of the centroid is then obtained through a symmetric aggregation function, typically max pooling:

$$h_j = \max_{p_i \in \mathcal{N}(q_j)} g_i \tag{3.7}$$

This operation produces a new set of points with learned local features. By stacking several set abstraction layers, PointNet++ progressively increases the receptive field and learns features from local geometric structures to higher-level object representations.

3.4.1.4 Local Feature Learning

In the ViL3DRef implementation, the PointNet++ backbone uses single-scale set abstraction layers to learn local geometric features from object point clouds. Although the original PointNet++ framework supports density-adaptive variants such as multi-scale grouping (MSG), which extracts features from multiple neighborhood radii around each centroid, and multi-resolution grouping (MRG), which combines features across different abstraction resolutions, ViL3DRef uses the standard set abstraction module. In this setting, each abstraction layer applies only one neighborhood scale at a time.

For each sampled centroid q_j, a local neighborhood is constructed using a single radius r and a fixed maximum number of sampled neighboring points. The points within each neighborhood are then processed using a PointNet-style local encoder. This encoder applies shared multi-layer perceptrons to the point features and aggregates them into a local feature representation, commonly using a symmetric max-pooling operation:

$$h_j = \max_{p_i \in \mathcal{N}_r(q_j)} \phi(p_i - q_j, f_i) \tag{3.8}$$

where ϕ denotes the shared point-wise feature transformation and f_i represents the input feature associated with point p_i. The relative coordinate $p_i - q_j$ allows the encoder to learn the local geometry of the neighborhood with respect to the centroid.

Unlike multi-scale grouping, where several neighborhoods with different radii are constructed around the same centroid and their features are concatenated, single-scale grouping extracts one local feature per centroid at each abstraction level. However, the overall PointNet++ hierarchy still captures increasingly larger spatial context across successive abstraction layers, since later layers operate on fewer centroids with larger receptive fields.

Therefore, in ViL3DRef, local object geometry is learned through hierarchical single-scale abstraction rather than explicit multi-scale grouping. This design allows the model to encode object-level point cloud features efficiently while still preserving the hierarchical structure of PointNet++.

3.4.1.5 Limitations of Object-Level PointNet++ Encoding

Although PointNet++ provides strong object-level geometric features, its use in the baseline MSR3D pipeline has an important limitation. Since objects are segmented and encoded independently, the 3D backbone does not directly observe the full scene during feature extraction. The feature of object is computed only from its own point set without explicitly considering neighboring objects. As a result, inter object spatial context is introduced only after object features have already been extracted, mainly through the situation encoder and multimodal fusion

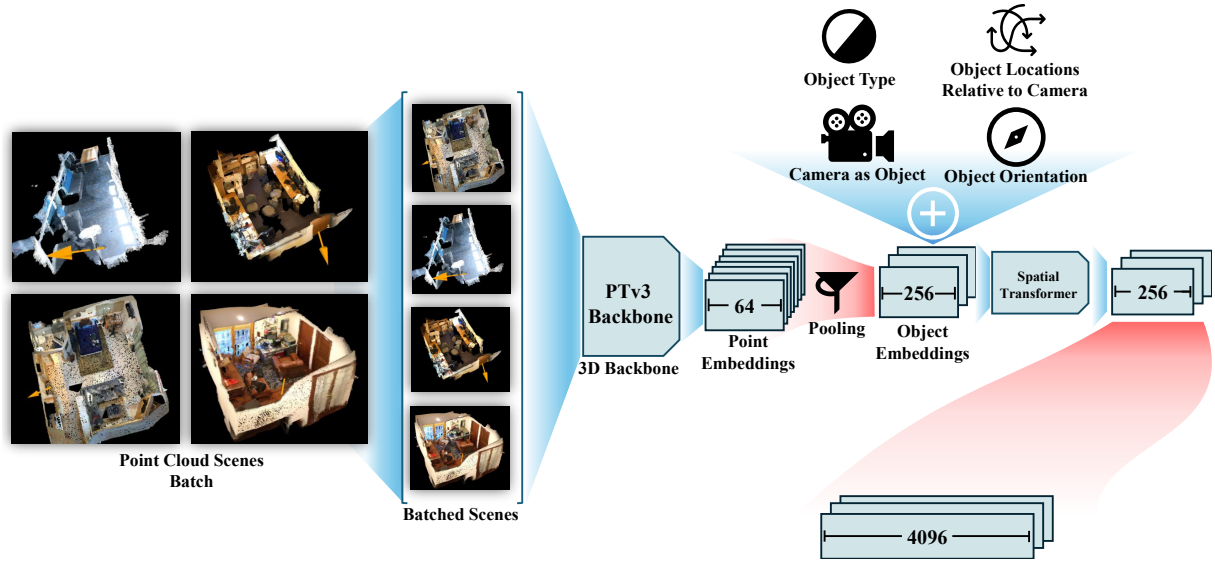

Figure 3.8: 3D backbone based on a PTv3 architecture, where batched point cloud scenes are encoded into point features, aggregated into object embeddings, and enriched with spatial information before being projected into a shared 4096 dimensional embedding space.

modules. This can be restrictive for questions that depend on relationships between objects, visibility, room layout, or navigation [43, 34].

For example, whether an object is close to the agent, partially occluded by another object, or located in a specific functional region of the room may not be fully captured by an object feature extracted in isolation. This limitation motivates the use of PTv3 as an alternative backbone, since PTv3 processes the entire scene before aggregating features into object-level embeddings [57].

3.4.2 PTv3

While the baseline pipeline operates on object-centric representations, where individual objects are segmented and processed independently, this work introduces a scene-level processing paradigm designed to better exploit global spatial context. The proposed pipeline leverages the PTv3 backbone to encode entire 3D scenes prior to object-level aggregation [57].

In contrast to the baseline, which isolates objects early in the pipeline, the proposed approach retains the full scene structure during feature extraction as shown in Figure 3.8. Each scene is represented as a unified point cloud tensor containing spatial coordinates, color information, and surface normals to the 3D backbone. This holistic representation allows the model to capture contextual relationships between objects, such as spatial proximity and layout dependencies, which are otherwise lost in object-centric processing. The processing pipeline consists of three high-level stages:

- **Scene Construction and Formatting:** Raw scene data is converted into a structured

format compatible with the PTv3 backbone, where each point is associated with geometric and semantic attributes. Instance identifiers are preserved to enable later aggregation into object-level representations.

- **Scene-Level Feature Encoding:** The entire point cloud is processed using the PTv3 backbone, which applies hierarchical attention mechanisms over the spatial domain. Data augmentations and spatial transformations are applied consistently at the scene-level to improve generalization. This stage produces dense point-wise feature embeddings that encode both local geometry and global context.
- **Object-Level Feature Aggregation:** Following scene encoding, object representations are derived by grouping points according to their instance labels and aggregating their features. A pooling operation is applied to the point feature representations, to obtain a fixed dimensional embedding per object. This preserves compatibility with downstream components that expect object-level inputs, while benefiting from context aware feature extraction.

To ensure computational consistency across scenes with varying object counts, a fixed maximum number of objects per scene is maintained, with padding applied where necessary. Additionally, binary masks are used to distinguish valid object embeddings from padded entries.

This design enables a seamless transition from scene-level understanding to object-level reasoning. By deferring object aggregation until after feature extraction, the proposed pipeline captures richer semantic and spatial information compared to the baseline object-centric approach.

Furthermore, the integration of PTv3 allows the model to operate directly on large-scale point clouds using efficient sparse convolution and attention mechanisms, improving both scalability and representational capacity. The resulting object embeddings are therefore conditioned not only on the object itself, but also on its surrounding environment.

Overall, the proposed pipeline enhances the representational power of the system by bridging the gap between local object features and global scene context, leading to more informative and robust 3D representations.

3.4.2.1 Voxelization and Grid Coordinate Mapping

Before serialization, PTv3 maps continuous 3D coordinates into discrete grid coordinates through voxelization [57]. Given a point coordinate $p_i = (x_i, y_i, z_i)$, the corresponding voxel coordinate is computed as

$$v_i = \left\lfloor \frac{p_i - p_{\text{min}}}{s} \right\rfloor \tag{3.9}$$

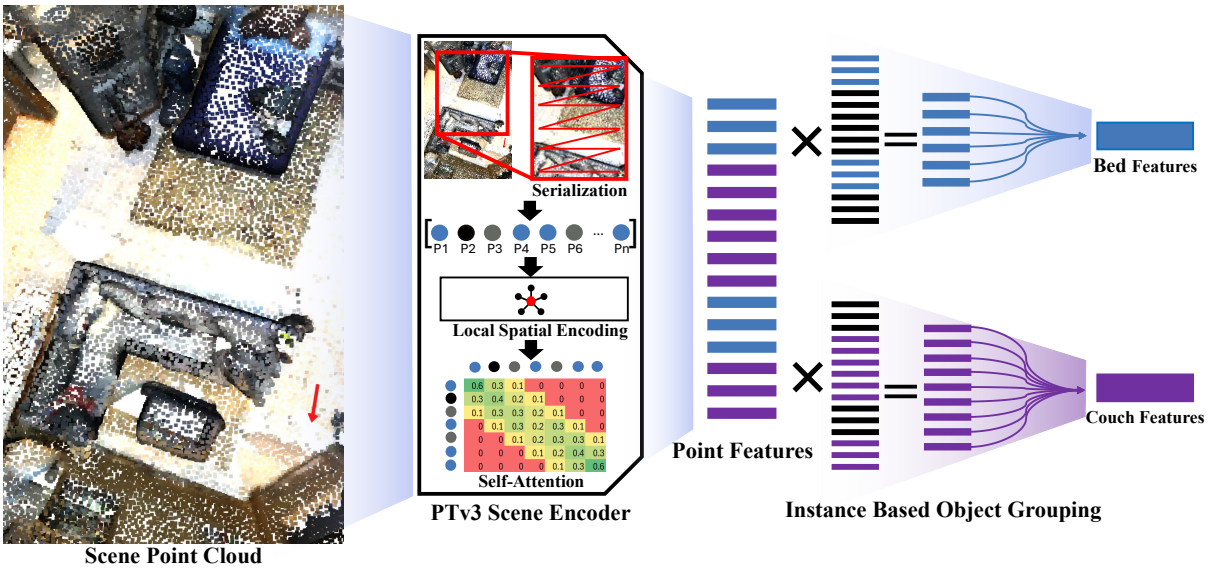

Figure 3.9: Proposed PTv3-based scene encoding pipeline. The entire scene point cloud is first processed by a Point Transformer V3 (PTv3) encoder, where points are serialized and enriched through local spatial encoding and self attention mechanisms. This enables the model to capture both local and long-range spatial relationships across the scene. The resulting point features are then grouped into object representations, producing context aware embeddings for each object.

where p_{min} is the minimum coordinate of the scene and s is the voxel size. The resulting voxel coordinate

$$v_i = (v_i^x, v_i^y, v_i^z) \tag{3.10}$$

places the point into a discrete 3D grid.

Voxelization serves two purposes. First, it provides a structured spatial representation for an otherwise unordered point cloud. Second, it enables efficient serialization by assigning each point a discrete grid location that can be mapped to a 1D index. Unlike dense voxel based methods, PTv3 does not need to construct a full dense voxel grid. Instead, it uses sparse grid coordinates only for the occupied points, preserving the efficiency of point based processing while introducing enough structure for scalable attention.

3.4.2.2 Point Cloud Serialization

PTv3 processes point clouds by converting the unordered set of points into an ordered sequence through point cloud serialization [57]. Given an input point cloud

$$\mathcal{P} = \{(p_i, f_i)\}_{i=1}^{N} \tag{3.11}$$

where $p_i \in \mathbb{R}^3$ denotes the 3D coordinates of point i, and f_i denotes its associated features, such as color and surface normals. After voxelization, each point is assigned to a voxel according to its spatial position, therefore the voxels are then ordered using a space filling curve, such as a Z-order or Hilbert curve.

This serialization step transforms the point cloud into a 1D ordered sequence while approximately preserving spatial locality. In other words, points that are close in 3D space tend to remain close in the serialized sequence. This allows PTv3 to avoid expensive explicit neighborhood construction, such as repeated kNN grouping, and instead apply attention over local regions of the serialized sequence [57].

The importance of this step is that the model can process large-scale point clouds more efficiently while still maintaining a meaningful spatial structure. For indoor scenes, where objects are arranged according to room layout and functional relationships, preserving this locality is essential for maintaining scene context before object-level aggregation.

3.4.2.3 Spatial Encoding and Point Attention

Although serialization imposes an order on the point cloud, the model still retains the original 3D coordinates and point attributes. In this work, each point was represented using spatial coordinates, color information, and surface normals, apart from its instance and segmentation labels:

$$x_i = [p_i, c_i, n_i] \tag{3.12}$$

where p_i denotes the point coordinates, c_i denotes color features, and n_i denotes the estimated normal vector.

PTv3 applies attention to point features using scalar dot-product attention over serialized point patches [57]. In this formulation, each point token updates its representation by computing scalar attention scores with nearby tokens in the serialized sequence and aggregating their transformed features. This design is computationally efficient because it follows the standard Transformer attention operation, making it compatible with optimized implementations such as FlashAttention. The attention update can be written as:

$$\mathbf{y}_i = \sum_{\mathbf{x}_j \in \mathcal{X}} \rho\left(\varphi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) + \delta\right) \alpha(\mathbf{x}_j) \tag{3.13}$$

where $\mathbf{y}_i$ is the updated feature of point i, $\mathcal{X}$ denotes the set of tokens within the local serialized window, φ and ψ are learnable query and key projections, α is a learnable value projection, d is the per-head feature dimension, δ_{ij} is a learnable positional bias conditioned on the relative po-

sition between points i and j, and ρ denotes softmax normalization applied over all $j \in \mathcal{X}$. This operation is applied independently across multiple attention heads, with outputs concatenated and projected to produce $\mathbf{y}_i$. In contrast to purely local geometric aggregation, attention allows the model to assign different importance to different neighboring points. This is useful in cluttered indoor scenes, where nearby points may belong to different objects or semantic regions. By combining geometric features with learned attention weights, PTv3 can produce point-wise embeddings that capture both local structure and broader contextual information [64, 58, 57].

3.4.2.4 Long Range Dependencies and Scene Context

A key advantage of PTv3 is that scene encoding is performed before object-level aggregation. Instead of extracting features from each object independently, the full point cloud is processed as a single scene. This allows information to flow between points belonging to different objects, preserving contextual relationships such as spatial proximity, room layout, and object co-occurrence.

PTv3 achieves this efficiently by applying attention within serialized patches. The serialized sequence is divided into local groups, and attention is computed within each group rather than across the entire point cloud. This reduces the computational cost compared to full global attention, while still allowing the receptive field to expand across multiple layers.

As features pass through successive PTv3 blocks, each point representation becomes influenced by increasingly larger regions of the scene. Therefore, even if two distant objects are not directly connected in a single attention operation, their information can still interact through intermediate points and deeper network layers. This enables the model to capture long range dependencies without requiring prohibitively expensive global attention.

This property is particularly important for MSR3D, since many questions require reasoning about relationships between multiple objects and the agent's position in the environment. For example, a query may depend on whether an object is visible from a certain viewpoint, whether one object is closer than another, or how objects are arranged within a room. By encoding the scene before aggregating object features, PTv3 produces object embeddings that are conditioned not only on the geometry of each object, but also on the surrounding spatial context.

3.4.2.5 Object-Level Aggregation after Scene Encoding

After PTv3 generates dense point-wise features, object-level representations are recovered using the instance annotations preserved during preprocessing. Let p_j denote the set of points belonging to object O_i. The object embedding can be obtained by pooling the features of all points assigned to that object, based on their instance ids l_i. In this work, this aggregation step allowed the PTv3 backbone to remain compatible with the MSR3D architecture, which expects

object-level embeddings as input to the situation encoder and multimodal fusion modules.

$$\mathcal{O}_i = \{\, p_j \in \mathcal{P} \mid l_j = i \,\} \tag{3.14}$$

The main difference from the original PointNet++ pipeline is that object aggregation was delayed until after scene-level encoding. As a result, each object representation incorporated information from the surrounding scene, rather than being computed only from the isolated points of the object. This design preserved the object-centric interface required by MSR3D while benefiting from the richer scene-level context captured by PTv3.

3.5 Interactive multimodal User Interface

An interactive visualization interface was developed to support qualitative analysis and model question answering within 3D scenes. The system is implemented using the Gradio framework [19] for rapid interface construction and Plotly [41] for rendering large-scale point clouds in a browser based environment. This design enables real time exploration of multimodal data, bridging geometric representations, semantic annotations, and natural language interaction.

The interface supports multiple datasets, including ScanNet [13], ARKitScenes [53], and 3RScan [52], which are used to train the model, each loaded using a unified abstraction that mirrors the preprocessing pipeline used during training. Scene data is represented using a structured feature tensor, combining spatial coordinates, normalized RGB values, and surface normals into a single representation. This aligns with established practices in 3D deep learning, where point-wise features are concatenated to capture both geometric and photometric information [20]. The interface also preserves instance-level and segment level annotations, enabling fine grained inspection of object-level structure within each scene.

To ensure efficient interaction with large-scale point clouds, the system employs a multi level caching strategy. Raw scene data is loaded once per scan and stored in memory, while downsampled representations are cached based on user defined parameters such as point count. Additional caches store precomputed color mappings and bounding box geometries, reducing redundant computation during repeated visualization updates. This design minimizes expensive disk I/O operations and avoids repeated preprocessing steps, allowing the interface to maintain responsiveness even when visualizing scenes containing hundreds of thousands of points. Similar strategies are commonly adopted in large-scale visualization systems to balance fidelity and interactivity [1].

The visualization component provides multiple rendering modes to facilitate analysis of different aspects of the scene. In addition to standard RGB coloring, the interface supports instance based and segment based color encodings, enabling users to distinguish object boundaries and seman-

tic regions. A dedicated normal-based visualization mode maps surface normals to color space, providing insight into local geometric structure. Surface normals are optionally re-oriented toward a specified viewpoint to improve interpretability, a technique frequently used in computer graphics and geometry processing [6]. Furthermore, the system includes the ability to render normal glyphs as line segments, offering an explicit representation of local surface orientation.

To enhance spatial understanding, several geometric overlays are integrated into the interface. Axis aligned bounding boxes are computed per instance and rendered as wireframe structures, allowing users to assess object extents and spatial relationships. A global coordinate frame can be visualized to provide orientation context, while a directional arrow represents the agent's position and viewing direction within the scene. The orientation is derived from quaternion or vector representations and converted into a normalized direction vector, enabling consistent visualization across datasets with differing pose formats. These overlays provide essential context for interpreting both the scene geometry and the model's reasoning process.

The interface also incorporates a multimodal interaction component, allowing users to query the underlying model using natural language and optional image inputs. Queries are processed through an interactive service layer built on top of the MSR3D framework [34], which dynamically adapts to the selected dataset and scene. Users may override the default scene context by specifying custom spatial anchors or uploading reference images, enabling controlled experimentation with different viewpoints and conditions. This functionality reflects broader trends in multimodal learning, where joint reasoning over vision, language, and spatial data is increasingly emphasized [49].

From a systems perspective, the interface is designed to decouple heavy computation from lightweight interaction. Expensive operations such as scene loading, feature extraction, and downsampling are performed only when necessary, while most user interactions, such as changing visualization modes or toggling overlays, operate directly on cached data. This separation ensures that the interface remains responsive and scalable, even when deployed on resource constrained environments. Overall, the system provides a flexible and extensible platform for analyzing 3D scene representations and evaluating multimodal reasoning behavior in a controlled and interactive setting as exaplained in Appendix I.

4 Experiments

This chapter focuses on the experimental evaluation of the proposed PTv3 integration within the MSR3D framework. The experiments are designed to assess how different 3D backbone configurations affect multimodal reasoning performance across indoor 3D scene datasets. The evaluation considers the MSQA dataset, the training setup, hardware and model configurations, task-specific evaluation metrics, qualitative examples, and quantitative comparisons between PointNet++ and PTv3 under both ScanNet-only and multi-dataset settings.

4.1 Experimental Setup

4.1.1 Dataset

This method utilizes the MSQA dataset, which comprises 3D scenes sourced from ScanNet [13], ARKitScenes [53], and 3RScan [52]. Each training and validation sample in the dataset contains the scene identifier of the corresponding 3D environment, a situation description and natural language question, the ground truth answer, the question category, and the spatial context of the agent, including its location and orientation within the scene. In addition, each sample includes a raw thought annotation, which specifies the key object instance identifiers required to answer the question correctly. This structure allows a single 3D scene to generate multiple distinct samples by varying the agent's situation and the associated query, enabling the model to reason about the same environment from different spatial perspectives and contextual conditions.

In addition to the question answer samples provided for each dataset split, the benchmark includes a large collection of metadata files used for scene reconstruction, object-level reasoning, and multimodal grounding. These files provide semantic mappings, spatial annotations, and object specific attributes required during preprocessing and model inference. For example, the dataset includes mappings between semantic identifiers and natural language labels, as well as a complete list of the 607 object categories present across the scenes. Furthermore, the train and validation scene identifiers are provided in predefined split files, including sorted versions that facilitate repeated access during training and evaluation, whereas the test split is stored only once since it is used exclusively for final evaluation.

To support object-centric scene representation, several instance-level annotation files are provided for each scene:

- `instance_id_to_name`: Contains the semantic class names of all object instances within a scene.

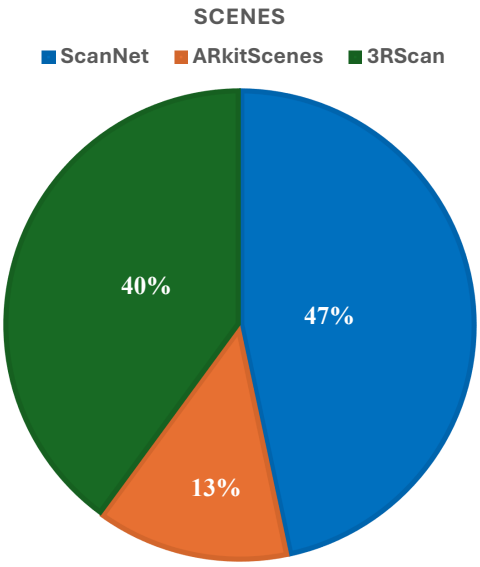

Figure 4.1: MSQA Dataset Scene Counts. A single scene can yield multiple samples by varying the situation and the corresponding question.

- `instance_id_to_label`: Associates each object instance identifier with its corresponding semantic category label.
- `instance_id_to_loc`: Stores the spatial properties of each object, including its location and orientation, represented using six values.
- `instance_id_to_gmm_color`: Provides an approximate representation of object appearance using a three component GMM. Instead of storing dense RGB distributions for all object points, each object is represented through mixture weights and mean RGB vectors. The weights indicate the relative prevalence of each dominant color component, while the mean vectors correspond to the principal RGB color distributions of the object.

The dataset also provides globally aligned scene point clouds through tensors in serialized PyTorch data files. These contain four aligned arrays describing the scene geometry and semantics: (1) the XYZ coordinates of each point, (2) the RGB color values of each point, (3) the object instance identifier to which each point belongs, and (4) the semantic class label associated with each point, where the value `-100` denotes ignored labels. Finally, additional files containing precomputed surface normal vectors for every point in the scene point cloud are included through the `pcd_normals` annotations. The computation and usage of these normals within the proposed pipeline are discussed in detail in Appendix B.

Each scene in the dataset is additionally accompanied by a collection of object RGB images corresponding to the individual object instances present within the 3D environment. This provides complementary multimodal information that enables the model to jointly reason over geometric structure and visual semantics.

4.1.2 Implementation Details

Each scene in the dataset is additionally accompanied by a collection of object-centric RGB images corresponding to the individual object instances present within the environment. These images are generated by extracting cropped visual regions around annotated objects from the original RGB observations of the scene. As a result, each object instance can be represented not only through its geometric point cloud representation, but also through its associated visual appearance in image space. This provides complementary multimodal information that enables the model to jointly reason over geometric structure and visual semantics. The object images vary in scale, viewpoint, and visual complexity depending on the position and visibility of the corresponding object within the scene.

The experimental pipeline was implemented using the MSR3D framework and trained using the `LeoTrainer` [25]. All experiments used a fixed random seed of 42 in order to improve reproducibility. The training configuration used a single GPU, and a per-GPU batch size of 2. Gradient accumulation was applied over 5 steps, resulting in a larger effective batch size while keeping memory usage manageable. The models were trained for 10 epochs, with evaluation performed after every epoch.

Optimization was performed using the AdamW optimizer with a learning rate of 3×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.05. A warmup cosine scheduler was used, with 400 warm up steps. Gradient norm clipping was applied with a maximum norm of 5.0. These settings were kept consistent across the main experimental runs in order to ensure a fair comparison between the PointNet++ and PTv3 object encoders [43][57].

The input scenes were processed using an object-centric representation. Each scene was padded or truncated to a maximum of 60 objects, and each object point cloud was sampled to 1024 points. The model used the MSQA annotations from ScanNet [13], 3RScan [52], and ARKitScenes [53], depending on the corresponding experimental configuration. For training and validation, the standard training and validation transformations were used, depending on which 3D backbone was used during that experiment.

The multimodal architecture consisted of a 2D vision encoder, a 3D object encoder, a situation encoder, and a large language model, as explained in the previous chapters. The 2D visual features were extracted using a BLIP-2 backbone based on `Salesforce/blip2-opt-2.7b` [32] . The language model was Vicuna-7B, loaded from the `huangjy-pku/vicuna-7b` [65] configuration. Low Rank Adaptation (LoRA) [24] was enabled for the language model with rank 16, alpha 16, and dropout 0.0. LoRA was applied to the query, key, value, output, gate, up, and down projection layers, namely `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`.

Across the experiments, different combinations of trainable modules were evaluated. In the main configurations, the image encoder and situation encoder were tuned, while additional runs also tuned the object encoder. This allowed the experiments to compare whether adapting the 3D object representation jointly with the rest of the multimodal pipeline improved performance. The same general training procedure was used for both the PointNet++ baseline and the PTv3-based variant, so that performance differences could be attributed primarily to the choice of 3D backbone [43][57] .

In addition to model accuracy, the computational cost of each configuration was recorded in terms of training time and inference time. In the ScanNet-only setting, the PointNet++ model required approximately 1 day and 15 hours to train for 10 epochs, whereas the PTv3 model required approximately 3 days and 11 hours for the same number of epochs. During inference, the PointNet++ model processed 414 validation examples in 5 minutes, corresponding to approximately 0.72 seconds per sample, while the PTv3 model processed the same 414 examples in 7 minutes and 40 seconds, corresponding to approximately 1.11 seconds per sample. In the multi-dataset setting, the PointNet++ model required approximately 2 days and 12 hours to train for 10 epochs, while the PTv3 model required approximately 8 days and 20 hours. For inference across 703 examples from all datasets, PointNet++ required 8 minutes, corresponding to approximately 0.68 seconds per sample. The PTv3 model required approximately 20 minutes and 40 seconds, corresponding to approximately 1.76 seconds per sample. These results show that although PTv3 introduces a substantially higher computational cost during both training and inference compared to PointNet++.

4.1.3 GPU and Transformer Specifications

All experiments were executed on the Prometheus high performance computing cluster provided by CYENS Centre of Excellence. Prometheus provides NVIDIA enterprise GPUs, AMD EPYC processors, Lustre based parallel storage, and Slurm based job scheduling [12]. The experiments in this thesis were run on a single NVIDIA A6000 Ada GPU.

The PTv3 based experiments used a PTv3 backbone with the `PT-v3m1` configuration [57]. The encoder received 6 input channels, corresponding to geometric and appearance related point features. The point cloud was voxelized using a grid size of 0.02 before being processed by the PTv3 encoder. The encoder used five hierarchical stages with depths $(3, 3, 3, 6, 3)$, channel dimensions $(48, 96, 192, 384, 512)$, and attention head counts $(3, 6, 12, 24, 32)$. The decoder used four stages with depths $(3, 3, 3, 3)$, channel dimensions $(64, 96, 192, 384)$, and attention head counts $(4, 6, 12, 24)$. The patch size was set to 1024 for all encoder and decoder stages.

The PTv3 backbone used an MLP ratio of 4, query-key-value bias, pre-normalization, and a drop path rate of 0.3. The model used multiple point serialization orders, namely `z`, `z-trans`,

`hilbert`, and `hilbert-trans`, with order shuffling enabled. The pretrained PTv3 weights were loaded from a model trained using the PPT configuration, and the resulting object features were reduced using max pooling before being passed to the MSR3D situation encoder.

The situation encoder operated over a maximum scene token length of 60 and used a hidden size of 256. Spatial attention was enabled, together with the embodied anchor token and orientation features. The spatial encoder consisted of 3 transformer layers with 8 attention heads, a feed forward dimension of 2048, dropout of 0.1, and GELU activation. Pairwise spatial relations were computed using object centers, and spatial distance normalization was enabled. This module transformed the object-level 3D features into situation aware scene representations before fusion with the image and language modalities.

The language component used Vicuna-7B [65] with a maximum output length of 256 tokens and a maximum context length of 256 tokens. Rather than fully fine-tuning all language model parameters, LoRA was used to adapt selected projection layers efficiently [24]. This reduced the number of trainable parameters while still allowing the model to adapt its language generation and reasoning behavior to the multimodal 3D question answering task.

4.2 Evaluation Protocol

For a fair and consistent comparison across experiments, the following evaluation metrics are adopted. Each metric is designed to calculate the accuracy of the model in answering different types of questions. The metrics include:

- **Counting**: The ability to accurately count objects within the 3D scene.
- **Existence**: Question that require the model to determine the presence or absence of specific objects.
- **Attribute Description**: Performance is assessed under two complementary criteria: Direct attribute comparison and semantic consistency of the corresponding description.
- **Spatial Relation**: The metric accounts for two evaluation categories: The geometric relationship between objects and the relative to the agent position of objects within the scene.
- **Navigation**: The model's ability to navigate in the scene and identify locations based on spatial cues.
- **Others**: This category encompasses two question types, the room type and affordance questions that requires the model to identify the type of room depicted in the scene and the possible interactions with objects respectively.

The evaluation is performed on three test splits, corresponding to ScanNet, 3RScan, and ARKitScenes. The ScanNet test split contains 827 question-answer pairs, distributed as 133 counting,

96 existence, 116 attribute, 25 description, 190 spatial relationship, 49 refer, 144 navigation, 43 affordance, and 31 room type questions. The 3RScan test split contains 315 question-answer pairs, distributed as 58 counting, 47 existence, 30 attribute, 15 description, 66 spatial relationship, 38 refer, 35 navigation, 25 affordance, and 1 room type question. Finally, the ARKitScenes test split contains 262 question-answer pairs, distributed as 63 counting, 50 existence, 29 attribute, 9 description, 31 spatial relationship, 15 refer, 47 navigation, 17 affordance, and 1 room type question. During evaluation, these fine-grained annotations are mapped into six reporting categories. Specifically, *attribute* and *description* questions are merged into the **Attribute Description** category, *spatial relationship* and *refer* questions are merged into the **Spatial Relation** category, and *affordance* and *room type* questions are grouped under **Others**. The remaining categories, namely **Counting**, **Existence**, and **Navigation**, are reported directly. This grouping follows the evaluation script, where metrics are first computed per dataset and per question type, and are then aggregated using the number of samples in each category as weights as shown in Figure 4.2. For each dataset, the evaluation is first performed at the level of the original question types. Let $s_{d,t}$ denote the average score obtained by the model on question type t of dataset d, and let $n_{d,t}$ denote the corresponding number of question-answer pairs. When multiple original question types are merged into a single reporting category c, the category score is computed as a weighted average over all question types assigned to that category:

$$S_{d,c} = \frac{\sum_{t \in \mathcal{T}_c} n_{d,t} s_{d,t}}{\sum_{t \in \mathcal{T}_c} n_{d,t}} \tag{4.1}$$

where $\mathcal{T}_c$ is the set of original question types included in category c. For example, the **Attribute Description** score is computed by combining the scores of the *attribute* and *description* questions, weighted by the number of samples of each type. Similarly, the **Spatial Relation** score combines *spatial relationship* and *refer* questions, while **Others** combines *affordance* and *room type* questions.

The final overall score is then computed as the weighted average over the six reporting categories:

$$S_{\text{overall}} = \frac{\sum_{c \in \mathcal{C}} N_c S_c}{\sum_{c \in \mathcal{C}} N_c} \tag{4.2}$$

where $\mathcal{C}$ is the set of reporting categories and N_c is the total number of question–answer pairs assigned to category c. This ensures that each individual question contributes equally to the final score, while still allowing the results to be reported using broader semantic categories.

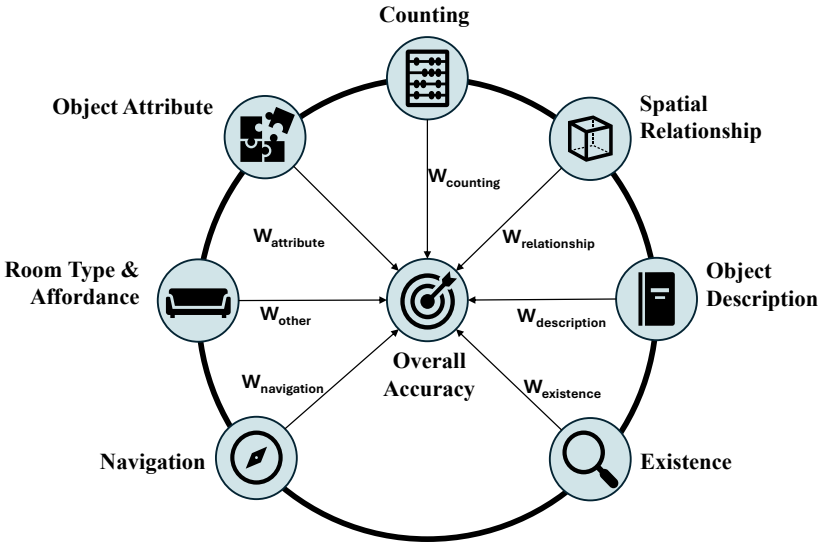

Figure 4.2: Evaluation metrics used to assess model performance across multiple task categories. Overall accuracy is the weighted sum of the accuracy of the model for each question type.

Table 4.1: PointNet++ baseline experiments with different training configurations on ScanNet.

Dataset	LoRA	Tuned Parameters	Counting	Existence	Attr.-Desc.	Spatial	Navigation	Overall
ScanNet	Disabled	Situation Encoder Only	0.30075	0.88542	0.26950	0.19247	0.17361	0.32769
ScanNet	Disabled	Object Encoder and Situation Encoder	0.27820	0.82292	0.27660	0.18410	0.16667	0.32164
ScanNet	Enabled	Object Encoder and Situation Encoder	0.37594	0.90625	0.34752	0.23013	0.18056	0.36276
ScanNet	Enabled	Object, Image and Situation Encoder	0.34586	0.91667	0.33333	0.24686	0.14583	0.36397
ScanNet	Enabled	Image and Situation Encoder	**0.30827**	**0.91667**	**0.31915**	**0.25105**	**0.22917**	**0.36638**

4.3 Experimental Results

This section presents the experimental results obtained from evaluating different 3D backbone and training configurations within the MSR3D framework. The experiments focus on three main comparisons. First, different experiments with PointNet++ as the 3D backbone in order to identify a suitable baseline model. Second, the proposed PTv3 3D backbone is compared against the matching PointNet++ configuration on ScanNet only. Finally, both models are evaluated in the multi dataset setting using ScanNet, 3RScan, and ARKitScenes [57, 43, 34, 13, 52, 53].

4.3.1 PointNet++ Baselines

The first set of experiments was conducted in order to determine the most appropriate PointNet++ baseline configuration for comparison. All experiments in this group use BLIP [32] as the 2D vision encoder, Vicuna-7B [65] as the language model, and PointNet++ [43] as the object encoder. The main variables are whether LoRA [24] is enabled and which modules are fine tuned during training.

Table 4.2: ScanNet-only comparison between the PointNet++ baseline and the proposed PTv3-based model.

Dataset	Object Encoder	Tuned Parameters	Counting	Existence	Attr.-Desc.	Spatial	Navigation	Overall
ScanNet	PointNet++	Image and Situation Encoder	0.30827	0.91667	0.31915	0.25105	0.22917	0.36638
ScanNet	PTv3	Image and Situation Encoder	**0.41353**	**0.89583**	**0.31206**	**0.26360**	**0.20139**	**0.37485**

As shown in Table 4.1, disabling LoRA consistently leads to weaker performance compared to the stronger LoRA enabled configurations. This suggests that adapting the language model through LoRA is important for preserving and improving the multimodal reasoning ability of the overall architecture [24]. Another important consideration in selecting the baseline configuration was determining which MSR3D encoders should remain frozen during training. Based on this criterion, the LoRA enabled PointNet++ configuration in which only the image and situation encoders were fine tuned was selected as the baseline for the subsequent comparisons.

The PointNet++ baseline also shows clear strengths in specific evaluation categories. Its strongest behavior is observed in existence questions, indicating that PointNet++ provides effective object-centric features for determining whether relevant entities are present in the scene. It also performs competitively on counting questions, which rely on the ability to encode object structure. This is consistent with the design of PointNet++, which hierarchically extracts local geometric features from point clouds and aggregates them into object representations [43]. However, the comparatively weaker performance on spatial and navigation questions suggests that PointNet++ is less effective at capturing broader scene relationships, motivating the investigation of transformer based point cloud encoders.

4.3.2 Experimental Result Comparison Between PointNet++ and PTv3

The next step in establishing the experimental baseline was to compare the proposed 3D architecture against the corresponding PointNet++ configuration by training and evaluating both models exclusively on ScanNet. This comparison was designed to examine whether replacing the PointNet++ object encoder with PTv3 improves performance when both models are evaluated within the same dataset domain.

Table 4.2 shows that the proposed model achieves slightly stronger overall performance than the corresponding PointNet++ configuration. Although the improvement is modest, it indicates that PTv3 can serve as an effective replacement for PointNet++ when the evaluation domain matches the data distribution on which the PTv3 backbone was trained. This result supports the motivation for using PTv3 as a stronger 3D backbone, since transformer based point cloud models are designed to capture richer contextual relationships through attention mechanisms [64, 57].

Their performance over the different evaluation metrics further clarifies the behavior of the two

Table 4.3: Full-dataset comparison between PointNet++ and PTv3 on ScanNet, 3RScan, and ARKitScenes.

Training Dataset	Object Encoder	Tuned Parameters	Counting	Existence	Attr.-Desc.	Spatial	Navigation	Overall
ScanNet, 3RScan, ARKitScenes	PointNet++	Image and Situation Encoder	**0.36614**	**0.89119**	**0.34821**	**0.24936**	**0.23894**	**0.39459**
ScanNet, 3RScan, ARKitScenes	PTv3	Image and Situation Encoder	0.34646	0.86528	0.33036	0.24679	0.21681	0.37678

models. PTv3 is stronger on metrics that require more explicit spatial and structural reasoning, especially counting, spatial reasoning, and navigation. These gains suggest that PTv3 is better able to encode contextual relationships across the scene, rather than relying only on object local geometric features. This is particularly relevant for situated reasoning tasks, where the model must reason about the environment relative to the question and situation context [34]. However, PointNet++ remains competitive, especially in existence based and attribute-description questions, showing that its object-centric representation remains a strong baseline for several question categories.

The final experiment evaluates the PointNet++ and PTv3 configurations in the full multi dataset setting, using ScanNet, 3RScan, and ARKitScenes. This setting is more challenging than the ScanNet experiment because it requires the model to generalize across different indoor scene datasets with different reconstruction properties, object distributions, and scanning conditions [13, 52, 53].

As shown in Table 4.3, the proposed model performs slightly worse than the matching PointNet++ configuration in the full dataset setting. This result contrasts with the ScanNet experiment, where PTv3 achieved a small advantage. A likely explanation is that the PTv3 backbone used in this work was trained only on ScanNet. As a result, while it can provide stronger representations when evaluated on ScanNet, its features may not transfer as effectively to 3RScan and ARKitScenes without additional training or domain adaptation.

This outcome suggests that PTv3 is a promising 3D backbone, but its effectiveness depends strongly on the alignment between the training domain and the target evaluation domain. In contrast, PointNet++ appears to provide more stable performance in the multi dataset setting, possibly because its object-centric hierarchical encoding is less specialized to a single dataset distribution. Therefore, while the ScanNet results demonstrate the potential of PTv3 for improving scene-level reasoning, the full dataset results indicate that further training on diverse datasets or additional adaptation strategies may be necessary for PTv3 to generalize robustly across heterogeneous 3D indoor scene benchmarks.

Overall, the experiments show that the proposed PTv3 integration can improve performance in a matched domain setting, particularly on categories that benefit from stronger spatial context modeling. However, the multi dataset results reveal that this advantage does not automatically transfer across datasets. This highlights an important limitation of the current implementation

and suggests that future work should investigate multi dataset PTv3 training, domain adaptation, or hybrid encoding strategies that combine the generalization stability of PointNet++ with the contextual modeling capacity of transformer based point cloud backbones.

4.3.3 Qualitative Examples

Figures 4.3, 4.4, 4.5, and 4.6 present qualitative examples of the model's responses to illustrate its reasoning capabilities in 3D scenes. The examples are selected from the multi-dataset and Scannet evaluation test sets and cover diverse question categories. For each example, we provide the input question, the ground truth answer, and the model predictions. When applicable, we compare the outputs of the PointNet++ baseline and the PTv3 based model in order to highlight differences in reasoning behavior. Across all examples, the important objects are highlighted with colored bounding boxes in the point cloud, making it easier to identify the ground truth answer in the scene.

These examples show that the PTv3-based model is particularly effective in cases where the answer depends on contextualized understanding of the surrounding 3D scene rather than isolated object recognition. In the counting examples, PTv3 correctly estimates the number of target objects within a specified egocentric direction, while PointNet++ underestimates or overestimates the count in cluttered environments. The attribute examples further show that PTv3 can distinguish object properties such as color and physical state, whereas PointNet++ often returns plausible but incorrect attributes. A similar pattern appears in the navigation and spatial-reference examples, where PTv3 more accurately associates the queried object with its location relative to the user and nearby objects. For existence and room-type questions, PTv3 also makes better use of surrounding contextual cues, such as furniture, fixtures, and room layout, to determine whether an object is present or to infer the functional category of the space. Taken together, these qualitative cases suggest that, although the overall quantitative improvement is not large, the PTv3-based model offers a clearer advantage in questions requiring broader scene awareness, including object co-occurrence, local spatial structure, egocentric direction, and semantic context. This behavior is observed in both the ScanNet-only and multi-dataset evaluations, suggesting that the model's benefit is most evident when reasoning depends on integrating the queried object with its surrounding 3D environment.

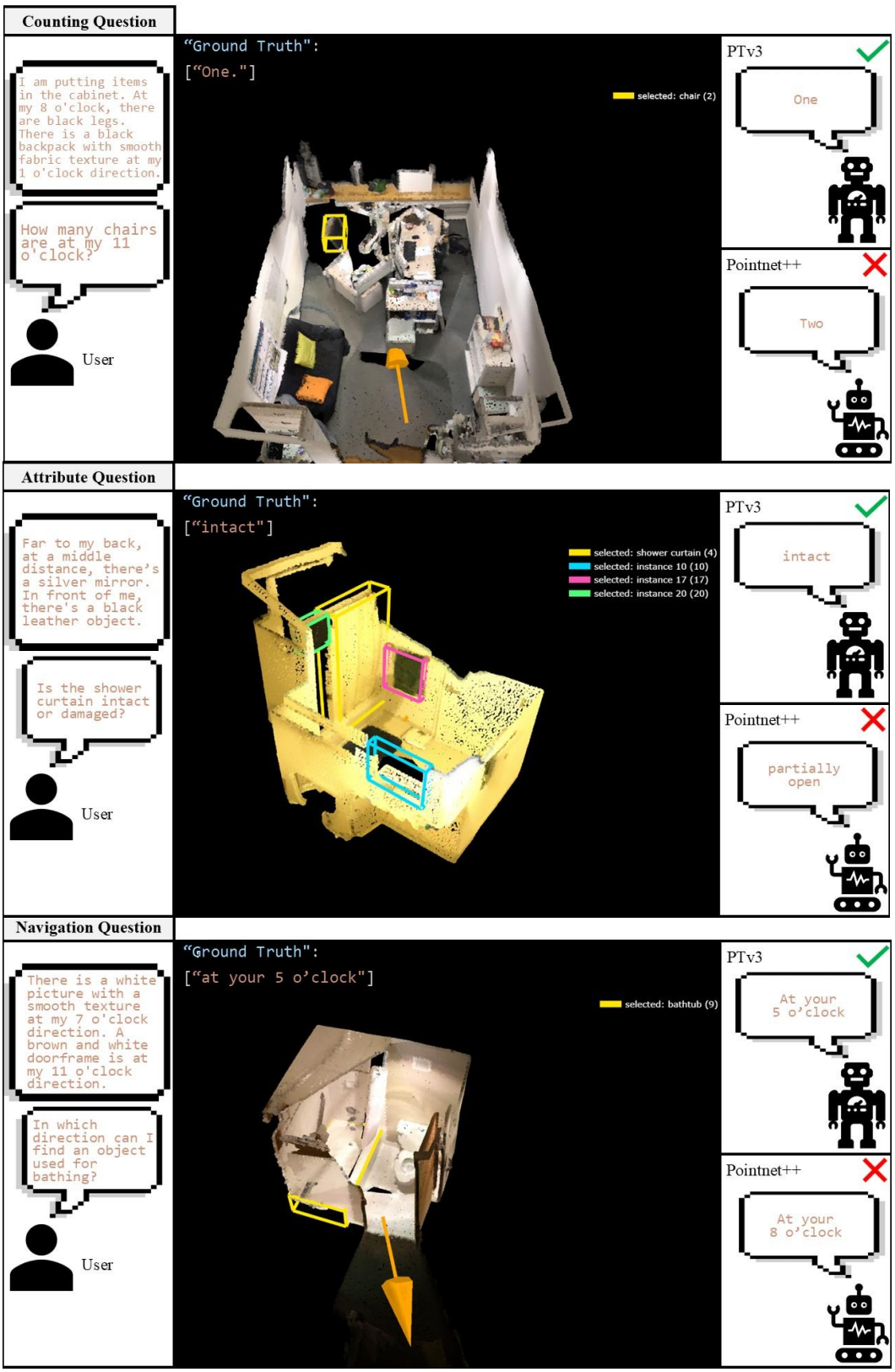

Figure 4.3: Qualitative comparison between the proposed PTv3 based model and the original PointNet++ baseline across the counting,navigation and attribute reasoning tasks from the MSQA benchmark. The reported examples are drawn from the evaluation results of the model trained jointly on ScanNet, ARKitScenes, and 3RScan.

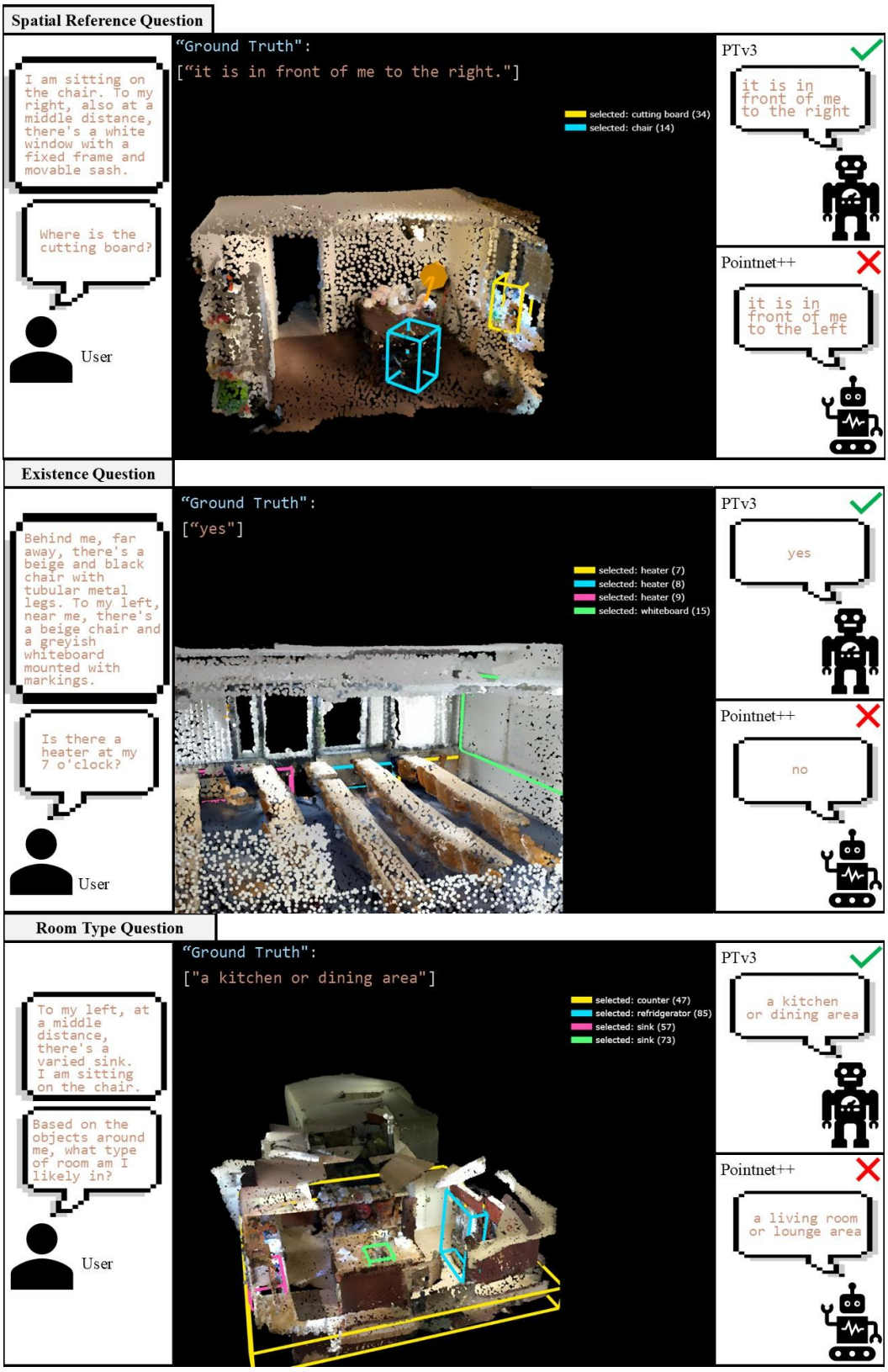

Figure 4.4: Qualitative comparison between the proposed PTv3 based model and the original PointNet++ baseline across the existence, spatial reference, and room type tasks from the MSQA benchmark. The reported examples are drawn from the evaluation results of the model trained jointly on ScanNet, ARKitScenes, and 3RScan.

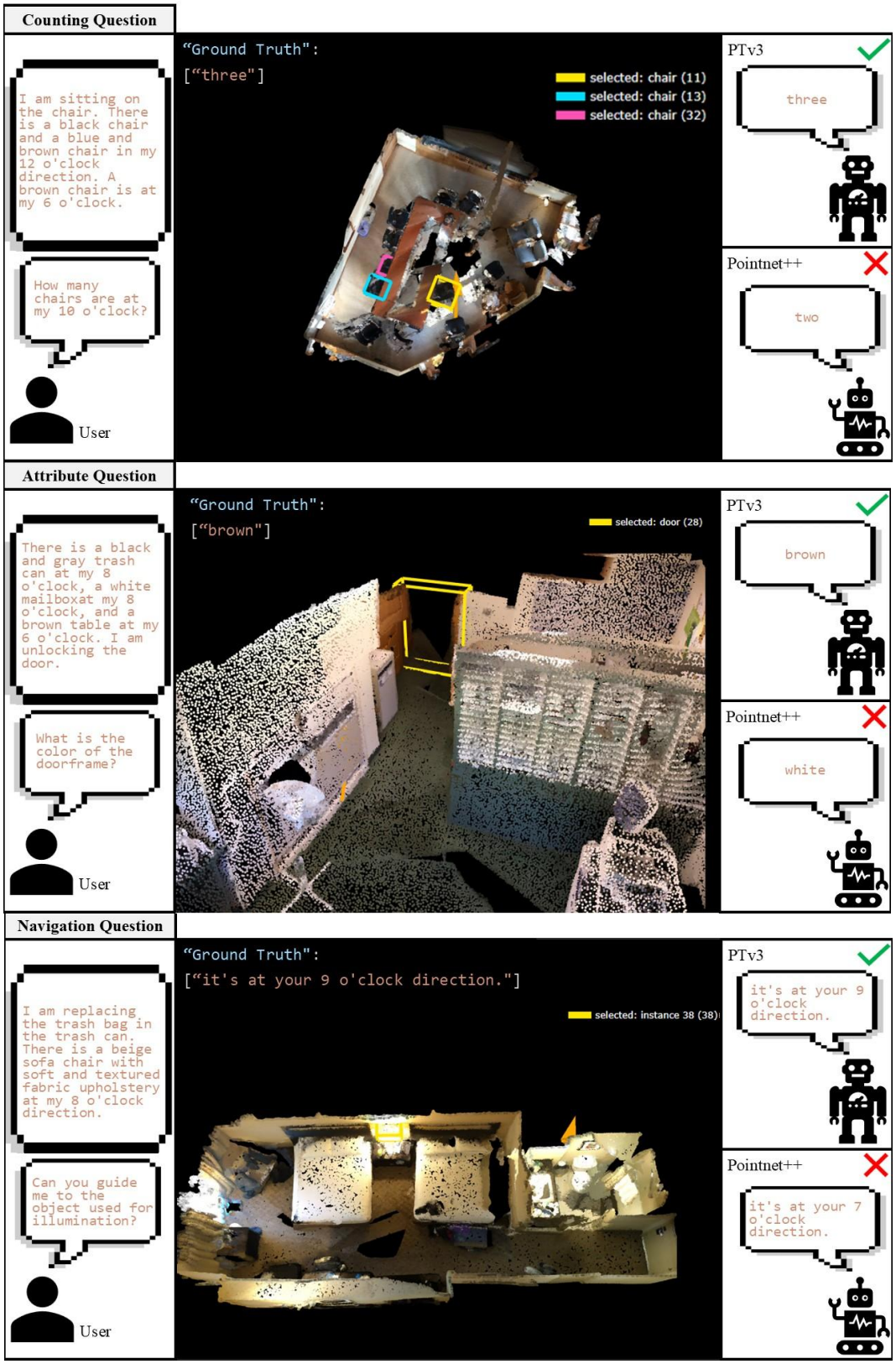

Figure 4.5: Qualitative comparison between the proposed PTv3 based model and the original PointNet++ baseline across the counting,navigation and attribute tasks from the MSQA benchmark. The reported examples are drawn from evaluation results for the model trained exclusively on the ScanNet dataset.

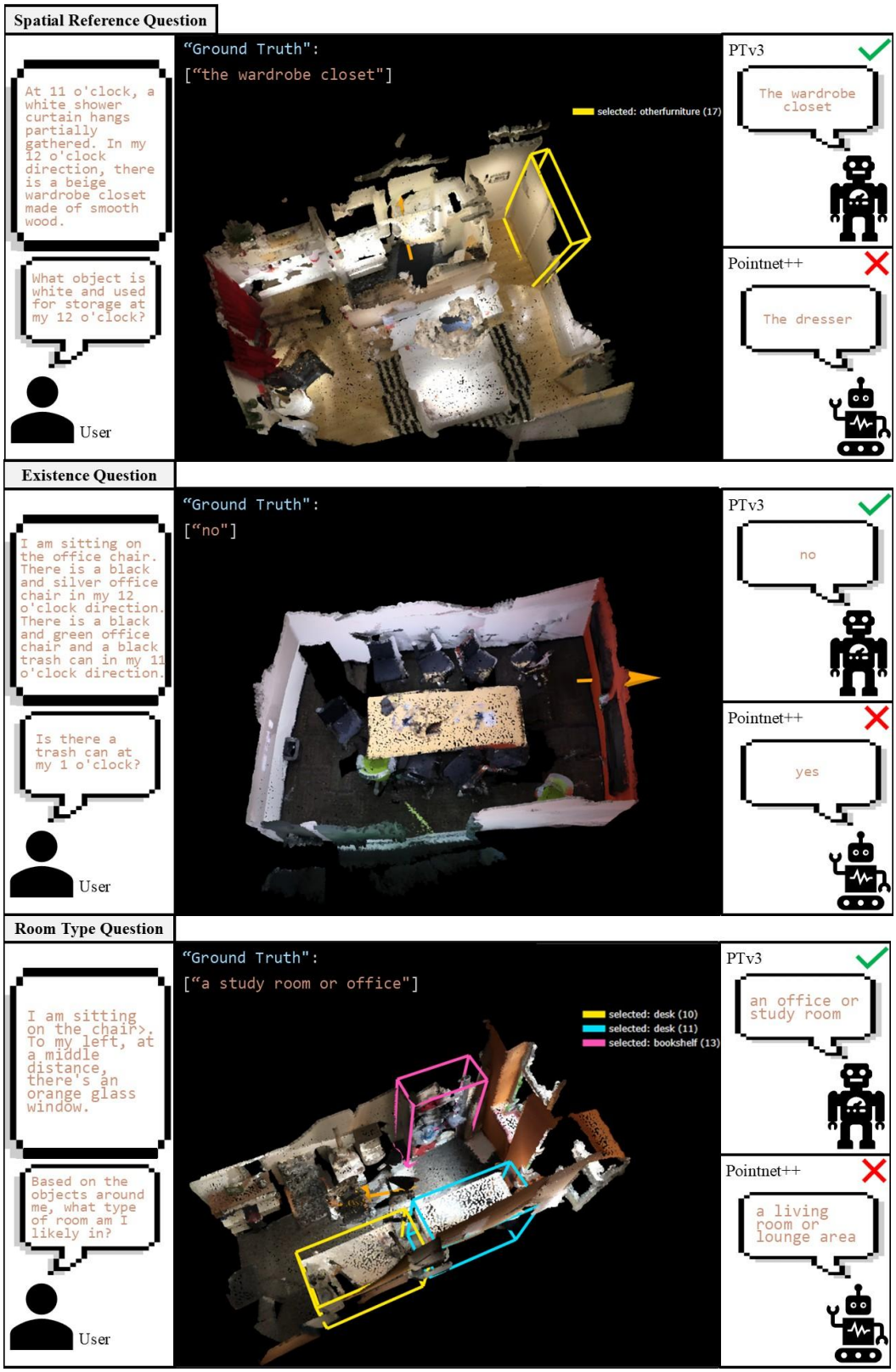

Figure 4.6: Qualitative comparison between the proposed PTv3 based model and the original PointNet++ baseline across the existence, spatial reference, and room type reasoning tasks from the MSQA benchmark. The reported examples are drawn from evaluation results for the model trained exclusively on the ScanNet dataset.

5 Conclusion

This work investigated the role of 3D feature extraction architectures in multimodal scene understanding and situated reasoning within indoor environments. In particular, the work focused on evaluating the effectiveness of transformer point cloud processing within the MSR3D framework for multimodal situated question answering tasks [25, 34].

The study was motivated by the growing importance of contextual and spatial reasoning in 3D environments, especially in applications such as robotics, embodied AI, autonomous systems, and interactive scene understanding [31, 18]. While traditional architectures such as PointNet and PointNet++ have demonstrated strong performance in learning geometric representations from unordered point clouds [42, 43], recent advances in transformer architectures have introduced new possibilities for modeling long range spatial dependencies and contextual relationships [64, 57].

Within this context, the primary contribution of this thesis was the integration of PTv3 [57] as a replacement for the original PointNet++ backbone inside the MSR3D pipeline. The proposed modification aimed to investigate whether scene encoding could improve the quality of object-level feature representations for multimodal reasoning tasks. Unlike the original PointNet++ encoder, which processes objects independently through hierarchical local aggregation [43], PTv3 enables the modeling of both local and global spatial interactions across the entire scene using attention mechanisms and serialized point processing.

To support this integration, several architectural and implementation adaptations were introduced. These included scene-level point cloud preprocessing, tensor serialization compatible with the PTv3 pipeline, inverse mapping for restoring object associations, and instance aware feature pooling for generating object-level embeddings compatible with the existing MSR3D architecture. Furthermore, the work introduced an interactive visualization framework for analyzing 3D scenes, object instances, segmentation outputs, and multimodal reasoning behavior. The interface provided tools for scene inspection, object highlighting, question answer interaction, and visualization of geometric properties such as surface normals and bounding boxes.

Experimental evaluation was conducted using the MSQA benchmark [34], which provides large-scale situated reasoning tasks over indoor 3D scenes derived from datasets such as ScanNet [13], 3RScan [52], and ARKitScenes [53]. The experiments compared the original PointNet++-based implementation against the proposed PTv3 approach under both exclusively ScanNet, as well as multi dataset training settings.

The results demonstrated that scene encoding can provide meaningful improvements in contextual feature learning and multimodal reasoning performance. In particular, PTv3 showed

stronger capability in capturing spatial relationships and producing context aware object embeddings, especially in situations requiring relational reasoning and broader environmental awareness. At the same time, the experiments also highlighted several practical trade offs associated with transformer architecture point cloud processing, including increased computational complexity, memory consumption, and training overhead.

Overall, this thesis demonstrates that the quality of geometric feature extraction plays a critical role in multimodal 3D scene reasoning systems. The findings support the growing trend toward transformer architectures for 3D understanding and suggest that scene-level contextual encoding can benefit situated reasoning tasks. More broadly, the work contributes to ongoing research in embodied AI and multimodal scene understanding by exploring how advances in point cloud transformers can be integrated into complex reasoning pipelines that combine geometry, vision, and language [50, 16, 20].

Finally, this study establishes a foundation for future research into scalable transformer architectures, improved multimodal fusion strategies, and more advanced embodied reasoning systems capable of operating in realistic and dynamic 3D environments.

5.1 Future Work

The findings of this thesis open several directions for further research and development. While the study demonstrates that replacing the PointNet++ backbone in MSR3D with PTv3 can influence geometric feature representation within a multi-modal reasoning pipeline, the experiments also surface concrete limitations that motivate further investigation. In particular, the integration of PTv3 revealed trade offs in preprocessing complexity, batching behavior, and scalability to large indoor scenes that were not present with the original backbone [34, 43, 57].

5.1.1 Exploration of Alternative 3D Backbone Architectures

While this work focuses on replacing the PointNet++ backbone with PTv3, the design space of 3D feature extractors is considerably broader, and further exploration of alternative architectures could provide deeper insight into the relationship between geometric representation and multimodal reasoning performance. Early approaches such as PointNet [42] and its hierarchical extension PointNet++ [43] established the foundation for learning directly from point clouds, while subsequent work has introduced increasingly expressive models based on graph structures [54] and attention mechanisms [64]. The progression toward transformer based architectures, culminating in models such as PTv3 [57], highlights the rapid evolution of this field and motivates a broader comparative analysis.

A particularly relevant direction is the evaluation of newer point transformer variants that are

designed to improve upon the efficiency limitations of PTv3. LitePT [62], discussed earlier as a hybrid convolution-transformer architecture, is especially suitable for this purpose. Rather than applying attention uniformly throughout the network, LitePT uses sparse convolutional blocks in the early high-resolution stages and attention blocks in deeper low-resolution stages. This design reflects the observation that convolution is well suited for extracting low-level local geometry, while attention is more useful for modeling high-level semantic context and long-range relationships. Since the experiments in this thesis identified memory consumption, batching behavior, and computational overhead as practical limitations of PTv3, LitePT represents a strong candidate for future integration into the MSR3D framework. The reported reductions in parameters, runtime, and memory usage compared with PTv3 make it especially relevant for improving scalability while preserving the benefits of scene-level contextual encoding [62].

Future work could also investigate whether pretrained 3D representations improve multimodal reasoning performance beyond what can be achieved through supervised backbone replacement alone. Sonata [56] is relevant in this context because it focuses on self-supervised learning of reliable point cloud representations. Instead of only modifying the network architecture, Sonata addresses the quality of learned point features by reducing the tendency of 3D models to rely on low-level geometric shortcuts. Incorporating a Sonata-pretrained encoder would therefore allow a controlled ablation that isolates the contribution of feature quality from that of architectural design, providing clearer insight into where the performance ceiling of the performance bottleneck of the current MSR3D pipeline lies.

Similarly, Concerto [63] provides a promising direction that is closely aligned with the multimodal nature of MSR3D. Concerto combines self-supervised learning within the 3D backbone with cross-modal alignment between 2D image features and 3D point cloud features. This is particularly relevant for MSR3D, where point cloud features, image features, and text embeddings must be projected into a shared reasoning space. A Concerto-style representation could therefore serve not only as a stronger 3D encoder, but also as a more semantically aligned bridge between the 3D backbone and the multimodal fusion module.

In addition to point transformer variants and pretrained representations, future work could systematically evaluate voxel-based and sparse convolutional approaches. Methods such as VoxNet [38] and OctNet [44] demonstrate how structured volumetric representations can be used for 3D learning. Sparse convolutional networks are particularly relevant to the limitations observed in this work, as they exploit the inherent sparsity of point cloud data while preserving spatial locality. Such approaches may provide a complementary balance between efficiency and representational power when integrated into the MSR3D framework.

Ultimately, expanding the range of evaluated 3D backbones would contribute to a more comprehensive understanding of how architectural choices influence both geometric feature quality

and multimodal reasoning effectiveness. This aligns with broader trends in geometric deep learning [7] and recent surveys on 3D point cloud methods [20], which emphasize that no single representation is universally optimal and that task-specific considerations should guide the selection of underlying architectures.

5.1.2 Improved multimodal Fusion Strategies

The current MSR3D framework employs a placeholder based fusion mechanism in which embeddings from different modalities are inserted into a shared token sequence and processed by the language model. While this approach is effective, it limits the depth of cross modal interaction to what the language model can infer implicitly from the concatenated sequence. This design may underutilize the geometric features produced by a stronger 3D backbone such as PTv3, since the language model has no mechanism to attend selectively to spatial features in response to the structure of a given query.

Future research could investigate cross attention fusion strategies in which geometric and visual features are queried by the language model dynamically, conditioned on the text representation. Architectures such as Q-Former, already used in BLIP-2 [32] for the 2D backbone, could be adapted to mediate between 3D point features and the language decoder. Incorporating temporal information, such as sequences of observations from a moving agent, could further extend the model toward dynamic scene understanding, enabling reasoning over changes in the environment rather than static snapshots.

5.1.3 Integration with Embodied AI Systems

A final and significant direction is the integration of the proposed approach into embodied AI systems, where a physical agent perceives and acts within a 3D environment in real time [25]. The situated reasoning setting studied in this thesis is a natural precursor to embodied applications: the model already conditions its responses on an agent's position and orientation, and the interactive interface demonstrates that the system can respond to user specified queries about a scene [36, 34]. Extending this to a fully embodied setting would require handling streaming point cloud data, maintaining a consistent scene representation as the agent moves, and coupling perception with action planning.

Such an extension could involve combining the current model with reinforcement learning or planning modules, enabling the agent not only to answer questions about the environment but also to take actions in response to goals specified in natural language [31]. This direction would represent a significant step toward practical deployment, bridging the gap between the static scene understanding studied in this thesis and interactive, real world applications.

Bibliography

[1] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman, "Real-time rendering," in CRC Press, 2018 (cit. on pp. 7, 40).

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015 (cit. on pp. 1, 5, 19, 20).

[3] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016 (cit. on pp. 10, 21).

[4] Daichi Azuma, Yuta Miyanishi, Shuhei Kurita, and Motoaki Kawanabe, "Scanqa: 3d question answering for spatial scene understanding," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022 (cit. on pp. 1, 2, 11, 20).

[5] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy, "A survey of polygon mesh processing," *Eurographics*, 2007 (cit. on pp. 6, 7).

[6] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy, "Polygon mesh processing," in AK Peters, 2010 (cit. on pp. 6, 7, 13, 41, 71, 73).

[7] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, 2017 (cit. on pp. 5, 11, 13, 14, 59).

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, "Spectral networks and locally connected networks on graphs," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014 (cit. on p. 13).

[9] Holger Caesar, Vora Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Gabriel Baldan, and Oscar Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020 (cit. on pp. 6, 10, 12, 14).

[10] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing 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," *arXiv preprint arXiv:1512.03012*, 2015 (cit. on p. 8).

[11] Dave Zhenyu Chen, Angel X. Chang, and Matthias Nießner, "Scanrefer: 3d object localization in rgb-d scans using natural language," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020 (cit. on p. 2).

[12] CYENS Centre of Excellence, *Prometheus documentation*, Accessed for hardware description of the Prometheus HPC cluster, 2025. [Online]. Available: https://prometheus-docs.cyens.org.cy/docs/hardware/ (cit. on p. 45).

[13] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 2, 5, 10–12, 14, 21, 25, 40, 42, 44, 48, 50, 56, 68).

[14] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness," in *Advances in Neural Information Processing Systems*, 2022 (cit. on p. 86).

[15] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi, "Objaverse: A Universe of Annotated 3D Objects," *arXiv preprint arXiv:2212.08051*, 2022 (cit. on p. 9).

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021 (cit. on pp. 3, 12, 20, 57).

[17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, 2013 (cit. on p. 10).

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012 (cit. on pp. 1, 6, 56).

[19] Gradio Team, *Gradio: Build machine learning web apps in python*, 2023. [Online]. Available: https://gradio.app (cit. on pp. 40, 68, 79).

[20] Yulan Guo, Huan Wang, Qingyong Hu, Hao Liu, Liqian Liu, and Mohammed Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2020 (cit. on pp. 5, 6, 8, 11, 12, 14, 40, 57, 59, 70).

[21] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or, "Meshcnn: A network with an edge," in *Proceedings of the ACM Special Interest Group on Computer Graphics and Interactive Techniques Conference (SIGGRAPH)*, 2019 (cit. on pp. 11, 13, 14).

[22] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant, "Array Programming with NumPy," *Nature*, 2020 (cit. on p. 79).

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016 (cit. on p. 3).

[24] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022 (cit. on pp. 44, 46, 48, 49).

[25] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang, "An Embodied Generalist Agent in 3D World," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024 (cit. on pp. 44, 56, 59).

[26] Shijia Huang, Yilun Chen, Jiaya Jia, and Liwei Wang, "Multi-view transformer for 3d visual grounding," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2022 (cit. on p. 30).

[27] ICLR Blogposts, *Multi-modal learning: A look back and the road ahead*, Accessed: 2026-04-08, 2025. [Online]. Available: https://iclr-blogposts.github.io/2025/blog/multimodal-learning/ (cit. on pp. 5, 22).

[28] Georg Klein and David Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007 (cit. on p. 1).

[29] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo, "ABC: A Big CAD Model Dataset for Geometric Deep Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 (cit. on p. 9).

[30] Taku Kudo and John Richardson, "SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing," *arXiv preprint arXiv:1808.06226*, 2018 (cit. on p. 27).

[31] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," in *International Symposium on Experimental Robotics (ISER)*, 2016 (cit. on pp. 1, 6, 56, 59).

[32] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi, "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2023 (cit. on pp. 26, 44, 48, 59).

[33] Zechuan Li, Hongshan Yu, Yihao Ding, Yan Li, Yong He, and Naveed Akhtar, "Embodied Intelligence for 3D Understanding: A Survey on 3D Scene Question Answering," *arXiv preprint arXiv:2502.00342*, 2025 (cit. on p. 20).

[34] Xiongkun Linghu, Jiangyong Huang, Xuesong Niu, Xiaojian Ma, Baoxiong Jia, and Siyuan Huang, "Multi-modal Situated Reasoning in 3D Scenes," in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2024 (cit. on pp. 3, 5, 11, 12, 21–25, 31, 35, 41, 48, 50, 56, 57, 59, 68, 69, 75, 82–84, 89).

[35] Nicolas von Lützow, Barbara Rössle, Katharina Schmid, and Matthias Nießner, "GaussianGPT: Towards autoregressive 3d gaussian scene generation," *arXiv preprint arXiv:2603.26661*, 2026 (cit. on p. 10).

[36] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang, "Sqa3d: Situated question answering in 3d scenes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023 (cit. on pp. 1, 11, 20, 21, 59).

[37] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCV Workshops)*, 2015 (cit. on pp. 13, 14).

[38] Daniel Maturana and Sebastian Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015 (cit. on pp. 5, 6, 12, 13, 58).

[39] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su, "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 (cit. on p. 9).

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2019 (cit. on pp. 70, 79).

[41] Plotly Technologies Inc., *Plotly: Interactive graphing library*, 2023. [Online]. Available: https://plotly.com/python/ (cit. on pp. 40, 68, 70, 79).

[42] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 2, 3, 5, 6, 8, 11, 12, 14, 31, 56, 57).

[43] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2017 (cit. on pp. 2, 3, 5, 6, 8, 12, 14–16, 25, 30, 32, 35, 44, 45, 48, 49, 56, 57).

[44] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017 (cit. on pp. 5, 7, 13, 58).

[45] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012 (cit. on pp. 70, 77).

[46] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on pp. 2, 10, 21).

[47] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-View Convolutional Neural Networks for 3D Shape Recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 945–953 (cit. on p. 6).

[48] Pei Sun, Henrik Kretzschmar, Xavier Dotiwalla, Aurélien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020 (cit. on pp. 6, 10).

[49] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill, "Multimodal few-shot learning with frozen language models," in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2021 (cit. on pp. 22, 41).

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2017 (cit. on pp. 3, 12, 57).

[51] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, 2020 (cit. on pp. 73, 76, 79).

[52] Johanna Wald, Armen Avetisyan, Nassir Navab, and Federico Tombari, "3rscan: A large-scale rgb-d scan dataset for 3d scene understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019 (cit. on pp. 2, 25, 40, 42, 44, 48, 50, 56, 68).

[53] Johanna Wald, Armen Avetisyan, Nassir Navab, and Federico Tombari, "Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2021 (cit. on pp. 2, 5, 25, 40, 42, 44, 48, 50, 56, 68).

[54] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay Sarma, Michael Bronstein, and Justin Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics*, 2019 (cit. on pp. 12, 57).

[55] Wikipedia contributors, *Types of mesh*, Accessed: 2026-04-02, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Types_of_mesh&oldid=1244155780 (cit. on p. 6).

[56] Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub, *Sonata: Self-supervised learning of reliable point representations*, 2025. arXiv: 2503.16429 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2503.16429 (cit. on p. 58).

[57] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao, "Point Transformer V3: Simpler, Faster, Stronger," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024 (cit. on pp. 3, 6, 12, 18, 24, 35–39, 44, 45, 48, 49, 56, 57, 82–84, 86, 89).

[58] Xiaoyang Wu, Yixing Lao, Li Jiang, Hengshuang Zhao, Jingyong Liu, and Jingdong Zhao, "Point transformer v2: Grouped vector attention and partition-based pooling," in *Proceedings of the Annual Conference on Neural Information Processing System (NeurIPS)*, 2022 (cit. on pp. 3, 12, 17, 18, 39).

[59] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015 (cit. on pp. 5–8, 11–13).

[60] Wei Xu, Chunsheng Shi, Sifan Tu, Xin Zhou, Dingkang Liang, and Xiang Bai, "UniSeg3D: A Unified Framework for 3D Scene Understanding," *arXiv preprint arXiv:2407.03263*, 2024 (cit. on p. 21).

[61] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyuan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas, "A Scalable Active Framework for Region Annotation in 3D Shape Collections," *ACM Transactions on Graphics*, 2016 (cit. on p. 9).

[62] Yuanwen Yue, Damien Robert, Jianyuan Wang, Sunghwan Hong, Jan Dirk Wegner, Christian Rupprecht, and Konrad Schindler, "LitePT: Lighter Yet Stronger Point Transformer," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026 (cit. on pp. 18, 19, 58).

[63] Yujia Zhang, Xiaoyang Wu, Yixing Lao, Chengyao Wang, Zhuotao Tian, Naiyan Wang, and Hengshuang Zhao, *Concerto: Joint 2d-3d self-supervised learning emerges spatial representations*, 2026. arXiv: 2510.23607 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2510.23607 (cit. on p. 58).

[64] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun, "Point Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021 (cit. on pp. 3, 6, 12, 16, 18, 39, 49, 56, 57).

[65] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yongji Zhuang, Zhenyu Lin, Zihao Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica, *Vicuna: An open-source chatbot impressing gpt-4 with 90% chatgpt quality*, 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/ (cit. on pp. 27, 28, 44, 46, 48).

[66] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer, "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017 (cit. on p. 6).

APPENDICES

APPENDIX I

Implementation of the Interactive User Interface

An interactive visualization interface was developed as an integral component of this work, serving two complementary purposes: enabling systematic qualitative analysis of the multimodal reasoning model, and providing an accessible environment for direct human interaction with 3D scenes. Rather than confining evaluation to aggregate metrics alone, the interface makes it possible to inspect individual scene question answer pairs, explore geometric structure at multiple levels of granularity, and probe the model's responses through natural language queries.

The system is implemented in Python, combining the Gradio framework [19] for web based interface construction with Plotly [41] for 3D point cloud visualization. This pairing allows large-scale point clouds containing hundreds of thousands of points to be rendered and displayed in a standard web browser, without requiring dedicated graphics software or specialized hardware on the client side. The interface is designed around the MSR3D multimodal reasoning pipeline [34] and natively supports all three datasets used in this work: ScanNet [13], ARKitScenes [53], and 3RScan [52].

I.1 System Architecture

The interface adopts a modular, three layer architecture that decouples user interaction from data processing and model inference. This separation improves maintainability and allows each layer to evolve independently as the underlying model or datasets change. The three primary layers are as follows:

- **User Interaction Layer.** Responsible for all input controls, including dataset and scene selection, visualization parameter controls, object filtering, natural language chat input, and optional multimodal pose overrides. This layer is implemented entirely within the Gradio framework [19], which exposes UI components as reactive Python callbacks.
- **Visualization Layer.** Handles the construction and rendering of 3D Plotly figures [41], including point cloud coloring, bounding box overlays, coordinate axes, agent orientation arrows, and surface normal glyphs. All rendering operations are performed on the server side and streamed as interactive HTML figures to the client browser.

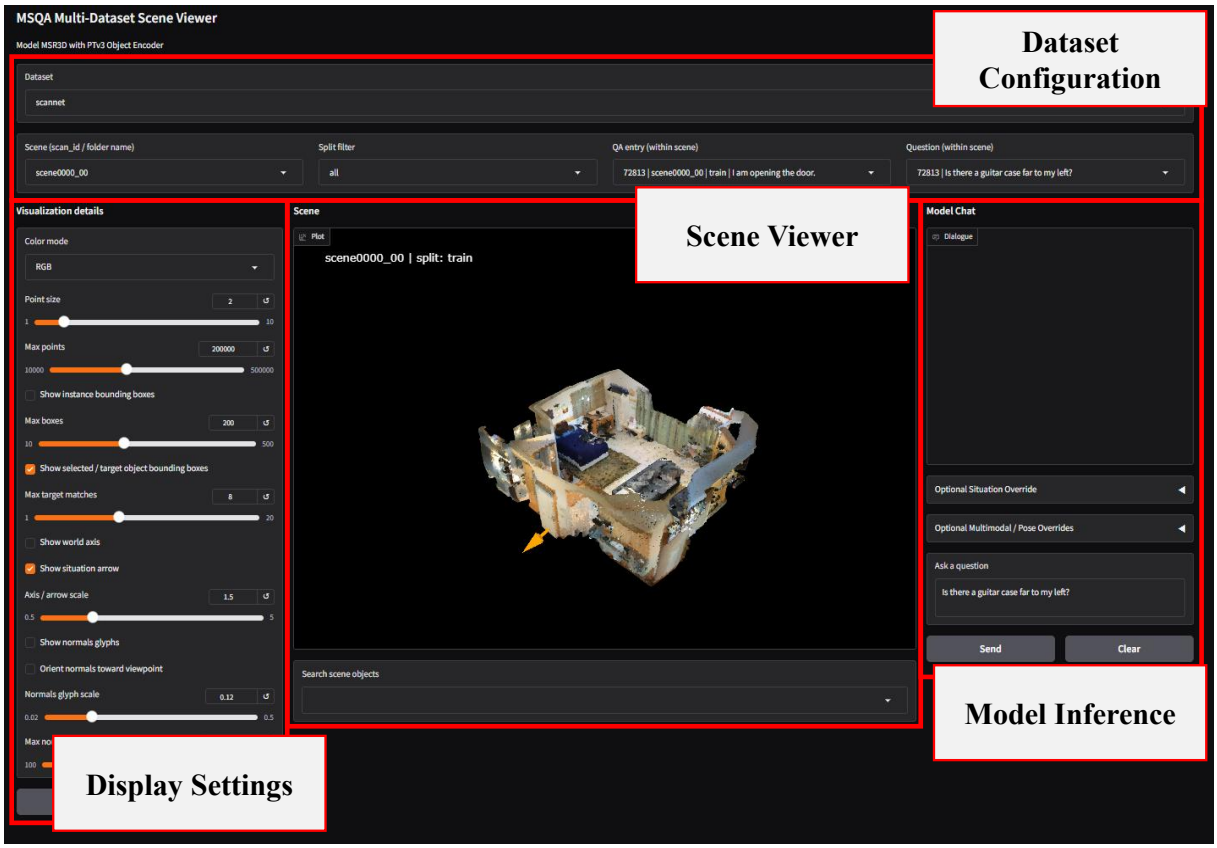

Figure I.1: Overview of the multi-dataset scene viewer interface, showing dataset configuration, visualization controls, scene rendering, and model inference panels.

- **Model Integration Layer.** Provides a thread safe interface to the MSR3D inference service [34], supporting concurrent scene loading while serializing model queries through a threading lock, thereby preventing race conditions during multi session access.

Data and control flow are managed through stateful Gradio components, which maintain the currently selected scene, and QA sample across all user interactions. Heavyweight operations, such as point cloud loading and downsampling, are explicitly separated from lightweight style updates. Therefore, simple visual adjustments such as toggling bounding boxes or changing the color mode do not trigger unnecessary scene reloading.

I.2 Scene Representation and Data Loading

All scenes are represented using a unified point cloud structure that mirrors the preprocessing pipeline employed during model training. Each scene is encoded as a structured feature tensor:

$$\mathbf{S} \in \mathbb{R}^{N \times 9} \tag{I.1}$$

where each of the N points is represented by a 9 dimensional feature vector comprising 3D spatial coordinates (x, y, z), normalized RGB color values in the range $[-1, 1]$, and estimated surface normal vectors (n_x, n_y, n_z). This representation aligns with established practices in 3D deep learning, where heterogeneous point attributes are concatenated into a single feature vector to capture both geometric and photometric properties [20].

Specific data loaders are utilized for each of the three supported datasets. The ScanNet loader reads pre aligned point cloud files in PyTorch [40] tensor format alongside separately stored surface normal estimates, and performs label remapping from the NYU-40 [45] taxonomy to the ScanNet-20 category set using a precomputed lookup table derived from the official annotation metadata. The ARKitScenes and 3RScan loaders follow analogous procedures tailored to the file formats and annotation conventions of their respective datasets.

In addition to the scene-level point cloud, instance identifiers and semantic labels are preserved for every point, enabling fine grained object-level analysis within the interface. Object point subsets are extracted by grouping points according to their instance annotations, producing per object feature arrays that are used to compute bounding boxes and support focused object inspection.

To ensure efficient interaction with large-scale scenes, a multi level caching strategy is employed. Raw scene data is loaded once per scan identifier and retained in memory across subsequent interactions. Downsampled point cloud representations are cached based on the user specified maximum points chosen, avoiding redundant subsampling during repeated visualization updates. Precomputed color mappings and axis aligned bounding box geometries are similarly cached, reducing the computational overhead of style switching operations. This design minimizes expensive disk I/O and maintains interface responsiveness even for scenes containing several hundred thousand points.

I.3 Visualization Functionality

The visualization component is designed to expose different structural properties of a scene through multiple complementary rendering modes. Users can switch between these modes without reloading the scene's geometry, as style updates operate directly on the point data. All rendering is performed via Plotly `Scatter3d` traces [41], which are streamed to the browser as self contained dynamic figures supporting rotation, panning, and zooming, allowing the user to inspect the scene thoroughly.

I.3.1 Color Encodings

Four color encoding modes are provided, each revealing a distinct aspect of the scene.

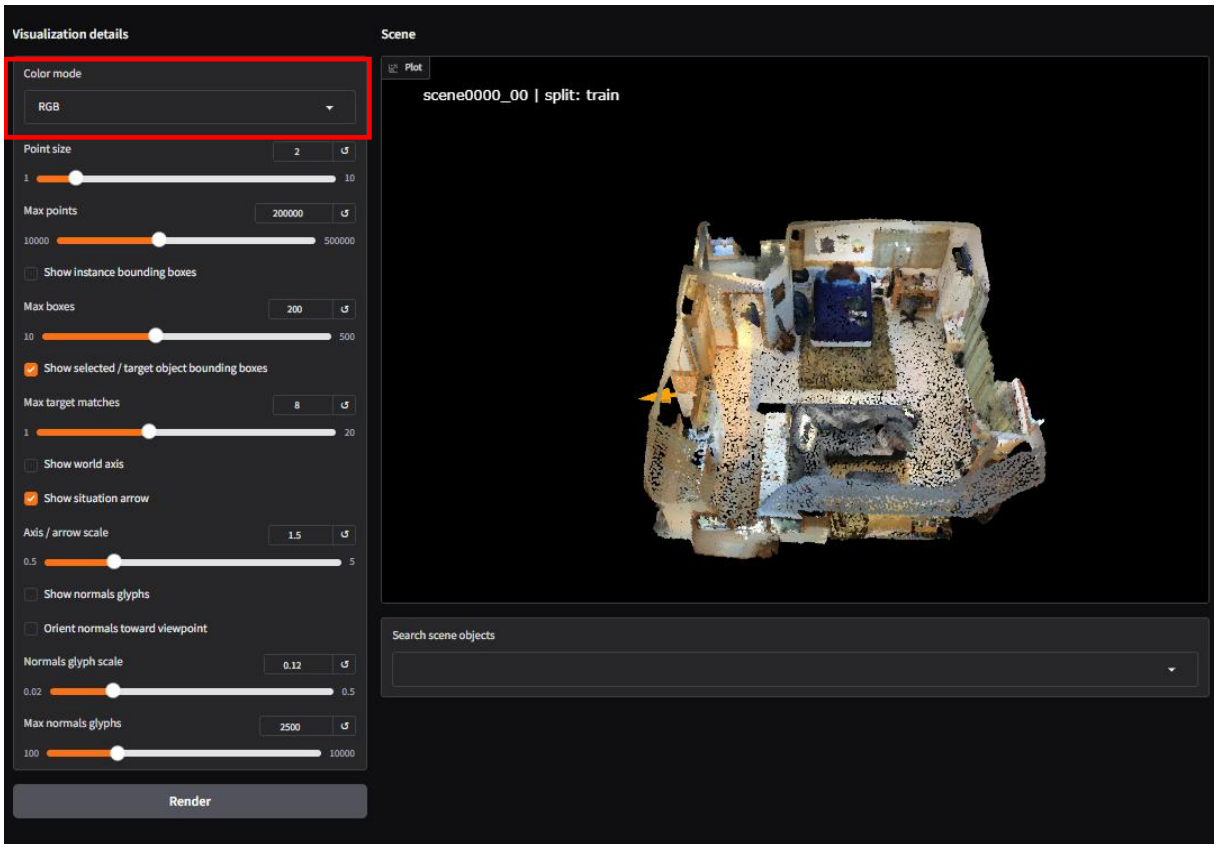

Figure I.2: Scene visualization using RGB color mode for rendering the reconstructed 3D environment.

- **RGB.** Standard photometric coloring derived from the sensor captured color values, providing a visually realistic representation of the scene as it appears in the real world.
- **Instance.** Each object instance is assigned a unique, randomly seeded color derived from a stable hash of its instance identifier. This encoding immediately makes the different instances distinct within the scene.
- **Semantic.** Points are colored according to their class level semantic label using the ScanNet-20 category set. This mode exposes the semantic structure of the scene and allows inspection of the point distributions of each semantic class.
- **Normals.** Surface normal vectors are mapped to color space via the transformation $\text{RGB} = 0.5 \times (\mathbf{n} + \mathbf{1})$, mapping the range $[-1,\ 1]$ per axis to $[0,\ 1]$ in each color channel. This encoding provides an intuitive visual representation of local surface orientation and is widely used in geometric analysis and normal map inspection [6].

I.3.2 Geometric Overlays

Several optional overlays supplement the point cloud rendering with spatial context.

- **Axis Aligned Bounding Boxes.** Instance bounding boxes are computed from the extremal

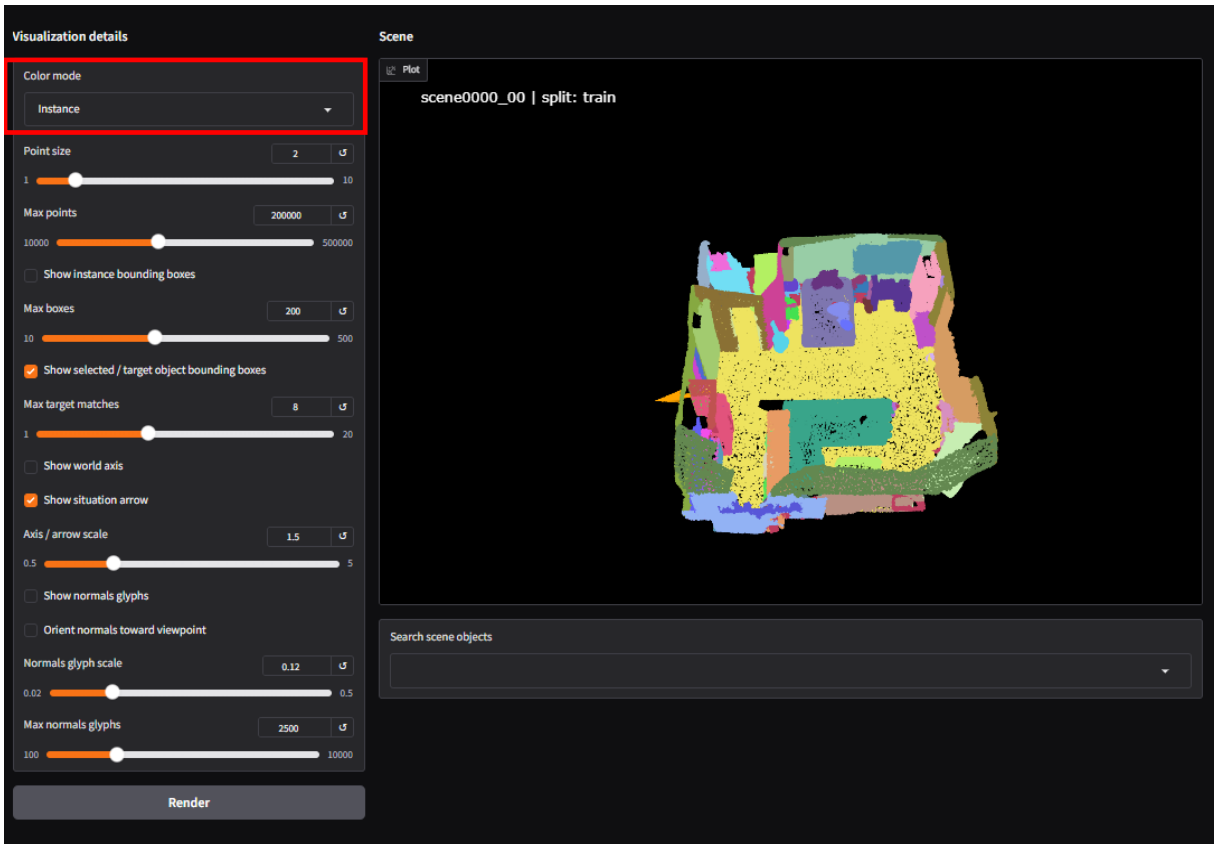

Figure I.3: Instance-level visualization where unique colors correspond to individual object instances.

coordinates of each object's point set and rendered as wireframe structures composed of twelve line segments. The number of displayed boxes is user configurable to manage visual clutter in densely populated scenes. Each box is colored consistently with the instance coloring scheme.

- **Target Object Highlighting.** In addition to AABB, when a QA sample is selected, objects mentioned in the associated question,situation and answer are identified through a tokenization and alias expansion procedure that maps question terms to instance labels and semantic class names. Matched objects are highlighted with a distinct bounding box color, yellow for instance matches and cyan for semantic class matches, drawing the analyst's attention to the scene elements most relevant to the query.
- **Global Coordinate Frame.** An optional RGB axis triad is rendered at the scene origin, providing absolute spatial reference for interpreting object positions and orientations across scenes.
- **Agent Orientation Arrow.** The agent's position and viewing direction, as encoded in the situation description of the selected QA sample, are rendered as a directed arrow composed of a shaft and a cone head. The orientation is derived from quaternion or Euler angle

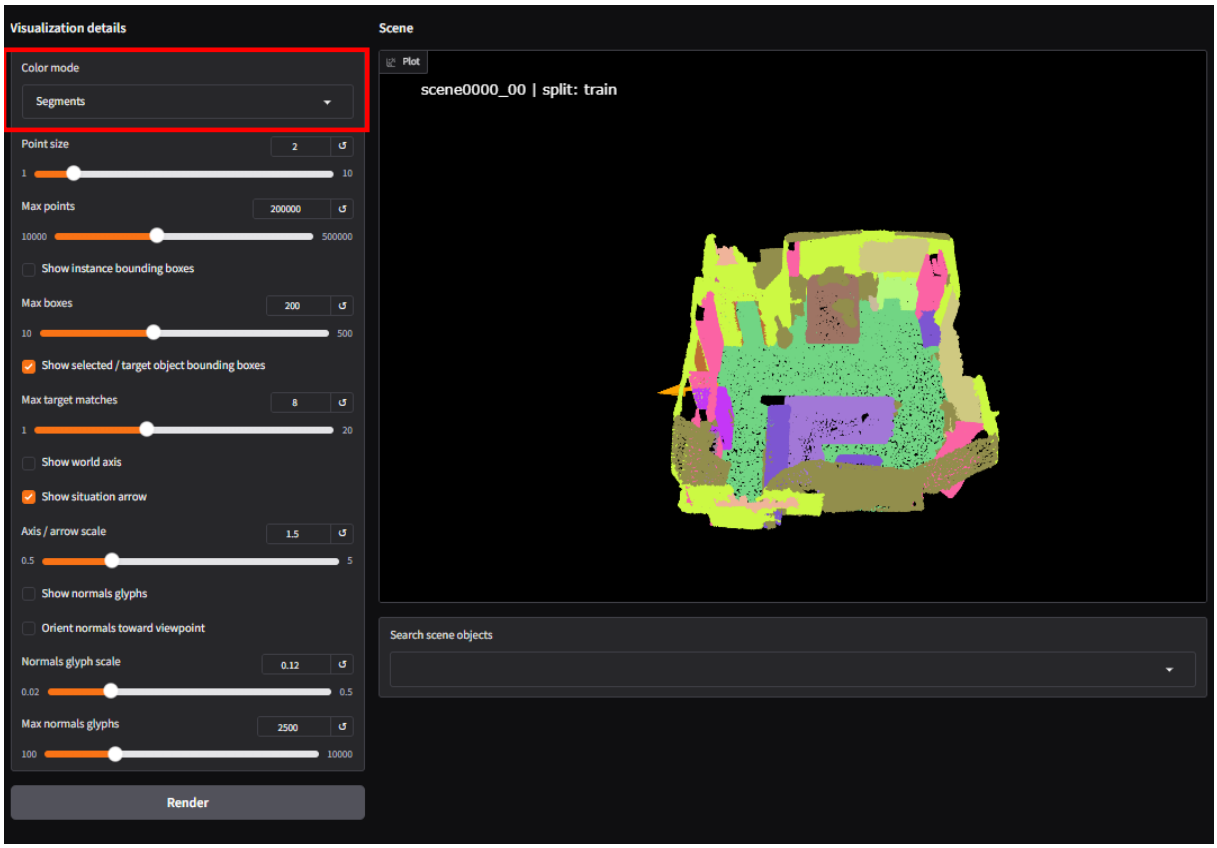

Figure I.4: Segment-based visualization highlighting different geometric regions within the scene.

representations via SciPy [51] rotation utilities and projected onto the horizontal plane to produce a consistent directional indicator. This overlay makes the situated nature of each query directly interpretable within the scene.

I.3.3 Surface Normal Visualization

In addition to normal based color encoding, the interface provides an explicit glyph representation of surface normals. When enabled, surface normals are rendered as short line segments, or glyphs, originating from a subsampled set of points and extending in the direction of the estimated normal. The number of displayed glyphs and their length are both configurable through slider controls.

An optional viewpoint oriented normal mode is also available. When activated, normals that face away from a specified viewpoint are automatically flipped so that all displayed glyphs point toward the observer. This technique, standard in surface reconstruction and geometry processing [6], resolves the inherent sign ambiguity of independently estimated normals and produces a more interpretable visualization of surface geometry.

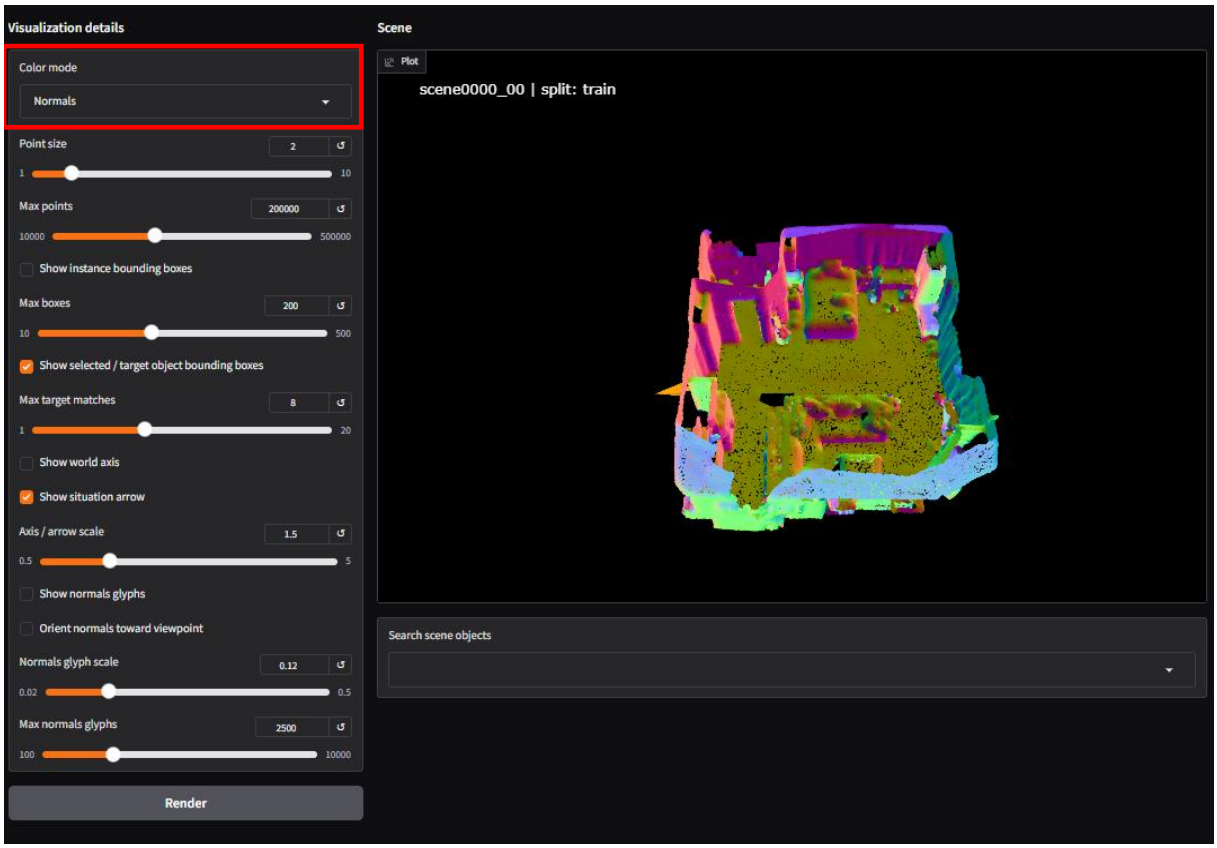

Figure I.5: Scene visualization in normals mode, where colors represent surface normal directions.

I.4 Object-level Interaction

The interface exposes interactive mechanisms for exploring individual objects within a scene. A searchable, multi select dropdown lists all annotated instances in the current scene, displaying each entry with its semantic class label, instance identifier, and point count. Users can search this list by class name or instance number and select one or more objects simultaneously.

Selected objects are immediately highlighted in the 3D view through colored bounding boxes drawn with a distinct palette separate from the global instance coloring, making it straightforward to locate and inspect specific objects of interest. Each selected box is labeled with its semantic class and instance identifier in the Plotly legend.

Complementing manual selection, the question driven automatic highlighting mechanism identifies instances semantically related to the currently selected question. This is accomplished through a tokenization pipeline that strips common stopwords, applies simple suffix normalization, and expands tokens using an alias table covering common naming variations, such as *fridge* for *refridgerator*. The resulting token set is matched against both instance-level semantic labels and the global ScanNet-20 category list. This functionality creates a direct visual link between natural language queries and their spatial referents within the scene.

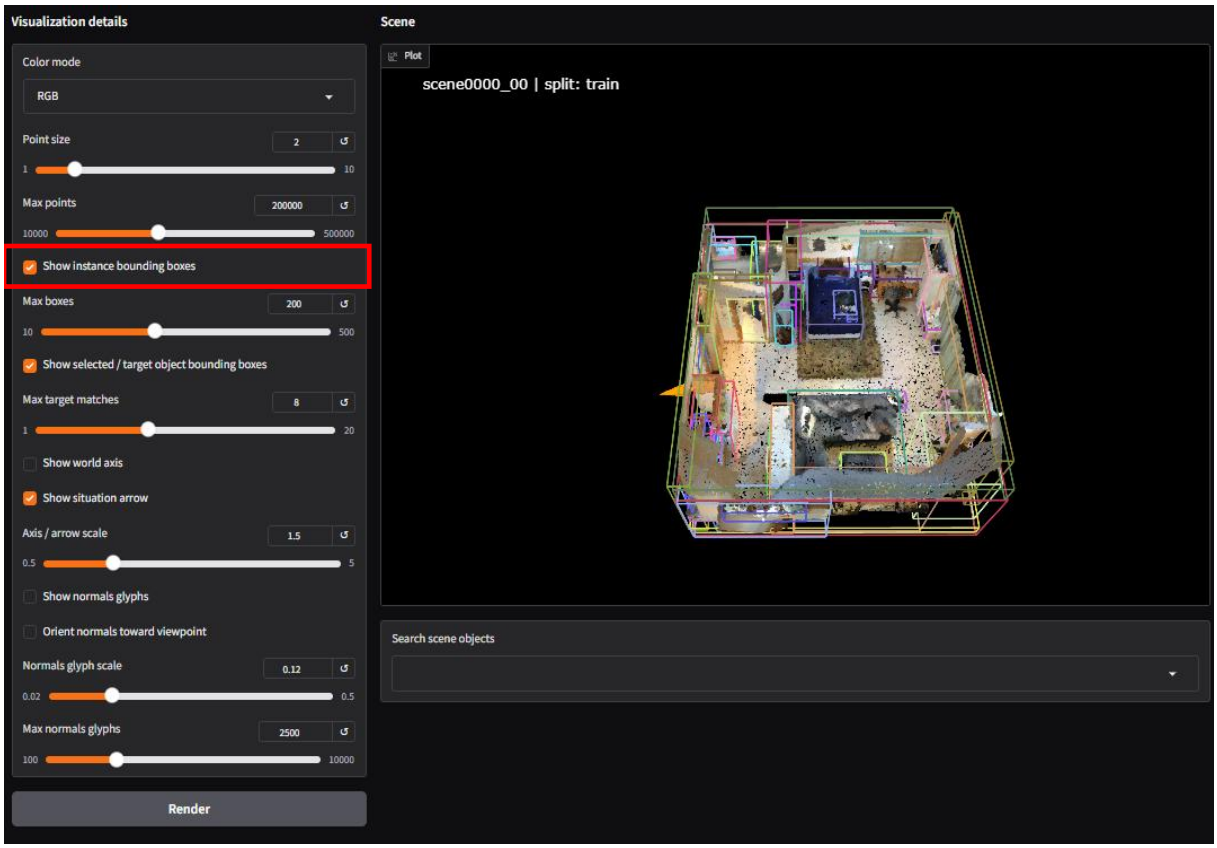

Figure I.6: Visualization of axis-aligned bounding boxes for detected object instances in the scene.

I.5 Model Interaction Pipeline

A chat interaction panel enables users to query the MSR3D model [34] using natural language, with the full scene and situation context automatically assembled and forwarded to the inference service. The conversation history is maintained across turns within a session, allowing the user to view the chat history and previous answers.

For each user query, the model interaction pipeline proceeds through the following stages.

1. **Scene and metadata retrieval.** The point cloud, object annotations, and QA metadata for the currently selected scene are retrieved from the cache, ensuring that inference operates on the same scene representation shown in the visualization.
2. **Context assembly.** The situation description associated with the selected QA sample is extracted and used to define the agent's spatial context. Users may optionally provide a custom situation override to replace the default annotation, enabling controlled experiments under user defined conditions.
3. **Pose override.** If the user provides custom location and orientation values, these are parsed from comma separated text inputs and used to redefine the agent anchor in the

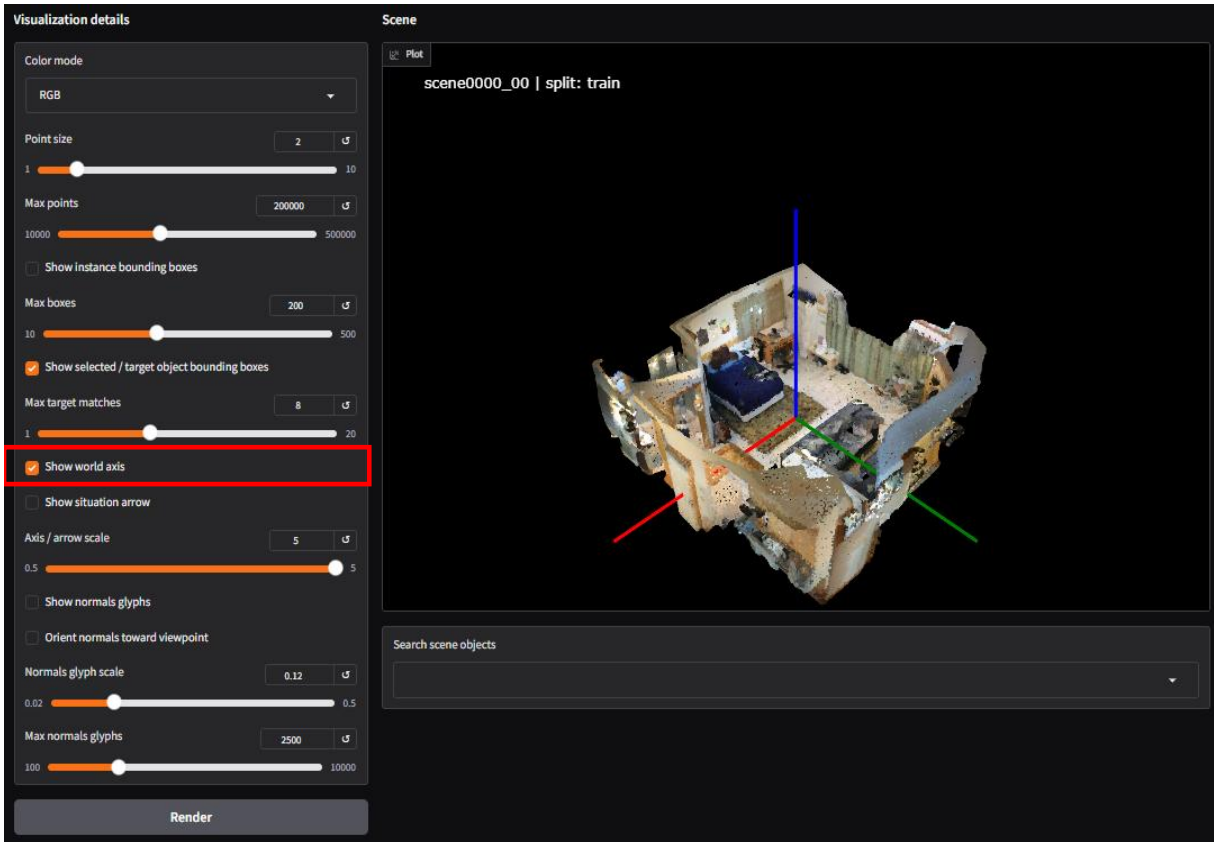

Figure I.7: World axis visualization showing the global coordinate frame within the 3D scene.

scene. Orientation may be specified as a quaternion (x, y, z, w) or as a 3D facing vector, with automatic conversion to the internal representation via SciPy rotation utilities [51].

4. **Multimodal image input.** Reference images may optionally be uploaded via the interface and forwarded to the model as additional visual context. This allows controlled experimentation with image conditioned queries that supplement the 3D scene information.

5. **Inference and response display.** The assembled query is forwarded to the MSR3D inference service, which runs model inference and returns a textual response. The response is appended to the chat dialogue and displayed alongside the user's question.

I.6 Multi Dataset Integration

Supporting three heterogeneous datasets within a single interface requires careful normalization of scene representations, annotation formats, and metadata conventions. A unified abstraction layer presents a consistent internal data structure to all downstream visualization and inference components, regardless of the originating dataset.

Each dataset is associated with a set of JSON annotation files for the training, validation, and

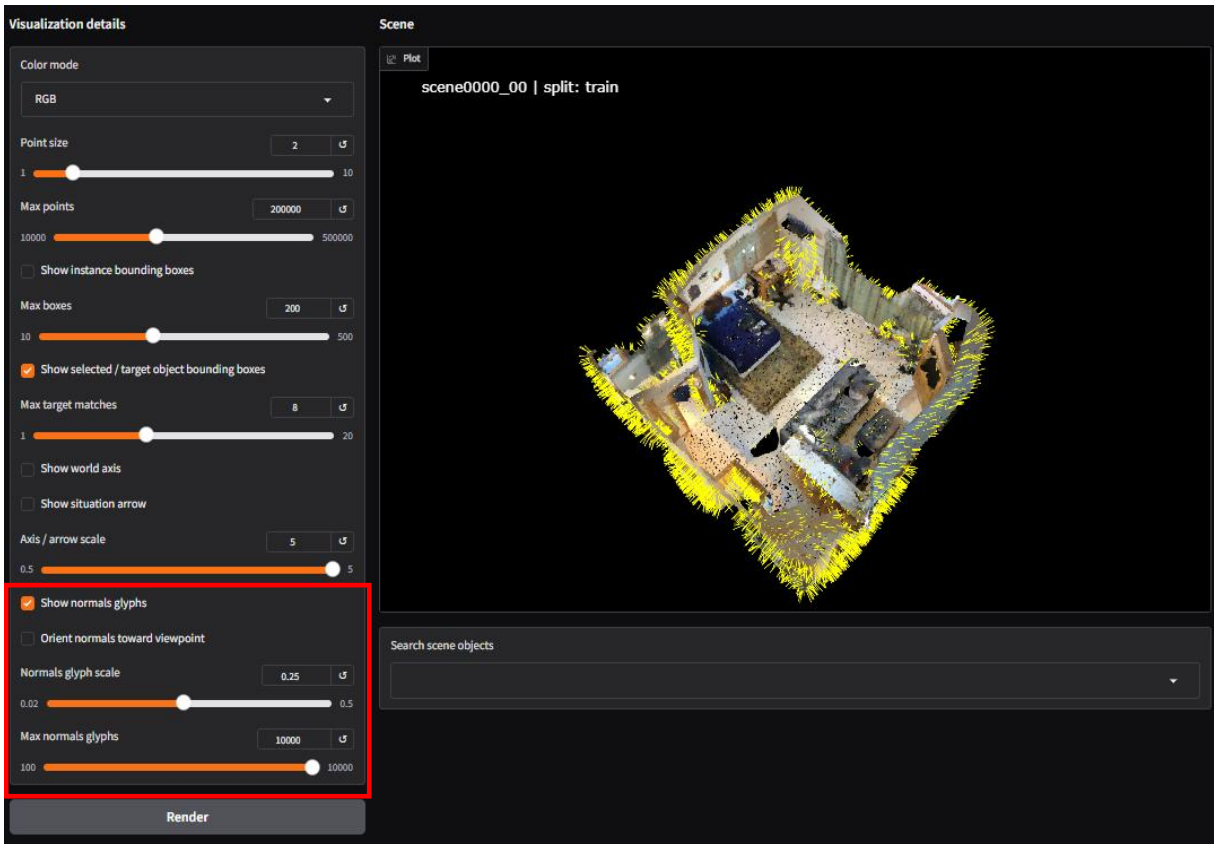

Figure I.8: Surface normal glyph visualization illustrating local orientation vectors across the scene geometry.

test splits, and a base directory containing the pre processed point cloud files. At application startup, all annotation files are loaded and indexed by scan identifier, enabling rapid retrieval of all QA samples associated with a given scene.

Dataset specific differences in label taxonomies are handled through explicit remapping procedures. For ScanNet, semantic labels are remapped from the NYU-40 [45] classification scheme to the ScanNet-20 subset using a precomputed lookup table derived from the official label mapping file. For 3RScan and ARKitScenes, instance to label mappings are loaded from dataset annotation files and normalized to a common string format. This ensures consistent labeling across all three datasets within the search and highlighting functionality.

Navigation across scenes is exposed through a cascading dropdown interface: selecting a dataset repopulates the scan identifier list, selecting a scan repopulates the QA sample list for that scene, and selecting a QA sample fills the question text field and updates the agent arrow position in the 3D view. A split filter allows restricting the QA list to the training, validation, or test partition of the selected dataset.

Figure I.9: Object search interface highlighting selected objects and their corresponding bounding boxes in the scene.

I.7 Performance Considerations

Efficient interaction with large indoor point clouds, which may contain several hundred thousand points, requires careful management of computational resources across both the server and client. Several complementary strategies are employed to maintain interface responsiveness under these conditions.

On the server side, scene loading and downsampling are treated as expensive operations that are triggered only when the user explicitly requests a scene refresh or changes the maximum point budget. All other user interactions, including color mode switching, bounding box toggling, normal glyph adjustments, and object selection, are handled as lightweight style updates that operate directly on the loaded scene. This separation is enforced through two distinct Gradio callback chains: a heavy render path that rebuilds the base figure, and a light update path that modifies only the overlay traces.

Dynamic point cloud downsampling is performed using uniform random subsampling controlled by the configurable maximum point count. A fixed random seed ensures deterministic downsampling, so that repeated renders of the same scene with the same parameters produce identical

visualizations. Surface normal glyphs are similarly subsampled, with the maximum glyph count independently adjustable to balance geometric detail against rendering performance.

Model weights are preloaded at application startup in a background thread, so that the first user query does not incur the full model initialization overhead. A status indicator in the interface reports whether the model has been loaded successfully, is currently loading, or encountered an initialization error, providing clear feedback to the user before any inference is attempted.

I.8 Implementation Details

The interface is implemented in Python using a combination of libraries for visualization, numerical computation, and model inference.

- **Gradio** [19]**.** Provides the web based user interface framework, including all input controls, reactive callbacks, and the chat dialogue component. The application is launched with public sharing enabled, making it accessible from remote machines without additional network configuration.
- **Plotly** [41]**.** Renders 3D point clouds, bounding box wireframes, coordinate axes, agent arrows, and normal glyphs as `Scatter3d` and `Cone` traces. In order to avoid full figure reconstruction, the figure state is maintained using Gradio state components.
- **PyTorch.** Handles loading of pre processed scene files stored in PyTorch tensor format, and provides utilities used throughout the data and model pipelines [40].
- **NumPy and SciPy.** Support all numerical processing operations, including point cloud subsampling, AABB computation, normal estimation, color mapping, and rotation conversion from quaternion and Euler representations [51, 22].

I.9 Example Interactions

To demonstrate the practical capabilities of the interface, three representative interaction examples are presented. These examples highlight how the system supports qualitative inspection of scene understanding, situated reasoning, and multimodal interaction within reconstructed 3D environments.

The first example illustrates standard question answering within a scene. A user query referring to objects located relative to the agent viewpoint is forwarded to the MSR3D model, while the interface automatically highlights the relevant target objects using semantic and instance based bounding boxes. This provides a direct visual connection between the natural language query and its spatial referents in the scene.

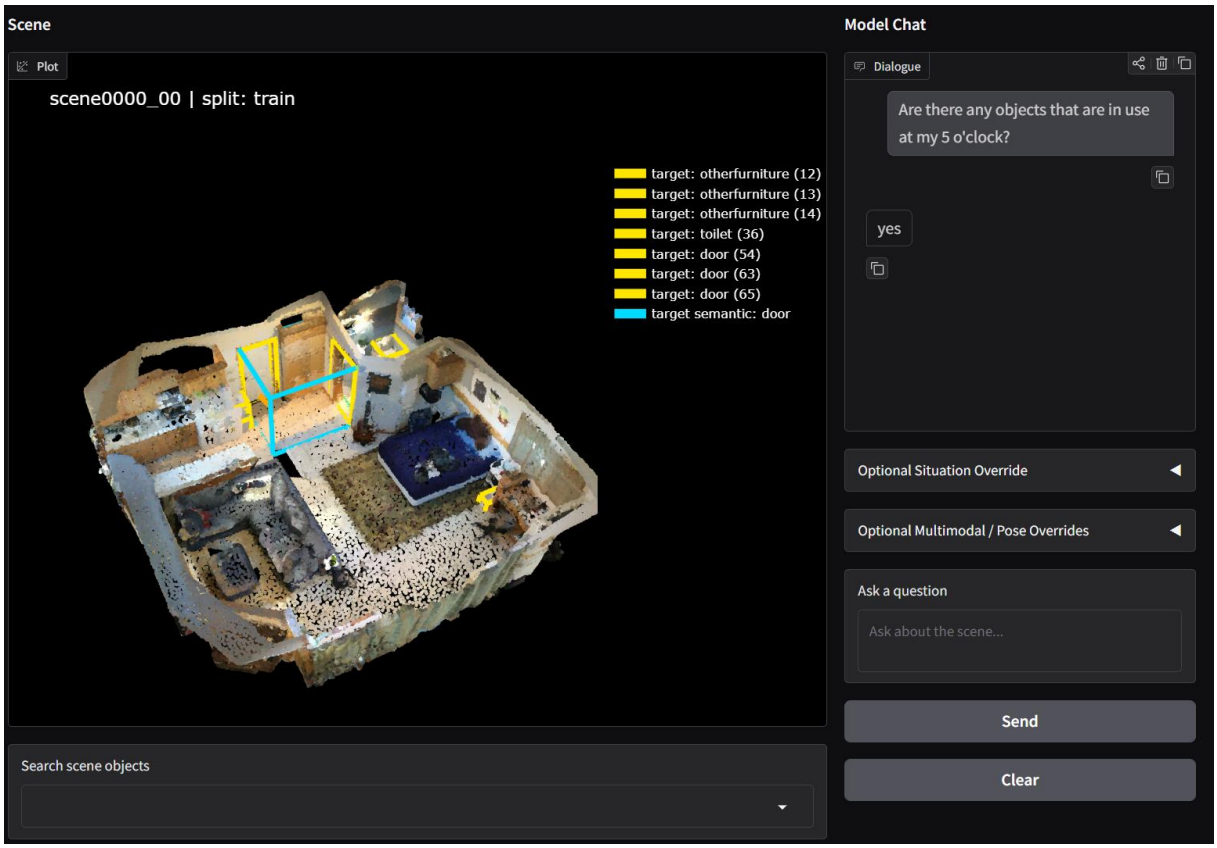

Figure I.10: Example question-answer interaction between the user and the scene understanding model.

The second example demonstrates custom situation overrides, where the default annotated situation description is replaced with a user defined context. This functionality enables controlled experimentation by allowing analysts to modify the agent's assumed state or position and observe how the model response changes under different contextual conditions.

The third example extends the interaction pipeline with multimodal inputs. In addition to the 3D scene and textual query, a reference image and custom pose specification are provided to the model. This allows the interface to support image conditioned scene reasoning, enabling richer forms of interaction that combine geometric, textual, and visual information within a unified framework. The overall design of the interface reflects a deliberate balance between expressiveness and performance. By decoupling heavy scene operations from visual updates, and by preloading the model before the first user interaction, the system provides a responsive and extensible platform for qualitative analysis of multimodal 3D scene reasoning.

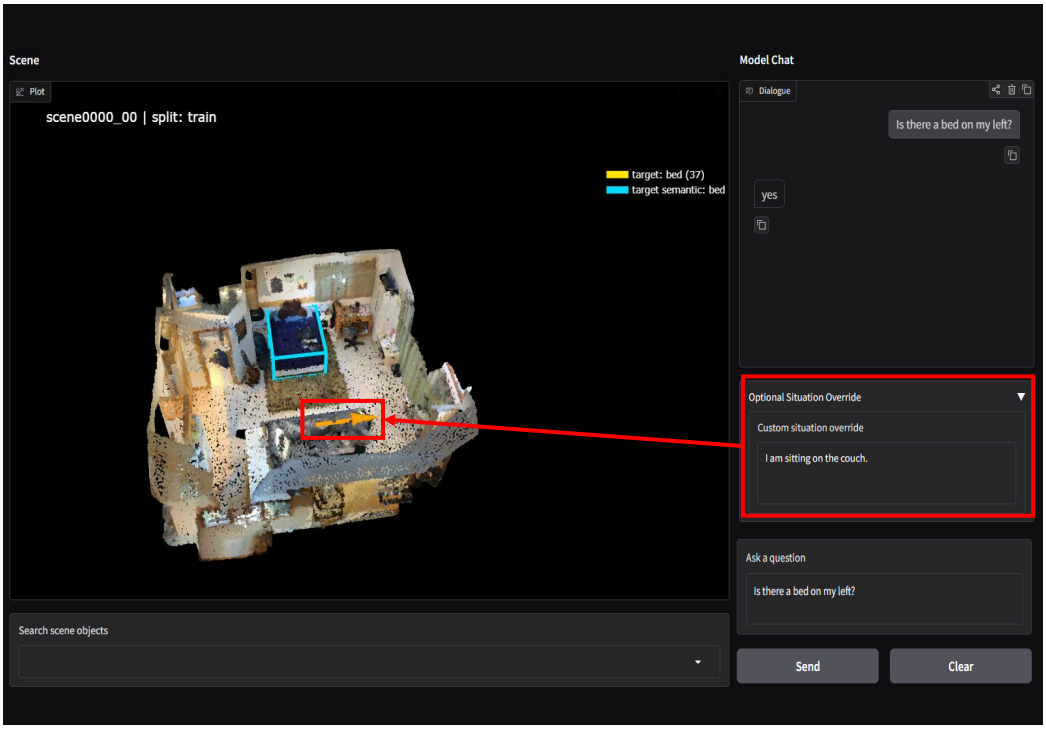

Figure I.11: Example of custom situation override input used to guide scene-based reasoning and question answering.

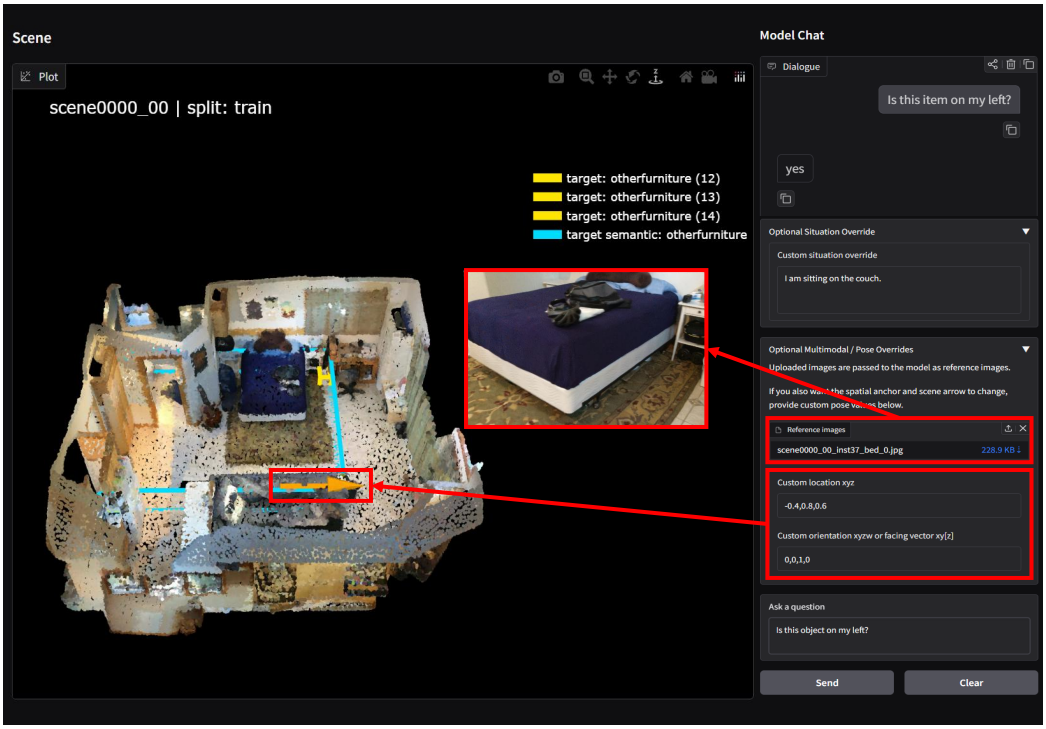

Figure I.12: Multimodal scene interaction example using reference images and custom pose/location overrides.

APPENDIX II

Point Transformer V3 Integration

Integrating PTv3 into the MSR3D pipeline required bridging two different processing assumptions: MSR3D represents 3D inputs through selected object point clouds, whereas PTv3, as implemented in Pointcept, operates on full scene point clouds. The integration layer therefore performs the necessary data conversion, scene batching, Pointcept preprocessing, feature restoration, and instance pooling needed to preserve compatibility with the downstream MSR3D modules [57, 34].

The integration is implemented mainly through two components. The first component, `PTV3DataProcessing`, converts MSR3D point cloud data into the dictionary format used by Pointcept and applies the corresponding Pointcept transformation pipeline. The second component, `PTv3PcdObjEncoder`, wraps the PTv3 model inside the MSR3D vision module interface. This design allows the internal 3D backbone to operate on full scenes while preserving the object-centric interface expected by the multimodal reasoning pipeline.

II.1 Input Tensor Conversion

The MSR3D dataloader provides the scene point cloud as a tensor

$$X \in \mathbb{R}^{N\times 9} \tag{II.1}$$

where each point is represented using coordinates, color, and surface normal attributes:

$$X_i = [x_i, y_i, z_i, r_i, g_i, b_i, n_i^x, n_i^y, n_i^z] \tag{II.2}$$

The first three channels correspond to the 3D coordinates, the next three channels correspond to RGB color, and the final three channels correspond to the estimated surface normal. Before the point cloud is passed to Pointcept, the tensor is decomposed into the fields

$$\texttt{coord} \in \mathbb{R}^{N\times 3}, \quad \texttt{color} \in \mathbb{R}^{N\times 3}, \quad \texttt{normal} \in \mathbb{R}^{N\times 3} \tag{II.3}$$

In addition to these geometric and appearance attributes, each point is associated with an instance

identifier and a semantic segment label. These are stored as

$$\texttt{inst\_id}, \quad \texttt{segment} \tag{II.4}$$

The instance identifiers are later used to pool point PTv3 features into object embeddings.

A practical compatibility issue arises from the color representation. In the MSR3D preprocessing pipeline, RGB values are stored in the normalized range $[-1, 1]$. However, the Pointcept transformation pipeline expects RGB values in the raw $[0, 255]$ range before applying its own color normalization. Therefore, the color channels are converted as

$$c_{\text{ptv3}} = \text{clip}((c_{\text{msr3d}} + 1) \cdot 127.5, 0, 255) \tag{II.5}$$

This conversion prevents the Pointcept normalization step from being applied to already normalized color values.

II.2 Surface Normals

The original MSR3D point representation contains coordinates and color information. For the PTv3 integration, this representation was extended by adding surface normal vectors, resulting in the nine channel point representation described above. The purpose of the normal channels is to provide local surface orientation information that is not directly represented by coordinates and color alone [57, 34].

Surface normals were computed as a preprocessing step before the PTv3 forward pass. For each point, a local neighborhood was selected using kNNs with $k = 30$. Given the neighboring points, a covariance matrix was computed from their coordinates. The normal direction was then estimated as the eigenvector corresponding to the smallest eigenvalue of the covariance matrix:

$$\mathbf{n}_i = \mathbf{v}_{\text{min}}(\mathbf{C}_i) \tag{II.6}$$

where $\mathbf{C}_i$ is the covariance matrix of the local neighborhood around point i. The smallest eigenvalue eigenvector corresponds to the direction of least local variance and therefore approximates the direction perpendicular to the local tangent plane.

The resulting nine channel representation is then preserved throughout the PTv3 data conversion stage. In the Pointcept dictionary, normals are stored explicitly under the `normal` field rather than being treated as generic additional features. This is important because Pointcept transformations may handle geometric vectors differently from scalar attributes. When geometric augmentation is applied, normal vectors must remain consistent with the transformed

coordinate frame [57].

II.3 Batching Difference from the Original MSR3D Encoder

The main implementation challenge was the difference between the batching assumptions of the original MSR3D 3D encoder and PTv3.

The original MSR3D point cloud encoder follows an object-centric batching strategy. Each scene is first decomposed into object point clouds, and each object is represented using a fixed number of sampled points. Conceptually, the input can be viewed as

$$B \times O \times P \times C \tag{II.7}$$

where B is the batch size, O is the maximum number of objects per scene, P is the maximum number of sampled points per object, and C is the number of point attributes. In this representation, each object can be encoded independently.

PTv3 uses a different assumption. Instead of processing each object separately, PTv3 is designed to operate on scene point clouds. Therefore, the PTv3 integration changes the batching strategy from object-centric batching to scene centric batching. The full scene is processed first, and only after PTv3 feature extraction are the point features grouped back into object embeddings [57, 34].

The MSR3D dataloader provides all points from the scenes in a batch as a concatenated tensor, together with a `scene_offset` field. This field marks the boundaries between scenes:

$$\texttt{scene}_b = X[\texttt{scene\_offset}_b : \texttt{scene\_offset}_{b+1}] \tag{II.8}$$

The encoder first reconstructs each scene independently using these offsets. For each scene, a Pointcept dictionary is created with the fields

$$\texttt{coord}, \texttt{color}, \texttt{normal}, \texttt{inst\_id}, \texttt{segment}, \texttt{name} \tag{II.9}$$

The important distinction is that Pointcept transformations are applied independently to each scene before collation. This avoids applying geometric transformations across an artificial concatenation of unrelated scenes. After each scene is transformed, the processed scene dictionaries are collated using Pointcept's `collate_fn`. The resulting batch is represented as a flat point

cloud with an `offset` tensor that identifies scene boundaries inside the collated Pointcept batch:

$$N_{\text{total}} = \sum_{b=1}^{B} N_b \tag{II.10}$$

Thus, the PTv3 batch is not a fixed object tensor. It is a variable length concatenation of scene point clouds, together with offsets that preserve scene membership.

II.4 Pointcept Transformation Pipeline

The adapter uses the transformation pipeline defined in the PTv3 Pointcept configuration file. During training, the training transform list is used. During validation or inference, the validation transform list is used. This separation is necessary because training may include stochastic augmentations, while validation and inference should remain deterministic.

The conversion process follows these steps:

1. Read the MSR3D batch containing `scene_fts`, `instance_ids`, `segments`, `selected_obj_ids`, and `scene_offset`.
2. Split the concatenated scene tensor into individual scenes using `scene_offset`.
3. Convert each scene into a Pointcept dictionary.
4. Apply the Pointcept train or validation transforms to each scene independently.
5. Collate the processed scenes into a single Pointcept batch.
6. Move the nested Pointcept dictionary to the active device.

This design keeps the PTv3 preprocessing compatible with Pointcept while still accepting the original MSR3D dataloader output.

II.5 PTv3 Backbone Execution

After collation, the Pointcept batch is passed to the PTv3 backbone. The backbone produces point features:

$$F = \{f_i\}_{i=1}^{N_{\text{total}}}, \quad f_i \in \mathbb{R}^d \tag{II.11}$$

In the current implementation, the feature dimension is

$$d = 64 \tag{II.12}$$

The encoder supports both frozen and trainable PTv3 execution. When the PTv3 backbone is

frozen, the forward pass is executed under `torch.no_grad()` and `torch.inference_mode()` to reduce memory usage. When CUDA is available, automatic mixed precision is used to further reduce memory requirements during the backbone forward pass. The underlying PTv3 configuration also enables FlashAttention [14] for the attention layers, which improves the efficiency of scene self attention by reducing memory usage and increasing throughput.

II.6 Inverse Mapping and Restoration of Original Point Structure

Some Pointcept transformations may downsample, reorder, or voxelize points. In such cases, Pointcept provides an `inverse` mapping that maps processed features back to the original point structure [57]. The PTv3 integration explicitly handles this case.

If an `inverse` field exists, the PTv3 features are remapped as

$$F_{\text{original}} = F_{\text{processed}}[\texttt{inverse}] \tag{II.13}$$

The original instance identifiers are also restored using `origin_inst`. This restoration is required because object pooling depends on the original instance labels. If pooling were applied directly after transformation without inverse remapping, the selected object identifiers from MSR3D could become misaligned with the transformed point set.

After inverse remapping, the original MSR3D `scene_offset` is reintroduced as the offset used for object pooling. This ensures that each scene is pooled according to its original point boundaries. If no inverse mapping is present, the Pointcept collated `offset` is used instead.

II.7 Instance Aware Object Feature Pooling

Although PTv3 produces point features, the downstream MSR3D modules expect object embeddings. Therefore, the point features must be aggregated into a fixed set of object embeddings.

For each scene in the batch, the restored offset tensor as shown in equation II.8 is used to isolate the corresponding point features. The instance identifiers for the same range are used to group points by object.

A batching issue arises because instance identifiers are locally defined within each scene. For example, object id 3 may appear in multiple scenes in the same batch. To avoid collisions, the implementation uses a globalized instance id:

$$g = b \cdot K + l \tag{II.14}$$

where b is the scene index, l is the local instance id, and K is a fixed constant large enough to separate the instance id ranges of different scenes.

For each selected object id, the corresponding global id is computed and used to retrieve the point features belonging to that object. The object embedding is then computed by mean pooling:

$$e_{b,o} = \frac{1}{|\mathcal{P}_{b,o}|} \sum_{i \in \mathcal{P}_{b,o}} f_i \tag{II.15}$$

where $\mathcal{P}_{b,o}$ is the set of points assigned to object o in scene b.

The pooled output has shape

$$E \in \mathbb{R}^{B \times M \times d} \tag{II.16}$$

where B is the batch size, M is the maximum number of selected object slots, and d is the PTv3 feature dimension. A boolean object mask:

$$M_{\text{obj}} \in \{0, 1\}^{B \times M} \tag{II.17}$$

is produced in parallel. A mask value of one indicates that the corresponding object slot was successfully matched to points after PTv3 preprocessing, while a mask value of zero indicates an invalid or unmatched object slot.

II.8 Forward Pass Summary

Algorithm 1 outlines the forward pass of the PTv3-based object encoder at implementation level. The input MSR3D batch is first decomposed into geometric and appearance attributes, namely coordinates, colors, and normals, while instance identifiers, segment information, and scene metadata are attached to each point. Since the batch contains multiple concatenated scenes, the points are split using `scene_offset` so that the Pointcept preprocessing pipeline can be applied independently to each scene.

After transformation, the scenes are collated again and passed through the PTv3 backbone to produce point-level features. When an `inverse` mapping is available, the features are mapped back to the original point ordering, ensuring that they remain aligned with the original instance identifiers. Finally, point features belonging to the same selected object instance are aggregated into a single object-level embedding, and a validity mask is returned to indicate which object slots correspond to valid objects.

Algorithm 1 PTv3 object encoder forward pass

Require: MSR3D batch containing `scene_fts`, `instance_ids`, `segments`, `selected_obj_ids`, and `scene_offset`
Ensure: Object embeddings and object mask
Convert `scene_fts` into `coord`, `color`, and `normal`
Attach `inst_id`, `segment`, and scene name fields
Split the concatenated batch into individual scenes using `scene_offset`
for each scene **do**
    Apply the Pointcept train or validation transform pipeline
Collate transformed scenes using Pointcept's `collate_fn`
Move the collated dictionary to the active device
Run the PTv3 backbone to obtain point-level features
if `inverse` mapping exists **then**
    Restore point features to the original point ordering
    Restore original instance identifiers using `origin_inst`
    Reconstruct offsets from the original `scene_offset`
else
    Use the Pointcept collated `offset`
Pool point features into object embeddings using instance identifiers and `selected_obj_ids`
Return object embeddings and valid object mask

II.9 Configuration and Runtime Adjustments

The PTv3 encoder is selected through the MSR3D vision configuration. The configuration specifies the encoder name, the PTv3 Pointcept configuration path, the pretrained checkpoint path, the feature dimension, and whether the backbone is frozen.

```
vision:
  name: PTv3PcdObjEncoder
  args:
    embedding_size: 64
    freeze: False
    grid_size: 0.02
    feat_reduce: "mean"
    sem_num_classes: 607
    ptv3_cfg_path: ".../configs/scannet/semseg-pt-v3m1-1-ppt-extreme.py"
    weight_path: ".../model_best.pth"
```

Several implementation details were required for stable execution. Firstly, checkpoint loading handles both standard and distributed checkpoints by adding or removing the `module.` prefix when necessary. This allows pretrained Pointcept weights to be loaded even when the checkpoint

and current model wrapper use different naming conventions.

Additionally, the encoder includes a recursive device transfer function for nested Pointcept dictionaries. This is necessary because Pointcept batches contain tensors inside dictionaries, lists, and other nested containers. Moving only the top level object to the GPU would not be sufficient.

II.10 Output Compatibility with MSR3D

The final purpose of the integration layer is to make a scene PTv3 backbone behave like an MSR3D object encoder from the perspective of the downstream model. Internally, PTv3 processes the full scene and produces contextualized point features. Externally, the module returns the same type of output expected by MSR3D [57, 34] as shown in equation II.16 together with a valid object mask as shown in equation II.17.

This preserves compatibility with the multimodal fusion pipeline. The language and image modules do not need to distinguish between embeddings produced by the original PointNet++ object encoder and embeddings produced by full scene PTv3 encoding followed by instance pooling. As a result, the integration changes the internal 3D feature extraction mechanism while keeping the downstream MSR3D interface unchanged.