Politecnico di Torino

Master's Degree in Computer Engineering

A.Y. 2025/2026

Graduation Session March 2026

RGB-only Active 3D Scene Graph Generation for Indoor Mobile Robots

Supervisors:

Daniele De Martini
Giuseppe Averta
Davide Buoso

Candidate:

Giorgia Modi

This thesis presents the results of a six-month research project conducted in collaboration with the **Oxford Robotics Institute** at the **University of Oxford**.

Abstract

Three-dimensional scene graphs provide robots with structured, queryable representations of their surroundings by encoding objects, their attributes, and spatial relationships in a unified graph. Existing methods for constructing such graphs typically require depth sensors and predefined exploration trajectories, limiting their applicability to platforms equipped with specialised hardware and carefully planned data collection.

This thesis presents a framework for *active*, *incremental* construction of 3D scene graphs from *RGB images only*. The framework is based on two main contributions. First, an *RGB-only 3D scene graph generation pipeline* that couples MapAnything [1], a state-of-the-art feed-forward 3D reconstruction model, with ConceptGraphs [2], an open-vocabulary semantic mapping system, replacing the original LLM-based edge inference with a deterministic geometric edge generator. Quantitative evaluation on the Replica dataset [3] shows that the proposed pipeline achieves scene graph quality comparable to the ground-truth-depth baseline, confirming that learned depth and pose predictions are sufficient for accurate scene graph construction. Second, an *active exploration framework* that drives a mobile robot to select viewpoints maximising scene understanding with minimal observations. Two complementary strategies are integrated: the Surface Edge Explorer (SEE) [4, 5], a geometric density-based method targeting under-observed surfaces, and Active Semantic Perception (ASP) [6], a semantic-driven method using large language models to analyse the current 3D scene graph and guide exploration toward informative viewpoints. Experiments on the ReplicaCAD dataset [7] show that ASP, by leveraging the structured semantic information in the evolving scene graph, significantly outperforms frontier-based exploration: in 30 steps it detects more than twice the number of objects compared to SEE, achieving scene graph quality comparable to that obtained from hundreds of pre-recorded images.

Additionally, the thesis investigates *fixed external cameras* as a low-cost complement to egocentric observations, showing that overhead RGB cameras can bootstrap the scene graph before exploration and serve as standalone sources for scene understanding when no robot is available.

Together, these results show that 3D scene graphs are not only a powerful structured representation of the environment, but also an effective guide for exploration, enabling a perception–action loop in which the scene graph drives viewpoint selection and each new observation refines the graph. By relying solely on RGB images, the framework also reduces hardware requirements, making active 3D scene graph generation accessible to a wider range of robotic platforms.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

Modern robotic systems increasingly depend on rich visual scene understanding to perceive, decide, and act effectively. For autonomous robots operating in human environments such as homes, offices, and hospitals, this understanding must go beyond simply detecting objects: robots require a structured representation that captures *what* is present, *where* it is in three-dimensional space, and *how* entities relate to one another. Object-centric outputs from standard perception modules (classification, detection, segmentation) are often insufficient, because they do not explicitly encode relational context and therefore struggle to distinguish between semantically different but visually similar situations.

3D scene graphs address this limitation by representing an environment as a graph whose nodes correspond to object instances annotated with semantic labels and 3D locations, and whose edges capture pairwise spatial relationships [8, 9]. This compact and queryable structure supports a broad range of downstream robotic capabilities, including language grounding and task planning [10], navigation [11], and manipulation in partially observed scenes [12]. In this view, the scene graph acts as an intermediate layer between low-level perception and high-level decision making.

Despite this promise, constructing accurate 3D scene graphs in realistic conditions remains challenging. First, most state-of-the-art systems [2, 13, 14] rely on RGB-D cameras or LiDAR to obtain depth and pose information for metric 3D reconstruction, restricting deployment to platforms equipped with dedicated depth sensors and excluding scenarios where only commodity RGB cameras are available, such as surveillance infrastructure, smartphones, or low-cost robots. Second, existing pipelines usually assume that a complete set of observations is given in advance (for example, a pre-recorded video trajectory) and focus solely

on how to build the scene graph from that data, without addressing the active perception question of which viewpoints would be most informative. This passive acquisition leads to redundant observations and may still leave important regions of the scene unobserved. Third, evaluation of open-vocabulary 3D scene graphs at scale is difficult: protocols such as the original ConceptGraphs assessment [2] rely on manual human annotation, which is labour-intensive, partially subjective, and not practical when many scenes and exploration strategies must be compared systematically.

Within this context, the central research question of this thesis is: *Can accurate 3D scene graphs be constructed incrementally (updating the graph step by step as new observations arrive) and actively (selecting viewpoints based on expected information gain) using only RGB images?*

1.2 Motivation

Recent advances in two distinct research areas create a unique opportunity to tackle the challenges outlined above.

On the *3D reconstruction* side, feed-forward models such as MapAnything [1] and its precursors DUSt3R [15] and VGGT [16] can now predict dense depth maps, camera poses, and a global metric scale factor from a set of uncalibrated RGB images in a single forward pass, achieving reconstruction quality competitive with classical Structure-from-Motion pipelines at a fraction of the computational cost. This makes it feasible to replace dedicated depth sensors with a learned depth-and-pose backbone without sacrificing the geometric consistency required for downstream scene graph construction.

On the *scene understanding* side, open-vocabulary 3D scene graph methods, notably ConceptGraphs [2], leverage large foundation models (Segment Anything Model (SAM) [17] for class-agnostic segmentation, Contrastive Language–Image Pre-training (CLIP) [18] for semantic embeddings, and Large Language Models (LLMs) for relationship inference) to construct flexible scene graphs that are not limited to a predefined set of object categories. However, ConceptGraphs was originally designed for RGB-D input and has not yet been evaluated in a purely RGB-based setting.

In parallel, active perception research [19, 20] has shown that intelligent viewpoint selection can dramatically reduce the number of observations needed to build accurate 3D representations. Existing approaches span purely geometric criteria, such as frontier-based exploration that explicitly targets the boundary between known free space and unknown regions [4, 5], as well as semantic-driven strategies that reason about object- and scene-level uncertainty, including recent LLM-guided scene completion methods [6]. Combining these exploration strategies with an

RGB-only scene graph pipeline promises to close the loop between perception and action: the robot observes, builds a scene graph, reasons about what is still unknown, moves to an informative viewpoint, and repeats.

Finally, the widespread availability of fixed RGB cameras in indoor environments (e.g. security or wall-mounted cameras) suggests an additional, largely unexplored modality for scene graph construction. Because the proposed pipeline operates solely on RGB images, it can be applied directly to external viewpoints. In this setting, fixed cameras provide complementary, often overhead views of the scene that are not accessible to the robot's onboard sensor, and thus offer additional support for initialising a global 3D scene graph and for maintaining it over time as the environment changes.

1.3 Contributions

The first contribution of this thesis is an **RGB-only 3D scene graph generation pipeline**. MapAnything is integrated with ConceptGraphs to construct 3D scene graphs from uncalibrated RGB images, eliminating the need for depth sensors. The pipeline includes a deterministic geometric edge generator that replaces the original LLM-based edge inference, removing dependence on proprietary models while producing reproducible, low-latency spatial relationships (*on top of*, *next to*, *inside*, *under/over*, *supported by*). Quantitative evaluation on the Replica dataset demonstrates that the RGB-only pipeline matches the ground-truth-depth baseline, validating the viability of learned depth and pose for scene graph construction.

The second contribution is a **study of active exploration strategies for 3D scene graph construction**. A central question addressed in this thesis is whether active viewpoint selection can improve scene graph quality over passive observation, and whether the 3D scene graph itself can serve as an effective representation for guiding exploration. To investigate this, two complementary next-best-view strategies are adopted and integrated into the RGB-only pipeline: the Surface Edge Explorer (SEE) [4, 5], a geometric density-based method that operates directly on the reconstructed point cloud, and Active Semantic Perception (ASP) [6], a semantic-driven method that reasons over the current 3D scene graph and uses LLM-sampled completions to maximise expected information gain. Both methods required several modifications to operate within the RGB-only setting and on a ground-based mobile robot, and the two strategies are evaluated in the Habitat simulator on the ReplicaCAD dataset [7] over multi-step exploration episodes with multiple starting positions.

The third contribution is a **study of the integration of external cameras** for scene graph construction. Fixed RGB cameras are studied in two complementary settings: combined with onboard sensing during active exploration, to assess their

contribution when additional viewpoints are available, and as a standalone source of observations, to evaluate how far scene graphs can be constructed from external infrastructure alone without any robot motion.

Finally, an **automated validation framework** is proposed for open-vocabulary 3D scene graphs. The framework performs joint semantic–geometric node matching and supports optional ICP-based alignment with scale estimation [21] and duplicate-node removal, enabling systematic comparison across scenes, pipelines, and exploration strategies without manual annotation.

1.4 Thesis Organisation

The remainder of this thesis is organised as follows. Chapter 2 reviews the literature on scene graphs in 2D vision and 3D robotics, 3D reconstruction from images (including classical, learning-based, and feed-forward methods), active perception and next-best-view planning, and the use of external cameras for robotic perception. Chapter 3 describes the proposed system in detail: the RGB-only pipeline (Section 3.2), including MapAnything-based reconstruction, ConceptGraphs-based scene graph construction, and geometric edge generation; the automated validation framework (Section 3.3); and the active exploration strategies, both geometric (SEE, Section 3.4.2) and semantic (ASP, Section 3.4.4), together with the modifications introduced to adapt them to the RGB-only setting and deployment on a ground-based mobile robot. Chapter 4 presents three experimental studies: static validation of the RGB-only pipeline on the Replica dataset, active exploration evaluation on the ReplicaCAD dataset in the Habitat simulator, and external-camera-only scene graph construction across 90 indoor scenes. Chapter 5 discusses and contextualises the experimental findings, while Chapter 6 summarises the main outcomes and outlines directions for future research.

Chapter 2

Related Work

This chapter reviews prior work that supports this thesis in four areas: (i) scene graphs in vision and robotics, (ii) 3D reconstruction from images, (iii) active perception and exploration, and (iv) the use of external cameras for robotic perception and control. Together, these topics provide the foundations and motivations for the methods proposed in Chapter 3.

2.1 Scene Graphs

2.1.1 Scene Graphs as Structured Representation for Scene Understanding

Research in visual scene understanding focuses on enabling machines to derive an interpretable description of a scene from sensory input, inspired by human-level perception. Humans can quickly grasp the essence of a scene and infer its deeper meaning, a capability that machines are striving to replicate. Initially, the field concentrated on object-centric tasks like image classification, object detection, and semantic segmentation. Over time, the focus has shifted toward more complex tasks, such as understanding relationships and interactions between objects [8, 9]. Purely object-centric representations often fall short, as they fail to distinguish between different scenarios that involve the same set of objects, which can only be clarified through explicit relational modeling (e.g., "a man feeding a horse" versus "a man standing by a horse") [22]. For this reason, several works identify effective scene representations as a key bottleneck for deeper scene understanding and motivate abstractions that couple object semantics with relational context for higher-level reasoning and downstream tasks [9].

Scene graphs were introduced as an explicit representation to address this limitation by modeling a scene as a graph of interacting entities. In a scene graph,

nodes correspond to object instances annotated with semantic labels and attributes, while edges capture pairwise relationships such as spatial configurations, actions, or functional interactions. Explicitly encoding relational context helps resolve ambiguities that arise in purely object-centric representations, as illustrated in Fig. 2.1 [22]. By transforming raw perceptual data into a compact, abstract, and interpretable structure, scene graphs act as an intermediate representation between visual input and semantic understanding [9, 23].

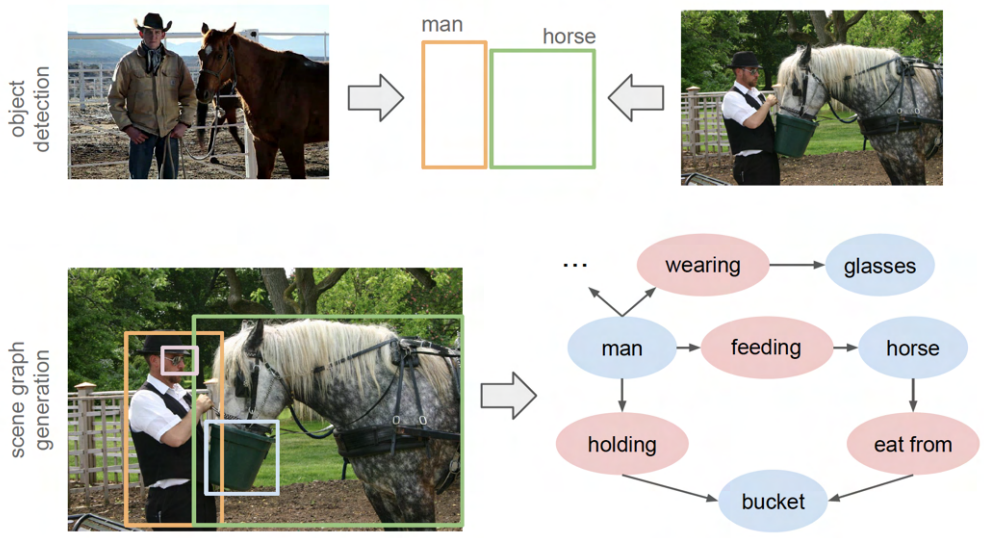


Figure 2.1: An object detector (top) may identify individual objects in a scene but fails to capture their relationships, leading to ambiguity. A scene graph (bottom) explicitly models objects as nodes and their relationships as edges, providing a richer understanding of the scene context (e.g., distinguishing between "man feeding horse" and "man standing by horse"). Image adapted from [22].

Early formulations of scene graphs primarily focused on 2D images, where relationships are inferred from visual appearance and projected spatial arrangements. While effective in image-centric tasks, 2D scene graphs are inherently limited by viewpoint dependency and ambiguities in spatial reasoning. Extending scene graphs to three dimensions preserves the same underlying abstraction while grounding entities and relationships in metric space, thereby resolving many of these ambiguities. This extension is particularly important in robotics, where autonomous agents must continuously interpret and act within complex, three-dimensional, and temporally dynamic environments. Robots require representations that not only recognize objects, but also capture their spatial, functional, and relational structure in support of navigation, manipulation, and task-level reasoning [24]. Scene graphs therefore provide a natural conceptual bridge between visual perception and

embodied intelligence, motivating their adoption as a foundational representation for robotic scene understanding.

Further discussion follows, first covering 2D scene graph generation and then 3D scene graphs in robotics.

2.1.2 Scene Graphs in 2D Vision

Although this thesis focuses on robotic perception and 3D scene graphs, the main conceptual and methodological foundations of scene graphs were established in 2D vision.

The notion of a scene graph was introduced by Johnson et al. [25] in the context of image-based scene understanding and retrieval, demonstrating that graph-structured representations of objects and their relations support more expressive, compositional queries (e.g., "find images with a man riding a bicycle") than flat keyword tags. The release of Visual Genome [26] then became a key incentive by providing large-scale annotations of objects, attributes, and relationships, effectively standardizing the *Scene Graph Generation* (SGG) problem: given an image, predict a set of object instances and labeled relations (predicates) for relevant object pairs.

Extensive surveys review the SGG problem and its main methodological trends [23, 9]. In the standard 2D setting, most SGG methods follow a two-stage pattern: object proposals (typically from detectors) are produced first, and realtionship classification is performed over object pairs using visual, geometric, and contextual features. Much of the progress has therefore focused on *context modeling* (i.e. enriching features from individual objects and object pairs with global information from the image) to resolve relational ambiguity, ranging from iterative message passing over candidate graphs [22] to global-context baselines that exploit recurring structural regularities in datasets [27]. Other approaches explicitly reduce the quadratic number of object pairs by first proposing a small set of likely relations and then refining them with graph reasoning, as in Graph R-CNN, which uses a Relation Proposal Network followed by an attentional GCN [28].

More recent 2D methods increasingly move toward *one-stage* and more *end-to-end* formulations, often leveraging attention/Transformer-style architectures to jointly predict objects and relations [9]. Here, *one-stage* denotes approaches that infer objects and relations within a single integrated model, rather than using a detector-first, multi-stage pipeline with separate modules for proposal generation and predicate classification. Transformer-based methods such as RelTR[29] and SGTR[30] formulate SGG as a set-prediction problem and use self-attention to jointly encode object representations and their relational context, achieving strong results on Visual Genome and related benchmarks[9]. These architectures simplify the pipeline and tend to yield more consistent relational predictions.

Overall, 2D scene graphs have become a widely used framework for relational

scene understanding, but they remain limited by their image-centric nature: they do not directly encode 3D geometry, metric consistency across viewpoints, or persistent world state, which are central requirements for embodied robotics, where an agent moves through and interacts with the environment over time, and motivate the transition to 3D scene graphs in the next sections.

2.1.3 3D Scene Graphs in Robotics

3D scene graphs (3DSGs) serve as a unified representation that bridges metric geometry with semantic and relational knowledge in a scene, enabling tasks ranging from navigation and manipulation to high-level planning. This section is based on the recent survey by Catalano et al. [24], which gives a comprehensive overview of 3DSG generation methods in robotics, categorizing them based on key design choices and methodological paradigms.

Taxonomy of 3D Scene Graph Generation Approaches.

A fundamental design choice is whether the scene graph is *hierarchical* or *flat*. Early works like Armeni et al.[31] introduced a multi-layer 3D scene graph for static indoor scans reconstructed offline, with nodes for buildings, rooms, objects, and cameras (top-down approach), and edges encoding relations like occlusion, physical support, containment and so on. In contrast, flat scene graphs like the one introduced by Kim et al. [32] focus on object instances and their pairwise relations, which can be particularly effective for object-focused reasoning or vision-question answering, but they struggle to represent large-scale structure (for example, a flat graph cannot directly answer a query like "find the nearest kitchen"). Figure 2.2 illustrates an example of flat and hierarchical 3D scene graphs.

Another example of hierarchical 3DSG is the Dynamic Scene Graph (Kimera-DSG) proposed by Rosinol et al. [33], which builds a 5-layer graph (metric mapping layer, objects, places, rooms, and agents) from sensor data using a bottom-up approach. This is achieved without requiring a prior map, by integrating semantic perception into a *Simultaneous Localization and Mapping* (SLAM) system, an approach that allows robots to construct a geometric and topological model of the environment while estimating their own position within it. Hybrid SLAM-based pipelines now form an important branch of 3DSG generation: they extend a SLAM backbone with semantic perception and graph structuring modules in a modular pipeline.

One representative of the SLAM-based approach is *Hydra* by Hughes et al. [13], a real-time spatial perception system that incrementally builds a hierarchical 3DSG from sensor data. Hydra explicitly represents multiple abstraction levels and focuses on real-time algorithms for incrementally constructing these layers as the robot

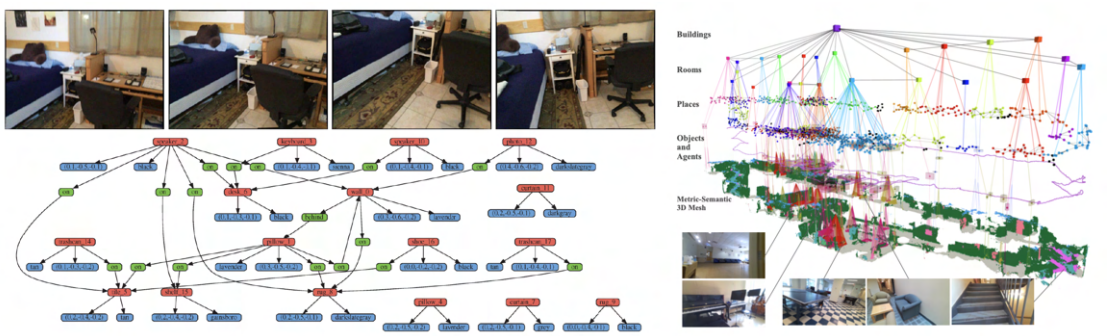


Figure 2.2: Flat vs. Hierarchical 3D Scene Graphs. A flat 3DSG (a) represents objects and their pairwise relations without higher-level structure [32], while a hierarchical 3DSG (b) organizes entities into multiple abstraction levels (e.g., buildings, rooms, places, objects, mesh), enabling richer spatial reasoning [13]. Image adapted from [8].

explores. Crucially, Hydra also studies loop closure *in the scene graph domain*: it introduces hierarchical descriptors for loop closure detection and proposes a deformation-graph-based optimization to coherently correct all layers of the 3DSG when loop closures occur, rather than rebuilding the representation from scratch.

In parallel, a different branch of research has explored learning-based 3DSG generation that minimizes hand-engineered mapping routines. Instead of sequentially detecting objects, assigning labels, and inserting them into a graph, these approaches train neural networks to directly predict the graph structure from sensory input. Wald et al. [34] were among the first to demonstrate an end-to-end 3DSG predictor: their framework takes an instance-segmented 3D point cloud as input and uses a PointNet-based backbone with Graph Convolutional Networks (GCNs) to jointly infer the graph's nodes and edges. In other words, the network outputs a structured graph (object categories and relationships) without explicit intermediate steps for data association or pose graph optimization. In addition, they introduce 3DSSG, a semi-automatically generated dataset, that contains semantically rich scene graphs of 3D scenes.

Another important development is SceneGraphFusion by Wu et al. [14], which brings learned scene graph prediction into an online RGB-D SLAM setting. SceneGraphFusion *incrementally* builds a scene graph in tandem with mapping: as new RGB-D keyframes arrive, a neural network predicts a local scene graph for the partial scene, and these subgraphs are then merged into a global graph. By learning a "same-part" relation to decide when two segmented regions belong to the

same object, the system can fuse observations over time and achieve instance-level segmentation on the fly.

A recent significant advancement in scene graph generation is the transition from close-vocabulary to open-vocabulary-based methods. Traditional scene graph generation approaches typically rely on pre-defined object categories and relationships, limiting their flexibility at inference time. Open-vocabulary methods, such as ConceptGraphs [2], tackle this challenge by leveraging class-agnostic segmentation and multimodal foundation models (LVLMs and LLMs) to generate flexible, object-centric 3D scene graphs in indoor environments. These models are able to adapt to new, previously unseen objects and relationships, marking a shift toward more adaptable, language-grounded scene representations. Expanding on this foundation, OpenGraph[35] extends the scope to hierarchical 3DSGs for outdoor driving environments, integrating 2D image-based features for higher-level semantic reasoning, while HOV-SG [11] focuses on geometric and topological structures for indoor navigation. This shift toward open-vocabulary models significantly increases the flexibility and versatility of scene graph generation. Since ConceptGraphs [2] is the paradigm used in this thesis, a more detailed discussion of its methodology and applications will follow in section 3.2.4.

In conclusion, 3D scene graph generation has progressed from traditional models to more flexible, open-vocabulary methods, enhancing adaptability and scalability. The following section explore how 3DSGs are applied in robotics.

Applications of 3D Scene Graphs in Robotics.

A key motivation for 3D scene graphs (3DSGs) is to provide embodied agents with a *persistent* and *queryable* world model that unifies semantics, geometry, and relations. Unlike point clouds or occupancy grids, a 3DSG explicitly encodes *what* entities exist (object/room semantics and attributes), *where* they are (3D pose/extent), and *how* they relate (spatial, functional, or interaction relations). Recent surveys emphasize that 3DSGs are not merely internal maps, but a substrate for reasoning, planning, and interaction across robotics tasks [24].

Language grounding and scene understanding. Scene understanding focuses on building a structured representation of the environment from visual and spatial data, while language grounding links natural language expressions to this representation. Together, they enable robots to interpret complex human instructions by mapping words and phrases to specific objects, relations, and configurations in the scene. This is a straightforward application of 3DSGs, where the structured graph representation naturally connects linguistic concepts to scene entities. ConceptGraphs [2] exemplifies this approach by constructing open-vocabulary 3D scene graphs that enable complex queries spanning multiple modalities. The

system supports *object retrieval* based on descriptive queries (e.g., "a potted plant"), affordance-based queries (e.g., "something to use for temporarily securing a broken zipper"), negation queries (e.g., "something to drink other than soda"), and complex visual-language queries that combine image context with text (e.g., "something this guy would play with" when shown an image). This capability enables robots to ground abstract linguistic instructions in the 3D world without task-specific training.

Task Planning and Manipulation Recent approaches integrate 3D scene graphs into task planners, particularly those leveraging large language models (LLMs) or symbolic planners, in order to bridge high-level instructions and low-level robot actions. SayPlan [10] exemplifies this by using a hierarchical 3DSG as a knowledge graph that the LLM can query. Instead of processing raw images or flat detections, the LLM accesses a structured representation, enabling semantic search over relevant subgraphs. For example, given the instruction "bring the first-aid kit from the bathroom to the living room," the planner can locate relevant nodes and objects through the graph's hierarchy, reducing search complexity and enabling scalability to large environments. Another line of work uses 3DSGs to break down natural-language instructions or high-level tasks into executable steps. Ni et al. propose GRID [36]: a Graph-based Robotic Instruction Decomposer that leverages scene graphs instead of raw images for task planning. The key idea is to use the scene graph as the perceptual input to an instruction interpretation model, rather than, say, a sequence of camera frames. In GRID, the current scene is represented as a graph (with object nodes and their relationships), which is fed into a Graph Attention Network together with the textual instruction encoding. By attending over the graph, the model can interpret references inthe instruction (e.g. "the cup on the table in the kitchen") and ground them to specific nodes in the scene graph. It then predicts a sequence of subtasks, each consisting of an action and a target object, effectively translating a complex instruction into a step-by-step plan. This approach highlights a general benefit of 3DSGs: they serve as a mediating representation between high-level reasoning (often done with language or symbolic AI) and the robot's perception/action space. By querying or updating the scene graph, planners ensure their decisions remain tethered to the physical reality of the environment.

Beyond instruction decomposition, several systems emphasize *interactive* manipulation under partial observability by constructing and traversing graph memories that encode occlusions and action effects. RoboEXP [12] builds an action-conditioned scene graph where edges capture how interactions change what can be observed or accessed (e.g., opening a cabinet reveals an object inside), enabling efficient multi-step retrieval and placement in cluttered scenes. CuriousBot [37] similarly maintains an actionable relational object graph (e.g., *inside/behind/under*)

and selects interaction primitives (push/open/lift) to uncover hidden objects and execute retrieval by graph traversal.

Robot Navigation and Localization In robotic navigation, 3D scene graphs provide a rich topological map augmented with semantics. Traditional navigation algorithms use metric maps or topological graphs of places; a scene graph can be seen as a topological map where nodes carry semantic labels (rooms, objects) and edges can denote connectivity or spatial relations. Kimera-DSG [33] pioneered the idea of building a dynamic metric-semantic scene graph during SLAM, organizing the environment into a multi-layer graph (e.g., objects, rooms, buildings) that enables hierarchical route planning and more informed decision-making than metric maps alone. More recent work focuses on using relational graphs for goal-directed navigation. For instance, Graph2Nav [38] generates an object-centric 3D scene graph with predicted semantic relationships in real time, and integrates it with an LLM-based planner to efficiently search for target objects – showing that encoding object relations (e.g. "table next to sofa") improves navigation success.

HOV-SG [11] enables language-grounded robot navigation in large-scale, multi-floor environments by allowing robots to understand and navigate based on natural language instructions. The system constructs a hierarchical scene graph consisting of floor, room, and object concepts, each enriched with open-vocabulary features derived from vision foundation models (CLIP, SAM). Beyond semantic abstraction, HOV-SG integrates an actionable navigational Voronoi graph spanning multiple floors, enabling high-level planning and low-level path execution. Evaluation demonstrates a 75% reduction in representation size compared to dense open-vocabulary maps.

In the context of localization, SceneGraphLoc [39] uses a 3DSG as a canonical map and localizes new camera views by matching a query scene's graph (constructed from detected objects) against the map's graph—achieving robust coarse localization across modalities. A similar approach for localization was proposed in ConceptGraphs [2], which uses a particle filter to localize a robot within a pre-built 3DSG by matching observed object nodes and relations against the graph. It's worth noting that early semantic mapping systems like S-Graphs [40] already demonstrated the localization potential of 3DSGs: S-Graphs built a three-layer LiDAR-based scene graph (objects, rooms, building) for indoor environments under Manhattan-world assumptions, allowing robots to localize themselves in a structured semantic map. These works highlight that 3DSGs can serve as a compact, queryable map for navigation tasks, in contrast to dense point clouds or occupancy grids.

2.1.4 Active and Incremental 3D Scene Graph Construction

Most of the works discussed above assume that a dataset or a fixed trajectory of sensor observations is given, and the focus is on how to build the best scene graph from that data. A relatively underexplored question is: *how should an autonomous agent actively choose where to look next in order to improve its scene graph?* In other words, *can a robot intelligently explore its environment so as to efficiently construct a more complete and correct 3D scene graph?* This touches on the field of active vision and autonomous exploration, intersecting with semantic mapping.

In fact, most existing 3D scene graph methods either run offline (taking a pre-reconstructed 3D scene or a dataset of images as input and computing the graph in batch) or remain passive (incrementally updating the graph from whatever sensor trajectory is provided without influencing where the robot looks next). For example, systems such as Hydra [13] and SceneGraphFusion [14] build 3D scene graphs incrementally and in real time from streaming RGB-D data, but they consume whatever trajectory is provided by SLAM or the dataset and do not optimize the robot's viewpoint selection themselves. In these pipelines, the challenge is how to update and maintain the scene graph given the incoming frames, rather than how to decide which observations would be most informative for the graph.

This stands in contrast to the *active perception* paradigm in robotics, where agents deliberately control their sensors to reduce uncertainty and gather task-relevant information. Work on active SLAM has long shown that robot actions can be chosen to minimize map uncertainty or to trigger loop closures for better localization [41], and next-best-view planning similarly frames sensor placement as an iterative decision problem for acquiring missing 3D information [42].

This thesis investigates whether these principles can be extended beyond purely geometric mapping to *semantic-aware exploration*, where exploration is guided not only by geometric uncertainty but also by the semantic state of the environment (see Section 2.3.4). In this setting, 3D scene graphs provide a structured representation that enables reasoning over objects, relationships, and unobserved regions, with the goal of achieving semantic completeness and correctness of the reconstructed 3D scene graph rather than geometric accuracy alone.

Recent research has started to address this. Li et al. [43] formally define the task of Embodied Semantic Scene Graph Generation, in which an agent must navigate an environment to build a scene graph rather than passively receive a fixed set of views. They explicitly argue that prior scene graph generation methods rely on data collected along pre-defined paths, which may be far from optimal for discovering objects and relations. Instead, their agent learns a navigation policy, via a combination of imitation learning and reinforcement learning, that chooses actions to expose unseen objects and relationships in the AI2-THOR simulator.

Quantitative results show that this active strategy yields more complete scene graphs with fewer frames than random or hand-crafted sweeps.

Concept-Guided Exploration [44] tackles a closely related problem in a real-robot setting, proposing a concept-first architecture in which asynchronous "concept agents" (e.g. for rooms and doors) maintain and update a 3D scene graph, while a central mission-monitoring agent sequences actions. Affordances are encoded as context-sensitive controllers that are triggered when their preconditions hold, so that task-directed motions (e.g. "find the next room") are interleaved with actions explicitly aimed at refining the internal scene representation, ensuring that the robot actively explores regions that can confirm or refute semantic hypotheses—such as whether a doorway actually leads to a new room—rather than passively accepting whatever the current trajectory reveals.

Tang and Chaudhari's Active Semantic Perception framework [6] takes a complementary approach, using large language models (LLMs) to reason about unobserved parts of the environment. Their system builds a hierarchical multi-layer scene graph over rooms, objects, structural elements (walls, doors, windows), and even "nothing" nodes that encode confirmed free space. Given partial observations, an LLM is prompted to generate plausible completions of the scene graph, effectively sampling multiple hypotheses about what unobserved rooms and objects might exist and where. These samples define an information-gain objective that scores candidate viewpoints according to how much they would reduce semantic uncertainty at both room and object level, allowing the robot to prioritize viewpoints that are likely to clarify, for example, which type of room lies behind a particular door. In simulation on HM3D environments, this semantic-driven strategy recovers the structure of the environment faster and with more accurate scene graphs than frontier-based or purely semantic-map baselines.

All three of these works share a common direction with this thesis: they treat the 3D scene graph not just as a passive product of perception, but as the object of the exploration strategy itself, using semantic priors and learned or model-based reasoning to guide where the agent looks next. The theoretical background on semantic-aware active perception and next-best-view planning will be discussed in more detail in Section 2.3.

2.2 3D Reconstruction

Three-dimensional reconstruction is a fundamental task in computer vision that aims to recover the spatial structure and geometry of scenes from two-dimensional images. In the context of this work, 3D reconstruction serves as an essential preprocessing step for 3D scene graph generation, as it provides both the geometric representation (point clouds) and camera poses required to extract semantic relationships between

objects. Specifically, depth maps and camera poses enable ConceptGraphs [2] to operate on RGB images without requiring expensive depth sensors, making the pipeline applicable to commodity cameras and broader real-world scenarios.

This section provides a comprehensive review of 3D reconstruction methods and it's organized into three main categories: traditional geometry-based methods, learning-based multi-view stereo, and recent feed-forward reconstruction models.

2.2.1 Classical Geometry-Based Methods

The classical approach to 3D reconstruction relies on well-established principles from photogrammetry and projective geometry [45]. The pipeline typically consists of two sequential stages: Structure-from-Motion (SfM) for sparse reconstruction and camera pose estimation, followed by Multi-View Stereo (MVS) for dense reconstruction [46, 45].

Structure-from-Motion

Structure-from-Motion (SfM) is the process of simultaneously estimating the 3D structure of a scene and the poses of the cameras from an unordered or ordered set of 2D images. The standard SfM pipeline includes feature extraction, correspondence search, geometric verification, and iterative reconstruction.

- **Feature Extraction and Matching**: The process begins with detecting "keypoints"—distinctive locations in an image that are invariant to scale, rotation, and illumination changes. The Scale-Invariant Feature Transform (SIFT) [47] was a seminal contribution, providing robust descriptors that allowed for reliable matching across widely separated views. Alternatives like Speeded-Up Robust Features (SURF) [48] were developed to reduce computational cost, enabling real-time applications. Matches between images are established by comparing these descriptors, often using a nearest-neighbor search followed by a ratio test to reject ambiguous matches.

- **Epipolar Geometry and Robust Estimation**: To filter out incorrect matches (outliers), SfM relies on epipolar geometry, which describes the geometric relationship between two views. The fundamental matrix F (for uncalibrated cameras) or the essential matrix E (for calibrated cameras) encapsulates this relationship:

$$x'^T E x = 0 \tag{2.1}$$

  where x and x' are corresponding normalized image coordinates. Algorithms like RANSAC (Random Sample Consensus) [49] are employed to robustly

estimate these matrices by iteratively selecting minimal subsets of points and verifying the consensus of the remaining set.

- **Incremental Reconstruction and Bundle Adjustment**: The most successful strategy for large-scale reconstruction has been Incremental SfM. Systems like Bundler [50] and COLMAP [46] exemplify this approach. Reconstruction typically starts with a carefully selected "seed" pair of images that have a wide baseline and a high number of verified matches. The system then incrementally adds new images using Perspective-n-Point (PnP) algorithms to estimate the pose of the new camera relative to the existing 3D structure. The core optimization engine of SfM is Bundle Adjustment (BA) [51], which refines the camera parameters (intrinsics and extrinsics) and 3D point coordinates to minimize the reprojection error (i.e. the distance between the projected 3D points and the observed 2D feature locations). COLMAP refined this process by introducing rigorous geometric verification and next-best-view selection strategies, making it the de facto standard for generating "ground truth" camera poses for training modern neural networks.

Multi-View Stereo

While SfM provides camera poses and a sparse point cloud, it does not reconstruct dense surfaces. Multi-View Stereo (MVS) algorithms take the output of SfM (images and calibrated poses) and aim to reconstruct a dense 3D representation of the scene [52].

A landmark in this domain is the Patch-based Multi-View Stereo (PMVS) algorithm proposed by Furukawa and Ponce [53]. PMVS operates on a "match, expand, and filter" paradigm. It begins with the sparse feature points from SfM and expands them into small rectangular patches oriented tangent to the local surface. Through an iterative process, it enforces local photometric consistency (ensuring the patch looks similar in all visible views) and global visibility constraints (ensuring no patch occludes another inconsistent with the view). PMVS was capable of reconstructing fine details and thin structures that volumetric methods of the time often smoothed away.

Later approaches shifted toward computing a *dense depth map* for every image and then fusing them, a process that involves plane-sweeping or patch-matching to find the depth value at each pixel that maximizes photo-consistency across neighboring views. The resulting depth maps are subsequently merged into a unified point cloud or volumetric mesh, often through techniques such as Poisson Surface Reconstruction [54], which is used to generate watertight models.

Despite their maturity and widespread adoption, classical methods face inherent limitations. They rely heavily on hand-crafted features and photometric consistency assumptions that fail under challenging conditions such as non-Lambertian surfaces,

repetitive textures, illumination changes, and insufficient viewpoint coverage [52]. These limitations motivate the transition to learning-based approaches.

2.2.2 Learning-Based Multi-View Stereo

The emergence of deep learning has revolutionized multi-view stereo by replacing hand-crafted features and matching metrics with learned representations that can capture semantic priors and handle challenging scenarios more robustly [55, 56].

MVSNet [57], introduced by Yao et al., pioneered the integration of deep learning into the MVS pipeline, formulating the problem as an end-to-end differentiable task, where its key innovation was the differentiable homography warping operation, which warps feature maps from source views onto the reference view's camera frustum at multiple depth planes, constructing a cost volume that encodes feature similarity across views. A variance-based cost metric flexibly aggregates features from an arbitrary number of source views into a single cost feature, and this cost volume is then regularized using 3D convolutional networks and regressed to produce per-pixel depth estimates. MVSNet's success generated numerous extensions addressing its computational limitations and reconstruction quality, as R-MVSNet [58] introduced recurrent neural networks to reduce memory consumption, enabling the reconstruction of larger scenes, while subsequent works incofrporated attention mechanisms [59], improved feature extraction modules, and coarse-to-fine cascade architectures to enhance both efficiency and accuracy, thereby establishing learning-based MVS methods as powerful tools for depth estimation, even though they still relied on the traditional pipeline structure—SfM for pose estimation followed by separate depth inference.

2.2.3 Neural Scene Representations

A fundamentally different approach to 3D reconstruction emerged from neural rendering research, which seeks to synthesize photorealistic novel views of scenes. Neural Radiance Fields (NeRF) [60], introduced by Mildenhall et al. in 2020, represent scenes as continuous volumetric functions encoded by multilayer perceptrons (MLPs). NeRF models the volumetric density and view-dependent radiance at any 3D location (x, y, z) and viewing direction (θ, ϕ) as a 5D function. By optimizing the network weights to minimize photometric error between rendered and observed images, NeRF learns an implicit 3D representation that supports high-quality novel view synthesis. While NeRF requires known camera poses—typically obtained from COLMAP—and substantial optimization time per scene, its success demonstrated that neural networks could learn continuous scene representations directly from images. This inspired extensive follow-up research exploring improvements in speed, quality, and generalization.

More recently, 3D Gaussian Splatting [61] proposed an alternative to NeRF's implicit neural representation. Instead of using neural networks to model radiance fields, Gaussian Splatting represents scenes as collections of 3D Gaussian primitives with learned positions, colors, opacities, and covariances. These Gaussians are rasterized using efficient splatting operations to render novel views. By avoiding neural network evaluation during rendering and leveraging traditional optimization techniques like stochastic gradient descent, Gaussian Splatting achieves real-time rendering speeds while producing photorealistic results that match or exceed NeRF quality. The explicit nature of the representation also facilitates easier manipulation and integration with downstream tasks.

2.2.4 Feed-Forward 3D Reconstruction Models

Most recently, a new paradigm has emerged that fundamentally reconceptualizes 3D reconstruction as a direct regression problem rather than a multi-stage geometric pipeline. These feed-forward models leverage large-scale transformer architectures trained on diverse datasets to directly predict 3D geometry from images in a single forward pass, bypassing traditional iterative optimization entirely [62].

DUSt3R [15] (Dense Unconstrained Stereo 3D Reconstruction) introduced a radically novel approach that operates without camera calibration or pose estimation, as rather than first estimating camera parameters and then computing depth, DUSt3R directly regresses *pointmaps* for pairs of input images, i.e. pixel-aligned 3D coordinates expressed in a common coordinate frame. The architecture employs standard transformer encoders and decoders trained on large-scale data, thereby enabling it to learn powerful geometric priors, while for multi-image scenarios, DUSt3R performs global alignment by optimizing camera poses and scene geometry directly in 3D space, avoiding traditional bundle adjustment based on 2D reprojection errors.

MASt3R [63], building upon DUSt3R's foundation, enhanced the matching capabilities by adding a dedicated head for dense local features trained with an explicit matching loss, which enables robust correspondence estimation even under extreme viewpoint changes and in texture-less regions, thereby achieving substantial improvements in matching accuracy while maintaining the feed-forward inference paradigm.

Visual Geometry Grounded Transformer (VGGT) [16] represents further progress toward unified feed-forward reconstruction. Unlike DUSt3R, which predicts pointmaps in local coordinate systems requiring subsequent alignment, VGGT directly regresses all 3D attributes (including camera intrinsics, extrinsics, depth maps, point maps, and 3D point tracks) in a globally consistent manner from one or many input views. The architecture employs alternating view-wise and global self-attention to aggregate information across images, enabling reconstruction of

scenes with hundreds of images in a single forward pass at 0.2 seconds, significantly faster than both classical methods and DUSt3R's optimization-based alignment.

In the same spirit, MapAnything [1] pushes this paradigm further by introducing a universal feed-forward backbone that jointly predicts metric depth maps, local ray directions, camera poses, and a global scale factor from flexible RGB inputs. By factorizing multi-view geometry into rays, depth, pose, and scale, MapAnything attains globally consistent metric reconstructions from uncalibrated images in a single pass, and supports a broad family of 3D vision tasks ranging from uncalibrated structure-from-motion to multi-view stereo and monocular depth estimation. MapAnything is the framework adopted in this thesis for 3D reconstruction, and it will be detailed in Section 3.2.3.

This paradigm shift in visual 3D reconstruction brings several advantages: first, feed-forward models replace explicit geometric constraints with implicit data-driven priors learned from diverse training data, enabling them to handle challenging scenarios where traditional methods struggle; second, the unified architecture enables information sharing and internal consistency across tasks: depth, pose, and matching are jointly predicted and mutually constrain each other. Third, inference is dramatically faster; where traditional pipelines require minutes to hours of iterative optimization, feed-forward models produce results in seconds or less. Finally, these models democratize 3D reconstruction by eliminating the need for camera calibration and controlled capture conditions, enabling reconstruction from arbitrary image collections including those from low-cost sensors.

2.3 Active Perception for Scene Reconstruction

Autonomous robots must not only perceive their environment but actively choose *where* to look next to build comprehensive and accurate representations of their surroundings. This capability, known as active perception or active vision, stands in contrast to passive perception approaches where sensor viewpoints are predetermined or controlled by human operators. Active perception enables robots to intelligently explore unknown environments by selecting observation strategies that maximize information gain, reduce uncertainty, and efficiently construct high-quality scene representations.

As previously discussed in section 2.1.4, this thesis aims to integrate active perception strategies with 3D scene graph construction, enabling an autonomous agent to not only build a semantic representation of its environment but also to actively decide where to look next in order to improve that representation.

This section provides an overview of active perception and autonomous exploration, followed by detailed descriptions of two complementary approaches used in this thesis: the Surface Edge Explorer (SEE), a geometric density-based method,

and Active Semantic Perception, which leverages scene graphs and large language models for semantic-driven exploration.

2.3.1 Foundations of Active Perception

The concept of active perception was pioneered by Bajcsy [64] in the late 1980s, introducing a paradigm shift from passive data analysis to purposeful sensor control. Active perception argues that perception should not be passive but rather involve *actively* changing sensor parameters according to sensing strategies. The key ingredients of this paradigm are taking multiple measurements, integrating them, and incorporating feedback between perception and action. Around the same time, Aloimonos et al. [65] formalized the theoretical foundations of active vision, proving that an active observer can solve basic vision problems more efficiently than a passive one.

A survey by Bajcsy et al. [19] has revisited these foundational ideas, arguing that with modern computational capabilities, active perception is more relevant than ever for robotic systems. An actively perceiving agent is one that dynamically determines not only the *why* of its behavior but also controls the *what*, *how*, *where*, and *when* of its sensing operations.

2.3.2 Next Best View Planning

A central problem in active perception is *Next Best View (NBV) planning*: determining the optimal sensor viewpoint that will provide the greatest improvement in scene observation or reconstruction quality. The NBV problem was first formalized by Connolly [66], who introduced octree-based methods to determine the "best" next view for object reconstruction.

Scott et al. [20] provide a comprehensive survey of view planning techniques for automated 3D object reconstruction and inspection. They categorize NBV approaches into two broad classes:

Model-based methods assume *a priori* knowledge of the scene or object geometry (e.g., CAD models) and plan viewpoints to achieve complete coverage with specified quality levels. These methods use representations such as visibility matrices, aspect graphs, or geometric decompositions to determine minimal sets of viewpoints that satisfy coverage and accuracy constraints.

Non-model-based methods (also called exploratory or online methods) do not assume prior scene knowledge and incrementally build representations as new observations are acquired. These methods plan next best views by evaluating the current observation state and selecting viewpoints that maximize expected information gain or uncertainty reduction.

This thesis focuses on non-model-based NBV approaches, as they are more applicable to unknown environments where robots must explore and build scene graphs from scratch.

Scene Representations for NBV Planning

The choice of scene representation fundamentally impacts NBV planning efficiency and the quality of resulting observations. Scott et al. [20] and subsequent work identify three main representation categories within model-free NBV methods, each with distinct advantages and limitations.

Volumetric representations Volumetric methods discretize the environment into a three-dimensional grid of voxels, often organized using occupancy grids or octree structures. Each voxel represents the state of space as free, occupied, or unknown based on sensor measurements. This representation is well suited for large-scale exploration and path planning, as it provides an explicit notion of free space and visibility. However, volumetric models tend to be memory-demanding and strongly dependent on voxel resolution: finer grids and denser view sampling improve geometric detail but significantly increase the computational cost of ray tracing and visibility checks. As a result, scaling to large environments typically requires either reducing resolution or accepting higher computational overhead. Early work by Connolly [66] proposed evaluating candidate viewpoints by counting the number of previously unseen voxels observable from directions sampled on a sphere surrounding the scene. Vasquez-Gomez et al. [67] extend this idea by ranking viewpoints according to multiple criteria, including reachability, distance, overlap with earlier observations, and visibility of unknown space. Delmerico et al. [68] further formalize view evaluation through Information Gain (IG) metrics that account for voxel visibility, observability, and proximity to existing measurements. To address scalability issues, later approaches reduce the search space of viewpoints using frontier-based strategies [69, 70] or sample viewpoints along feasible trajectories generated by Rapidly-Exploring Random Trees (RRTs) [71, 72].

Surface representations Surface-based approaches model the environment as a continuous mesh composed of connected triangles, directly capturing the geometry of observed surfaces with high fidelity. View planning in this context focuses on extending incomplete surface regions and improving the quality of existing observations. Some methods incrementally update the surface model as new sensor data becomes available [73, 74], while others adopt a multistage strategy in which an initial mesh is progressively refined through successive exploration steps [75, 76].

Dierenbach et al. [73] reconstruct surface geometry using a Growing Neural Gas (GNG) algorithm to derive a simplified mesh from sensor data. Local point density around mesh vertices is then used to guide viewpoint selection, with new views proposed to expand the surface while meeting a target density. Khalfaoui et al. [74] instead cluster sensor observations based on spatial density and generate viewpoints aimed at observing the boundaries of these clusters until a predefined compactness criterion is satisfied. Multistage methods [75, 76] typically start from an initial surface mesh—often obtained through manual modeling or a predefined trajectory—and iteratively improve it. Hollinger et al. [75] model surface uncertainty using Gaussian processes and select viewpoints that maximize the expected improvement in surface estimation. Roberts et al. [76] sample candidate views within a bounded distance from the surface, choose a minimal subset that ensures full coverage, and compute an efficient path connecting them.

While capable of producing highly detailed reconstructions, surface-based representations often rely on carefully tuned parameters or multiple exploration stages, which can complicate their practical deployment, especially in dynamic or unstructured environments.

Unstructured representations work directly with point cloud measurements without imposing external structure, and the Surface Edge Explorer (SEE) [4, 5] exemplifies this approach by using measurement density to identify observation boundaries and propose next best views without requiring additional data structures. SEE iteratively expands the observed surface by selecting viewpoints that target low-density regions, thereby effectively guiding exploration toward unobserved areas while maintaining computational efficiency, and it constitutes one of the core methods adopted in this thesis for geometric exploration, as described in detail in Section 3.4.2.

2.3.3 Exploration Strategies

Beyond selecting individual next best views, robots must employ *exploration strategies* that guide a systematic coverage of unknown environments, where frontier-based exploration, introduced by Yamauchi [77], remains one of the most influential approaches. The central idea is to move the robot to the boundary (frontier) between open and unexplored space, thereby constantly expanding the known territory while maintaining efficiency in coverage.

Frontier-based methods have been extensively developed for both 2D [77] and 3D environments [71], as Bircher et al. [71, 78] proposed the Receding Horizon Next-Best-View Planner (RH-NBVP), which combines rapidly-exploring random trees (RRT) with volumetric information gain to perform autonomous 3D exploration in a receding horizon fashion, thus balancing local exploitation of promising viewpoints

with global exploration to escape local minima.

More recent approaches integrate deep reinforcement learning to learn exploration policies from data [79, 80]. These learning-based methods can generalize across different environments and adapt exploration strategies based on task-specific objectives, though they typically require substantial training data and computational resources.

While frontier exploration focuses on completeness of coverage, more advanced techniques incorporate information gain and uncertainty reduction into the decision. Instead of just aiming for unknown space, they evaluate the expected information that would be gained by taking candidate actions. Active SLAM (A-SLAM), for example, extends exploration to simultaneously reduce both map uncertainty and localization uncertainty [81, 41]. A-SLAM systems balance exploration actions (visiting new areas) with loop closure actions (revisiting known areas to correct accumulated drift), ensuring both coverage and map accuracy.

2.3.4 Semantic-Aware Active Perception

Traditional NBV planning focuses primarily on geometric coverage and reconstruction quality, yet in many robotic applications not all regions of the environment are equally important, as *semantic-aware* active perception methods instead leverage semantic information about objects, scene structure, and task relevance to guide viewpoint selection more intelligently. Early semantic-aware exploration work focused on integrating object detection into frontier-based exploration, and Jaramillo et al. [82] introduced S-AVE (Semantic-Based Active Vision Exploration), which explicitly uses detected objects and their semantic labels to drive exploration through active vision, so that rather than maximizing geometric coverage alone, S-AVE reasons about objects in the environment to improve 3D semantic reconstruction. More recent work has integrated semantic NBV planning with modern perception systems, as Burusa et al. [83, 84] developed semantic-aware NBV planners that use object detection confidence scores and semantic information gain to plan viewpoints for targeted perception tasks, demonstrating that incorporating semantics can significantly improve task-relevant information gathering compared to purely geometric approaches.

An emerging direction combines active exploration with semantic scene understanding through predictive models. Chen et al. [85] proposed ActiveSGM, an active semantic mapping framework built upon 3D Gaussian Splatting that predicts the informativeness of potential observations before execution. By quantifying both semantic and geometric uncertainty, the system strategically selects viewpoints that enhance mapping completeness and robustness to noisy semantic data, effectively *understanding while exploring*. Similarly, Ding et al. [86] introduced SEA (Semantic Map Prediction for Active Exploration), which uses semantic scene

completion to predict unobserved regions of the environment. Rather than relying on single-step waypoint prediction, SEA employs reinforcement learning to guide long-term exploration toward low-confidence (uncertain) map areas, iteratively refining a global semantic map.

As discussed in Section 2.1.4, 3D scene graphs offer a rich hierarchical representation of environments, making them particularly suitable for semantic-aware active perception. By leveraging the structured information encoded in scene graphs, LLMs can reason about unobserved regions, infer likely spatial and semantic configurations, and steer exploration toward semantically informative areas. This paradigm is exploited in Active Semantic Perception [6], one of the core frameworks adopted in this thesis, briefly introduced in Section 2.1.4 e described in details in Section 3.4.4.

These semantic-aware methods complement geometric approaches by enabling task-relevant, context-aware exploration. While geometric methods excel at comprehensive surface coverage, semantic approaches can prioritize viewpoints that maximize understanding of object semantics, spatial relationships, and scene structure, directly using the evolving 3D scene graph as a guide for exploration.

2.4 External Cameras for Scene Graph Construction

While most 3D scene graph construction approaches rely exclusively on egocentric views from mobile robots or handheld sensors, the integration of external cameras, like wall-mounted or surveillance cameras, could offer significant practical advantages for active scene understanding. In the context of this work, the use of external RGB cameras is explored as a cost-effective and powerful instrument to assist in scene graph construction. For this reason, here is provided a review of related work on the use of external cameras for robot control, navigation, and scene understanding.

2.4.1 External-Cameras in Perception

Using external cameras for perception offers several opportunities. First, offloading perception to fixed infrastructure can reduce the computational and sensing burden on the robot. Instead of every robot carrying an expensive 3D sensor, a set of wall cameras with an external computer could do the heavy perception and just send results to relatively "dumb" robots. Second, external views can cover a larger area or provide top-down coverage, which is ideal for multi-robot coordination (e.g., an overhead camera can see two robots and help them avoid collisions even if they can't see each other). Third, external cameras can facilitate global localization of a

robot without reliance on onboard SLAM; for example, an overhead camera system can always tell the robot its absolute location in the room by observing it.

Early work on external camera perception has explored coordinating multiple cameras for object tracking and surveillance. Li et al. [87] coordinate multiple cameras to improve the performance of an active tracker through pose-assisted multi-camera collaboration, while Esterle et al. [88] present a system for switching object tracking tasks between a network of cameras using socio-economic vision graphs.

2.4.2 External-Cameras Visual Servoing

Not only can external cameras passively feed information to robots, but they can also actively control robot operations through visual servoing, where the robot's motion is guided by visual feedback from external cameras. This "eye-to-hand" configuration offers advantages in scenarios where precise navigation is required or where onboard sensing is limited. In this scenario, Robinson et al. have developed a comprehensive framework for calibration-free visual servoing using external camera networks [89, 90].

Their initial work presents a weakly supervised pipeline for accurate robot orientation estimation from external camera images without assuming prior knowledge about the robot's physical characteristics or deployment environment [89]. The pipeline employs a neural network trained with weak supervision, where orientation labels are automatically derived from the robot's odometry rather than requiring manual annotation.

Building on this foundation, the authors developed Robot-Relay, a building-wide visual servoing system that manages networks of remote viewpoints without requiring explicit camera calibration [90]. Traditional multi-camera systems demand precise knowledge of camera intrinsics and extrinsics, which is labor-intensive to obtain and maintain across large-scale deployments. Robot-Relay circumvents this limitation by learning sensor handover networks that memorize soft pixel-wise topological connections between viewpoints from simultaneous robot detections across overlapping camera fields of view. During an initial deployment phase, the robot traverses the environment while being observed by multiple cameras; when the robot appears simultaneously in overlapping camera views, the system learns spatial correspondences at the pixel level. During the operational phase, the robot is visually servoed across the camera network by leveraging these learned handovers: when the robot approaches the boundary of one camera's field of view, control seamlessly transfers to the next camera based on the memorized topological map. Experimental validation in productive office environments with six cameras demonstrates that Robot-Relay achieves robust building-wide navigation with deployment times as short as 30 minutes.

2.4.3 External Cameras in High-Level Planning

Beyond low-level visual servoing, external cameras have been integrated into high-level robot planning and decision-making frameworks, particularly in the context of vision-language models for semantic navigation.

Buoso et al. introduce Select2Plan (S2P), a training-free framework for high-level robot planning that leverages Vision-Language Models (VLMs) for autonomous navigation without requiring task-specific fine-tuning [91]. The framework employs structured Visual Question-Answering (VQA) and In-Context Learning (ICL) to drastically reduce data collection requirements. The system is evaluated in two distinct sensing configurations: First-Person View (FPV), where the robot uses its onboard camera, and Third-Person View (TPV), where external infrastructure cameras provide the visual input. In the TPV configuration, overhead or wall-mounted cameras observe the robot and its environment from an external perspective, feeding bird's-eye view images to the VLM-based planner.

The experimental results reveal a clear performance gain in both sensing configurations. In the TPV scenario, S2P improves the performance of a baseline VLM by about 40%, while in the FPV scenario it surpasses end-to-end trained models by approximately 24% when navigating towards unseen objects in novel scenes. This demonstrates that external cameras can provide valuable contextual information that enhances the VLM's reasoning capabilities for high-level planning and navigation tasks.

For 3D scene graph construction, this suggests that external cameras could similarly enhance the robot's understanding of the environment by providing complementary perspectives that inform semantic labeling and relationship inference.

2.4.4 Implications for Scene Graph Construction

In summary, external cameras have proved effective across a range of robotic tasks, even though they are not yet widely adopted in robotic perception pipelines. In the context of 3D scene graph construction, several key advantages of external cameras can be identified:

- **Reduced exploration requirements**: External cameras can initialize the scene graph with a global view of the environment, significantly reducing the number of robot viewpoints needed for complete coverage. While small objects may not be fully captured, the initial graph structure provides a strong foundation for subsequent refinement.
- **Enhanced view overlap**: For approaches requiring image overlap (such as methods based on MapAnything for 3D reconstruction) external cameras with bird's-eye perspectives naturally facilitate overlap between views, improving reconstruction quality and consistency.

- **Continuous updates**: A network of external cameras can continuously update a shared global scene graph that multiple robots can access, adding nodes for newly observed objects and maintaining scene consistency across agents.
- **Cost-effectiveness**: Since the developed pipeline for 3D scene graph construction relies solely on RGB data, simple external cameras provide a powerful and economical instrument for scene graph construction without requiring specialized depth sensors or expensive hardware.

These advantages motivate the integration of external cameras into our active scene graph construction framework. This enables us to explore how external viewpoints can complement the robot's egocentric observations to build more complete scene graphs with fewer required viewpoints.

Chapter 3

Methodology

This chapter describes the methodological framework developed in this thesis for active, incremental construction of 3D scene graphs from RGB images. The work is organized into two main contributions: (i) an RGB-only pipeline for 3D scene graph generation that integrates state-of-the-art feed-forward reconstruction with open-vocabulary semantic mapping, and (ii) an active exploration framework that leverages both geometric and semantic information to efficiently construct scene graphs with minimal viewpoints.

Section 3.1 provides an overview of the complete system architecture. Section 3.2 details the RGB-only 3D scene graph generation pipeline, including depth and pose estimation, scene graph construction, and the geometric edge generation approach. Section 3.3 presents the automated validation framework developed to evaluate scene graph quality. Finally, Section 3.4 describes the active exploration strategies, including both geometric density-based and semantic-driven approaches.

3.1 System Overview

The proposed system enables autonomous robots to construct 3D scene graphs through active exploration of indoor environments using only RGB cameras. Figure 3.1 illustrates the complete architecture.

The system operates as an incremental perception–action loop: at each exploration step, the robot acquires an RGB image from its current pose (1). The RGB-only 3D scene graph pipeline described in Section 3.2 processes this observation to update both the reconstructed point cloud and the 3D scene graph (2). Given this updated representation, the active exploration module described in Section 3.4 selects a next-best-view (NBV) that maximizes expected information gain, using either semantic-based or frontier-based exploration strategies (3). The navigation module then drives the robot to the selected viewpoint (4), where a new

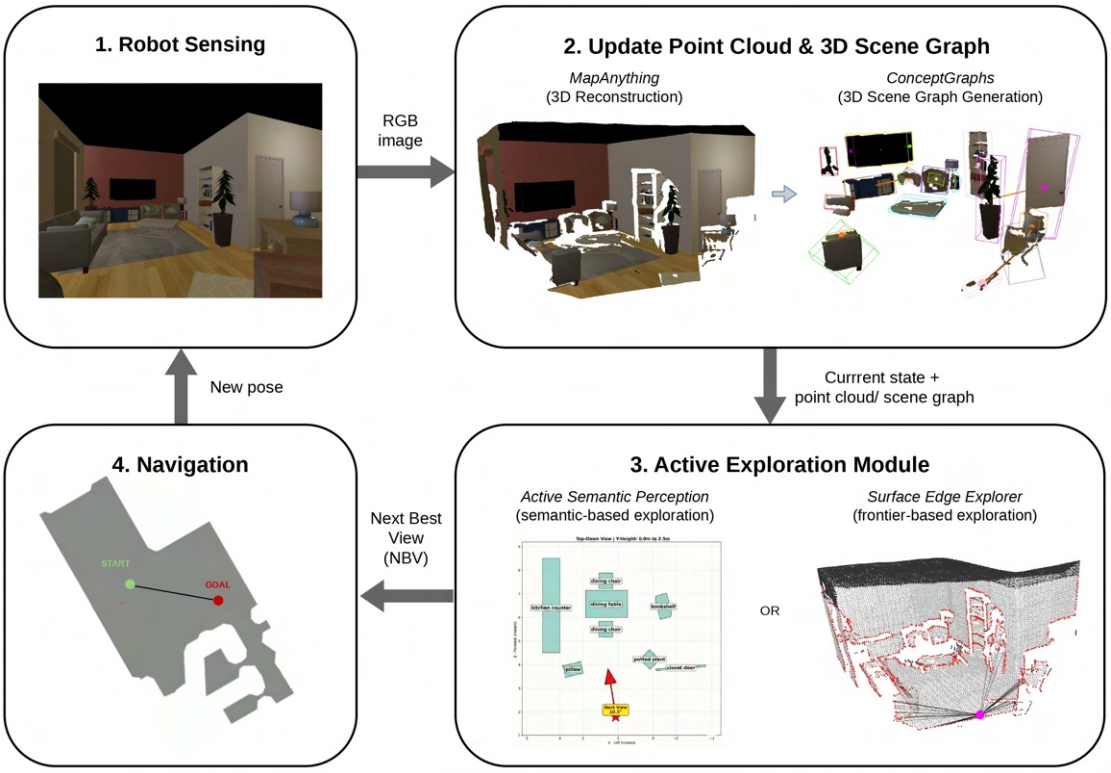


Figure 3.1: Overview of the active 3D scene graph generation system. The robot captures RGB images, which are processed by the RGB-only pipeline (MapAnything + ConceptGraphs) to update the current 3D scene graph and point cloud. An active exploration module selects the next best view based on the current graph and point cloud, guiding the robot's navigation to efficiently build a comprehensive scene graph.

RGB observation is captured and the cycle repeats until a stopping condition is met (e.g., a fixed number of views or sufficient scene coverage).

This architecture decouples perception and planning from robot control, enabling the computationally intensive reconstruction and scene graph generation to run on remote hardware while the robot handles navigation and image acquisition. The design also facilitates integration of external cameras, which can provide complementary viewpoints without requiring robot motion.

The key design principles underlying the system are:

- **RGB-only operation**: The pipeline requires only RGB images as input, eliminating the need for depth sensors and enabling deployment with commodity cameras or external surveillance systems.

- **Incremental construction**: The scene graph is built progressively as new observations are integrated, rather than requiring a complete dataset upfront.
- **Active view selection**: Rather than following predetermined trajectories or random exploration, the system actively selects viewpoints that maximize reconstruction quality or semantic understanding.
- **Modularity**: The architecture separates 3D reconstruction, semantic mapping, and exploration planning into distinct modules, facilitating experimentation with alternative methods for each component.

3.2 RGB-Only 3D Scene Graph Generation

3.2.1 Pipeline Overview

The RGB-only scene graph generation pipeline consists of two main stages: (i) depth and pose estimation together with 3D point cloud reconstruction from RGB images using MapAnything [1], and (ii) scene graph extraction using ConceptGraphs [2] with geometric edge generation. Figure 3.2 provides an overview of the pipeline architecture.

3.2.2 Motivations and Applications

The RGB-only pipeline developed in this section addresses several limitations of existing 3D scene graph generation methods. By relying solely on RGB images, it removes the need for depth sensors or LiDAR, which simplifies the hardware requirements and lowers cost. This makes the approach suitable for platforms that only carry a single color camera and for environments that already have fixed RGB cameras installed. In practice, even a short handheld video of a room recorded with a standard smartphone provides enough information for this pipeline to reconstruct a metric 3D point cloud of the scene and to build a corresponding 3D scene graph. The same pipeline can therefore be applied to surveillance streams, wall-mounted cameras, or offline collections of images and videos, greatly broadening the set of scenarios in which 3D scene graphs can be constructed without any additional sensing infrastructure.

3.2.3 3D reconstruction using MapAnything

The first stage of the pipeline focuses on reconstructing 3D geometry from RGB images. MapAnything [1], the method employed in this thesis, represents the current state-of-the-art in unified feed-forward 3D reconstruction. MapAnything directly regresses the geometric quantities needed for multi-view metric 3D reconstruction

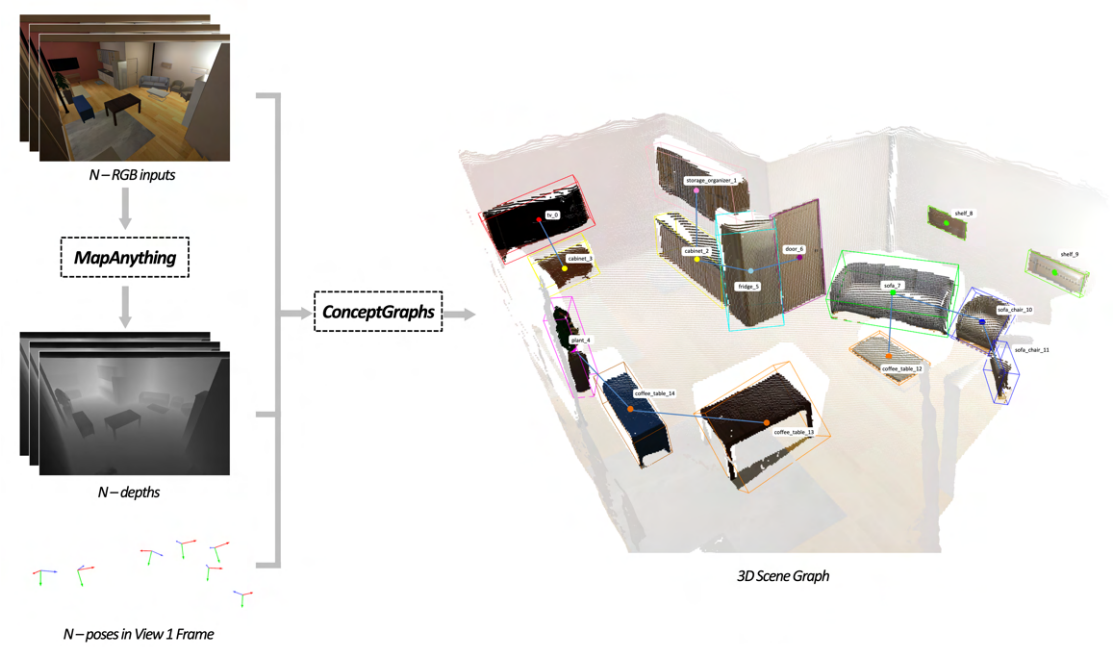


Figure 3.2: Overview of the RGB-only 3D scene graph generation pipeline. The system takes as input a sequence of RGB images and processes them through MapAnything to estimate depth and camera poses. RGB-images, depths, and poses are then fed into ConceptGraphs to extract object-centric nodes and geometric edge generation to infer spatial relationships, resulting in a comprehensive 3D scene graph representation.

from a flexible set of inputs: in its most general form, it ingests N RGB images and can optionally incorporate partial geometric cues (e.g., camera intrinsics, poses, sparse/degraded depth, or partial reconstructions), while still producing a consistent metric reconstruction in a single forward pass.

Factored output representation. Rather than directly predicting a single coupled 3D structure (e.g., a point cloud), MapAnything [1] outputs a *factored* set of geometric components. Given N RGB images $\hat{\mathcal{I}} = (\hat{I}_i)_{i=1}^{N}$ and optional geometric inputs (ray directions, poses, and/or depths for a subset of views), the model predicts

$$f_{\text{MapAnything}}\big(\hat{\mathcal{I}}, [\hat{\mathcal{R}}, \hat{\mathcal{Q}}, \hat{\mathcal{T}}, \hat{\mathcal{D}}]\big) = \Big\{ m,\ \big(R_i, \tilde{D}_i, \tilde{P}_i\big)_{i=1}^{N} \Big\}, \tag{3.1}$$

where $m \in \mathbb{R}$ is a *global metric scaling factor* and for each view i, the network predicts: (i) local ray directions $R_i \in \mathbb{R}^{3\times H\times W}$ (a per-pixel calibration in the generic central camera model), (ii) an *up-to-scale* ray depth map $\tilde{D}_i \in \mathbb{R}^{1\times H\times W}$, and (iii)

the pose $\tilde{P}_i \in \mathbb{R}^{4\times4}$ of view i in the reference frame of $\hat{I}_1$ (first image), represented as a quaternion Q_i and an up-to-scale translation $\tilde{T}_i \in \mathbb{R}^3$.

The factored outputs are composed to obtain metric 3D point maps (in the frame of $\hat{I}_1$):

$$\tilde{L}_i(\mathbf{u}) = R_i(\mathbf{u})\,\tilde{D}_i(\mathbf{u}), \tag{3.2}$$

$$\tilde{X}_i(\mathbf{u}) = O_i\,\tilde{L}_i(\mathbf{u}) + \tilde{T}_i, \tag{3.3}$$

$$X_i^{\text{metric}}(\mathbf{u}) = m\,\tilde{X}_i(\mathbf{u}), \qquad i \in \{1,\dots,N\}, \tag{3.4}$$

where O_i is the rotation matrix obtained from Q_i. Intuitively, $\tilde{L}_i$ back-projects pixels along rays to obtain an up-to-scale local point map, $\tilde{X}_i$ aligns each view to the reference frame of the first image, and m converts the reconstruction to metric scale.

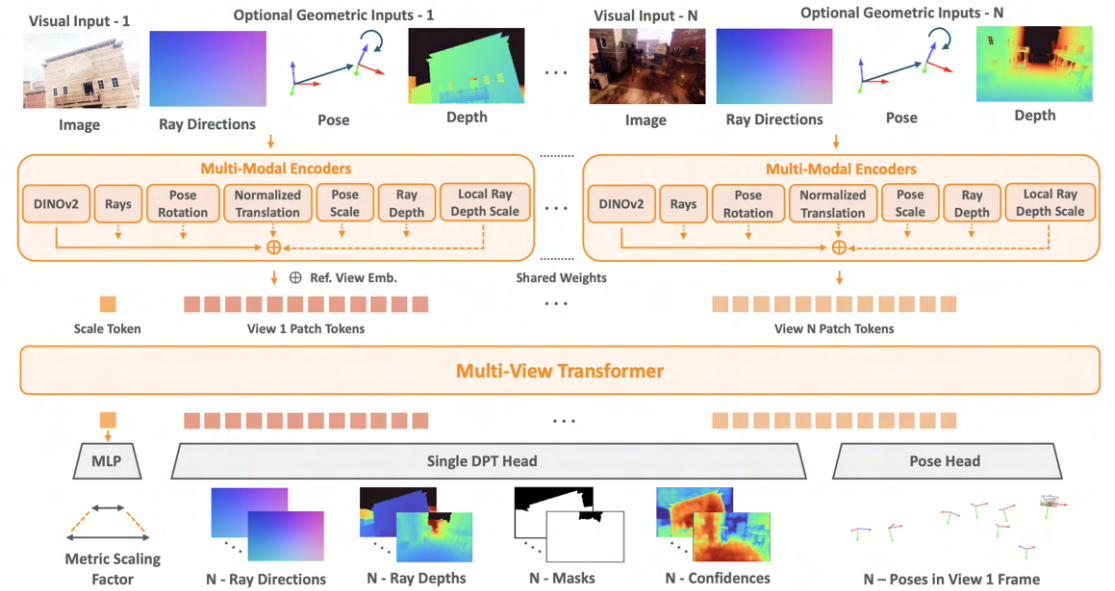


Figure 3.3: Overview of the MapAnything architecture. The model takes as input a variable number of RGB images along with optional geometric inputs (camera intrinsics, extrinsics, depth maps) and produces a factored representation of multi-view scene geometry consisting of per-view depth maps, ray maps, camera poses, and a global scale factor. Figure adapted from [1].

Architecture MapAnything's architecture, illustrated in Figure 3.3, follows an "encode–fuse–decode" design tailored to variable numbers of views:

- **Input encoding (images + optional geometry).** Each RGB image is encoded into patch tokens using a strong Vision Transformer (ViT) image

backbone (specifically, they adopted DINOv2 [92]), while optional *dense* geometric inputs (e.g., ray directions and normalized depths when provided) are encoded with a lightweight convolutional encoder and mapped to the same spatial resolution and feature dimension as the image patch features, and optional *global* inputs (e.g., rotations, translation directions, and scale-related quantities) are embedded with an MLP and broadcast to the patch grid so that all modalities live in a common latent space, with the resulting per-view features finally combined by summation and normalization before tokenization.

- **Multi-view fusion via a transformer.** The per-view patch tokens from all views are concatenated, a fixed reference-view embedding is added to tokens of view 1, and a *learnable scale token* is appended, after which a multi-view transformer performs cross-view information propagation (supporting N from a single image up to very long image sequences) and produces updated patch tokens for each view together with an updated scale token.

- **Decoding the factored representation.** A Dense Prediction Transformer (DPT) head [93] decodes the view-specific patch tokens into dense per-pixel outputs (ray directions and up-to-scale depths, and auxiliary confidence/mask signals). In parallel, a compact Pose Head pools patch tokens to regress per-view rotations and translations relative to the reference view. Finally, an MLP maps the scale token to the global metric scale factor (with an exponential parameterization to cover wide scene-scale ranges). This explicit separation between *up-to-scale geometry* and *metric scale* is crucial for stable, universal metric inference.

Training strategy. MapAnything is trained end-to-end in a supervised manner using a diverse collection of indoor, outdoor, and in-the-wild datasets, each providing different subsets of geometric annotations. Thanks to its factored output representation, the model can be trained with partial supervision, i.e., using images alone, or images augmented with camera intrinsics, poses, depth maps, or metric scale when available. Losses are defined separately for scale-invariant quantities (ray directions and rotations) and up-to-scale quantities (depths and translations), while a dedicated loss supervises the prediction of the global metric scale. This unified training scheme enables a single model to learn multiple 3D reconstruction tasks jointly, without task-specific fine-tuning.

Experiments. The experiments assess performance on dense multi-view reconstruction, camera pose estimation, ray direction prediction, and metric depth estimation, using from two up to hundreds of input views. Results demonstrate that MapAnything achieves state-of-the-art or competitive performance compared

to specialized feed-forward models such as VGGT [16] and DUSt3R [15], while significantly outperforming classical optimization-based pipelines (e.g., COLMAP [46]) in terms of inference speed and reconstruction completeness. Importantly, the evaluations show that a single universally trained model can handle more than twelve different 3D reconstruction tasks, and that reconstruction quality consistently improves when auxiliary geometric inputs (e.g., camera calibration, poses, or sparse depth) are provided.

How MapAnything is used in this thesis

In this work, MapAnything is used in the *image-only* setting: given RGB frames from commodity, uncalibrated cameras, it produces per-view depth and relative camera poses along with a global metric scale estimate. These outputs are then back-projected into a metric point cloud (Equations 3.2, 3.3, 3.4) to obtain a unified 3D representation of the scene geometry.

The output of MapAnything is exploited in three complementary ways:

1. The estimated depth maps and camera poses are fed to ConceptGraphs to project 2D object masks into 3D and to maintain a consistent global map coordinate frame.

2. The reconstructed point cloud, expressed in a consistent metric scale, serves as a geometric map of the scene that can be directly used for downstream robotic tasks beyond the 3D scene graph representation.

3. The same point cloud is provided as input to the Surface Edge Explorer (SEE) algorithm for active exploration (Section 3.4.2), where it is used to estimate surface density and to identify under-observed regions of the environment.

3.2.4 3D Scene Graph Generation using ConceptGraphs

The second stage of the pipeline focuses on extracting a 3D scene graph from the reconstructed geometry. For this purpose, ConceptGraphs [2] is adopted as the core scene graph generation framework, with a custom geometric edge generation module (Section 3.2.5) replacing the original LLM-based edge inference.

ConceptGraphs [2] is an open-vocabulary 3D scene graph generation framework that constructs semantically rich, object-centric representations from RGB-D sequences without requiring task-specific training. In Figure 3.4 is illustrated an overview of the ConceptGraphs system that will be described in the following.

Object-based 3D mapping. The foundation of ConceptGraphs is an incremental object-centric mapping system that constructs a 3D scene graph $\mathcal{M}_t = \langle \mathcal{O}_t, \mathcal{E}_t \rangle$

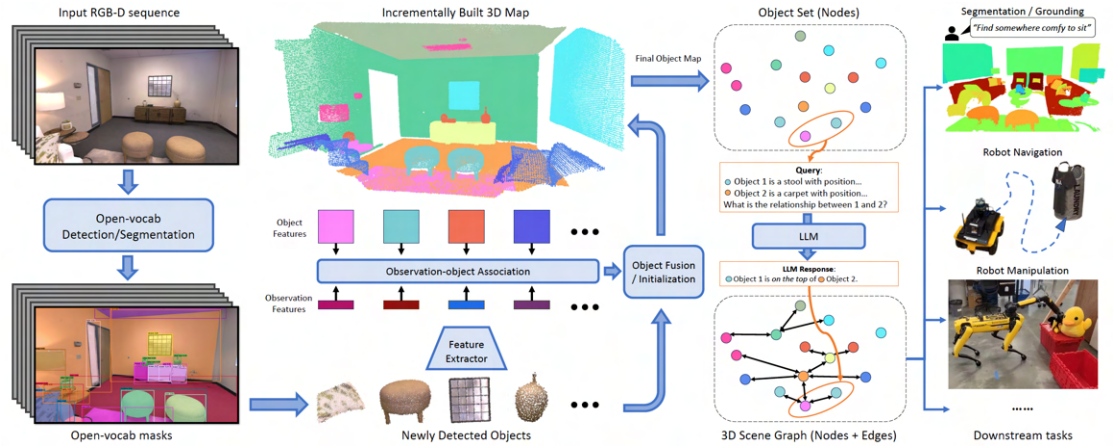


Figure 3.4: The ConceptGraphs pipeline for open-vocabulary 3D scene graph construction. The system processes a sequence of posed RGB-D images through: (1) class-agnostic segmentation to extract object masks, (2) semantic feature extraction via CLIP [18], (3) multi-view geometric and semantic association to incrementally build a 3D object map, (4) vision-language model captioning of detected objects, and (5) LLM-based inference of spatial relationships to construct graph edges. The resulting scene graph enables downstream robotic tasks including navigation, manipulation, and natural language querying. Figure taken from [2].

from a sequence of posed RGB-D observations $\mathcal{I} = \{I_1, \ldots, I_t\}$. Here, t denotes the current time step (i.e., the index of the latest frame), $\mathcal{O}_t = \{o_j\}_{j=1\ldots J}$ is the set of object nodes in the map at time t, and $\mathcal{E}_t = \{e_k\}_{k=1\ldots K}$ is the set of edges encoding spatial relations between these objects. Each observation $I_t = \langle I_t^{\text{rgb}}, I_t^{\text{depth}}, \theta_t \rangle$ consists of a color image, a depth map, and the corresponding camera pose, which together enable metric-scale reconstruction. In our pipeline, the depth maps and camera poses are predicted from RGB images using MapAnything, providing the posed RGB-D inputs required by ConceptGraphs.

The mapping process begins with *class-agnostic 2D segmentation*, wherein each incoming frame I_t^{rgb} is processed through Segment Anything Model (SAM) [17] to extract instance masks $\{m_{t,i}\}_{i=1\ldots M}$ corresponding to candidate objects. This class-agnostic approach is fundamental to the open-vocabulary capability, as it avoids constraining detection to predefined categories. For each extracted mask $m_{t,i}$, the system computes a semantic feature descriptor $f_{t,i} = \text{Embed}(I_t^{\text{rgb}}, m_{t,i})$ using the CLIP image encoder [18], which maps the masked region into a semantically rich embedding space that captures visual and linguistic correspondences. The masked region is then projected to 3D using the associated depth map I_t^{depth} and camera pose θ_t, yielding a 3D point cloud $p_{t,i}$. To mitigate sensor noise and outliers, the projected point cloud undergoes denoising via DBSCAN clustering, followed by

transformation into the global map coordinate frame.

Multi-view object association constitutes the core challenge in maintaining consistent object identities across frames. The geometric similarity $\phi_{\text{geo}}(i, j)$ measures point cloud overlap, while semantic similarity $\phi_{\text{sem}}(i, j)$ measures normalized cosine distance between CLIP features. The overall similarity $\phi(i, j)$ is the sum of these two components. Object association employs a greedy assignment strategy where in each detection is matched to the existing object with maximal similarity score, provided $\phi(i, j)$ exceeds threshold δ_{sim}; otherwise, a new object is instantiated. Upon successful association, the system performs object fusion by updating both geometric and semantic representations. The semantic feature vector is updated via exponential moving average, and the point cloud is updated as $p_{t,i} \cup p_{o_j}$, followed by voxel-based downsampling to remove reduntant points.

Node Captioning with Vision-Language Models ConceptGraphs generates natural language descriptions for each detected object through a two-stage captioning pipeline. The system identifies the ten most informative views per object and passes them to LLaVA-7B [94] with the prompt "describe the central object in this image," yielding a set of initial captions $\hat{c}_j = \hat{c}_{j,1}, \hat{c}_{j,2}, \ldots, \hat{c}_{j,10}$. To consolidate noisy or inconsistent multi-view captions, GPT-4 [95] is employed as a caption summarizer, producing a coherent object tag c_j that identifies repeatedly mentioned objects and accounts for containers or surfaces.

Scene Graph Construction and Edge Inference The final stage constructs the relational structure of the scene graph by inferring spatial relationships between objects. The system computes 3D bounding box intersection-over-union (IoU) for every object pair, then prunes the resulting dense graph via minimum spanning tree (MST) to retain only salient relationships. For each retained edge, GPT-4 determines the semantic relationship label given object captions and 3D locations, describing spatial relationships (e.g., "object A on object B") and their reasoning. This LLM-based approach extends naturally to open-vocabulary predicates (e.g., "a backpack may be stored in a closet") without training relationship classifiers.

Implementation and Evaluation The standard configuration uses SAM for segmentation, CLIP for feature extraction, LLaVA-7B for vision-language captioning, and GPT-4 for caption refinement and edge inference, and Scene Graph Construction by ConceptGraphs has been evaluated by human evaluators on the Replica Dataset [3], identifying 23–60 valid objects per scene with only 0–5 duplicates, achieving approximately 70% label accuracy and 88% edge accuracy [2]. Furthermore, the framework achieves 40.63% mean accuracy on open-vocabulary 3D semantic segmentation without fine-tuning [2], while real-world deployments

on Clearpath Jackal and Boston Dynamics Spot robots demonstrate applications including open-vocabulary object retrieval (100% R@1 on complex queries), traversability reasoning, and dynamic map updates [2].

3.2.5 Geometric Edge Generation

In the original ConceptGraphs pipeline, spatial relationships between objects are inferred by a large language model that reasons jointly over object captions and 3D positions. While this yields rich, language-level predicates, it also makes edge construction one of the most computationally expensive components of the system and introduces a heavy dependence on proprietary models such as GPT-4, with associated latency and economic costs. In this thesis, the focus is on reliable spatial reasoning for navigation and active exploration rather than on open-ended linguistic predicates. For this reason, the edge generation module is replaced with a deterministic, purely geometric procedure that operates directly on 3D bounding boxes.

Implementation details

Each object o_i is modeled by an oriented 3D bounding box with center, extent, and, when available, rotation. From this box, the algorithm derives (i) the vertical interval along the Y axis and (ii) a footprint on the floor plane, obtained by projecting the box vertices onto the XZ plane and taking their convex hull. The reference system adopted is a right-handed coordinate system with Y pointing upwards, as used in all experiments. These quantities are then used to compute a small set of spatial relations:

- **on top of / supported by**: two objects are in an *on top of* relation if the bottom of the upper object is within a small vertical distance of the top of the lower one and their floor-plane footprints significantly overlap. The inverse relation is recorded as *supported by*. These relations capture physical contact between objects, e.g., a book on a table.
- **under / over**: an object is *under* another one when its center lies below the other along Y, the vertical gap between them is within a given range, and their footprints overlap on the floor plane. The inverse relation is recorded as *over*. This relation captures vertical adjacency without requiring direct contact, e.g., a chair under a table.
- **inside**: containment is detected by approximating the overlap between the two bounding-box volumes; if the overlap volume covers most of the smaller box, the smaller object is considered *inside* the larger one. This relation captures cases such as a cup inside a cabinet.

- **next to**: two objects are *next to* each other if the minimum distance between their floor-plane footprints is below a threshold and their vertical extents overlap sufficiently, indicating that they are side by side at similar height. This relation captures horizontal adjacency, e.g., a sofa next to a coffee table.

For each ordered pair of objects, these predicates are evaluated in a fixed priority order (from the most specific (like *supported by*) to the more general (like *next to*)), and the first satisfied relation is assigned, ensuring at most one edge per pair and keeping the graph sparse. All the thresholds involved in the relation definitions (e.g., maximum vertical gap for *supported by*, minimum footprint overlap for *next to*, etc.) are exposed as parameters that can be tuned per dataset or application.

Advantages and trade-offs of the geometric approach

Compared to the original LLM-based edge inference, the proposed geometric edge generator:

- eliminates all external API calls and associated costs and latency;
- is fully deterministic and reproducible across runs and environments;
- degrades gracefully under noisy captions or segmentation, since it relies only on 3D geometry;
- exposes explicit thresholds that can be tuned per dataset or application.

On the other hand:

- it cannot infer abstract or functional relations that are not directly grounded in geometry;
- it's heavily affected by the quality of the 3D bounding boxes.

This trade-off is acceptable in the context of this thesis, where the primary requirement is a reliable spatial structure to support active exploration and scene understanding, rather than rich language-level reasoning. For applications requiring richer semantics, the geometric edges can be augmented with LLM-based inference in a hybrid approach.

3.3 Automated Validation Framework

To quantitatively evaluate the quality of generated scene graphs, this work introduces an automated validation framework to replace the original human annotation-based evaluation used in ConceptGraphs [2].

3.3.1 Motivation

Evaluating open-vocabulary methods is intrinsically challenging. Unlike closed-set benchmarks, where predictions can be compared against a fixed list of class names, open-vocabulary 3D scene representations must handle free-form textual labels and relationships whose wording varies across annotators and datasets. Two descriptions may refer to the same object with different levels of specificity (e.g., "chair", "office chair", "swivel chair"), or may be only partially correct but still acceptable in context, and there is no single canonical way to map such labels back to a small discrete taxonomy.

The original ConceptGraphs evaluation [2] relies on human annotators to manually identify ground-truth objects in each scene and then judge whether detected objects match these annotations in terms of position and label; however, this procedure is labor-intensive, does not scale to many scenes or experimental settings, and remains partly subjective and difficult to reproduce, since different annotators or annotation sessions may yield slightly different "ground truth".

In the context of this thesis, where many experiments must be run across multiple environments and exploration strategies, such a manual protocol would make systematic comparison practically infeasible, and for this reason a fully automatic validation pipeline was developed instead. Given that the proposed edge-generation module is completely geometry-based, fully deterministic, and has been extensively tested, the evaluation focuses on node quality, while edges are not quantitatively validated and are instead treated as a reliable by-product of the underlying geometric reasoning.

3.3.2 Ground Truth

The validation framework assumes access to a ground truth set of objects for each scene. In the case of the Replica and ReplicaCAD datasets [3, 7], the ones adopted in this work, this information is provided directly by the dataset annotations: for every object, the 3D position, semantic label, bounding box extent, and bounding box rotation are available. These quantities are exactly the ones required by the proposed pipeline and therefore serve as ground truth for node evaluation, without any additional manual annotation.

3.3.3 Node Evaluation

Single-scene matching. For a given scene, ground-truth nodes and predicted nodes are loaded and, for each node, the 3D center and textual label are extracted. The labels are encoded with a SentenceTransformer model (`all-MiniLM-L6-v2` [96]), yielding L2-normalized embeddings, and a cosine-similarity matrix is computed between all pairs of ground-truth and predicted labels. Matching is formulated as a

joint semantic–geometric problem: for every ground-truth node, all still-unmatched predictions are considered, and the candidate that simultaneously satisfies a minimum cosine similarity (parameter `MIN_SIM`) and a maximum Euclidean distance between centers (parameter `MAX_DIST`) is selected. Among candidates that pass both thresholds, the pair with highest similarity (and, in case of ties, smallest distance) is chosen. This greedy one-to-one assignment produces a set of matched pairs together with their similarity and distance values. From these matches, the numbers of true positives TP (matched pairs), false positives FP (predicted nodes that remain unmatched), and false negatives FN (ground-truth nodes that remain unmatched) are obtained. Precision, recall, and F1-score are defined in terms of TP, FP, and FN as:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{3.5}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3.6}$$

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.7}$$

where precision measures the fraction of predicted nodes that are correct, recall measures the fraction of ground-truth nodes that are detected, and F1-score provides a balanced summary of both.

Variants with Iterative Closest Point (ICP) and post-processing. To analyse specific failure modes of the pipeline, two implementation variants of the node evaluation are introduced.

The first variant examines the effect of **global pose and scale errors** using Iterative Closest Point (ICP) [21]. Both ground-truth and predicted geometry are first expressed in the same world coordinate frame (that of the ground truth), and ICP is initialised with the identity transform, treating the predicted cloud as source and the ground-truth cloud as fixed target. Starting from the ground-truth and predicted node centers, Open3D's point-to-point ICP algorithm then iteratively alternates between finding closest-point correspondences and updating a single global rigid or similarity transform so as to minimise the squared Euclidean distance between matched points, with an option to estimate an additional isotropic scale factor; concretely, this is implemented via Open3D's `TransformationEstimationPointToPoint` with optional scaling and a call to `registration_icp` using the identity matrix as initial transform, with the predicted cloud as source and the ground-truth cloud as target [97]. In an alternative configuration, ICP is run on the full predicted and ground-truth scene point clouds rather than on node centres; however, in practice this variant brought only marginal improvements in alignment metrics

and did not noticeably change downstream evaluation trends, so the node-centre configuration is adopted as the default for the experiments. The algorithm outputs a 4×4 homogeneous transformation matrix encoding the estimated rotation, translation, and (when enabled) scale, together with standard ICP fitness and inlier RMSE scores that quantify alignment quality. When scale estimation is enabled, the resulting similarity transform is further decomposed into a pure rotation, a translation, and an isotropic scale factor, which is also reported as a diagnostic quantity. This variant is particularly informative for evaluating MapAnything's metric scale prediction: in scenarios with few input images (for example, when only few external camera views are available), scale can be poorly constrained, and comparing metrics before and after ICP reveals how much error is due to a global similarity mismatch rather than local node detection failures, while also providing a direct metric for scale prediction quality.

The second variant focuses on **duplicate removal**. ConceptGraphs-style pipelines can occasionally produce multiple nodes for the same physical object when multi-view association is imperfect, an effect that can be exacerbated when depths are estimated. To study this effect, a post-processing step is applied to predicted nodes: labels are embedded again, pairwise cosine similarities between predicted labels are computed, and pairs of nodes with both high semantic similarity (over a threshold, usually 0.7) and very small center distance (below a threshold, usually 0.3m) are treated as duplicates. For each such pair, only the instance with higher confidence (e.g., higher average detection score across views) is retained, which reduces duplicate nodes and makes precision values more reflective of true object discovery.

Dataset-level validation pipeline. At dataset scale, for each scene, the following steps are executed:

1. **World-frame alignment.** Predicted detections are transformed from the camera frame into the world frame. Since MapAnything predicts poses relative to the pose of the first input image, the world-frame camera pose for that first view is required to align the reference system and express both predictions and ground truth in the same coordinate system.

2. **Baseline matching.** Single-scene matching, as described above, is run once on the aligned ground-truth and predicted nodes, yielding baseline values of TP, FP, FN, precision, recall, and F1-score.

3. **ICP-based refinement (optional).** If ICP evaluation is enabled, an ICP registration between predictions and ground truth is performed (either using only centers or using full point clouds, with or without scale estimation). The resulting transform is applied to the predicted nodes, and the matching

procedure is repeated on these ICP-refined predictions to obtain metrics that factor out residual global pose and scale errors.

4. **Post-processing and matching (optional).** If duplicate removal is enabled, the post-processing step is applied to the (possibly ICP-refined) predictions, and single-scene matching is run again on this de-duplicated node set, producing a third set of metrics.

These variants are exploited in the Experiments and Discussion chapters to disentangle the influence of global alignment quality, scale estimation, and duplicate nodes on the overall accuracy of the generated 3D scene graphs.

3.4 Active Exploration for 3D Scene Graph Construction

While the RGB-only pipeline enables scene graph generation from arbitrary RGB images, the quality and completeness of the resulting graph depend critically on the choice of viewpoints. Most existing 3D scene graph methods assume that a comprehensive dataset or trajectory is provided (e.g., a video walkthrough of the environment), but this passive approach is inefficient: many redundant views may be captured while informative regions remain unobserved.

This section presents an active exploration framework that intelligently selects viewpoints to efficiently construct high-quality scene graphs with minimal observations. Two complementary strategies are adopted:

- **Geometric density-based exploration** (Section 3.4.2): Uses the Surface Edge Explorer (SEE) [5] algorithm to maximize 3D reconstruction coverage.
- **Semantic-driven exploration** (Section 3.4.4): Uses Active Semantic Perception (ASP) [6] to reason about unobserved objects and room structure.

3.4.1 Problem Formulation

Let $\mathcal{V}$ denote the set of all possible viewpoints in the environment, where each viewpoint $v \in \mathcal{V}$ is parameterized by 3D position and orientation $(x, y, z, \text{yaw}, \text{pitch}, \text{roll})$. Given a current scene graph G_t and point cloud $\mathcal{P}_t$ constructed from observations $\{I_1, \ldots, I_t\}$, the next-best-view (NBV) planning problem is to select the viewpoint v^*_{t+1} that maximizes expected improvement in scene graph quality:

$$v^*_{t+1} = \arg\max_{v \in \mathcal{V}} \text{Gain}(v \mid G_t, \mathcal{P}_t) \tag{3.8}$$

where $\text{Gain}(v \mid G_t, \mathcal{P}_t)$ quantifies the expected information gain from observing the scene from viewpoint v. The definition of information gain depends on the exploration strategy:

- **Geometric strategies** maximize coverage of unobserved or sparsely observed surfaces.
- **Semantic strategies** maximize reduction in uncertainty about object identities, locations, and room structure.

After selecting v^*_{t+1}, the robot navigates to this viewpoint, captures image I_{t+1}, and updates the scene graph and point cloud to obtain G_{t+1} and $\mathcal{P}_{t+1}$. This process repeats for a fixed number of iterations or until a coverage criterion is satisfied.

3.4.2 Geometric Density-Based Exploration: Surface Edge Explorer (SEE)

The Surface Edge Explorer (SEE) algorithm, proposed by Border and Gammell [4, 5], has been adopted in this thesis as the primary geometric exploration strategy.

SEE is a Next Best View (NBV) planning method that operates directly on unstructured point cloud data, without relying on intermediate representations such as voxel grids or surface meshes. This makes it particularly well suited to the pipeline adopted in this thesis: since MapAnything [1] already produces point cloud reconstructions, SEE can be integrated directly, without introducing additional geometric data structures or preprocessing stages.

Methodology

SEE operates on an unstructured point cloud representation, classifying measurements based on local point density into three categories:

- **Core points**: densely observed surface regions where sufficient measurements have been acquired.
- **Frontier points**: sparsely observed boundary regions at the edge of known surfaces, indicating areas requiring additional observation.
- **Outlier points**: isolated measurements likely caused by sensor noise or transient objects.

Point classification uses two key parameters: an ϵ-radius defining the neighborhood for density queries, and a density threshold ρ determining the minimum number of neighbors required for a point to be considered a core point. Points with

a local density of neighbors less than ρ within radius ϵ are classified as frontier points (if adjacent to core points) or outliers (if isolated).

For each frontier point $\mathbf{f}$, SEE estimates the local surface geometry by fitting a plane to nearby measurements through eigendecomposition of the neighborhood covariance matrix. This produces three orthogonal vectors: a surface normal $\mathbf{e}_n$, a frontier vector $\mathbf{e}_f$ pointing toward partially observed regions, and a boundary vector $\mathbf{e}_b$ tangent to the density boundary. A candidate viewpoint is then proposed at distance d along the estimated surface normal direction from $\mathbf{f}$. This view proposal strategy naturally orients the sensor to observe the incompletely scanned surface region.

A key innovation in SEE is **proactive occlusion detection**. Before selecting a next best view, SEE performs occlusion queries for the τ nearest view proposals to the current sensor position. For each candidate view $\mathbf{v}$, the algorithm determines which frontier points would be visible by checking if ray paths from $\mathbf{v}$ to frontier points intersect with known occupied surfaces.This occlusion awareness prevents the robot from moving to viewpoints that would not actually observe the intended frontier regions.

SEE constructs a *frontier visibility graph* where nodes represent (frontier point, view proposal) pairs, and directed edges from a parent node to a child node indicate that the child frontier point is visible from the parent view. This graph quantifies the predicted surface coverage improvement from each view as the number of visible frontier points (the node's outdegree). The next best view is selected to maximize the ratio of visible frontier points to travel distance from the current sensor position. Formally, the NBV $\mathbf{m}^*$ is chosen as:

$$\mathbf{m}^* = \arg\max_{\mathbf{m}\in\mathcal{M}'} \frac{\deg(\mathbf{m})}{\|\mathbf{x} - \mathbf{x}_c\|} \tag{3.9}$$

where $\mathcal{M}'$ is the set of permissible view proposals, $\deg(\mathbf{m})$ is the outdegree (number of visible frontiers from the view proposal), $\mathbf{x}$ is the view position, and $\mathbf{x}_c$ is the current sensor position. To prevent the sensor from selecting distant views and then needing to return to observe closer surfaces, SEE imposes a minimum connectivity constraint: the selected view must have an edge to the frontier point associated with the closest view proposal.

Advantages and Performance

SEE's measurement-direct approach offers several advantages over volumetric and surface-based representations. Computationally, it requires no maintenance or updating of volumetric grids or surface meshes; point density queries use efficient spatial data structures (kd-trees), and the computational cost scales with the number of measurements captured rather than scene volume or structural resolution. This

enables SEE to achieve high observation fidelity without loss of detail due to voxel resolution limits, because the observation quality is determined solely by the user-specified density threshold ρ. Additionally, view selection operates directly on measurements without assumptions about downstream data processing (e.g., mesh reconstruction, object segmentation), decoupling sensing from final representation.

Border and Gammell [5] demonstrate through extensive simulated and real-world experiments that SEE achieves similar or better surface coverage compared to several state-of-the-art volumetric NBV approaches, while requiring less observation time, fewer views, and shorter travel distances. Real-world experiments on a UR10 robotic arm observing complex objects (e.g., a deer statue with varied geometry and self-occlusions) confirm SEE's effectiveness with real sensors and measurement noise.

SEE as a Baseline

In the context of scene graph construction, SEE serves as a geometric baseline: it does not leverage semantic information from the scene graph and instead focuses purely on achieving complete surface coverage. By maximizing point cloud density, SEE indirectly discovers new objects, but it does not explicitly reason about object identities or semantic uncertainty.

Comparing SEE with semantic-driven exploration (Section 3.4.4) quantifies the value added by semantic reasoning for scene graph construction tasks.

3.4.3 Modifications to Original SEE

To adapt SEE to the specific requirements of this thesis, key modifications were implemented to the original code. Some figures illustrating the effect of these modifications are included in the following sections, and the code snippets provided are taken from the actual implementation used in the experiments. For all the figures below, blue arrows indicate candidate view proposals, magenta arrow indicate the selected view and black lines connect the selected view with the visible frontier points from that view.

Integration with Vision-Based Reconstruction

In the original SEE formulation, the point cloud is assumed to be provided directly by a depth sensor (e.g., RGB-D camera or LiDAR). In this work, SEE operates on the point cloud $\mathcal{P}_t$ reconstructed by MapAnything from RGB images at step t. At each exploration step, SEE receives the point cloud that corresponds to the current observation and incrementally updates its own global point cloud representation. This allows SEE to avoid re-classifying all points (as core, frontier or outlier) at every step and instead focus on newly observed regions, as the original code does.

Reduced Distance Weighting for Exploration

In the original SEE formulation, the next best view is selected to maximize the ratio of visible frontier points to travel distance (Equation 3.9). This formulation strongly penalizes distant views, prioritizing local coverage completion. Since perfect 3D reconstruction is not the goal of this active exploration framework, broader environment coverage is prioritized over exhaustive local scanning. To reflect this objective, a configurable distance weight parameter was introduced to reduce the distance penalty in the view selection criterion, encouraging exploration of distant regions rather than dense local reconstruction. The original Equation 3.9 was modified to:

$$\mathbf{m}^* = \arg\max_{\mathbf{m}\in\mathcal{M}'} \frac{\deg(\mathbf{m})}{\|\mathbf{x}-\mathbf{x}_c\|^{\alpha}} \tag{3.10}$$

where $\alpha = 0.5$ in this implementation. This modification encourages the robot to explore more distant regions rather than exhaustively scanning nearby surfaces.

Field-of-View Constrained Visibility

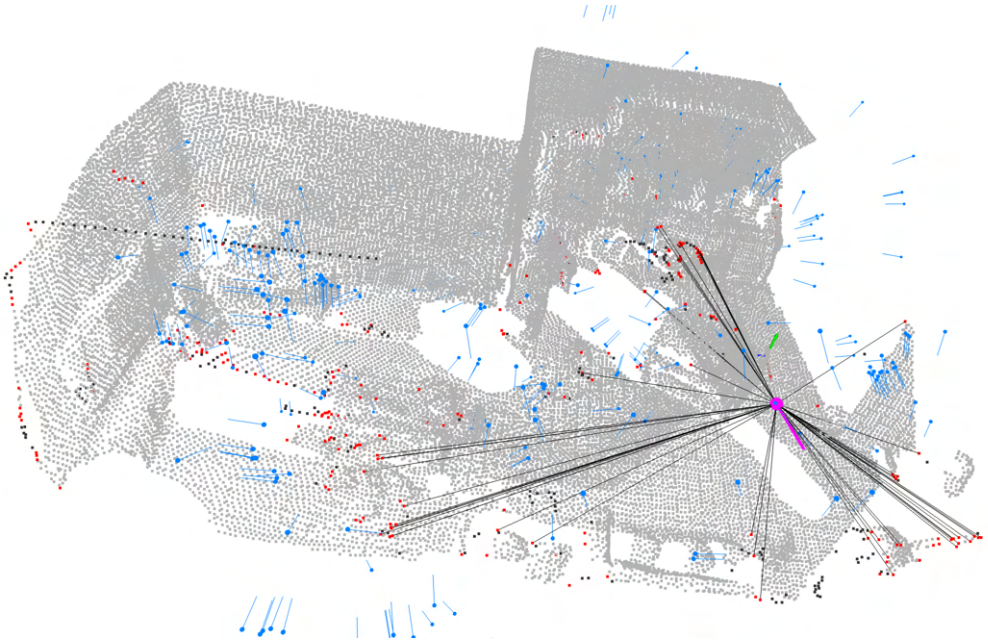
The original SEE implementation performs visibility checks assuming a 360-degree omnidirectional sensor. To account for the limited field of view (FoV) of a standard RGB camera, the visibility test was modified to include frustum culling. For each candidate view $\mathbf{v}$, a frontier point $\mathbf{f}$ is considered visible only if it lies within the camera's rectangular frustum. Figure 3.5 illustrates the effect of this modification.

Height Band Filtering

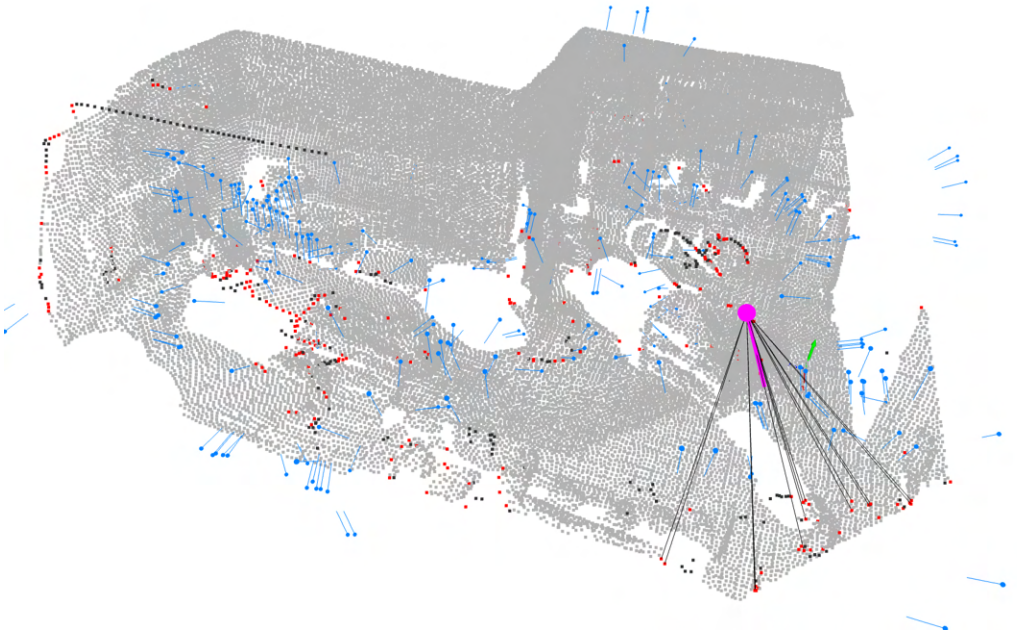
To exclude irrelevant geometric structures such as ceilings, which do not contribute to ground-level scene understanding but increase computational cost, a vertical filtering mechanism was introduced. . Points with Y coordinates (height) outside a predefined band $[y_{\min}, y_{\max}]$ are classified as outliers during the point processing stage, preventing them from being considered as core or frontier points. This simple modification focuses the algorithm on the navigable environment and reduces unnecessary mapping of overhead surfaces. Figure 3.6 illustrates the effect of this modification.

Hemisphere-Split Density Classification

A critical issue in the original SEE implementation occurs when a point has sufficient neighbors within radius ϵ to exceed the density threshold ρ, but all neighbors are concentrated in one hemisphere, as illustrated in Figure 3.7a. This causes boundary points to be incorrectly classified as core points rather than frontiers, preventing



(a) Original Visibility Check



(b) FoV Constrained Visibility Check

Figure 3.5: Comparison of visibility checks. The original visibility check (a) considers all frontier points visible from the candidate view, while the modified check (b) correctly identifies only those within the camera's FoV as visible.

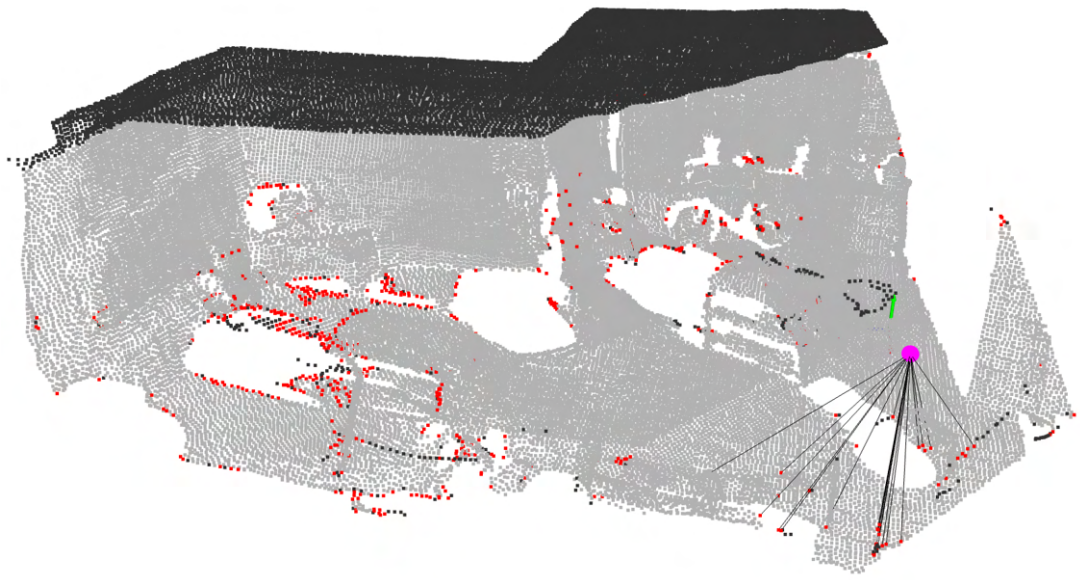
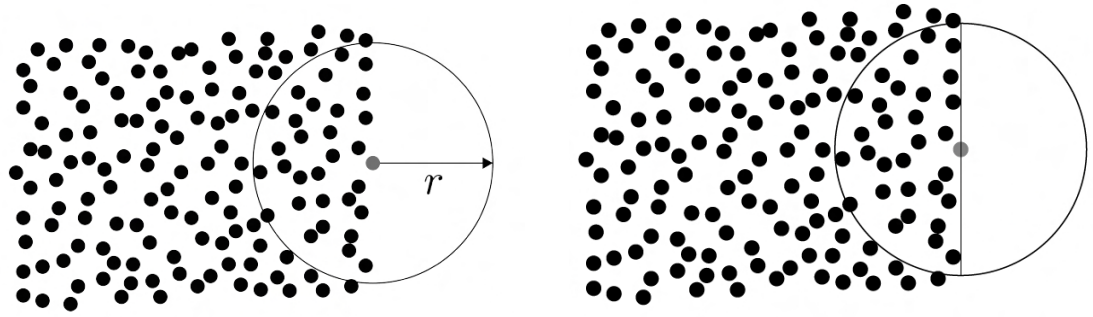


Figure 3.6: Effect of height band filtering. Points outside the specified height range (e.g ceiling points) are classified as outliers (black). Gray points are core points, red points are frontier points, and black points are outliers.

the algorithm from recognizing incomplete surface coverage. A possible solution would be to increase the density threshold ρ, but this would significantly increase computational cost and may still fail in cases where the density is sufficient but unbalanced.

To address this, the original core point classification was modified to require balanced density in both hemispheres. The neighborhood sphere is split into two hemispheres by a plane whose normal aligns with the local frontier direction $\mathbf{e}_f$, as shown in Figure 3.7b. A point is classified as core only if both hemispheres contain at least $\rho/2$ neighbors. This ensures that points at the edge of observed surfaces, which have neighbors only on one side, are correctly classified as frontiers.

Aligning the splitting plane with the frontier direction ensures that the core region (already well observed) is effectively separated from the frontier direction (requiring additional observation). This modification significantly improves frontier detection at surface boundaries without increasing the point density threshold, thereby avoiding additional computational cost. Figure 3.8 illustrates the impact of this adjustment on point classification. Compared to the previous result in Figure 3.6, the improvement is evident: all boundary points are now correctly identified as frontier points (in red).



(a) Original classification: boundary point. **(b)** Correct classification: frontier point.

Figure 3.7: Illustration of the hemisphere-split density classification issue. In (a), the point in the center has enough neighbors to be classified as a core point, but all neighbors are located in one hemisphere, indicating that the surface is not fully observed and this point should actually be classified as a frontier. In (b), the modified classification correctly identifies this point as a frontier by requiring balanced neighbor distribution across both hemispheres.

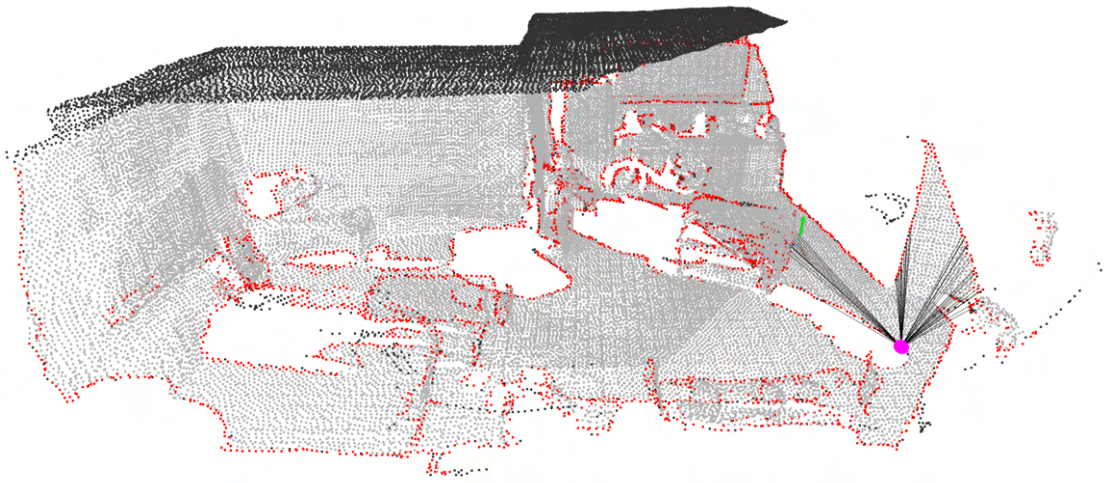


Figure 3.8: Effect of hemisphere-split density classification. The original classification (previous images) incorrectly classifies boundary points as core points due to unbalanced neighbor distribution, while the modified classification correctly identifies frontier points (like those at the border of unknown regions) by requiring balanced density in both hemispheres.

Kinematically Feasible View Proposal

The original SEE algorithm proposes candidate viewpoints at distance d along the surface normal direction:

$$\mathbf{v} = \mathbf{f} - d \cdot \mathbf{e}_n \tag{3.11}$$

This is appropriate for a generic sensor but problematic for a ground robot: for horizontal surfaces such as the floor or table tops, the normal is approximately vertical, so the corresponding view lies directly above the surface and looks strictly downward. Such viewpoints are typically unreachable for a mobile robot with a camera mounted at a limited height, and even when technically reachable, they require unrealistically large tilts of the camera. To ensure that all view proposals are physically realisable, the view generation step was modified to enforce simple kinematic constraints on both the camera position and orientation:

```
1  void SeeCore::EnforceKinematics(Eigen::Vector3f &view,
2                                  Eigen::Vector3f &pose,
3                                  const Eigen::Vector3f &frontier)
     {
4    float max_cam_height = 1.4f;
5    float min_cam_height = 0.0f;
6    float max_pitch_rad = 45.0f * (M_PI / 180.0f);
7
8    bool height_bad = (view.y() > max_cam_height || view.y() <
      min_cam_height);
9    float current_pitch = asin(pose.y());
10   bool pitch_bad = (std::abs(current_pitch) > max_pitch_rad);
11
12   if (height_bad || pitch_bad) {
13     // Flatten the normal to the ground plane (XZ)
14     Eigen::Vector3f flat_dir(pose.x(), 0.0f, pose.z());
15
16     // Construct new safe orientation
17     float safe_pitch = (view.y() > frontier.y()) ? -max_pitch_rad
      : max_pitch_rad;
18     float cos_p = cos(safe_pitch);
19     float sin_p = sin(safe_pitch);
20     pose = Eigen::Vector3f(flat_dir.x() * cos_p, sin_p, flat_dir.z
      () * cos_p);
21
22     // Recalculate view position
23     view = frontier - d_val * pose;
24
25     // Safety clamp for height
26     if (view.y() > max_cam_height) {
27       view.y() = max_cam_height;
28       pose = (frontier - view).normalized();
29     } else if (view.y() < min_cam_height) {
30       view.y() = min_cam_height;
31       pose = (frontier - view).normalized();
32     }
```

```
33     }
34 }
```

Listing 3.1: Kinematically Feasible View Proposal in SEE

Given an initial view position $\mathbf{v}$ and view direction (camera optical axis) $\mathbf{p}$, the algorithm first checks whether either the height or pitch constraints are violated. If the view is infeasible, the normal-based proposal is "flattened" into a kinematically valid configuration in two steps:

- **Projection onto the ground plane**: The vertical component of the view direction is removed, and only the horizontal components in the xz-plane are retained. This defines a horizontal direction along which the robot can move while maintaining a camera height compatible with its mounting.
- **Safe pitched orientation**: A new camera pitch $\theta_{\text{safe}} \in -\theta_{\max}, \theta_{\max}$ is chosen depending on whether the view is currently above or below the frontier. This yields a view direction with a bounded vertical component. The camera thus remains tilted towards the frontier point but never more than $\theta_{\max}$ away from the horizontal. The final step clips the resulting camera height into the admissible band for robot height and updates the viewing direction to continue pointing at the frontier.

This ensures all proposed views are physically reachable by a ground-based mobile robot with camera mounted at a fixed height, as clearly visible in Figure 3.9.

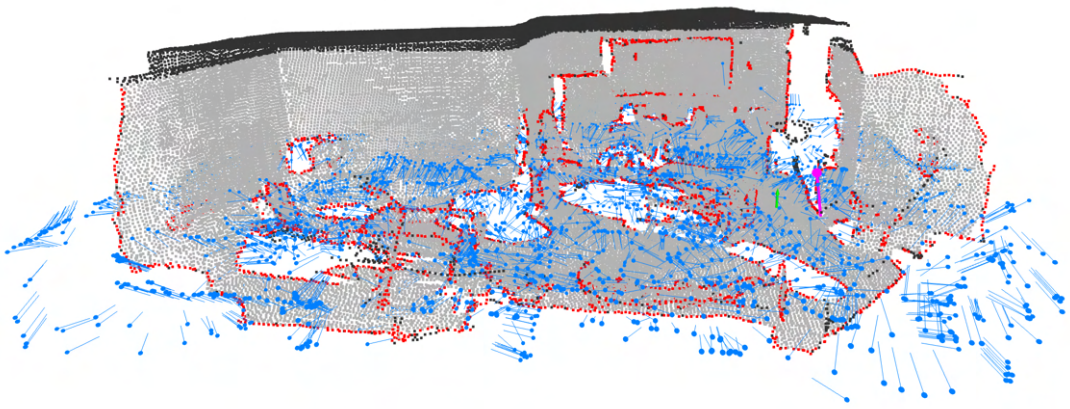


Figure 3.9: Effect of kinematically feasible view proposal. All the proposed views (blue arrows) are reachable by a ground-based mobile robot with a camera mounted at a fixed height.

3.4.4 Semantic-driven Exploration: Active Semantic Perception (ASP)

While geometric approaches like SEE efficiently achieve complete surface coverage, they do not leverage semantic knowledge about the environment. In the context of constructing 3D scene graphs, semantic information can guide exploration more intelligently by reasoning about spatial relationships, object locations, and likely scene configurations, directly using the evolving scene graph representation.

For this reason, the Active Semantic Perception (ASP) framework proposed by Tang and Chaudhari [6] is adopted as semantic-driven exploration in this thesis.

Methodology

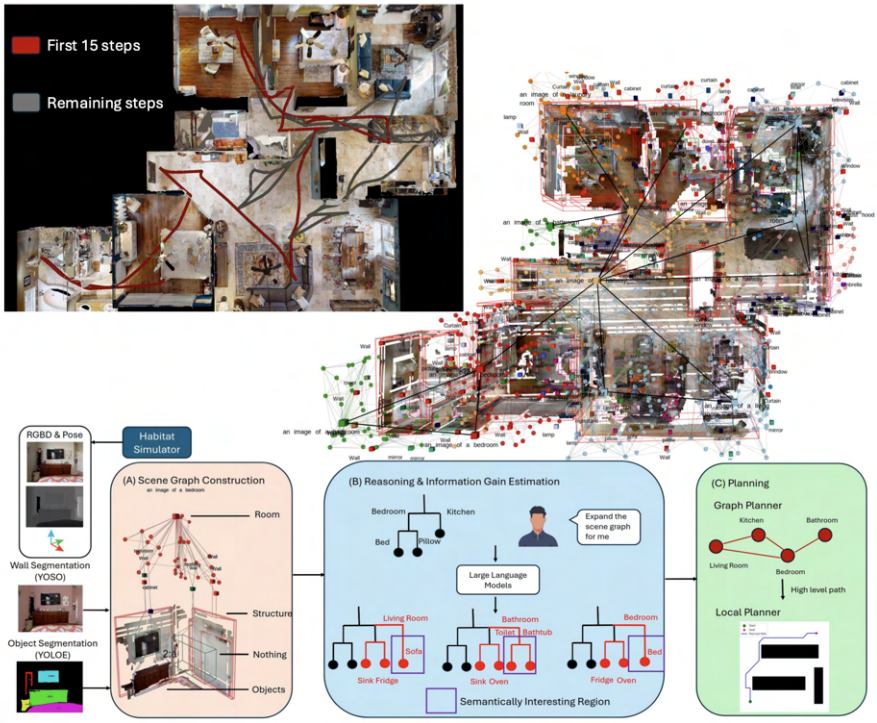


Figure 3.10: Active Semantic Perception (ASP) framework. The robot incrementally constructs a hierarchical scene graph from observations, uses an LLM to sample plausible completions of the unobserved scene, and selects the next viewpoint by maximizing expected information gain about semantic uncertainty. Figure adapted from [6].

As illustrated in Figure 3.10, ASP builds a compact, hierarchical multi-layer scene graph representation that captures the environment at multiple levels of

abstraction. Nodes in the scene graph correspond to rooms, objects, walls, windows, and other semantic entities, while edges encode spatial relationships (e.g., "adjacent to", "inside", "on top of") and semantic associations.

As the robot explores, it incrementally constructs the scene graph from sensor observations using semantic segmentation and object detection. The key innovation is the use of *LLM-based scene graph completion* to reason about unobserved regions. Given the partial scene graph constructed from observations so far, the LLM samples plausible completions of the full scene graph that are consistent with the partial observations.

For example, if the robot has observed a living room with two doors, the LLM might generate hypotheses such as: "door 1 likely leads to a kitchen" or "door 2 likely leads to a bedroom", based on typical spatial configurations in indoor environments learned from large-scale language corpora.

Information Gain for Spatial Reasoning

Multiple plausible completions of the partially observed scene graph are sampled using the LLM, inducing a distribution over possible unobserved scene configurations. For each candidate waypoint (i.e., potential next viewpoint), Active Semantic Perception evaluates how informative an observation from that viewpoint would be in reducing semantic uncertainty.

Formally, let x denote a candidate viewpoint, G_k the current scene graph encoding past observations, and Y_{k+1} the future observation. The next viewpoint is selected by maximizing the expected information gain, defined as the mutual information

$$I(Y_{k+1}; G_k \mid x) = H(Y_{k+1} \mid x) - H(Y_{k+1} \mid x, G_k), \tag{3.12}$$

where $H(\cdot)$ denotes Shannon entropy. The first term captures the uncertainty of the predicted observation from viewpoint x, while the second term measures the remaining uncertainty conditioned on the current scene graph.

Since the true scene graph is only partially known, the predictive distribution over observations is approximated by sampling a set of plausible scene graph completions $\{G^{(i)}\}_{i=1}^{m} \sim p(G \mid G_k)$. Under this approximation,

$$p(y_{k+1} \mid x) \approx \frac{1}{m} \sum_{i=1}^{m} \delta\Big(y_{k+1} - Y(G^{(i)}, x)\Big), \tag{3.13}$$

where $Y(G^{(i)}, x)$ denotes the observation rendered from viewpoint x in the sampled scene graph $G^{(i)}$. To model observation uncertainty conditioned on a single graph, small geometric perturbations are applied to each sampled graph when estimating $H(Y_{k+1} \mid x, G_k)$.

In practice, information gain is computed separately for object-level and room-level semantics and combined as

$$I(x) = I_{\text{object}}(x) + \lambda\, I_{\text{room}}(x), \tag{3.14}$$

where λ balances fine-grained object uncertainty against higher-level spatial ambiguity. The next viewpoint is then selected as

$$x_{k+1} = \arg\max_{x} I(x). \tag{3.15}$$

This formulation prioritizes viewpoints that resolve semantic disagreements across sampled scene graphs. For example, if different hypotheses predict that a door leads to either a kitchen or a bedroom, observing beyond that door yields high information gain. This enables efficient exploration driven by high-level spatial reasoning rather than purely geometric novelty [6].

Advantages and Evaluation

Active Semantic Perception offers several key advantages for scene graph construction. Exploration is driven by semantic understanding rather than purely geometric coverage, allowing the robot to prioritize viewpoints that resolve semantically ambiguous or informative regions. By leveraging Large Language Models (LLMs), the system incorporates prior knowledge about typical spatial layouts, enabling contextual reasoning and informed hypotheses about unobserved areas. Moreover, the resulting scene graph representation is directly usable for downstream tasks such as object search, navigation, and task planning.

Tang and Chaudhari [6] evaluate Active Semantic Perception in simulation on large-scale and realistic indoor environments from the HM3D dataset [98] using the Habitat simulator [7, 99, 100]. Both qualitative and quantitative results show that the proposed approach recovers scene semantics more quickly and accurately than baseline methods that rely on purely geometric criteria for viewpoint selection.

Limitations and Challenges

The approach depends critically on the quality of LLM predictions and the ability of the semantic perception system (object detection, segmentation) to reliably recognize entities in the scene. LLM hallucinations or incorrect predictions about scene structure could lead to suboptimal exploration.

3.4.5 Modifications to Original ASP

The first key modification to the original ASP framework is that the ASP 3D scene graph extraction pipeline is replaced with the RGB-only 3D scene graph extraction pipeline described in Section 3.2.

In addition, the original ASP framework [6] assumes a hierarchical scene graph with explicit room nodes and room-object relationships. However, the ConceptGraphs-based pipeline developed in Section 3.2 produces a flat scene graph where nodes represent objects and edges encode pairwise spatial relationships. There is no explicit notion of rooms or higher-level spatial structure.

To adapt ASP for flat scene graphs, the following modifications are made.

1. The scene graph is treated as consisting only of object nodes with associated 3D positions, dimensions, and orientations obtained from the ConceptGraphs detections; explicit room nodes are not assumed in the raw representation.

2. When room-level reasoning is required, an optional preprocessing step infers pseudo-room nodes by clustering object centers in the horizontal plane, and represents each cluster by a synthetic room node whose position is the centroid of its member objects.

3. If pseudo-rooms are enabled, each object is connected to its assigned synthetic room node to provide the room–object edges required by ASP's high-level policies; if they are disabled, all objects are instead linked to a single default room node.

This allows the ASP framework to operate on the flat scene graph produced by the RGB-only pipeline while still enabling room-level reasoning when desired. An example of the resulting scene graph structure with two pseudo-rooms created is shown in Figure 3.11.

No modifications were made to the core LLM-based scene graph completion or information gain computation components of ASP, as these are agnostic to the specific structure of the scene graph and can operate on the adapted representation without change.

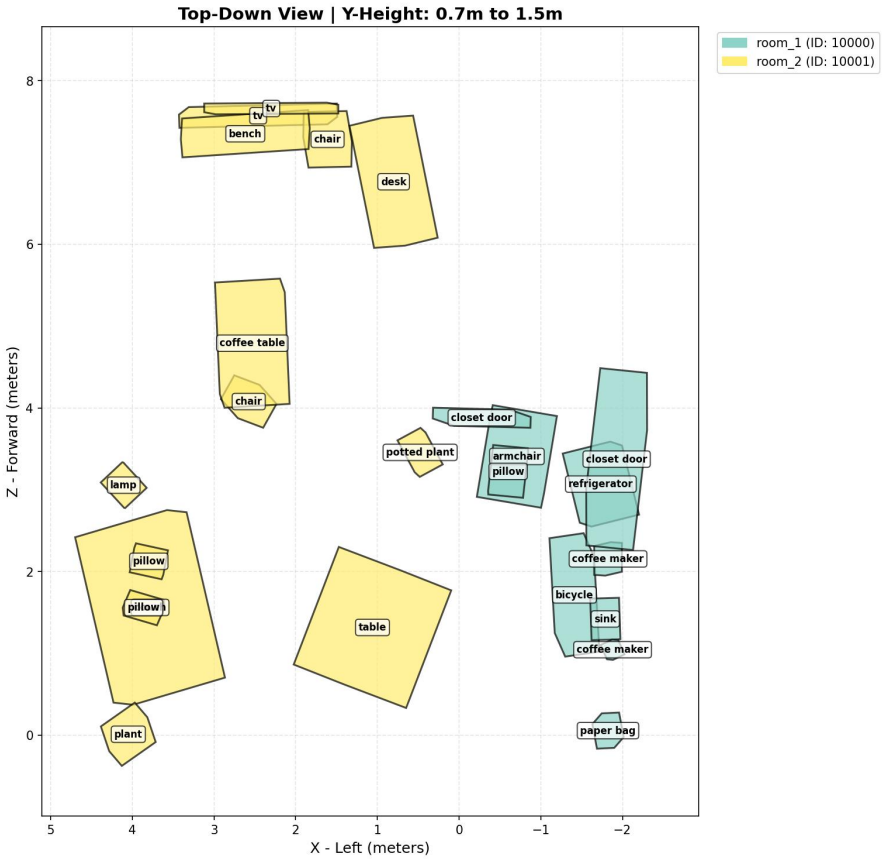


Figure 3.11: Top-down view of the scene graph structure adapted for ASP. Object nodes (e.g., chair, coffee table) are connected to synthetic room nodes (room 1, room 2) created by clustering object positions.

Chapter 4

Experiments and Results

This chapter presents a comprehensive experimental evaluation of the RGB-only 3D scene graph generation pipeline and active exploration framework developed in Chapter 3. The experiments are organized into three main studies: (i) validation of the RGB-only pipeline against the original ConceptGraphs method using ground-truth depth and pose, (ii) evaluation of active exploration strategies for scene graph construction, and (iii) investigation of external camera integration for scene understanding. For each experimental study, the motivation, dataset selection, simulator configuration, experimental protocol, and quantitative results are described in detail.

4.1 RGB-Only Pipeline Validation

4.1.1 Motivation and Objectives

The first set of experiments aims to validate the RGB-only 3D scene graph generation pipeline proposed in Section 3.2 by quantifying the accuracy loss introduced when creplacing ground-truth depth maps and camera poses with predictions from MapAnything [1]. This validation is essential to establish a performance baseline and to understand the trade-offs inherent in eliminating depth sensor requirements.

Specifically, these experiments address the following research questions:

- How much scene graph accuracy is lost when using estimated depth and pose instead of ground truth?
- Does the MapAnything reconstruction maintain sufficient geometric consistency for ConceptGraphs to build accurate 3D scene graphs?
- How do post-processing steps (ICP alignment and duplicate removal) affect scene graph quality?

4.1.2 Dataset Selection: Replica

For this validation study, the Replica dataset [3] was selected as the evaluation benchmark. Replica is a dataset of 18 high-quality reconstructions of indoor environments, featuring dense geometry, high-resolution HDR textures, and comprehensive semantic annotations including per-object ground-truth positions, labels, and bounding boxes [3]. Replica was specifically chosen because it is the primary dataset used in the original ConceptGraphs evaluation [2], enabling direct comparison with published results.

ConceptGraphs reports results on a subset of seven Replica scenes: `room_0`, `room_1`, `room_2`, `office_0`, `office_1`, `office_2`, and `office_3`, reported in Figure 4.1. To maintain consistency with the original evaluation protocol, the same eight scenes were used in these experiments.

Figure 4.1: Visualization of the eight Replica scenes used for RGB-only pipeline validation.

4.1.3 Data Preparation and Sampling

Dataset Acquisition

The Replica dataset was downloaded and prepared following the official ConceptGraphs instructions[1].

[1] `https://github.com/concept-graphs/concept-graphs?tab=readme-ov-file#prepare-dataset-replica-as-an-example`

Instead of using the original Replica dataset directly, the scanned RGB-D trajectories provided by Nice-SLAM[2] were downloaded and prepared. These trajectories consist of rendered sequences generated from the mesh models of the original Replica dataset.

This procedure provides 2000 RGB-D frames per scene, together with ground-truth camera poses for each frame.

Frame Subsampling Due to Memory Constraints

MapAnything loads all input frames into GPU memory simultaneously during inference, as it processes the entire image sequence in a single forward pass through the transformer architecture. Preliminary experiments revealed that processing 2000 frames exceeded the memory capacity of the available GPU hardware (NVIDIA GPU with 24GB VRAM on the remote server used for experiments).

To address this constraint while maintaining sufficient scene coverage, a uniform subsampling strategy was adopted. Since the original 2000 frames constitute a continuous video trajectory through the environment, sampling every 13th frame yields 154 frames per scene, which remained within the GPU memory budget while preserving adequate spatial coverage.

This subsampling does not fundamentally compromise the experimental validity for two reasons. First, 154 uniformly distributed frames still provide comprehensive coverage of each scene, as the original trajectory was designed to observe all regions thoroughly. Second, both the RGB-only pipeline and the baseline ConceptGraphs method operate on exactly the same 154-frame subset, ensuring a fair comparison.

4.1.4 Experimental Protocol

Two pipeline configurations were evaluated on each of the seven Replica scenes:

Baseline: ConceptGraphs with Ground Truth The original ConceptGraphs pipeline was executed using the ground-truth depth maps and camera poses provided by the Replica dataset. This configuration represents the upper bound on scene graph quality achievable with perfect geometric input.

Proposed: RGB-Only Pipeline The pipeline developed in Section 3.2 was executed using only the RGB images as input. MapAnything [1] predicted depth maps and camera poses for all 154 frames, which were then provided to ConceptGraphs for scene graph construction.

[2] `https://github.com/cvg/nice-slam/blob/master/scripts/download_replica.sh`

Both pipelines used identical ConceptGraphs hyperparameters, including segmentation model (SAM [17]), feature extractor (CLIP [18]), and object association thresholds. In particular, the merge thresholds were fixed to `merge_overlap_thresh = 0.7` and `merge_visual_sim_thresh = 0.7`. The first requires a sufficient overlap between the 3D bounding boxes of two hypotheses before they can be merged, while the second enforces a minimum cosine similarity between their CLIP image embeddings. These values were chosen empirically after several trials: when the thresholds were set lower, distinct objects were incorrectly merged into a single node, whereas higher thresholds significantly reduced merging and led to many duplicate nodes for the same physical object, degrading precision.

The only difference between the two configurations was the source of depth and pose information.

4.1.5 Evaluation Metrics and Variants

Scene graph quality was assessed using the automated validation framework described in Section 3.3. To isolate different sources of error, three evaluation variants were defined:

Original The predicted scene graph was transformed from the MapAnything coordinate frame to the Replica world frame using the known ground-truth pose of the first camera view, and evaluated directly without any refinement. This variant captures the raw performance of the RGB-only pipeline.

Post-Processing A post-processing step was applied to the predicted scene graph to remove duplicate object detections. Duplicate removal was performed by clustering object nodes based on their spatial proximity and semantic similarity, as described in 3.3.3. In particular, nodes were merged if they were within a distance threshold of 0.3 meters and if the cosine similarity between the embeddings of their predicted labels was above 0.7. These thresholds were determined empirically based on preliminary experiments to balance precision and recall in duplicate detection. This variant isolates errors due to duplicate detections, which can occur when MapAnything's depth and pose predictions lead to multiple fragmented reconstructions of the same object.

Iterative Closest Point (ICP) with scale ICP registration was applied to align the predicted scene graph to the ground truth using object centers as correspondence points. A similarity transformation (rotation, translation, and isotropic scale) was computed and applied to the predictions before evaluation. Since MapAnything predicts a global metric scale factor, comparing original performance with those obtained with scale refinement provides insight into the quality of MapAnything's

scale prediction. Note that scale estimation used only object centers, not full point clouds, as preliminary experiments showed negligible accuracy improvement with full point clouds while incurring significantly higher computational cost.

For all variants, precision, recall, and F1-score were computed following Equations 3.5, 3.6, and 3.7 in Section 3.3.

4.1.6 Results

Table 4.1 presents the scene graph node evaluation results for all seven Replica scenes. *CG* refers to the original ConceptGraphs pipeline with ground-truth depth and pose, while *CG-RGB* refers to the RGB-only pipeline using MapAnything predictions. The post-processing (*Post*) and *ICP* variants are also reported for each scene. *GT* represents the number of ground-truth object nodes, while Pred represents the number of predicted object nodes. *Scale* reports the scale factor estimated by MapAnything, which is applied in the ICP variant. *Precision*, *Recall*, and *F1-score* are computed based on node-level matching between predicted and ground-truth nodes. The average results across all scenes are reported at the end.

Just to provide a visual example of the experiments, Figure 4.2 shows a qualitative comparison between the original mesh and the MapAnything reconstruction for `office3`, while Figure 4.3 compares the resulting scene graphs obtained with the original ConceptGraphs pipeline and the RGB-only pipeline for the same scene. The visualizations confirm that MapAnything's reconstruction is generally accurate, although with some noise for small objects, and that the RGB-only pipeline produces a scene graph that captures the main objects and their relationships, although with some differences in node count and connectivity compared to the ground-truth-based pipeline.

4.1.7 Discussion

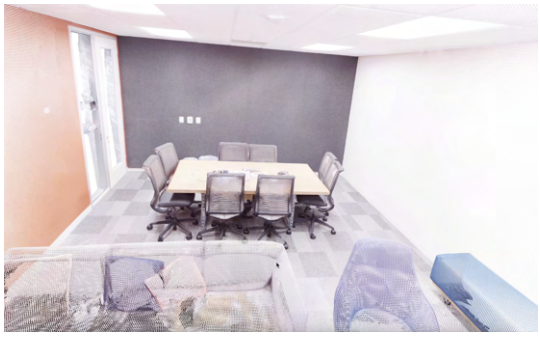
The results in Table 4.1 reveal that the RGB-only pipeline achieves scene graph quality that is comparable to that of the original ConceptGraphs pipeline with ground-truth depth and pose, despite relying entirely on MapAnything's learned depth, pose, and scale predictions.

Overall F1-score parity In terms of overall F1-score, the gap between the two pipelines is negligible. The raw CG-RGB pipeline achieves an average F1-score of 0.500, which is almost identical to the CG baseline of 0.499. When post-processing is applied to CG-RGB, the average F1 increases to 0.508, slightly exceeding the baseline by approximately 1.8%. This is a notable finding: replacing ground-truth geometric data with a learned feed-forward model introduces no measurable

Table 4.1: Scene graph node evaluation results for RGB-only 3D Scene Graph Generation pipeline validation.

Scene	Exp.	GT	Pred	Scale	Prec	Rec	F1
room0	CG	85	53	–	0.736	0.459	0.565
	CG-RGB	85	58	–	0.724	0.494	0.587
	CG-RGB + Post	85	57	–	0.737	0.494	0.592
	CG-RGB + ICP	85	57	1.002	0.737	0.494	0.592
room1	CG	47	39	–	0.692	0.575	0.628
	CG-RGB	47	48	–	0.604	0.617	0.611
	CG-RGB + Post	47	46	–	0.630	0.617	0.624
	CG-RGB + ICP	47	46	0.970	0.630	0.617	0.624
room2	CG	55	39	–	0.615	0.436	0.511
	CG-RGB	55	56	–	0.500	0.509	0.505
	CG-RGB + Post	55	52	–	0.539	0.509	0.523
	CG-RGB + ICP	55	52	0.943	0.539	0.509	0.523
office0	CG	57	27	–	0.704	0.333	0.452
	CG-RGB	57	27	–	0.667	0.316	0.429
	CG-RGB + Post	57	27	–	0.667	0.316	0.429
	CG-RGB + ICP	57	27	0.996	0.667	0.316	0.429
office1	CG	39	24	–	0.667	0.410	0.508
	CG-RGB	39	29	–	0.655	0.487	0.559
	CG-RGB + Post	39	29	–	0.655	0.487	0.559
	CG-RGB + ICP	39	29	1.113	0.621	0.462	0.529
office2	CG	85	35	–	0.743	0.306	0.433
	CG-RGB	85	48	–	0.562	0.318	0.406
	CG-RGB + Post	85	46	–	0.587	0.318	0.412
	CG-RGB + ICP	85	46	1.058	0.565	0.306	0.397
office3	CG	94	42	–	0.643	0.287	0.397
	CG-RGB	94	49	–	0.592	0.309	0.406
	CG-RGB + Post	94	44	–	0.659	0.309	0.420
	CG-RGB + ICP	94	44	1.040	0.659	0.309	0.420
Average	CG	66	37	–	0.686	0.401	0.499
	CG-RGB	66	45	–	0.615	0.436	0.500
	CG-RGB + Post	66	43	–	0.639	0.436	0.508
	CG-RGB + ICP	66	43	1.017	0.631	0.430	0.502

(a) Original mesh for `office3`.

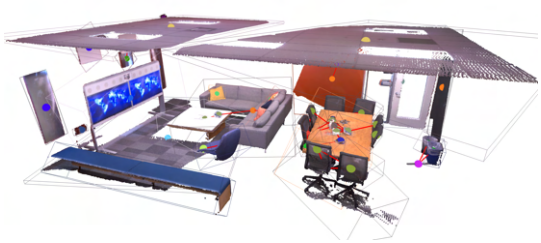


(b) Original mesh for `office3`.

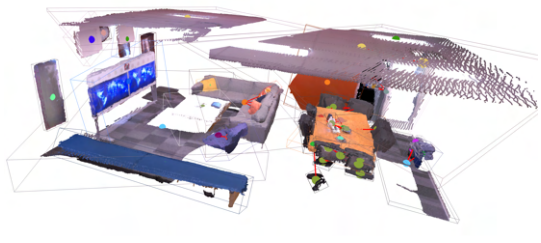
(c) MapAnything reconstruction for `office3`.

(d) MapAnything reconstruction for `office3`.

Figure 4.2: Original mesh vs MapAnything reconstruction for `office3`.



(a) CG



(b) CG-RGB

Figure 4.3: Scene graphs for `office3` obtained with the original ConceptGraphs pipeline (left) and the RGB-only pipeline (right).

degradation in the overall quality of the resulting scene graph, as measured by node-level F1-score.

Precision–recall trade-off Looking beyond F1-score, the results reveal a characteristic precision–recall trade-off driven by depth estimation errors, that can be

seen in Figure 4.4. CG consistently achieves higher precision across all scenes, while CG-RGB delivers higher recall in six out of seven scenes (except `office0`). On average, CG-RGB shows a 10.3% precision drop (0.686 to 0.615) but an 8.7% recall gain (0.401 to 0.436) compared to CG.

This behavior stems from MapAnything's depth predictions introducing small geometric inconsistencies across views. In the original CG pipeline, ground-truth depth ensures perfect multi-view consistency, so ConceptGraphs can reliably merge multiple observations of the same object into a single node based on spatial proximity of 3D back-projected masks. With MapAnything's approximate depths, these back-projections can be slightly misaligned, causing ConceptGraphs' merging criterion to fail more often and producing more fragmented predictions (45 nodes per scene vs 37 for CG, a 22% increase).

These fragments have two opposing effects: first, they increase false positives: spatially close fragments from the same object are not merged and counted as separate detections, lowering precision; second, they create a beneficial "lucky noise" effect that boosts recall. With perfect ground-truth depths, ConceptGraphs sometimes performs overly aggressive merging: two geometrically close but actually distinct objects can appear visually similar enough to pass both the spatial overlap and visual similarity thresholds, getting incorrectly fused into a single node. MapAnything's depth errors introduce small geometric inconsistencies that *prevent* this over-merging in some cases, allowing both objects to be detected separately. This increased detection coverage explains CG-RGB's recall advantage despite its lower precision.

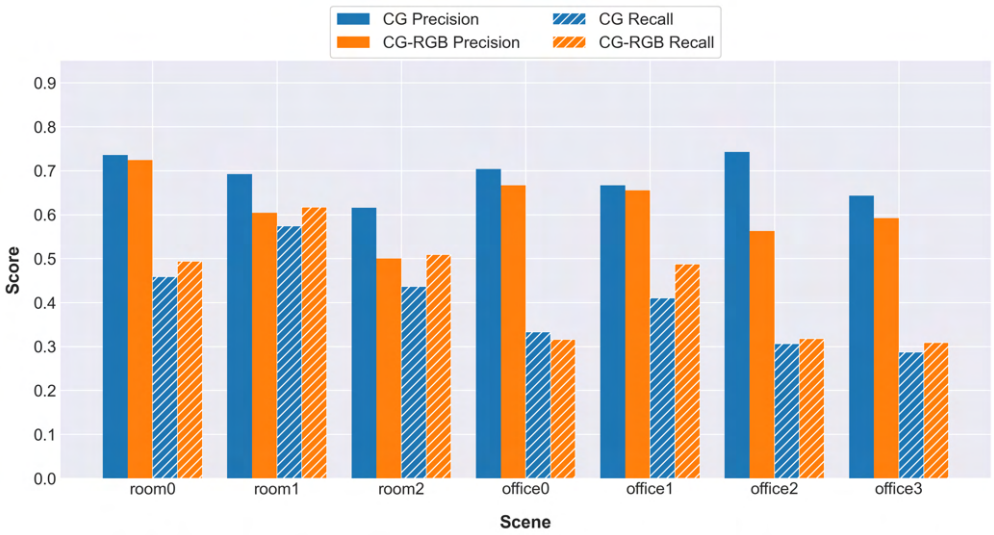
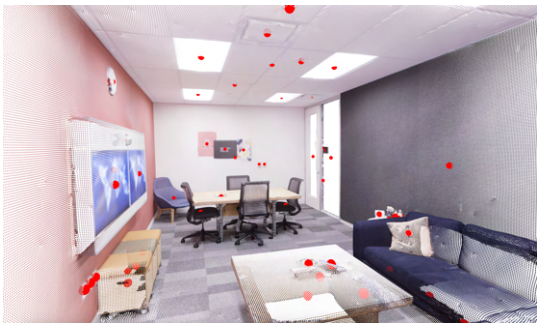
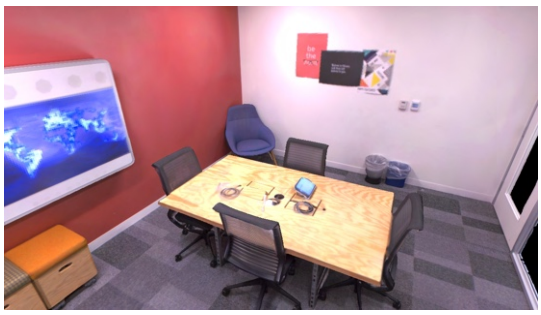


Figure 4.4: Precision–recall trade-off between CG and CG-RGB pipelines.

The relatively low recall observed in some scenes (notably `office0`, `office2`, and `office3`) can be attributed to limitations of the underlying data rather than to MapAnything. First, the pre-defined Replica trajectories do not always provide sufficient coverage of the ceiling area, so objects mounted on or close to the ceiling (e.g., lamps, ceiling fixtures) may never be visible in the input images and are therefore systematically missed by both pipelines. As can be seen in Figure 4.5a, the number of ceiling-mounted objects is significant. Second, several small objects are only weakly visible or heavily aliased in the RGB frames (Figure 4.5b), which leads to low-confidence mask predictions that are filtered out by ConceptGraphs' confidence thresholds. Together, these factors reduce the number of ground-truth instances that are even theoretically detectable, explaining why both CG and CG-RGB exhibit low recall in these scenes and why the gap between them remains relatively small.



(a) Point cloud from `office3` showing numerous ceiling objects (red points) absent from the Replica dataset trajectory.



(b) Small objects on the table not clearly recognizable in Replica RGB frames. Example from `office2`.

Figure 4.5: Two typical failure modes explaining low recall in `office0`, `office2`, and `office3`.

Effect of post-processing The post-processing step, which merges semantically and spatially similar duplicate nodes, partially recovers the precision lost due to fragmentation, as on average post-processing improves precision by 2.4 percentage points (from 0.615 to 0.639) without affecting recall, confirming that the removed nodes were indeed duplicates of already-detected objects rather than unique detections. The most impressive precision improvement is observed in `office3`, where post-processing increases precision by 4.5 percentage points (from 0.562 to 0.587) while maintaining the same recall of 0.318, resulting in a 6.2% increase in F1-score (from 0.406 to 0.412), which suggests that post-processing is particularly effective in scenes where MapAnything's depth predictions lead to more fragmented reconstructions, as it can successfully merge these fragments into coherent object

nodes.

MapAnything scale prediction The ICP-estimated scale factors provide insight into the quality of MapAnything's metric scale prediction, as across the seven scenes the estimated scales range from 0.943 (`room2`) to 1.113 (`office1`), with a mean of 1.017 and a standard deviation of 0.053, indicating that MapAnything's global scale prediction is generally accurate, with most scenes falling within ±6% of the ground-truth scale. The fact that the mean scale is very close to 1.0 further confirms that there is no systematic over- or under-estimation of metric scale across scenes, and interestingly, applying ICP with scale estimation does not improve average F1-score compared to post-processing alone; in fact, it slightly decreases it (from 0.508 to 0.502). This can be explained by noting that the "Original" evaluation variant already aligns the predicted scene graph, and the scale discrepancy is small enough that applying an additional rigid+scale transformation based on only a few object centers can occasionally introduce noise rather than improve alignment.

Summary The RGB-only pipeline produces 3D scene graphs of comparable quality to the ground-truth-based ConceptGraphs pipeline, with an average F1-score difference of less than 2% in favor of the RGB-only variant after post-processing. The main effect of replacing ground-truth depth and pose with MapAnything predictions is a precision–recall rebalancing: more objects are detected (higher recall), at the cost of more duplicate predictions (lower precision), which post-processing can partially mitigate. MapAnything's metric scale predictions are accurate to within approximately 5% on average, and explicit ICP-based scale correction offers no consistent benefit when the first-view pose is known. These results validate the RGB-only pipeline as a viable alternative to sensor-based approaches for 3D scene graph generation, enabling deployment in scenarios where depth sensors are unavailable or impractical.

4.2 Active Exploration Evaluation

4.2.1 Motivation and Objectives

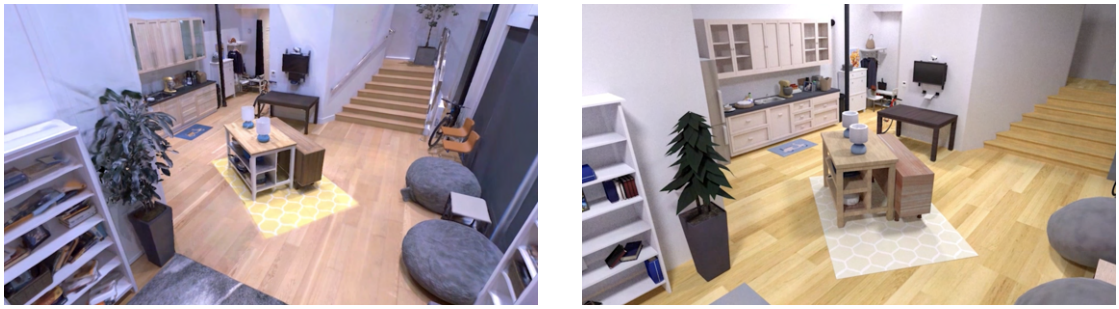
The second set of experiments evaluates the active exploration strategies explained in Section 3.4 for efficient incremental construction of 3D scene graphs. While the previous experiments validated static scene graph generation from a fixed set of observations, these experiments address the dynamic problem of viewpoint selection: given a partial scene graph constructed from previous observations, which viewpoint should the robot visit next to maximize scene understanding?

Specifically, these experiments aim to:

- Compare geometric density-based exploration (Surface Edge Explorer (SEE), Section3.4.2) against semantic-driven exploration (Active Semantic Perception (ASP), Section 3.4.4) for scene graph construction tasks
- Quantify the effect of incorporating external camera views at the initialization stage
- Evaluate scene graph completeness and accuracy as a function of exploration steps across diverse indoor environments
- Assess the robustness of exploration strategies to different starting positions

4.2.2 Dataset Selection: ReplicaCAD

For active exploration experiments, the ReplicaCAD dataset [7] was selected instead of the original Replica dataset. The ReplicaCAD dataset is an artist recreation of the scanned "FRL apartment" variations from the Replica dataset [3] (Figure 4.6), designed specifically for interactive simulation in Habitat [99].



(a) Replica Scan. **(b)** ReplicaCAD.

Figure 4.6: Comparison between a scanned Replica scene (a) and its corresponding ReplicaCAD recreation (b). The ReplicaCAD scenes are artist-created 3D environments designed for interactive simulation, while the original Replica scenes are based on real-world scans. Images adapted from the original ReplicaCAD demo video[3].

ReplicaCAD includes a single empty scene and a set of six detailed re-creations of the scanned "FRL apartment", where both large furniture and many small everyday objects are placed and the environments are configured for dynamic simulation in Habitat-Sim. In addition, the dataset provides 84 artist-authored rearrangements that focus mainly on large furniture: these are organized as five

[3]Demo video: `https://aihabitat.org/datasets/replica_cad/`

macro layout variations (reflecting how different tenants might arrange the same apartment), each further refined into 20 micro variations where only a few furniture pieces are moved or swapped.

For the experiments in this thesis, only the six FRL apartment re-creations (`apt_0`–`apt_5`) were used, since the goal is to study scene graph construction in object-rich environments; the remaining 84 scenes mainly contain large furniture items and lack the density of smaller objects required for a challenging semantic evaluation.

ReplicaCAD's apartment layouts are sized appropriately for room-level scene graph construction, matching the scale assumptions of ConceptGraphs [2]. Each apartment contains multiple areas with realistic furniture arrangements and significant object clutter, offering challenging but still manageable scenarios for active exploration and 3D scene graph generation.

4.2.3 Simulation Environment: Habitat-Sim

Simulator Selection and Configuration

All active exploration experiments were conducted in Habitat-Sim [7, 99, 100], a high-performance 3D simulator for embodied AI. Habitat-Sim was chosen because it provides:

- **Native ReplicaCAD integration:** ReplicaCAD scenes can be loaded directly from their scene dataset configuration files, together with the precomputed navigation meshes and semantic metadata, providing an out-of-the-box setup for realistic indoor navigation experiments.
- **Configurable agents and sensors:** Agents, articulated robots, and multiple RGB cameras (onboard and external) can be instantiated and controlled, which is essential for implementing custom exploration loops and evaluating different active perception strategies within the same virtual environment.

Robot and Visualization Setup

A Fetch mobile manipulator robot from Habitat-Lab was instantiated in each ReplicaCAD apartment. In this work, the exact choice of robot is not critical: the robot primarily serves as a moving camera platform to acquire egocentric RGB images and as a convenient handle for visualizing the exploration process. Manipulation capabilities are not used; the arm is kept in a fixed "side" configuration to minimize self-occlusions.

For monitoring and qualitative analysis, an additional fixed overhead camera was configured to provide a bird's-eye view of the environment and the robot trajectory.

Figure 4.7 shows an example of this setup for `apt_0`, where the left panel (a) shows the egocentric view from the robot's onboard camera, and the right panel (b) shows the top-down view from the external camera (from which the robot is visible).

(a) Egocentric view from the robot's onboard camera.

(b) Bird's-eye view from the fixed external camera.

Figure 4.7: Visualization of the two camera views used for monitoring the active exploration experiments in ReplicaCAD (`apt_0`).

Camera Configurations

Two camera configurations were evaluated for each exploration strategy:

- **Onboard Only:** Only the robot's egocentric camera observations are used. At step 0 (initialization), a single RGB image from the robot's starting position is captured and processed.
- **Onboard + External:** In addition to the robot's egocentric camera, observations from external cameras are included at initialization (step 0). A single fixed external camera was positioned to provide a bird's-eye view of a significant portion of the environment. This configuration emulates scenarios where pre-existing surveillance cameras or ceiling-mounted cameras can bootstrap the scene graph before the robot starts exploring.

Navigation Implementation

At each exploration step, after the next-best-view (NBV) is selected by the exploration strategy, the robot navigates to the target viewpoint using Habitat's built-in navigation functions:

- The target position (x, y, z) and orientation (yaw angle and pitch angle) are provided as a goal pose

- If the target position lies outside navigable regions, the navigator automatically snaps it to the closest reachable position on the navigation mesh
- Habitat's navigation mesh pathfinding computes a collision-free path
- The robot moves along this path at a constant velocity
- Upon reaching the goal pose, a new RGB observation is captured

4.2.4 Experimental Protocol

To validate these strategies, the complete active exploration system described in Section 3.1 was used: at each exploration step, the robot acquires an RGB image (1), which is processed by the RGB-only 3D scene graph pipeline to update the point cloud and scene graph (2), the active exploration module selects the next-best-view (3), and the navigation module drives the robot to this viewpoint (4), repeating until the stopping condition is met.

Experimental Design

A full factorial experimental design was employed with the following factors:

- **Scene:** 6 levels (`apt_0` through `apt_5`)
- **Starting Position:** 10 levels (uniformly sampled from navigable regions per scene)
- **Camera Configuration:** 2 levels (onboard only, onboard + external)
- **Exploration Strategy:** 2 levels (SEE, ASP)

This yields a total of $6 \times 10 \times 2 \times 2 = 240$ experimental trials: 120 trials for SEE and 120 trials for ASP.

Execution Procedure

For each experimental trial, the following procedure was executed:

1. **Initialization (Step 0):**
   - Load the ReplicaCAD scene in Habitat simulator
   - Spawn the Fetch robot at the assigned starting position
   - Capture the initial onboard RGB observation
   - If external camera configuration: capture external camera observation

- Process all initial observations through the RGB-only pipeline (MapAnything + ConceptGraphs) to construct the initial scene graph G_0 and point cloud P_0

2. **Exploration Loop (Steps 1–30):**
   - Provide the current representation to the exploration strategy, i.e. the reconstructed point cloud P_t when using SEE and the 3D scene graph G_t when using ASP
   - Exploration strategy selects next-best-view (NBV) pose $(x, y, z, \text{yaw}, \text{pitch})$ based on the current representation and the strategy's selection criteria
   - Navigate robot to NBV pose using Habitat navigation
   - Capture RGB observation from NBV
   - Update scene graph and point cloud: $G_{t+1}, P_{t+1} \leftarrow \text{RGBPipeline}(G_t, P_t, I_{t+1})$
   - Increment step counter: $t \leftarrow t + 1$
   - If $t = 30$: terminate exploration loop
3. **Data Logging:** Per-step 3D scene graph (objects and relations) and point cloud are logged for subsequent evaluation.

The maximum number of exploration steps was set to 30 to balance between allowing sufficient exploration for meaningful scene graph construction and keeping the total experiment duration manageable.

ConceptGraphs Hyperparameters

ConceptGraphs hyperparameters were re-tuned for this experiment compared to the previous ones. Specifically, the merge thresholds were lowered to `merge_overlap_thresh` = 0.2 and `merge_visual_sim_thresh` = 0.2 (from 0.7 in the Replica experiments). This adjustment was necessary because duplicate detections were far more prevalent in the ReplicaCAD scenes. Unlike Replica dataset, which consists of photorealistic scans with natural lighting and shadows, ReplicaCAD is an artist-authored CAD environment with simpler materials and no baked lighting effects. This lack of photorealistic cues makes it harder for MapAnything to estimate accurate depths, resulting in more fragmented and spatially inconsistent 3D back-projections across views. By lowering the merge thresholds, ConceptGraphs applies more aggressive merging during scene graph construction, absorbing most of these spurious fragments into existing object nodes and substantially reducing duplication.

Active Semantic Perception Hyperparameters

ASP was configured to generate three scene graph completions per planning step using the LLM `gemini-2.5-pro`.

Metric Computation and Aggregation

At each step t of each trial, scene graph node quality was evaluated using the validation framework with the **Post-Processing** variant (as defined in Section 3.3), computing precision, recall, F1-score, and total node count against the ground-truth scene graph for the corresponding ReplicaCAD scene.

For each combination of (scene, camera configuration, exploration strategy), metrics were aggregated across the 10 starting positions by computing:

- **Mean:** Average metric value across 10 trials at each step
- **Standard Deviation:** Standard deviation across 10 trials at each step

This aggregation yields learning curves showing how scene graph quality evolves as exploration progresses, with error bars indicating variability due to starting position.

4.2.5 Results

Since all the rsults across the different scenes yeld similar trends, here only the learning curves for one representative scene (`apt_3`) are presented, while the complete set of results for all scenes and configurations is provided in the Appendix A.

Geometric vs Semantic-based exploration

Figure 4.8 shows the evolution of precision, recall, and F1-score across exploration steps for `apt_3` with onboard camera only, while Figure 4.9 shows the corresponding evolution of the number of predicted nodes.

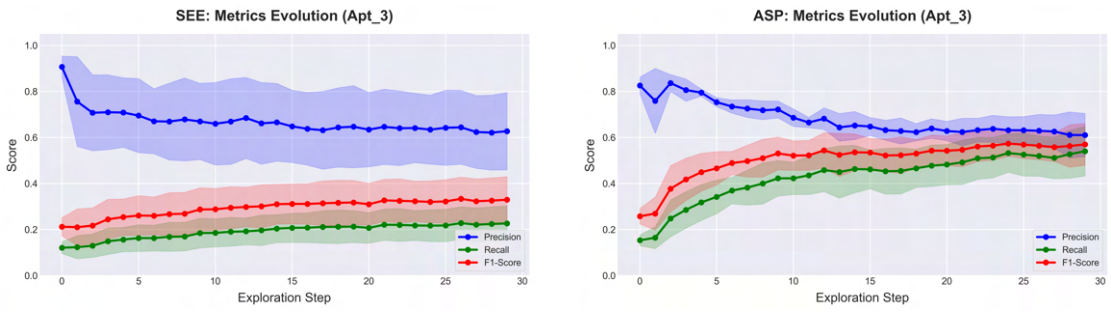


Figure 4.8: Evolution of precision, recall, and F1-score for `apt_3` with onboard camera only. Left: SEE (geometric-based exploration). Right: ASP (semantic-based exploration).

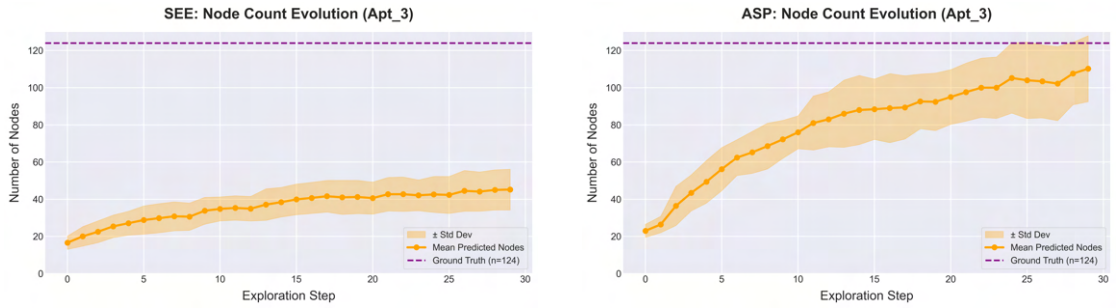


Figure 4.9: Evolution of number of predicted nodes for `apt_3` with onboard camera only. Left: SEE (geometric-based exploration). Right: ASP (semantic-based exploration).

Contribution of external cameras

Figure 4.10 shows the evolution of precision, recall, and F1-score for `apt_3` with external cameras, while Figure 4.11 shows the corresponding evolution of the number of predicted nodes.

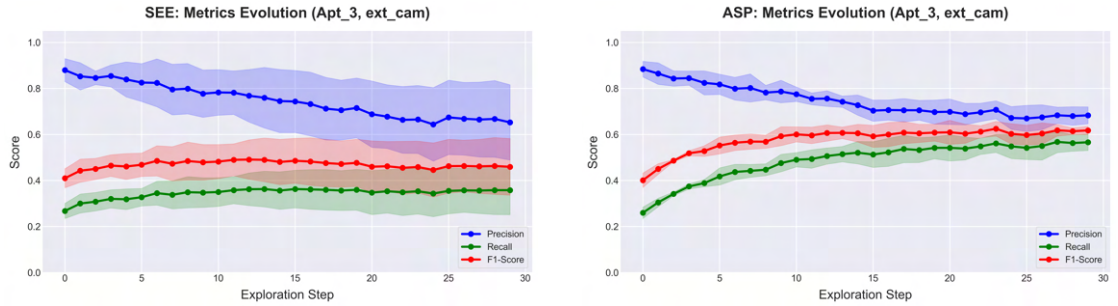


Figure 4.10: Evolution of precision, recall, and F1-score for `apt_3` with external cameras. Left: SEE (geometric-based exploration). Right: ASP (semantic-based exploration).

4.2.6 Discussion

ASP Achieves Significantly Higher Scene Graph Completeness

The most striking result is the large performance gap between semantic-driven exploration (ASP) and geometric frontier-based exploration (SEE). Comparing the onboard-only configurations in Figures 4.8 and 4.9, ASP detects approximately 110 object nodes by step 30, approaching the ground-truth count of 124, whereas SEE reaches only about 45 nodes (less than half). This translates directly into recall:

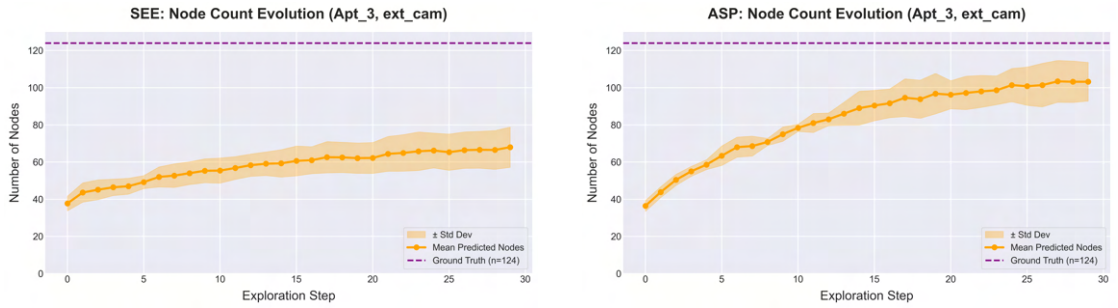


Figure 4.11: Evolution of number of predicted nodes for `apt_3` with external cameras. Left: SEE (geometric-based exploration). Right: ASP (semantic-based exploration).

ASP achieves a recall of approximately 0.54 at step 30, more than double SEE's 0.22. In terms of F1-score, ASP reaches approximately 0.57, compared to 0.33 for SEE.

The reason for this gap lies in the fundamental difference between the two exploration strategies. SEE selects next-best-views by identifying geometric frontiers, i.e. regions at the boundary of the reconstructed point cloud where surface coverage is incomplete. While this effectively expands spatial coverage, it does not prioritize areas that are semantically rich or where the scene graph is incomplete. As a result, SEE may spend several steps mapping geometrically unexplored but semantically uninteresting regions (e.g., empty walls or floors), resulting in slow growth of both node count and recall.

ASP, on the other hand, leverages the current scene graph to reason about which areas are likely to contain unobserved objects. By using an LLM to hypothesize plausible scene completions and computing information gain over these hypotheses, ASP directs the robot toward semantically informative viewpoints (for example, prioritizing a partially observed kitchen counter over an empty corridor). This semantic awareness leads to a much steeper growth in node count and recall, particularly in the first 10–15 steps where the scene graph is still sparse and many objects remain undiscovered.

Precision follows a similar declining trend for both strategies, starting high ($\sim$0.83–0.91) and gradually decreasing to approximately 0.61–0.63 by step 30. This behavior is expected: with few initial observations, most predicted nodes are correct, yielding high precision. As more views are added, MapAnything's depth errors produce fragmented reconstructions that introduce duplicate detections, progressively lowering precision. Notably, both strategies converge to a similar final precision, indicating that the precision decline is driven primarily by the RGB-only pipeline's reconstruction characteristics rather than by the exploration strategy itself. c

External Cameras Provide a Strong Initialization

Comparing Figures 4.8–4.9 (onboard only) with Figures 4.10–4.11 (onboard + external), the benefit of incorporating a single external camera at initialization is evident. For SEE, the initial node count at step 0 rises from approximately 16 to 37 (+130%), and initial recall jumps from 0.12 to 0.27. For ASP, the initial node count increases from approximately 23 to 36 (+57%), and initial recall from 0.15 to 0.26.

This strong initialization means that fewer exploration steps are needed to reach the same level of scene graph quality. For instance, the F1-score that SEE with external cameras achieves at step 0 ($\sim$0.41) is only reached by SEE without external cameras around step 25. Similarly, ASP with external cameras starts at an F1 of approximately 0.40 and reaches 0.62 by step 30, compared to 0.57 for ASP without external cameras—a consistent advantage maintained throughout the exploration.

It is worth emphasizing that this integration of external cameras is straightforward precisely because the pipeline is RGB-only: a single overhead image from a fixed camera is processed by MapAnything in exactly the same way as the robot's onboard frames, requiring no additional calibration, depth sensors, or pipeline modifications.

Robustness to Starting Position

The shaded standard deviation bands in the figures reveal differences in robustness across strategies and configurations. SEE exhibits noticeably higher variance, particularly in precision. This sensitivity arises because SEE's frontier-based decisions are heavily influenced by the initial point cloud geometry: different starting positions produce different frontier distributions, leading to divergent exploration trajectories.

ASP shows tighter confidence bands across all metrics, indicating greater robustness to initialization. This is because ASP's semantic reasoning provides a more stable exploration objective: regardless of where the robot starts, the LLM consistently identifies similar high-priority semantic targets (e.g., tables, shelves, counters with expected objects), leading to more consistent exploration behavior across different starting positions.

The addition of external cameras further reduces variance for both strategies, as the overhead view provides a consistent global context that partially decouples the initial scene representation from the robot's specific starting location.

Summary

These results demonstrate that semantic-driven active exploration significantly outperforms purely geometric approaches for 3D scene graph construction. By

leveraging the structured semantic information stored in the evolving scene graph, ASP can direct the robot toward the most informative viewpoints, achieving scene graph quality comparable to what ConceptGraphs obtains from hundreds of images in just 30 exploration steps. Furthermore, the combination of ASP with a single external camera provides the best overall performance, suggesting that even minimal infrastructure support can substantially accelerate scene graph construction in practical deployments.

4.3 External Camera Integration Study

4.3.1 Motivation and Objectives

The third set of experiments investigates the potential of using only fixed external cameras for 3D scene graph construction, without relying on any robot-driven exploration. After observing in the previous experiments that a single overhead view can significantly accelerate and improve active exploration, this section focuses on the scenario in which infrastructure cameras are the sole source of observations.

This setting is motivated by practical deployment scenarios where RGB cameras are already installed in indoor environments (e.g., surveillance systems, ceiling-mounted monitoring cameras) and no mobile robot may be available or it may be desirable to minimize robot motion. Since the proposed pipeline operates purely on RGB images, such external cameras can be leveraged directly to build 3D scene graphs with minimal additional sensing or hardware cost, providing a lightweight alternative to embodied exploration.

In these experiments, only the RGB-only 3D scene graph generation pipeline (Section 3.2) is applied to the images captured by the external cameras. No active exploration or robot navigation is involved: the scene graph is constructed in a single pass from the available camera views.

Specifically, these experiments address:

- How much scene understanding can be achieved using only external cameras with wide fields of view?
- How does scene graph quality scale with the number of external cameras (1, 2, or 3)?
- How do environmental characteristics (apartment complexity vs. simple furnished rooms) affect external-camera-only performance?

4.3.2 Dataset and Scene Characteristics

These experiments used the same ReplicaCAD dataset as the active exploration experiments (section 4.2.2), but all 90 scenes were included to evaluate a wider

range of environmental characteristics. As already discussed, these scenes can be broadly categorized into two groups based on their complexity and object density:

- **Apartments (apt_0–apt_5, 6 scenes)**: These are the same complex apartment environments used in the active exploration experiments. They contain a high density of objects (mean ≈ 123 objects per scene) including many small items such as books, kitchen utensils, and decorative objects that are challenging to detect from external cameras due to size and occlusion. Figure 4.12 shows examples of this type of scene.
- **Furnished Rooms (v3_*, 84 scenes)** The remaining 84 ReplicaCAD scenes are simpler single-room or small multi-room environments with lower object density (mean ≈ 25 objects per scene). Objects in these scenes are predominantly large furniture items (sofas, tables, beds, cabinets) that are more easily visible from overhead or wall-mounted cameras. Figure 4.13 shows examples of this type of scene.

This natural division in scene complexity enables evaluation of how environmental characteristics affect the feasibility of external-camera-only scene graph construction.

4.3.3 External Camera Configuration

For each scene, three external camera positions were manually selected to provide complementary coverage of the environment, with the goal of maximizing the number of visible objects while maintaining realistic placements (e.g., ceiling-mounted, wall-mounted, or corner-mounted). Figures 4.12 and 4.13 illustrate these placements: the top row shows the three camera viewpoints for an apartment scene (`apt_2`), while the bottom row shows the viewpoints for a furnished room scene (`v3_sc0_staging_02`). The exact camera positions were kept consistent across all trials to ensure comparability.

(a) External Camera 1 **(b)** External Camera 2 **(c)** External Camera 3

Figure 4.12: External camera placements for an apartment scene.

(a) External Camera 1 **(b)** External Camera 2 **(c)** External Camera 3

Figure 4.13: External camera placements for a furnished room scene.

4.3.4 Experimental Protocol

For each scene (total of 90), three experimental trials were conducted with different numbers of external cameras, from 1 to 3, in the positions showed in the previous section. This yields $90 \times 3 = 270$ total experiments.

For each scene and camera count configuration:

1. Load scene in Habitat simulator
2. Position the specified number of external cameras
3. Capture RGB images from all external cameras
4. Process images through the RGB-only pipeline (MapAnything predicts depth, pose, and scale from the camera images and ConceptGraphs constructs the 3D scene graph)
5. Evaluate the predicted scene graph against ground truth using the validation framework (Section 3.3) under the three evaluation variants described in Section 3.3.3: first the raw predicted scene graph (CG-RGB), then after duplicate removal (CG-RGB + Post), and finally after ICP registration with scale estimation (CG-RGB + ICP)

4.3.5 Results

Table 4.2 summarizes the results of the external camera integration experiments. *CG-RGB* denotes the RGB-only pipeline, while *CG-RGB + Post* and *CG-RGB + ICP* additionally include post-processing and ICP-based alignment, respectively. *#Cameras* indicates the number of views integrated for each scene. *GT* and *Pred* denote the average number of ground-truth and predicted object nodes, respectively, while *Post* reports the average number of nodes removed by post-processing. *Scale* is the average scale factor estimated by MapAnything and used in the ICP variant.

Precision, *Recall*, and *F1-score* are computed through node-level matching between predicted and ground-truth scene graph nodes. Results are averaged across 6 apartment scenes and 84 furnished room scenes.

Table 4.2: Results of external camera integration experiments for 3D scene graph construction.

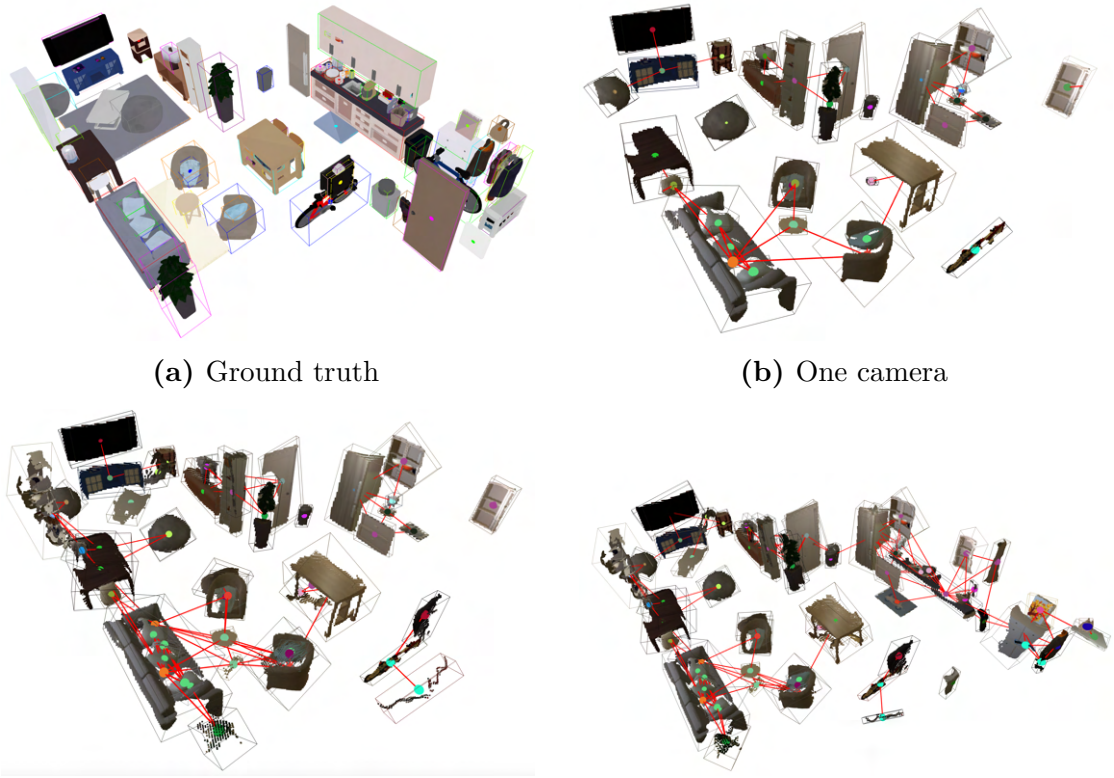
Scene type	Exp	#Cameras	GT	Pred	Post	Scale	Precision	Recall	F1-score
Apartments	CG-RGB	1	123	34	0	-	0.636	0.176	0.276
		2	123	45	0	-	0.643	0.235	0.344
		3	123	65	0	-	0.543	0.289	0.377
	CG-RGB + Post	1	123	32	3	-	0.662	0.170	0.270
		2	123	39	7	-	0.713	0.224	0.340
		3	123	53	13	-	0.619	0.268	0.373
	CG-RGB + ICP	1	123	32	3	0.866	0.770	0.198	0.315
		2	123	39	7	0.883	0.741	0.232	0.353
		3	123	53	13	0.862	0.698	0.301	0.421
Furnished Rooms	CG-RGB	1	25	19	0	-	0.531	0.397	0.452
		2	25	26	0	-	0.470	0.479	0.472
		3	25	34	0	-	0.405	0.551	0.465
	CG-RGB + Post	1	25	18	1	-	0.562	0.396	0.462
		2	25	23	3	-	0.529	0.477	0.500
		3	25	29	6	-	0.479	0.548	0.509
	CG-RGB + ICP	1	25	18	1	0.947	0.566	0.398	0.465
		2	25	23	3	0.930	0.540	0.487	0.510
		3	25	29	6	0.926	0.486	0.556	0.516

In addition to quantitative results, Figure 4.14 shows a qualitative example of a scene graph generated from three external cameras in an apartment scene, while Figure 4.15 shows the result for a furnished room scene.

4.3.6 Discussion

Feasibility of External-Camera-Only Scene Understanding

The first question to address is whether fixed external cameras alone can provide meaningful scene understanding. The results show that precision is consistently high across all configurations (reaching 0.698 in apartments and 0.486 in furnished rooms with three cameras and ICP alignment) indicating that the objects detected by the pipeline are largely correct. However, recall remains the limiting factor: with at most three views, only a fraction of the scene can be observed, and many objects are inevitably missed due to occlusion, distance, or limited angular coverage. In apartments, the best recall achieved is 0.301 (three cameras, ICP), compared to approximately 0.6 obtained with 30 steps of active exploration in the previous experiments (Section 4.2). This gap is entirely expected: three static views cannot



(a) Ground truth

(b) One camera

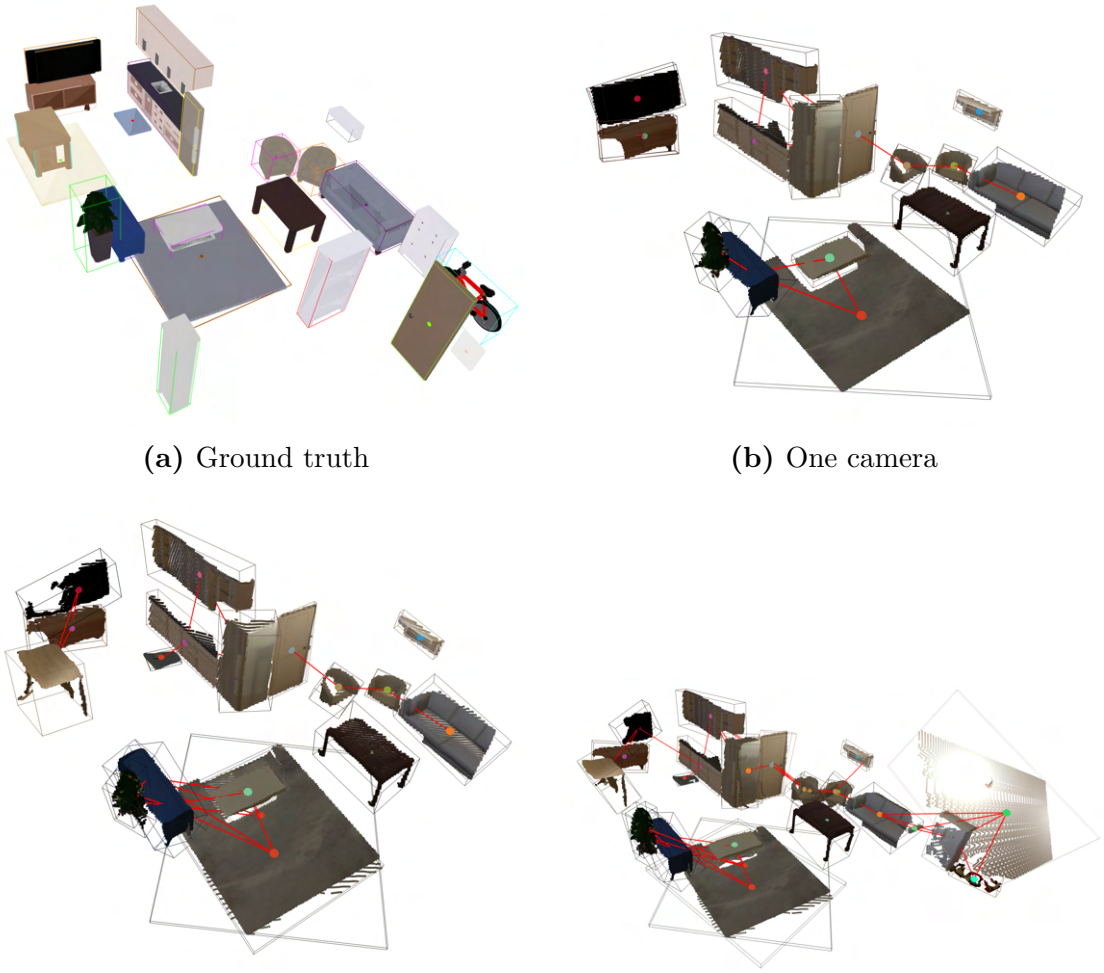
(c) Two cameras

(d) Three cameras

Figure 4.14: Qualitative results for apartment scene `apt_2` with increasing number of cameras.

compete with 30 strategically selected viewpoints that observe the scene from close range and multiple angles.

Nevertheless, these results demonstrate that even a small number of external cameras can construct a scene graph that captures the dominant structure of the environment. In furnished rooms, where objects are large and fewer, three cameras recover more than half of the ground-truth objects (recall of 0.556). In apartments, despite the lower recall, the detected objects tend to be the most prominent and spatially significant one (tables, sofas, shelves) providing a solid structural skeleton of the scene. Crucially, this initial scene graph is obtained at virtually zero cost: no robot motion, no depth sensors, and no active planning are required. In deployments where surveillance or monitoring cameras are already installed, the RGB-only pipeline can directly leverage these existing views to bootstrap a 3D scene graph, which can then be refined through active exploration if a mobile robot becomes available.



(a) Ground truth

(b) One camera

(c) Two cameras

(d) Three cameras

Figure 4.15: Qualitative results for furnished room scene `v3_sc0_staging_00` with increasing number of cameras.

Effect of Camera Count on Scene Graph Quality

Across both scene categories, adding more external cameras consistently increases recall: more viewpoints cover a larger portion of the environment, making previously occluded or out-of-view objects visible. In apartments, recall grows from 0.198 (1 camera) to 0.301 (3 cameras) with ICP alignment; in furnished rooms, from 0.398 to 0.556. This confirms the intuitive expectation that wider visual coverage leads to more complete scene graphs.

However, precision follows the opposite trend, decreasing as the number of cameras increases. In apartments, precision drops from 0.770 to 0.698 (CG-RGB +

ICP); in furnished rooms, from 0.566 to 0.486. This occurs because each additional camera introduces new object detections, but also produces duplicate predictions of objects already observed from other viewpoints (due to the depth estimation problem, already discussed in previous sections). The post-processing column confirms this: the number of removed duplicates grows from 3 to 13 in apartments and from 1 to 6 in furnished rooms as cameras are added.

The net effect on F1-score is positive in most configurations, since the recall gain outweighs the precision loss, and the largest F1 improvement typically occurs when moving from one to two cameras; for example, apartment F1 (CG-RGB + Post) jumps from 0.270 to 0.340 (+0.070). The marginal gain from the third camera is smaller, and in furnished rooms F1 nearly plateaus between two and three cameras (e.g., CG-RGB + ICP: 0.510 → 0.516, +0.006), indicating diminishing returns. This diminishing return in V3 scenes occurs because two well-placed cameras already cover most of the relatively simple environment with few, large objects, while the third camera mainly adds coverage of regions that contain little additional semantic content.

Apartments vs. Furnished Rooms

Furnished rooms achieve substantially higher F1-scores than apartments across all configurations, and with three cameras and ICP alignment they reach an F1 of 0.516 compared to 0.421 for apartments; this gap is primarily driven by recall, as furnished rooms achieve 0.556 recall versus only 0.301 for apartments.

Two factors explain this difference: first, apartments contain on average 123 ground-truth objects versus 25 in furnished rooms, many of which are small items (books, utensils, decorative objects—on average 22 objects below $1\,\mathrm{m}^3$ per apartment, versus essentially none in furnished rooms), and these small objects are difficult to resolve from distant external viewpoints because they occupy few pixels and produce weak segmentation masks. Second, the higher object density in apartments leads to more cluttered scenes where overlapping objects are harder to separate, further reducing detection accuracy.

In contrast, furnished rooms are dominated by large, distinct objects (sofas, tables, beds) that are easily visible from elevated camera positions, and the combination of fewer objects and larger average object size explains why external cameras alone can recover more than half of the ground-truth scene graph in furnished rooms.

Scale Estimation and ICP Alignment

The ICP-estimated scale factors reveal that MapAnything consistently overestimates the scene scale (i.e., the predicted 3D reconstruction is larger than ground truth), with scale factors below 1.0 in all configurations. Apartments exhibit a mean

scale of 0.870 (a 13% deviation from the ideal value of 1.0), while furnished rooms show a more accurate mean scale of 0.934 (6.6% deviation). This suggests that MapAnything's scale prediction is more reliable for simpler scenes with fewer, larger objects that provide stronger geometric cues for metric estimation.

Applying ICP with scale correction consistently improves F1-score, with the largest gains observed in apartments; for example, with three cameras, apartment F1 improves from 0.373 (Post only) to 0.421 (Post + ICP), a gain of +0.048. In furnished rooms the improvement is more modest (0.509 → 0.516, +0.007), reflecting the already more accurate scale prediction, while ICP primarily improves precision by correcting spatial misalignment between predicted and ground-truth object positions, allowing more predicted nodes to fall within the matching distance threshold.

Post-Processing Effectiveness

Duplicate removal via post-processing improves precision in all configurations without significantly affecting recall. The proportion of removed nodes increases with camera count, from approximately 9% (1 camera) to 20% (3 cameras) in apartments, confirming that multi-view redundancy is a primary source of duplicate detections; in furnished rooms the removal rate is lower (5% to 18%), consistent with the simpler scene structure producing fewer ambiguous overlapping detections.

Summary

These experiments demonstrate that fixed external RGB cameras, combined with the proposed RGB-only pipeline, can produce meaningful 3D scene graphs. While three static viewpoints are not sufficient to construct a complete scene graph and recall therefore remains limited, the resulting graphs capture the most prominent objects in the environment, providing a solid structural skeleton of the scene.

This makes external cameras particularly well suited for two practical roles. First, they can serve as a strong initialization for active exploration: as shown in the previous experiments (Section 4.2), bootstrapping the scene graph with a single overhead view already accelerates and improves robot-driven exploration significantly. Second, external cameras can be used to periodically update the scene graph over time—for instance, detecting when furniture is moved or new objects appear—without requiring any robot deployment.

The key advantage of this approach lies in its minimal requirements: since the pipeline operates on RGB images alone, any existing surveillance or monitoring camera can be leveraged directly, with no depth sensors, calibration procedures, or hardware modifications, making external cameras a simple and low-cost complement to robot exploration in real-world scenarios.

Chapter 5

Discussion

The experimental evaluation presented in Chapter 4 provided detailed analyses of each individual study: the RGB-only pipeline validation, the active exploration comparison, and the external camera integration. This chapter steps back from the per-experiment perspective to discuss the broader insights that emerge when considering the three studies together, the limitations of the current work, and the open challenges that remain.

5.1 The Role of Monocular Depth and Pose Estimation

The design choice at the core of this thesis is the replacement of ground-truth or sensor-based depth and camera pose with learned monocular predictions from MapAnything [1]. All three experiments ultimately depend on the quality of this replacement: the RGB-only pipeline validation measures its direct effect on scene graph accuracy, the active exploration experiments build incrementally on top of it at every step, and the external camera study pushes it to an extreme setting with few wide-angle overhead views. In this sense, monocular depth and pose estimation is the foundation of the entire system: if it fails, every downstream component inherits and amplifies those errors.

The experiments show that, in the settings tested, this foundation holds. The RGB-only pipeline achieved an average F1-score of 0.5, nearly identical to the 0.499 obtained by ConceptGraphs with ground-truth depth and pose. The main observable effect was not an overall quality degradation but a rebalancing of the precision–recall trade-off: estimated depths introduced small geometric inconsistencies across views, causing ConceptGraphs' merging criterion to fail more frequently (producing 22% more predicted nodes on average), which lowered precision while simultaneously preventing over-merging and thereby increasing recall. This pattern

(more fragments, lower precision, higher recall) recurred consistently in the active exploration experiments, where precision declined from approximately 0.83–0.91 at initialization to 0.61–0.63 after 30 steps regardless of the exploration strategy used, confirming that the precision decay is an intrinsic characteristic of the RGB-only reconstruction rather than a consequence of exploration choices. In the external camera experiments, the same mechanism manifested as an increasing number of duplicates removed by post-processing as more viewpoints were added.

Beyond depth accuracy, an important practical consideration emerged: MapAnything requires sufficient visual overlap between input frames to establish consistent geometric relationships across the sequence. In the Replica validation, the predefined camera trajectories inherently provided dense overlap between consecutive frames, even after subsampling to 154 frames. In the active exploration setting, however, each new observation is captured from a next-best-view that may be spatially distant from previous viewpoints, and in the external camera experiments, a small number of fixed cameras with limited overlap was used. The fact that the pipeline still produced usable scene graphs in these challenging overlap conditions is encouraging, but it also signals that real-world deployments must consider camera placement and trajectory design to ensure that MapAnything receives inputs with adequate inter-frame overlap (particularly for external camera installations, where cameras are fixed and cannot be repositioned after deployment).

The nature of the depth estimation challenge also varied across experimental domains. MapAnything achieved scale predictions within approximately 5% of ground truth on the photorealistic Replica scenes (mean ICP scale factor 1.017), whereas the artist-authored ReplicaCAD environments, with simpler materials and no baked lighting, required substantially more aggressive merge thresholds in the ConceptGraphs multi-view association and exhibited a systematic scale discrepancy (mean scale factor of 0.870 for apartments). This domain sensitivity implies that depth estimation quality is not a fixed property of the model but instead depends on the visual characteristics of the input, with direct implications for deployment in diverse real-world environments.

Crucially, the motivation for accepting these trade-offs is not merely academic. The entire rationale for replacing depth sensors with learned estimates is to lower the hardware requirements of the pipeline, removing the need for specialized depth sensors (RGB-D cameras, LiDAR) and enabling scene graph construction from standard RGB cameras alone; this capability is what makes the external camera integration possible in the first place, since a ceiling-mounted surveillance camera or a smartphone can serve as input to the pipeline without any hardware modification. The experiments demonstrate that this relaxation of sensor requirements comes at a manageable cost in scene graph quality, while opening the door to deployment scenarios—such as leveraging existing building infrastructure—that would be infeasible with a depth-sensor-dependent pipeline.

5.2 Complementarity of Exploration and External Cameras

The active exploration and external camera experiments, taken together, reveal a natural complementarity between robot-driven perception and infrastructure-based observation. External cameras alone achieved an F1-score of 0.421 in apartments (three cameras, with ICP alignment), capturing the dominant spatial structure but missing many small objects due to distance and occlusion. Active exploration with ASP reached an F1-score of approximately 0.57 after 30 steps using only the onboard camera—substantially higher, but starting from a very sparse initial representation. When a single external camera was incorporated at initialization, ASP's final F1-score improved to approximately 0.62, and the F1-score that SEE with external cameras achieved at step 0 (0.41) was only reached by SEE without external cameras around step 25.

This complementarity suggests a practical deployment model: external cameras provide a broad, low-cost structural "skeleton" of the scene, while active exploration fills in the details through close-range, semantically guided observation. The RGB-only design is what makes this integration seamless: a single overhead image from a fixed camera is processed by MapAnything in exactly the same way as the robot's onboard frames, requiring no additional calibration, depth sensors, or pipeline modifications. In practice, this means that environments already equipped with surveillance cameras can benefit from scene graph construction at no additional hardware cost, with a mobile robot optionally deployed to refine the representation where higher completeness is needed.

The experiments also suggest a spectrum of deployment configurations depending on the available resources and the required level of detail. At the lightest end, external cameras alone can provide coarse scene monitoring suitable for tasks such as furniture tracking or room layout verification. At the other end, the combination of external cameras with ASP-based active exploration achieved the highest overall performance and represents the recommended configuration when comprehensive scene understanding is required. Even without any external cameras, ASP-based active exploration alone remains viable, producing strong results, even if the initial exploration phase is less efficient.

5.3 The Scene Graph as Both Product and Instrument of Exploration

The active exploration experiments revealed a finding that extends beyond the specific strategies tested: the 3D scene graph is not merely a passive output of the perception pipeline, but serves a dual role as both the *product* and the *instrument*

of exploration.

In the SEE strategy, exploration is guided by the reconstructed point cloud, a purely geometric representation that encodes where surfaces have been observed but carries no semantic structure. The robot selects next-best views by identifying frontiers at the boundary of the known geometry, thereby directing exploration toward unmapped space regardless of its semantic content; as a result, SEE may spend multiple exploration steps mapping geometrically unexplored but semantically uninformative regions (such as empty walls, floors, or ceilings), leading to slow growth in object discovery (approximately 45 nodes detected after 30 steps, less than half the ground truth of 124).

ASP, by contrast, operates directly on the evolving scene graph, using structured semantic information—namely which objects have been detected, their spatial arrangement, and the rooms in which they appear—to reason about what is *likely missing* from the current representation. By querying an LLM with the current scene graph contents and asking it to hypothesize plausible scene completions, ASP effectively treats the scene graph as a semantic map of what is known and, by implication, what remains to be discovered, thereby transforming the scene graph from a passive record of past observations into an active planning tool that shapes future exploration decisions.

The quantitative results reflect this difference directly: ASP achieves approximately double the recall of SEE (0.54 vs. 0.22) and nearly double the F1-score (0.57 vs. 0.33), with the gap opening rapidly in the first 10–15 steps when the scene graph is still sparse and the potential for semantically informed exploration is greatest; ASP also exhibits substantially lower variance across starting positions, because its semantic reasoning provides a more stable exploration objective—regardless of where the robot starts, the LLM consistently identifies similar high-priority targets (tables, shelves, counters with expected objects), leading to convergent exploration trajectories.

This dual role of the 3D scene graph, both as the *final goal* and as the *guiding representation for exploration*, represents an important conceptual insight, suggesting that structured, semantically rich representations such as scene graphs are particularly well suited to active perception tasks because their structure encodes not only what has been observed but also what *should* be observed. A point cloud or occupancy map cannot express the concept "there should be a toaster near the microwave," whereas a scene graph, combined with commonsense reasoning, can; this positions 3D scene graphs as a natural interface between perception and planning in embodied AI systems and motivates further research into exploration strategies that exploit the relational and compositional structure of scene graphs more deeply.

5.4 Limitations

While the experimental results are encouraging, several limitations should be acknowledged.

Simulation-Only Evaluation All experiments were conducted in simulated environment using the Habitat simulator. While simulation enables controlled, reproducible experimentation with ground-truth annotations, it does not capture the full complexity of real-world deployment. Real environments exhibit uncontrolled and varying lighting conditions, reflective and transparent surfaces, motion blur, and sensor noise, factors that can significantly affect both monocular depth estimation and object segmentation. The sim-to-real gap remains a well-known challenge in embodied AI, and the performance reported in this thesis should be interpreted as an upper bound on what might be achieved in uncontrolled real-world settings.

Furthermore, the domain sensitivity observed across experiments (the need to substantially re-tune merge thresholds between Replica and ReplicaCAD, and the different scale estimation accuracy across environments) already indicates that transferring the pipeline to real-world settings would likely require environment-specific calibration or the development of adaptive mechanisms that can adjust to varying visual conditions without manual intervention.

Computational Considerations MapAnything's requirement to load all input frames into GPU memory simultaneously imposed practical constraints on the experiments. The Replica validation required subsampling from 2000 to 154 frames due to the 24 GB VRAM limit, and each exploration step in the active exploration experiments involved a full forward pass through the transformer architecture. While the experiments did not focus on computational efficiency, the latency of the RGB-only pipeline is a relevant consideration for real-time or near-real-time deployment. The ASP strategy additionally requires LLM inference at each exploration step to generate scene completion hypotheses, adding further computational overhead. Reducing this latency would be necessary to deploy the system on resource-constrained robotic platforms for real-time applications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis addressed the research question:

> *Can accurate 3D scene graphs be constructed incrementally (updating the graph step by step as new observations arrive) and actively (selecting viewpoints based on expected information gain) using only RGB images?*

The work presented across the preceding chapters provides an affirmative answer, subject to the trade-offs and limitations discussed below.

RGB-Only 3D Scene Graph Generation Pipeline The first contribution was the development of an RGB-only 3D scene graph generation pipeline, integrating MapAnything [1], a state-of-the-art feed-forward 3D reconstruction model, with ConceptGraphs [2], an open-vocabulary scene graph framework originally designed for posed RGB-D input. A deterministic geometric edge generator was introduced to replace the original LLM-based edge inference, removing dependence on proprietary models while ensuring reproducible spatial relationships. Quantitative evaluation on seven Replica [3] scenes demonstrated that this RGB-only pipeline achieves an average F1-score of 0.500, nearly identical to the 0.499 obtained by ConceptGraphs with ground-truth depth and pose—confirming that learned monocular depth and pose estimation can effectively replace dedicated depth sensors for scene graph construction without measurable degradation in overall quality. The main observable effect was a rebalancing of the precision–recall trade-off: estimated depths introduced small geometric inconsistencies that increased fragmentation (22% more predicted nodes on average), lowering precision while simultaneously preventing over-merging and thereby boosting recall.

Active Exploration Strategies for Scene Graph Construction The second contribution was a comparative study of active exploration strategies, evaluating the geometric frontier-based Surface Edge Explorer (SEE) [5] against the semantic-driven Active Semantic Perception (ASP) [6] framework for incremental scene graph construction. Across 240 experimental trials on six ReplicaCAD apartments [7], ASP achieved approximately double the recall (0.54 vs. 0.22) and nearly double the F1-score (0.57 vs. 0.33) compared to SEE after 30 exploration steps. This large performance gap demonstrated that leveraging the structured semantic information encoded in the evolving scene graph, rather than relying solely on geometric coverage, leads to substantially more efficient and complete scene graph construction. ASP also exhibited lower variance across starting positions, indicating greater robustness to initialization conditions. A key conceptual insight emerged from this comparison: the 3D scene graph serves a dual role as both the *product* and the *instrument* of exploration, enabling the robot to reason about what is likely missing from the current representation and to direct future observations accordingly. This is a particularly valuable property because it effectively closes the perception–action loop: the same representation that the robot builds through observation is also the one it reasons over to decide where to look next. Unlike dense representations such as point clouds or occupancy maps, the scene graph provides a compact and structured abstraction of the environment in which object identities and spatial relationships are explicitly represented. This structure enables efficient reasoning about what elements may still be missing and allows the system to translate such reasoning into targeted exploration actions.

Integration of External Cameras The third contribution was the study of external camera integration in two complementary settings: as a supplement to robot-driven active exploration and as the sole source of observations. When a single overhead camera was incorporated at initialization, ASP's final F1-score improved from approximately 0.57 to 0.62, and the F1-score that SEE with external cameras achieved at step 0 was only reached by SEE without external cameras around step 25. In the standalone external camera study across 90 ReplicaCAD scenes, three fixed cameras achieved an F1-score of 0.421 in complex apartments and 0.516 in simpler furnished rooms (with ICP alignment), capturing the dominant spatial structure of each environment. These results established that fixed RGB cameras can provide a low-cost structural skeleton of the scene, which active exploration can then refine through close-range, semantically guided observation. The RGB-only design is what makes this integration seamless: external camera images are processed by MapAnything in exactly the same way as the robot's onboard frames, requiring no additional calibration, depth sensors, or pipeline modifications.

Automated Validation Framework The fourth contribution was the design and implementation of an automated validation framework for open-vocabulary 3D scene graphs. Evaluating such representations is inherently challenging because predicted labels are free-form text that cannot be compared against a fixed taxonomy, and the original ConceptGraphs evaluation relied on manual human annotation [2], a labour-intensive and partially subjective process that does not scale to high numbers of experiments. The proposed framework addresses this challenge through joint semantic–geometric node matching: predicted and ground-truth nodes are paired by requiring both high cosine similarity between SentenceTransformer label embeddings [96] and small Euclidean distance between their 3D centres, with configurable thresholds for both criteria.

Two complementary evaluation variants were integrated into the framework: ICP-based alignment with optional scale estimation [21], which separates global pose and scale errors from local detection failures, and duplicate-node removal, which isolates the effect of fragmented multi-view association on precision. Together, these variants enabled fine-grained diagnostic analysis across all three experimental studies, supporting systematic comparisons across 7 Replica scenes, 240 active exploration trials across 6 ReplicaCAD apartments, and 270 external camera experiments without manual annotation.

Edge evaluation was not included for two reasons: the geometric edge generator introduced in this thesis is fully deterministic and relies solely on 3D bounding-box geometry, and neither the Replica nor ReplicaCAD datasets provide ground-truth edge annotations, making quantitative edge evaluation infeasible.

Taken together, these four contributions demonstrate that accurate 3D scene graphs can indeed be constructed incrementally and actively from RGB images alone.

6.2 Future Work

While the results presented in this thesis are encouraging, several directions remain open for future investigation.

Real-World Validation The most immediate and critical next step is to validate the proposed pipeline in real-world environments. All experiments in this thesis were conducted in simulation using the Habitat simulator, which, despite its photo-realistic rendering capabilities, does not capture the full complexity of real-world deployment. Real environments present challenges that are absent or attenuated in simulation, including uncontrolled and time-varying lighting conditions, reflective and transparent surfaces (glass tables, mirrors, windows), motion blur from the

robot's own movement, sensor noise inherent to commodity cameras, and dynamic elements such as moving people.

The domain sensitivity already observed within simulation (the need to re-tune ConceptGraphs merge thresholds between the photorealistic Replica scenes and the artist-authored ReplicaCAD environments, and the different MapAnything scale estimation accuracy across domains) strongly suggests that transferring the pipeline to real-world settings will require careful parameter tuning.

Crucially, MapAnything's depth and pose estimation lies at the foundation of the entire system: errors at this stage propagate to all downstream components, including point cloud reconstruction, ConceptGraphs' multi-view object association, scene graph construction, and ultimately active exploration decisions. In simulation, images are noise-free and rendered under controlled conditions, providing near-ideal inputs; in real-world settings, however, factors such as uncontrolled lighting, reflective or transparent surfaces, motion blur, and sensor noise may significantly degrade depth and pose estimates.

Additionally, MapAnything requires sufficient visual overlap between frames to maintain geometric consistency. In active exploration, where next-best-view selections may be far from previous viewpoints, such overlap is not guaranteed and may need to be enforced, for example by capturing intermediate frames along the navigation path or incorporating overlap constraints into the next-best-view selection process.

Replacing the LLM in the Exploration Loop The Active Semantic Perception (ASP) framework uses a large language model (LLM) at each exploration step to generate plausible completions of a partially observed scene graph. While this semantic reasoning significantly improves exploration performance, relying on an LLM introduces several practical limitations.

First, computational cost: each exploration step requires an LLM inference call, increasing latency and making real-time deployment on resource-limited robots difficult. Second, lack of reproducibility: LLM outputs are stochastic, so identical inputs can produce different scene completions and exploration trajectories. Third, external dependency: cloud-based LLMs require network connectivity and API availability, which may not be reliable in real-world deployment scenarios.

A promising direction is to replace the LLM with a **learned scene completion model** trained specifically on indoor scene graph data. Such a model could take as input the current partial scene graph (node types, positions, and spatial relationships) and predict a distribution over plausible missing objects and their likely locations, effectively distilling the commonsense spatial knowledge that the LLM currently provides into a compact, fast, and deterministic module. Training data for such a model could be derived from large-scale 3D scene datasets (e.g., ScanNet [101], HM3D [98]) by constructing partial scene graphs through random

or systematic removal of objects and training the model to predict the removed elements. Graph neural networks (GNNs) or transformer architectures operating on graph-structured inputs are natural candidates for this task, as they can capture the relational and compositional structure of scene graphs directly.

An alternative approach is to replace the LLM with a **retrieval-based system** that maintains a database of known scene graph patterns (e.g., "kitchens typically contain a stove, refrigerator, and sink in spatial proximity") and retrieves relevant patterns based on the current scene graph content. This would be fully deterministic, require no learned model, and could be easily extended or customized for specific deployment domains. However, it may lack the generalization capability of learned models when encountering novel or unusual scene configurations.

Regardless of the specific approach, the key requirement is that the replacement module must preserve the core capability that makes ASP effective: the ability to reason about *what is likely missing* from the current scene graph based on what has already been observed, and to translate this reasoning into spatially grounded predictions that can guide viewpoint selection.

Dynamic Environments and Temporal Scene Graph Maintenance All experiments in this thesis assume a static environment, where objects remain fixed and the scene graph is built once without later updates. In reality, environments are dynamic: furniture can be rearranged, objects moved, added, or removed, and occupants continuously modify the scene. Extending the pipeline to handle such dynamics would require detecting changes between new observations and the existing scene graph (e.g., moved or newly appeared objects), selectively updating affected nodes and edges without rebuilding the entire graph, and maintaining temporal consistency so that transient events (such as a person walking through the scene) are not permanently incorporated. External cameras are particularly suitable for this task, as they can continuously monitor the environment and trigger scene graph updates when significant changes occur. Combining periodic external monitoring with on-demand robot exploration for detailed inspection would represent a natural extension of the deployment strategy investigated in this thesis.

Improving Multi-View Object Association A consistent finding across all three studies was precision decay caused by fragmented multi-view association: depth estimation errors introduce geometric inconsistencies that prevent ConceptGraphs from merging observations of the same object, producing duplicate nodes. The post-processing step partially mitigates this issue but acts as a retrospective correction rather than addressing the root cause.

Future work could investigate more robust multi-view association strategies that are specifically designed to tolerate the geometric noise introduced by learned depth

estimation. Possible directions include learning a similarity metric that jointly considers semantic, geometric, and appearance features while being explicitly trained to handle depth-estimation noise, or adopting probabilistic data association methods that maintain a distribution over possible object identities rather than committing to hard assignments.

Downstream Task Integration The 3D scene graphs constructed in this thesis were evaluated exclusively in terms of node-level precision, recall, and F1-score. While these metrics provide a rigorous quantitative assessment of scene graph quality, they do not directly measure the utility of the representation for downstream robotic tasks. An important direction for future work is to evaluate how the generated scene graphs perform when used as input for practical applications such as object search and retrieval (e.g., "find the red mug in the kitchen"), task and motion planning (e.g., "pick up the cup from the table and place it in the sink"), or natural language grounding and human–robot interaction. Furthermore, integrating the active exploration loop with a specific downstream task could enable task-driven exploration, where the robot prioritizes viewpoints that are most informative for the task at hand rather than for general scene graph completeness (for example, focusing on the kitchen area when asked to retrieve a mug, rather than exhaustively mapping the entire apartment).

6.3 Final Remarks

More broadly, this thesis sits at the intersection of robotic perception and planning, and its central finding is that the choice of scene representation matters profoundly for both. By adopting 3D scene graphs—a structured, semantically rich, yet lightweight representation—the system gains the ability not only to understand the environment but also to reason about what remains unknown and to act on that reasoning. This tight coupling between representation, understanding, and action suggests that future progress in embodied AI will increasingly depend on representations that are designed not just to describe the world, but to guide interaction with it.

At the same time, the decision to build the entire pipeline on RGB images alone lowers the barrier to deployment, making these capabilities accessible in settings where depth sensors, precise calibration, or specialized hardware are impractical or unavailable, from low-cost mobile platforms to existing CCTV infrastructures. In this way, the work illustrates how semantically structured representations built from simple visual input can support both rich scene understanding and practical deployment in real-world environments.

Appendix A

Active Exploration Results - Complete Figures

This appendix presents the complete set of results for all active exploration experiments described in Chapter 4. Results are organized by experimental configuration: first, the comparison between geometric-based (SEE) and semantic-based (ASP) exploration strategies without external cameras, followed by the contribution of external camera integration.

For each apartment scene (`apt_0`–`apt_2`), figures show the evolution of scene graph quality metrics (precision, recall, and F1-score) and node counts over 30 exploration steps. Each plot displays the mean value across 10 different starting positions, with shaded regions indicating ± 1 standard deviation to show variability due to initial robot placement.

A.1 Geometric vs. Semantic-based Exploration (Onboard Camera Only)

This section presents results for active exploration using only the robot's onboard egocentric camera. The left column shows results for the SEE (geometric-based) strategy, and the right column shows results for the ASP (semantic-based) strategy.

A.1.1 Metrics Evolution

Here, the evolution of precision, recall, and F1-score is shown for each apartment scene.

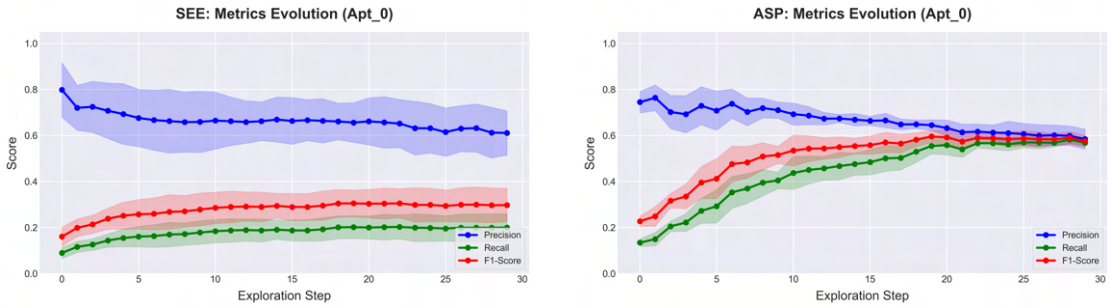


Figure A.1: Precision, Recall and F1-score for `apt_0`.

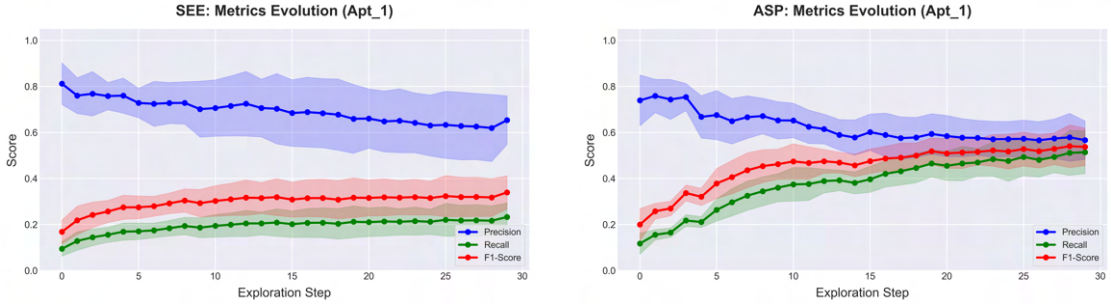


Figure A.2: Precision, Recall and F1-score for `apt_1`.

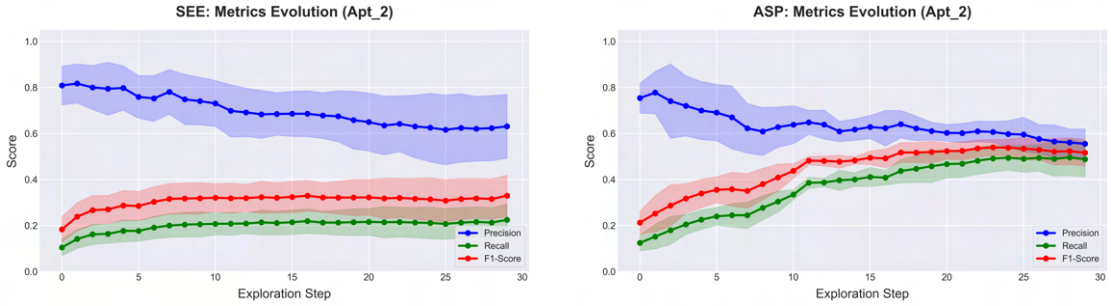


Figure A.3: Precision, Recall and F1-score for `apt_2`.

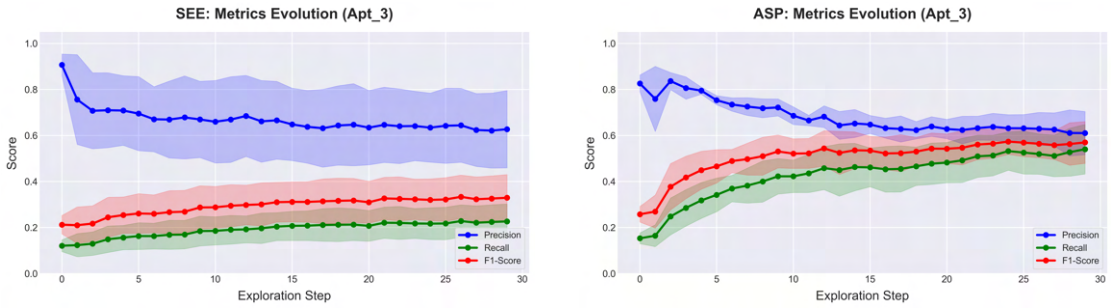


Figure A.4: Precision, Recall and F1-score for `apt_3`.

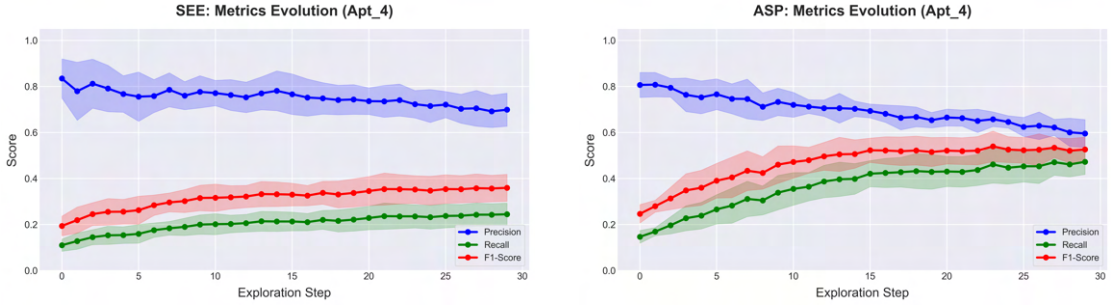


Figure A.5: Precision, Recall and F1-score for `apt_4`.

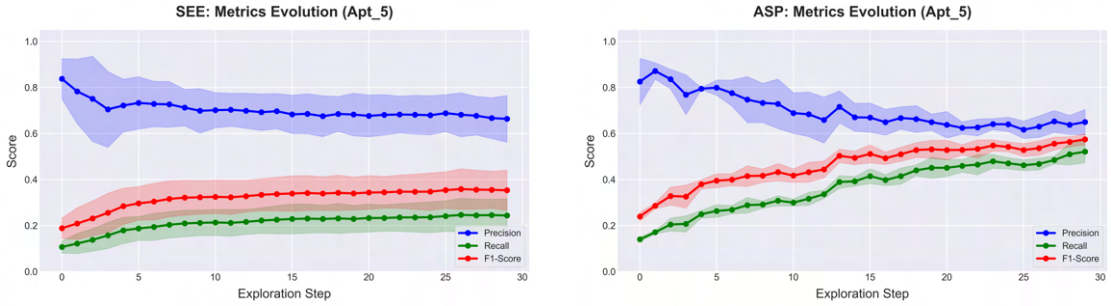


Figure A.6: Precision, Recall and F1-score for `apt_5`.

A.1.2 Node Count Evolution

Here, the evolution of the number of predicted nodes is shown for each apartment scene.

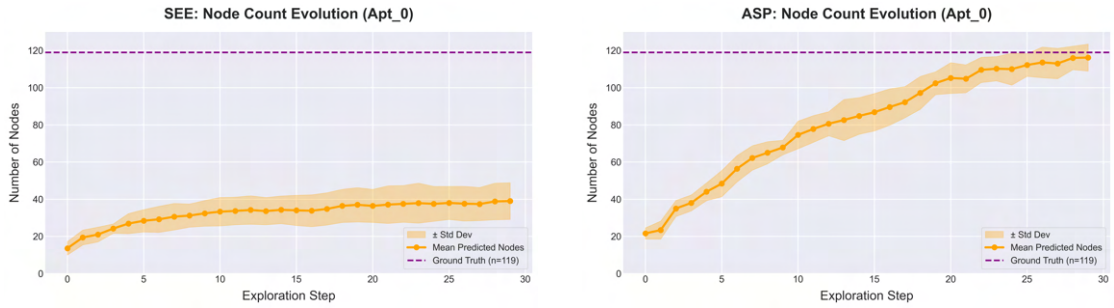


Figure A.7: Node count evolution for `apt_0`.

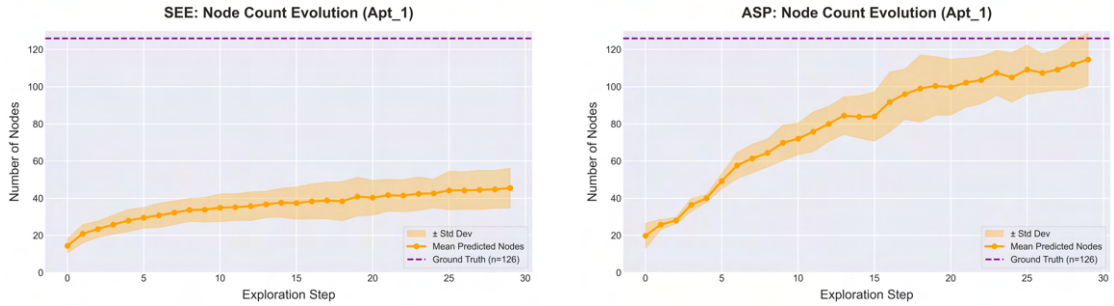


Figure A.8: Node count evolution for `apt_1`.

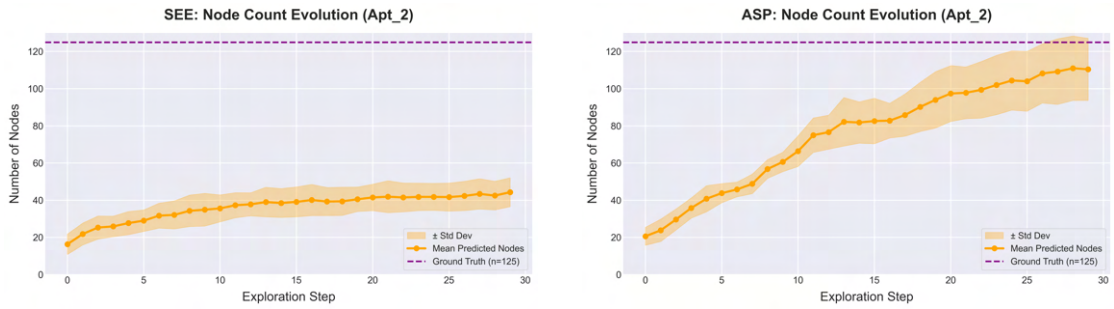


Figure A.9: Node count evolution for `apt_2`.

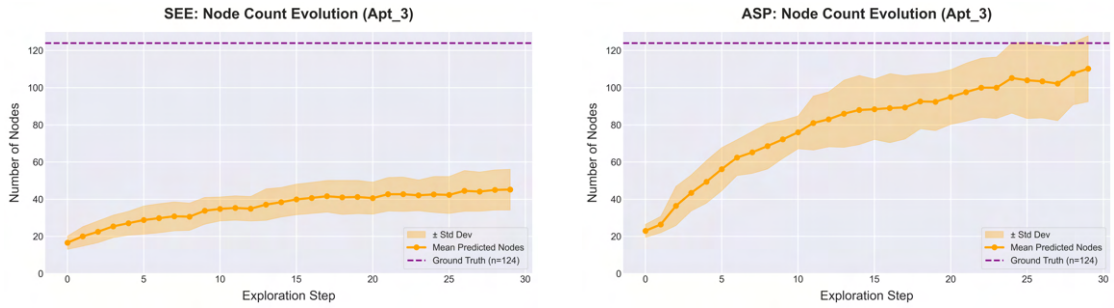


Figure A.10: Node count evolution for `apt_3`.

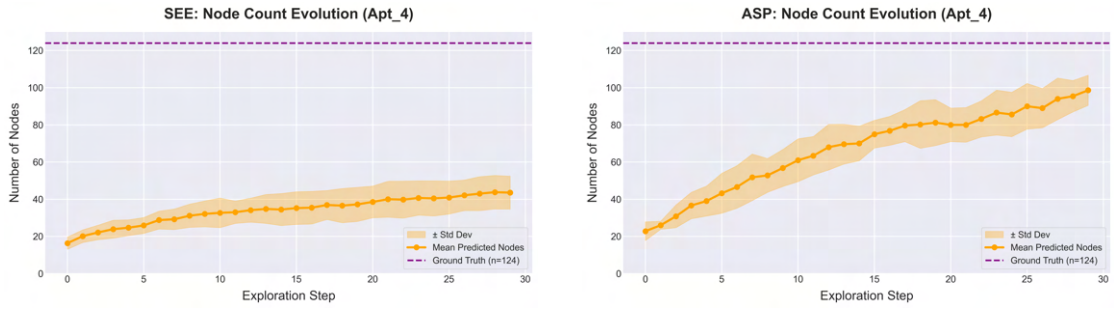


Figure A.11: Node count evolution for `apt_4`.

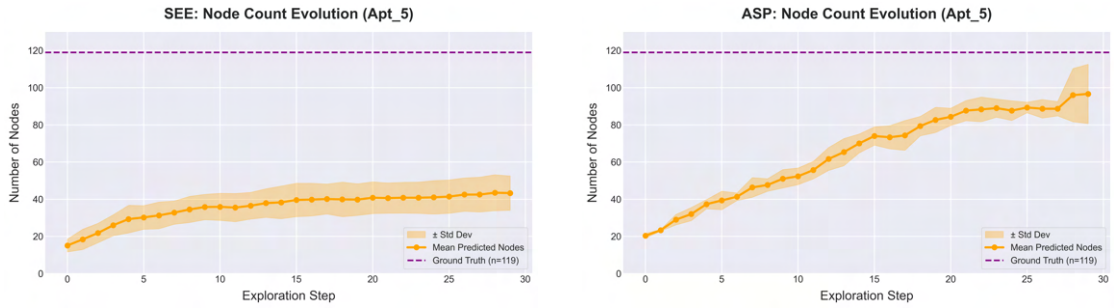


Figure A.12: Node count evolution for `apt_5`.

A.2 External Camera Contribution

This section presents results for active exploration with the addition of an external overhead camera at initialization (step 0), in addition to the robot's onboard camera. The left column shows results for the SEE (geometric-based) strategy, and the right column shows results for the ASP (semantic-based) strategy.

A.2.1 Metrics Evolution with External Camera

Here the evolution of precision, recall, and F1-score is shown for each apartment scene with the external camera configuration.

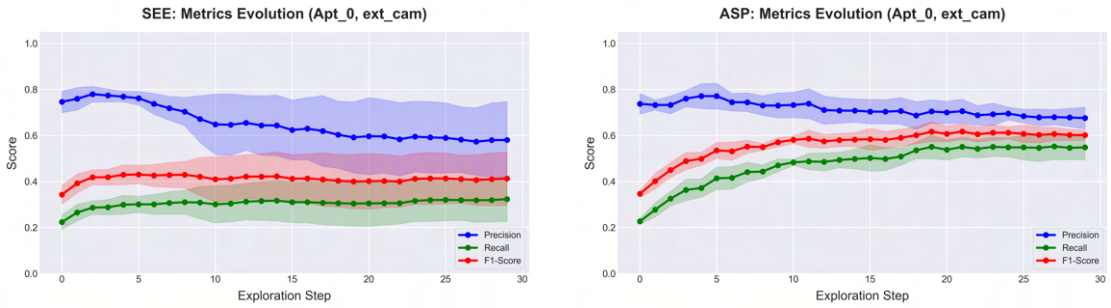


Figure A.13: Precision, Recall and F1-score for `apt_0` with external camera.

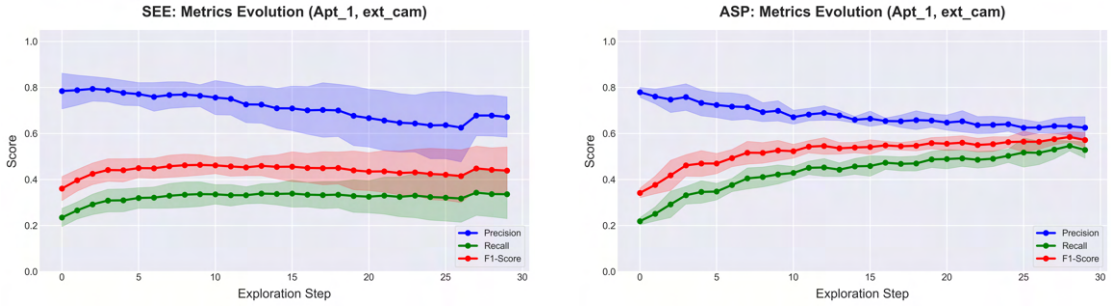


Figure A.14: Precision, Recall and F1-score for `apt_1` with external camera.

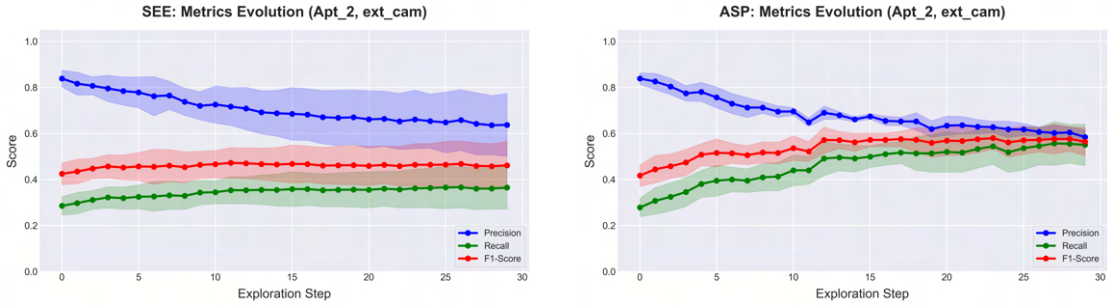


Figure A.15: Precision, Recall and F1-score for `apt_2` with external camera.

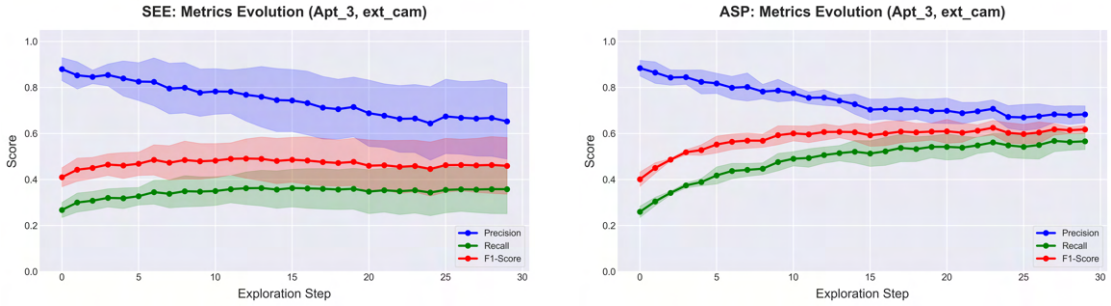


Figure A.16: Precision, Recall and F1-score for `apt_3` with external camera.

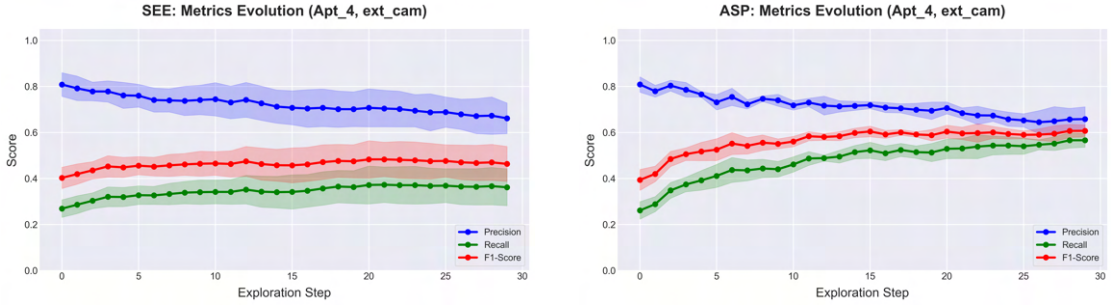


Figure A.17: Precision, Recall and F1-score for `apt_4` with external camera.

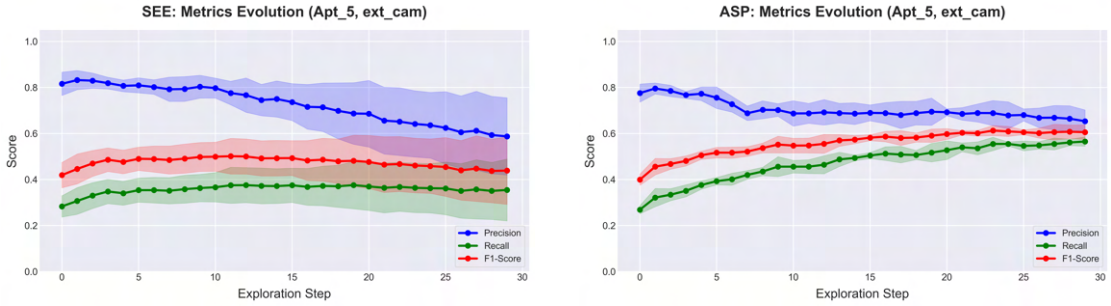


Figure A.18: Precision, Recall and F1-score for `apt_5` with external camera.

A.2.2 Node Count Evolution with External Camera

Here, the evolution of the number of predicted nodes is shown for each apartment scene with the external camera configuration.

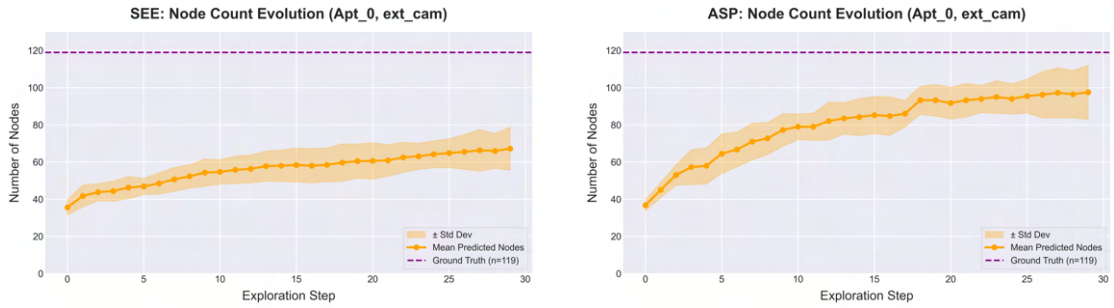


Figure A.19: Node count evolution for `apt_0` with external camera.

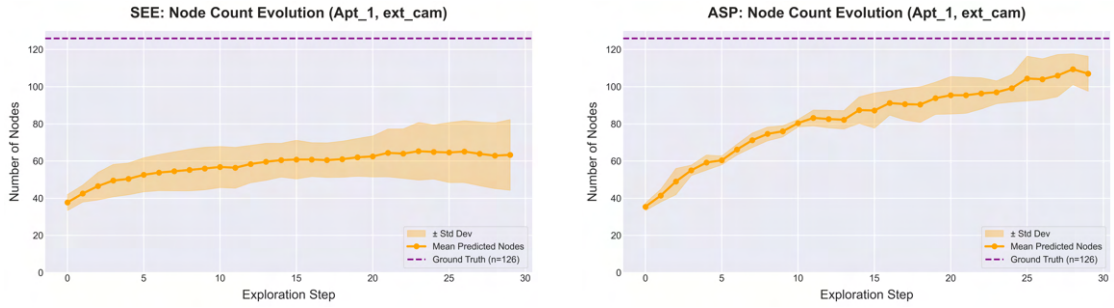


Figure A.20: Node count evolution for `apt_1` with external camera.

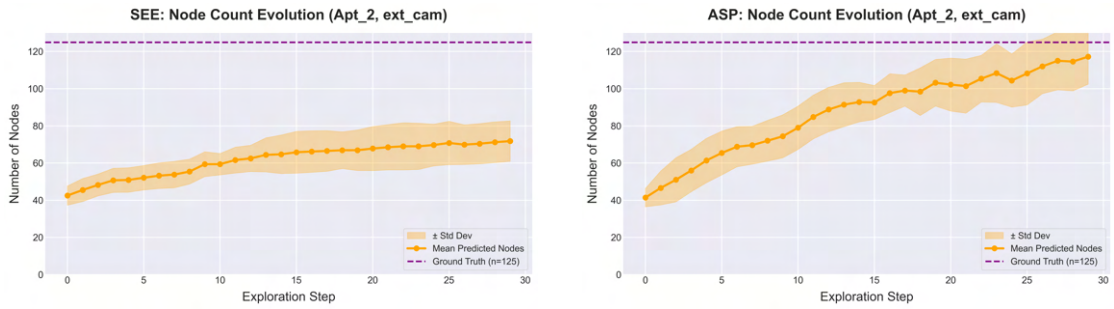


Figure A.21: Node count evolution for `apt_2` with external camera.

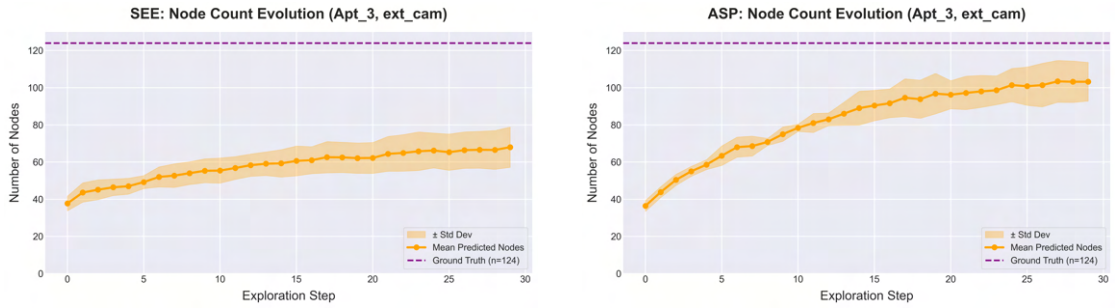


Figure A.22: Node count evolution for `apt_3` with external camera.

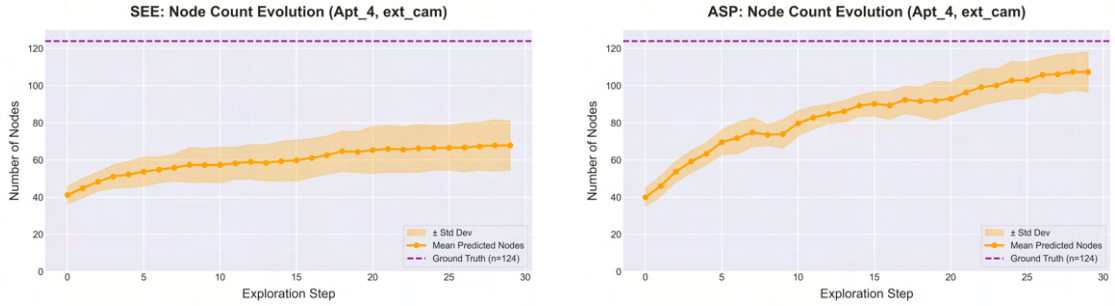


Figure A.23: Node count evolution for `apt_4` with external camera.

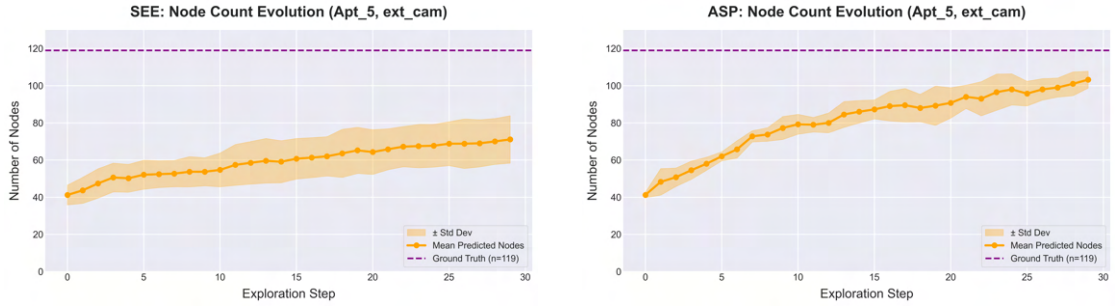


Figure A.24: Node count evolution for `apt_5` with external camera.

Bibliography

[1] Nikhil Keetha et al. «MapAnything: Universal Feed-Forward Metric 3D Reconstruction». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2024 (cit. on pp. i, 2, 19, 30–32, 43, 57, 59, 84, 89).

[2] Qiao Gu et al. «ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2024 (cit. on pp. i, 1, 2, 10, 12, 15, 30, 34–39, 58, 68, 89, 91).

[3] Julian Straub, Thomas Whelan, Lingni Ma, et al. *The Replica Dataset: A Digital Replica of Indoor Spaces*. 2019. arXiv: 1906.05797 [cs.CV]. URL: https://arxiv.org/abs/1906.05797 (cit. on pp. i, 36, 39, 58, 67, 89).

[4] Rowan Border, Jonathan D. Gammell, and Paul Newman. «Surface Edge Explorer (see): Planning Next Best Views Directly from 3D Observations». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2018 (cit. on pp. i, 2, 3, 22, 43).

[5] Rowan Border and Jonathan D. Gammell. «The surface edge explorer (SEE): A measurement-direct approach to next best view planning». In: *The International Journal of Robotics Research (IJRR)* 43.10 (2024), pp. 1506–1532 (cit. on pp. i, 2, 3, 22, 42, 43, 45, 90).

[6] Huayi Tang and Pratik Chaudhari. *Active semantic perception*. 2025. arXiv: 2510.05430 [cs.RO]. URL: https://arxiv.org/abs/2510.05430 (cit. on pp. i, 2, 3, 14, 24, 42, 52, 54, 55, 90).

[7] Andrew Szot et al. «Habitat 2.0: Training Home Assistants to Rearrange their Habitat». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021 (cit. on pp. i, 3, 39, 54, 67, 68, 90).

[8] Jaewoo Bae, Donghyun Shin, Kibeom Ko, Jaehyun Lee, and Uichin H. Kim. «A Survey on 3D Scene Graphs: Definition, Generation and Application». In: *Robot Intelligence Technology and Applications 7*. Springer, 2023, pp. 181–196 (cit. on pp. 1, 5, 9).

[9] Hongsheng Li et al. «Scene Graph Generation: A comprehensive survey». In: *Neurocomputing* 566 (2024) (cit. on pp. 1, 5–7).

[10] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. «SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning». In: *Proceedings of the Conference on Robot Learning (CoRL)*. 2023 (cit. on pp. 1, 11).

[11] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. «Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation». In: *Proceedings of Robotics: Science and Systems (RSS)*. 2024 (cit. on pp. 1, 10, 12).

[12] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. «RoboEXP: Action-Conditioned Scene Graph via Interactive Exploration for Robotic Manipulation». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2024 (cit. on pp. 1, 11).

[13] Nathan Hughes, Yun Chang, and Luca Carlone. «Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization». In: *Proceedings of Robotics: Science and Systems (RSS)*. 2022 (cit. on pp. 1, 8, 9, 13).

[14] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. «SceneGraphFusion: Incremental 3D Scene Graph Prediction From RGB-D Sequences». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 1, 9, 13).

[15] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. «DUSt3R: Geometric 3D vision made easy». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024 (cit. on pp. 2, 18, 34).

[16] Gunhee Bae, Ignas Budvytis, and Roberto Cipolla. «VGGT: Visual Geometry Grounded Transformer». In: *Proceedings of the International Conference on 3D Vision (3DV)*. 2025 (cit. on pp. 2, 18, 34).

[17] Alexander Kirillov et al. «Segment Anything». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023 (cit. on pp. 2, 35, 60).

[18] Alec Radford et al. «Learning Transferable Visual Models From Natural Language Supervision». In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2021 (cit. on pp. 2, 35, 60).

[19] Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. «Revisiting active perception». In: *Autonomous Robots* 42.2 (2018), pp. 177–196 (cit. on pp. 2, 20).

[20] William R. Scott, Gerhard Roth, and Jean-François Rivest. «View planning for automated three-dimensional object reconstruction and inspection». In: *ACM Computing Surveys* 35.1 (2003), pp. 64–96 (cit. on pp. 2, 20, 21).

[21] P.J. Besl and Neil D. McKay. «A method for registration of 3-D shapes». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 14.2 (1992), pp. 239–256 (cit. on pp. 4, 40, 91).

[22] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. «Scene Graph Generation by Iterative Message Passing». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 5–7).

[23] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. «A Comprehensive Survey of Scene Graphs: Generation and Application». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 45.1 (2023), pp. 1–26 (cit. on pp. 6, 7).

[24] Iacopo Catalano, Carlos Cueto Zumaya, Julio A Placed, Javier Civera, Wallace Moreira Bessa, and Jorge Peña-Queralta. «3D scene graphs in robotics: A unified representation bridging geometry, semantics, and action». In: *Authorea Preprints* (2025) (cit. on pp. 6, 8, 10).

[25] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. «Image Retrieval Using Scene Graphs». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (cit. on p. 7).

[26] Ranjay Krishna et al. «Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations». In: *International Journal of Computer Vision (IJCV)* 123.1 (2017), pp. 32–73 (cit. on p. 7).

[27] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. «Neural Motifs: Scene Graph Parsing with Global Context». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 7).

[28] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. «Graph R-CNN for Scene Graph Generation». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 7).

[29] Yuren Cong, Michael Ying Yang, and Bodo Rosenhahn. «RelTR: Relation Transformer for Scene Graph Generation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 45.9 (2023), pp. 11119–11136 (cit. on p. 7).

[30] Rongjie Li, Songyang Zhang, and Xuming He. «SGTR: End-to-end Scene Graph Generation with Transformer». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022 (cit. on p. 7).

[31] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. «3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 5664–5673 (cit. on p. 8).

[32] Ue-Hwan Kim, Jin-Man Park, Taek-Jin Song, and Jong-Hwan Kim. «3-D Scene Graph: A Sparse and Semantic Representation of Physical Environments for Intelligent Agents». In: *IEEE Transactions on Cybernetics* 50.12 (2020), pp. 4921–4933 (cit. on pp. 8, 9).

[33] Antoni Rosinol, Marco Abate, Yun Chang, and Luca Carlone. «Kimera: From SLAM to Spatial Perception with 3D Dynamic Scene Graphs». In: *The International Journal of Robotics Research (IJRR)* (2021) (cit. on pp. 8, 12).

[34] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. «Learning 3D Semantic Scene Graphs From 3D Indoor Reconstructions». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 (cit. on p. 9).

[35] Lianghao Xia, Ben Kao, and Chao Huang. *OpenGraph: Towards Open Graph Foundation Models*. 2024. arXiv: `2403.01121 [cs.LG]` (cit. on p. 10).

[36] Zhe Ni, Xiaoxin Deng, Cong Tai, Xinyue Zhu, Qinghongbing Xie, Weihang Huang, Xiang Wu, and Long Zeng. «GRID: Scene-Graph-based Instruction-driven Robotic Task Planning». In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024 (cit. on p. 11).

[37] Yixuan Wang, Leonor Fermoselle, Tarik Kelestemur, Jiuguang Wang, and Yunzhu Li. *CuriousBot: Interactive Mobile Exploration via Actionable 3D Relational Object Graph*. 2025. arXiv: `2501.13338 [cs.RO]` (cit. on p. 11).

[38] Tixiao Shan, Abhinav Rajvanshi, Niluthpol Mithun, and Han-Pang Chiu. *Graph2Nav: 3D Object-Relation Graph Generation to Robot Navigation*. 2025. arXiv: `2504.16782 [cs.RO]` (cit. on p. 12).

[39] Yang Miao, Francis Engelmann, Olga Vysotska, Federico Tombari, Marc Pollefeys, and Dániel Béla Baráth. «SceneGraphLoc: Cross-Modal Coarse Visual Localization on 3D Scene Graphs». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024 (cit. on p. 12).

[40] Hriday Bavle, Jose Luis Sanchez-Lopez, Muhammad Shaheer, Javier Civera, and Holger Voos. «S-Graphs+: Real-time Localization and Mapping leveraging Hierarchical Representations». In: *IEEE Robotics and Automation Letters (RA-L)* 8.8 (2023), pp. 4927–4934 (cit. on p. 12).

[41] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A. Castellanos. «A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers». In: *IEEE Transactions on Robotics (T-RO)* 39.3 (2023), pp. 1686–1705 (cit. on pp. 13, 23).

[42] Bashar Alsadik, Hussein Alwan Mahdi, and Nagham Amer Abdulateef. «A structured review and taxonomy of next-best-view strategies for 3D reconstruction». In: *ISPRS Open Journal of Photogrammetry and Remote Sensing* 17 (2025), p. 100098 (cit. on p. 13).

[43] Xinghang Li, Di Guo, Huaping Liu, and Fuchun Sun. «Embodied Semantic Scene Graph Generation». In: *Proceedings of the Conference on Robot Learning (CoRL)*. 2022 (cit. on p. 13).

[44] Noé José Zapata Cornejo, Gerardo Pérez, Alejandro Torrejón, Pedro Núñez, and Pablo Bustos. «Concept-Guided Exploration: Building Persistent, Actionable Scene Graphs». In: *Applied Sciences* 15.20 (2025), p. 11084 (cit. on p. 14).

[45] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd. Cambridge University Press, 2003 (cit. on p. 15).

[46] Johannes Lutz Schönberger and Jan-Michael Frahm. «Structure-from-Motion Revisited». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 15, 16, 34).

[47] David G. Lowe. «Distinctive image features from scale-invariant keypoints». In: *International Journal of Computer Vision (IJCV)* 60.2 (2004), pp. 91–110 (cit. on p. 15).

[48] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. «SURF: Speeded up robust features». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2006, pp. 404–417 (cit. on p. 15).

[49] Martin A. Fischler and Robert C. Bolles. «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography». In: *Communications of the ACM* 24.6 (1981), pp. 381–395 (cit. on p. 15).

[50] Noah Snavely, Steven M. Seitz, and Richard Szeliski. «Photo tourism: exploring photo collections in 3D». In: *ACM SIGGRAPH*. 2006, pp. 835–846 (cit. on p. 16).

[51] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. «Bundle Adjustment — A Modern Synthesis». In: *Vision Algorithms: Theory and Practice*. Springer, 2000, pp. 298–372 (cit. on p. 16).

[52] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. «A comparison and evaluation of multi-view stereo reconstruction algorithms». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2006, pp. 519–528 (cit. on pp. 16, 17).

[53] Yasutaka Furukawa and Jean Ponce. «Accurate, dense, and robust multiview stereopsis». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 32.8 (2010), pp. 1362–1376 (cit. on p. 16).

[54] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. «Poisson surface reconstruction». In: *Proceedings of the Eurographics Symposium on Geometry Processing (SGP)*. 2006 (cit. on p. 16).

[55] Yiqi Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. «3D reconstruction using deep learning: a survey». In: *Communications in Information and Systems* 20.4 (2020), pp. 367–415 (cit. on p. 17).

[56] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. «Learning-based multi-view stereo: A survey». In: *arXiv preprint arXiv:2408.15235* (2024). URL: `https://arxiv.org/abs/2408.15235` (cit. on p. 17).

[57] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. «MVSNet: Depth inference for unstructured multi-view stereo». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 767–783 (cit. on p. 17).

[58] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. «Recurrent MVSNet for high-resolution multi-view stereo depth inference». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5525–5534 (cit. on p. 17).

[59] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. «Cascade cost volume for high-resolution multi-view stereo and stereo matching». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2495–2504 (cit. on p. 17).

[60] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. «NeRF: Representing scenes as neural radiance fields for view synthesis». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020, pp. 405–421 (cit. on p. 17).

[61] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. «3D Gaussian Splatting for real-time radiance field rendering». In: *ACM Transactions on Graphics (SIGGRAPH)* 42.4 (2023), pp. 1–14 (cit. on p. 18).

[62] Wanzhou Zhang. «Review of feed-forward 3D reconstruction: From DUSt3R to VGGT». In: *Journal of Advances in Information and Computer Science* (2025) (cit. on p. 18).

[63] Vincent Leroy, Yohann Cabon, and Jerome Revaud. «Grounding image matching in 3D with MASt3R». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2024 (cit. on p. 18).

[64] Ruzena Bajcsy. «Active perception». In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005 (cit. on p. 20).

[65] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. «Active vision». In: *International Journal of Computer Vision* 1.4 (1988), pp. 333–356 (cit. on p. 20).

[66] C. Connolly. «The determination of next best views». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. 1985, pp. 432–435 (cit. on pp. 20, 21).

[67] J. Irving Vasquez-Gomez, L. Enrique Sucar, and Rafael Murrieta-Cid. «View/state planning for three-dimensional object reconstruction under uncertainty». In: *Autonomous Robots* 41.1 (2017), pp. 89–109 (cit. on p. 21).

[68] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. «A comparison of volumetric information gain metrics for active 3D object reconstruction». In: *Autonomous Robots* 42.2 (2018), pp. 197–208 (cit. on p. 21).

[69] Luke Yoder and Sebastian Scherer. «Autonomous exploration for infrastructure modeling with a micro aerial vehicle». In: *Field and Service Robotics*. Springer, 2016, pp. 427–440 (cit. on p. 21).

[70] Ziyang Meng, Hao Qin, Zheng Chen, Xudong Chen, Hui Sun, Feng Lin, and Marcelo H. Ang. «A two-stage optimized next-view planning framework for 3-D unknown environment exploration, and structural reconstruction». In: *IEEE Robotics and Automation Letters (RA-L)* 2.3 (2017), pp. 1680–1687 (cit. on p. 21).

[71] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. «Receding horizon "next-best-view" planner for 3D exploration». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1462–1468 (cit. on pp. 21, 22).

[72] Suyoung Song and Sungmin Jo. «Online inspection path planning for autonomous 3D modeling using a micro-aerial vehicle». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 6217–6224 (cit. on p. 21).

[73] Katja Orestin Dierenbach, Martin Weinmann, and Boris Jutzi. «Next-best-view method based on consecutive evaluation of topological relations». In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. III-3. 2016, pp. 11–19 (cit. on pp. 21, 22).

[74] Sami Khalfaoui, Ralph Seulin, Yohan Fougerolle, and David Fofi. «An efficient method for fully automatic 3D digitization of unknown objects». In: *Computers in Industry* 64.9 (2013), pp. 1152–1160 (cit. on pp. 21, 22).

[75] Geoffrey A. Hollinger, Brendan Englot, Franz S. Hover, Urbashi Mitra, and Gaurav S. Sukhatme. «Active planning for underwater inspection and the benefit of adaptivity». In: *The International Journal of Robotics Research (IJRR)* 32.1 (2013), pp. 3–18 (cit. on pp. 21, 22).

[76] Mike Roberts, Alex Truong, Debadeepta Dey, Sudipta Sinha, Ashish Kapoor, Neel Joshi, and Pat Hanrahan. «Submodular trajectory optimization for aerial 3D scanning». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5334–5343 (cit. on pp. 21, 22).

[77] Brian Yamauchi. «A Frontier-Based Approach for Autonomous Exploration». In: *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 1997 (cit. on p. 22).

[78] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. «Receding horizon path planning for 3D exploration and surface inspection». In: *Autonomous Robots* 42.2 (2018), pp. 291–306 (cit. on p. 22).

[79] Mihir Naazare, Mihir Dharmadhikari, Mihir Kulkarni, and Kostas Alexis. «Online next-best-view planner for 3D-exploration and inspection with a mobile manipulator robot». In: *IEEE Robotics and Automation Letters (RA-L)* 7.2 (2022), pp. 3779–3786 (cit. on p. 23).

[80] Xiao Chen, Ran Xu, Shikun Wu, Liang Pan, Ziwei Liu, Jingyi Li, Christian Theobalt, Wenping Wang, and Yu Liu. «GenNBV: Generalizable next-best-view policy for active 3D reconstruction». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 16937–16948 (cit. on p. 23).

[81] Iñaki Lluvia, Elena Lazkano, and Ander Ansuategi. «Active mapping and robot exploration: A survey». In: *Sensors* 21.7 (2021), p. 2445 (cit. on p. 23).

[82] Juan Vegara Jaramillo, Carsten Rother, and Daniel D. Lee. «Semantic active vision exploration and mapping of indoor environments». In: *CEUR Workshop Proceedings*. Vol. 2594. 2020, pp. 41–46 (cit. on p. 23).

[83] Akshay K. Burusa, Ali Tavakoli, and Danica Kragic. «Semantics-aware next-best-view planning for efficient 3D object reconstruction». In: *Computers and Electronics in Agriculture* 223 (2024), p. 109094 (cit. on p. 23).

[84] Akshay K. Burusa, Francesco Verdoja, and Danica Kragic. «Gradient-based local next-best-view planning for improved perception of targeted plant nodes». In: *IEEE Robotics and Automation Letters (RA-L)* 9.2 (2024), pp. 1580–1587 (cit. on p. 23).

[85] Liyan Chen, Huangying Zhan, Hairong Yin, Yi Xu, and Philippos Mordohai. *Understanding while Exploring: Semantics-driven Active Mapping*. 2025. arXiv: 2506.00225 [cs.RO] (cit. on p. 23).

[86] Hongyu Ding et al. *SEA: Semantic Map Prediction for Active Exploration of Uncertain Areas*. 2025. arXiv: 2510.19766 [cs.RO] (cit. on p. 23).

[87] Jing Li, Jing Xu, Fangwei Zhong, Xiangxiang Kong, Yu Qiao, and Yizhou Wang. «Pose-Assisted Multi-Camera Collaboration for Active Object Tracking». In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 01. 2020, pp. 759–766 (cit. on p. 25).

[88] Lukas Esterle, Peter R. Lewis, Xin Yao, and Bernhard Rinner. «Socio-Economic Vision Graph Generation and Handover in Distributed Smart Camera Networks». In: *ACM Transactions on Sensor Networks* 10.2 (2014), 20:1–20:24 (cit. on p. 25).

[89] Luke Robinson and Daniele De Martini. «Visual Servoing on Wheels: Robust Robot Orientation Estimation in Remote Viewpoint Control». In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2024 (cit. on p. 25).

[90] Luke Robinson, Matthew Gadd, Paul Newman, and Daniele De Martini. «Robot-Relay: Building-Wide, Calibration-Less Visual Servoing with Learned Sensor Handover Networks». In: *Experimental Robotics: The 18th International Symposium (ISER)*. Springer, 2024, pp. 129–140 (cit. on p. 25).

[91] Davide Buoso, Luke Robinson, Giuseppe Averta, Philip Torr, Tim Franzmeyer, and Daniele De Martini. «Select2Plan: Training-Free ICL-Based Planning through VQA and Memory Retrieval». In: *IEEE Robotics and Automation Letters (RA-L)* 10.2 (2025) (cit. on p. 26).

[92] Maxime Oquab et al. «DINOv2: Learning Robust Visual Features without Supervision». In: *Transactions on Machine Learning Research (TMLR)* (2024) (cit. on p. 33).

[93] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. «Vision transformers for dense prediction». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2021, pp. 12179–12188 (cit. on p. 33).

[94] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. «Visual Instruction Tuning». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023 (cit. on p. 36).

[95] OpenAI. *GPT-4 Technical Report*. Tech. rep. OpenAI, 2023. URL: `https://arxiv.org/abs/2303.08774` (cit. on p. 36).

[96] Nils Reimers and Iryna Gurevych. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks». In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019 (cit. on pp. 39, 91).

[97] *ICP registration - Open3D Documentation*. `https://www.open3d.org/docs/latest/tutorial/pipelines/icp_registration.html` (cit. on p. 40).

[98] Santhosh K. Ramakrishnan et al. «Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021 (cit. on pp. 54, 92).

[99] Manolis Savva et al. «Habitat: A platform for embodied ai research». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9339–9347 (cit. on pp. 54, 67, 68).

[100] Xavi Puig et al. *Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots*. 2023 (cit. on pp. 54, 68).

[101] 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 conference on computer vision and pattern recognition*. 2017, pp. 5828–5839 (cit. on p. 92).