

MASTER THESIS

Thesis submitted in fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Mechatronics/Robotics

SAGE: Multi object semantic aware guided exploration with persistent memory

By: Kevin Eppacher, BSc

Student Number: 2310331013

Supervisor: Simon Schwaiger, MSc

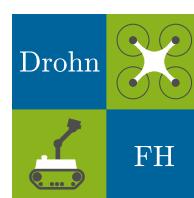
Vienna, January 22, 2026

This work was conducted in the context of the project “Stadt Wien Kompetenzteam für Drohnentechnik in der Fachhochschulausbildung” (project number MA23 35-02, financed by the Department MA23 for Economic Affairs, Labour and Statistics of the City of Vienna).

Funded by



Economic Affairs,
Labour and Statistics



Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, January 22, 2026

Signature

Kurzfassung

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Schlagworte: Keyword1, Keyword2, Keyword3, Keyword4

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Keywords: Keyword1, Keyword2, Keyword3, Keyword4

Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1	Introduction	1
1.1	Language-Guided Exploration	2
1.2	Language-Embedded Semantic Mapping	3
1.3	Scientific Contribution	6
1.4	Thesis Structure	7
2	State of the Art	9
2.1	Frontier-Based Exploration	10
2.2	RL-based Semantic Exploration	11
2.3	Foundation-Model-Based Semantic Exploration	14
2.4	Map Reconstruction and Persistent Semantic Mapping	18
2.5	Identified Research Gaps in State of the Art	26
3	Methods	29
3.1	System Overview	29
3.2	Semantic Frontier Exploration	30
3.3	Persistent Semantic 3D Mapping	40
3.4	Promptable Zero-Shot Detection	44
3.5	Fusion Strategy	48
3.6	Behavior Tree for Semantic-Guided Exploration	54
4	Implementation	55
4.1	Simulation Environment	55
4.2	Dataset	56
4.3	Used Software	57
4.4	Used Hardware	57
4.5	Evaluation Pipeline Overview	58
4.6	Evaluation Metrics	58
5	Discussion and Results	65
5.1	Experiment 1: Benchmarking on Matterport Scenes	65
5.2	Experiment 2: Impact of Exploration-Memory Weighting	65
5.3	Experiment 3: Sensitivity to Semantic Map Granularity	67
5.4	Experiment 4: Effect of Multi-Source Semantic Fusion	68
5.5	Experiment 5: System Efficiency and Real-World Validation	69
6	Summary and Outlook	70

Bibliography	71
List of Figures	81
List of Tables	83
List of source codes	84
A Methodology details	85
A.1 Behavior Tree Structure	85

1 Introduction

The introduction of Transformer-based architectures [1] has opened new opportunities for integrating high-level semantic reasoning with low-level geometric navigation in robotics. Traditional robotic exploration methods have primarily focused on mapping unknown environments using geometric cues, often neglecting the rich semantic information available in visual and linguistic modalities [2]. However, recent advances in Large Language Models (LLMs) [3] and Vision-Language Models (VLMs) [4] have enabled robots to interpret and act upon complex, open-ended instructions expressed in natural language [5]. These developments mark a transition from deterministically modeled and explicitly programmed robotic systems toward zero-shot generalizable behavior, enabling robots to reason about previously unseen concepts beyond fixed, task-specific datasets.

Consequently, new applications have emerged that require robots not only to explore and map their surroundings but also to reason about the semantic structure and relationships within them. In service robotics, for instance, a mobile agent may be instructed to locate a specific object based on high-level descriptions such as *“find the red chair in the living room”*, rather than relying on a limited set of predefined categories such as those from Common Objects in Context (COCO) [6]. Similarly, in Search and Rescue (SAR) operations [7], robots may be tasked with locating missing persons based on vague or incomplete contextual information, such as the assumption that an individual might be found *“in the bathroom”*. In industrial inspection, autonomous agents must identify structural anomalies or specific components within unstructured and partially observable environments, while in warehouse automation, robots must locate items or storage units that may not be consistently labeled or fully visible.

Across these domains, the integration of semantic understanding with autonomous navigation is an active area of research [8–11]. Robots must be capable of interpreting abstract human instructions. Traditional geometric exploration approaches [2, 12, 13] focus on exploring unknown environments by maximizing information gain about the spatial structure. However, in semantic exploration tasks, the robot must reason jointly about spatial and semantic context to strategically locate target objects or regions of interest [9, 10, 14, 15].

Current research increasingly leverages pretrained VLMs to extract semantic cues from RGB images, enabling zero-shot reasoning about novel objects and scenes [8, 9, 11]. However, many of these approaches rely on short-term or episodic semantic representations and lack mechanisms for persistent spatial memory or principled integration of semantic information into long-term exploration decisions. Furthermore, the dynamic and partially known nature of real-world environments necessitates efficient search strategies that balance exploration of unknown areas with exploitation of previously acquired knowledge [16–19].

These challenges motivate the development of a unified framework that bridges geometric exploration with semantic scene understanding by balancing semantic frontier-based ex-

ploration with long-term semantic memory, enabling autonomous agents to perform open-vocabulary, goal-directed exploration guided by high-level semantic input.

1.1 Language-Guided Exploration

Traditional geometric exploration techniques are widely used for mapping unknown environments by identifying frontiers in either two or three dimensions and navigating toward the frontier with the highest expected information gain [13, 20]. Such methods, including those based on occupancy grids [21] or point cloud representations [22], are particularly effective for coverage and mapping tasks and have been successfully extended to multi-robot systems for large-scale exploration [23]. However, these approaches remain primarily geometry-driven and do not incorporate semantic understanding of the environment. Consequently, they are sub-optimal for goal-directed exploration tasks, where the objective is to locate specific objects or regions of interest defined by high-level semantic criteria rather than unexplored geometry [15].

To address this limitation, recent research has focused on integrating semantic perception into exploration frameworks [8–11, 14, 24–26]. Instead of relying solely on LiDAR or depth sensors for geometric mapping, the use of RGB imagery enables semantic reasoning about the scene and the objects contained within it. Table 1 summarizes representative works that leverage either pretrained VLMs or learning-based navigation policies trained via Behavioral Cloning (BC) or Reinforcement Learning (RL), to guide robots toward regions that are semantically relevant to a given target description in natural language.

Approach	Training Required	Real-Time	Semantic Reasoning Model
VLFM [9]	X (zero-shot)	✓	BLIP-2 [27] + GroundingDINO [28] + SAM [29]
SemUtil [8]	X (training-free)	✓	Mask R-CNN [30] + CLIP [4] + BERT [31]
ESC [10]	X (zero-shot)	✓	GLIP [32] + DeBERTa [31] / ChatGPT reasoning
LGX [24]	X (zero-shot)	✓	GPT-3 [3] + GLIP [32] + BLIP-2 [27]
CoW [14]	X (zero-shot)	✓	CLIP similarity scoring [4]
ZSON [11]	✓ (Reinforcement Learning pretraining)	✗	CLIP-based RL policy
PONI [25]	✓ (supervised)	✗	Learned potential-field network
PIRLNav [26]	✓ (Behavior Cloning + Reinforcement Learning)	✗	DINO-based CNN-RNN policy [33]

Table 1: Overview of semantic zero-shot and trained exploration approaches. All of these methods are capable of navigating to a goal described in natural language, however, they differ in zero-shot applicability to new scenes and real-time capability.

Approaches such as ZSON [11], PONI [25], and PIRLNav [26] employ deep reinforcement

learning (DRL) or supervised training to develop navigation policies capable of generalizing to unseen objects. Although these methods achieve promising results in simulation, they require extensive offline training and exhibit limited adaptability to previously unseen environments or object categories not encountered during training. In contrast, zero-shot methods such as Vision-Language Frontier Maps for Zero-Shot Semantic Navigation (VLFM) [9], SemUtil [8], ESC [10], LGX [24], and CoW [14] leverage pretrained VLMs to perform semantic exploration without additional training. These models enable real-time decision-making by exploiting semantic cues extracted from RGB imagery, guiding robots toward areas likely to contain the target object or region.

Some approaches, such as ESC [10] and LGX [24], further integrate LLMs for common-sense reasoning and high-level task interpretation, enabling a more contextual understanding of complex instructions. However, this comes at the cost of increased computational demand and potential inference latency, which limits their applicability on resource-constrained mobile platforms. While VLFM [9] achieves real-time performance, it relies on multiple computationally expensive foundation models (GroundingDINO [28], Segment Anything Model (SAM) [29], and Bootstrapped Language Image Pretraining 2 (BLIP-2) [27]) that require high-end GPUs with up to 16 GB of VRAM, which is impractical for embedded robotic systems.

Overall, these language-guided exploration methods primarily focus on short-term semantic reasoning and lack persistent memory. They do not maintain long-term storage or recall mechanisms for previously acquired semantic knowledge, leading to redundant exploration and reduced efficiency in multi-object and chained search tasks.

1.2 Language-Embedded Semantic Mapping

A language-embedded semantic map serves as a persistent spatial memory that jointly encodes the geometric structure of the environment and its semantic content. Semantic information can be incorporated either by associating discrete object classes [34], inferred by a visual perception backbone, with their spatial locations, or by embedding high-dimensional visual representations into a spatial map [17, 35, 36], such as a projected voxel grid. In the latter case, the visual embeddings capture semantic properties of objects and regions beyond fixed category labels [9, 17, 35]. This abstraction allows semantic information to be reused across tasks and time horizons, rather than being tied to a single perception or navigation episode [16].

Such semantic representations can be queried using natural language prompts, enabling robots to reason about the presence, distribution, and spatial relationships of objects or regions of interest based on high-level descriptions [10, 15]. By leveraging either object class information or continuous visual embeddings, a robot can guide its navigation toward areas with a high likelihood of containing a specified target [16, 18], thereby improving search efficiency and task success rates. Furthermore, language-embedded semantic maps can be combined with high-level reasoning modules, such as Large Language Models (LLMs), to infer object relationships, contextual cues, and action sequences required to accomplish more complex, multi-step goals [10, 24].

Maintaining such semantic representations persistently over time enables robots to exploit

past observations, recall previously detected objects, and avoid redundant exploration of already known regions. This form of long-term global memory improves navigation efficiency, scalability, and robustness in open-vocabulary, real-world environments [16, 19]. Table 2 summarizes representative works that incorporate persistent or memory-based semantic mapping to enhance exploration capabilities.

Approach	Training Required	Real-Time	Memory Representation	Exploration Integration
OneMap [16]	✗ (zero-shot)	✓	2D probabilistic feature field	✓ (frontier-based)
ConceptGraphs [17]	✗ (pretrained models)	✗	3D scene graph	✓ (LLM-planner)
SemExp [34]	✓ (RL + supervised)	✗	2D semantic occupancy map	✓ (learned policy)
GeFF [37]	✓ (ScanNet pretrain)	✓	Implicit 3D feature field	✗ (passive)
RayFronts [15]	✗ (foundation model)	✓	Hybrid voxel + ray field	✗ (planner-agnostic)
VLMaps [18]	✗ (pretrained LSeg/CLIP)	✗	2.5D open-vocab grid	✓ (frontier-compatible)
Pigeon [19]	✓ (RLVR fine-tune)	✓	Point-of-Interest snapshot memory	✓ (reasoning-aware)

Table 2: This table provides an overview of representative persistent or memory-based semantic mapping approaches, comparing their training requirements, real-time capability, underlying memory representations, and the extent to which semantic memory is integrated into exploration or planning.

methods such as SemExp [34], Pigeon [19], and GeFF [37] rely on offline training to construct persistent semantic representations. In the case of policy-learning approaches, this dependence on extensive training limits adaptability to previously seen environments and unseen object categories. Other approaches, such as Open-Vocabulary 3D Scene Graphs for Perception and Planning (ConceptGraphs) [17] and Visual Language Maps for Robot Navigation (VLMaps) [18], construct persistent open-vocabulary maps using pretrained foundation models but often require pre-mapping and lack real-time performance, which restricts their use in dynamic or large-scale settings.

While One Map to Find Them All (OneMap) [16] demonstrates real-time onboard operation on embedded platforms such as the Jetson Orin AGX, its overall map update rate of approximately 2 Hz may limit applicability to high-speed navigation scenarios. Moreover, the quality of the probabilistic semantic belief map remains sensitive to depth sensing noise, as feature localization uncertainty increases with distance, directly affecting the spatial accuracy of semantic information. Although OneMap incorporates a consensus-filtering mechanism combining CLIP-based similarity maps with open-vocabulary object detectors to mitigate false positives, its semantic reasoning ultimately relies on vision-language similarity, which remains inherently ambiguous under open-vocabulary conditions.

Learning Generalizable Feature Fields for Mobile Manipulation (GeFF) [37] provides a compact implicit 3D representation by distilling CLIP-aligned features into a neural field, enabling both geometric and semantic understanding. However, it requires pretraining on large-scale datasets such as ScanNet, limiting its direct generalization to arbitrary environments. RayFronts [15] introduces a hybrid 3D representation that combines voxel-based semantics with ray-based frontier expansion, offering real-time operation and high efficiency for open-set semantic search. Nevertheless, its computational complexity grows with environment size, and its planner-agnostic design prevents it from actively guiding exploration toward semantically relevant regions.

Table 3 summarizes the key limitations observed across existing semantic exploration frameworks. These gaps illustrate the need for a unified approach that combines zero-shot semantic understanding, persistent spatial memory, and real-time exploration to achieve robust, scalable autonomy in complex environments.

Limitation	Example Works	Implication
No persistent memory	VLFM [9], CoW [14], LGX [24], ESC [10]	No long-term fusion or recall; repeated exploration of known areas.
Offline training required	ZSON [11], PONI [25], PIRLNavi [26], SemExp [34]	Heavy RL/supervised training; poor adaptability to new scenes.
No explicit exploration-memory trade-off	OneMap [16], RayFronts [15], VLMaps [18]	Either passive mapping or short-term exploration; inefficient search.
No zero-shot exploration	CoW [14], LGX [24]	Detect novel objects but fail to explore unseen regions strategically.
Premapping needed	ConceptGraphs [17], GeFF [37]	Depend on pre-recorded data; not suited for online autonomy.
Limited robustness	PONI [25], Object Goal Navigation using Goal-Oriented Semantic Exploration (SemExp) [34], PIRLNavi [26]	Closed-set categories; fragile under real-world variation.
Low real-world applicability	ConceptGraphs [17], VLMaps [18]	High GPU cost or simulation-only evaluation; limited deployability on mobile robots.

Table 3: This table summarizes key limitations of existing semantic exploration frameworks, providing representative example works for each limitation and outlining their practical implications for autonomous navigation and exploration.

These observations reveal several fundamental challenges that are not yet adequately addressed by existing semantic exploration frameworks.

First, many approaches lack the ability to perform zero-shot exploration while maintaining a persistent and incrementally updated semantic representation of the environment [8–11, 14, 25, 26]. As a result, semantic information is often either discarded after individual navigation

episodes or requires pre-mapped environments, limiting applicability in unknown or changing scenes.

Second, a strong reliance on computationally expensive deep reinforcement learning or supervised training pipelines remains common. This dependence restricts adaptability to new environments and object categories and poses significant challenges for deployment on resource-constrained robotic platforms [11, 14, 25].

Third, existing methods struggle to robustly handle the reliability of semantic information during exploration. Semantic maps constructed from open-vocabulary perception are inherently noisy and uncertain [4, 38], and indiscriminate reliance on semantic memory can lead to inefficient navigation [18], as robots may pursue low-confidence or spurious cues instead of exploring informative regions. In such cases, poorly calibrated use of semantic memory may negate the potential benefits of semantic guidance [25].

Conversely, approaches that rely exclusively on semantic-driven exploration may overlook structurally plausible regions suggested by prior observations, resulting in reduced task success. This challenge is further amplified in dynamic environments, where previously stored semantic information may become outdated or misleading over time.

Fourth, real-world deployment introduces additional challenges related to computational efficiency, robustness to sensor noise, and environmental variability. Many current systems struggle to sustain reliable real-time performance under embedded hardware constraints, leading to increased latency or unstable inference behavior [9, 17].

Finally, several existing approaches rely on single-source detection or similarity pipelines, which are prone to false positives and semantic ambiguities under open-vocabulary conditions. The lack of multi-source semantic validation and memory-aware reasoning limits robustness and consistency during long-term autonomous operation [9, 14, 16, 24].

1.3 Scientific Contribution

This work contributes to the state of the art by introducing a hybrid semantic exploration framework that integrates zero-shot semantic frontier scoring with persistent 3D scene representation, enabling autonomous robotic search guided by open-vocabulary text queries. The proposed method combines real-time semantic reasoning during exploration with a long-term spatial memory, allowing the robot to dynamically balance between discovering new information and exploiting previously acquired knowledge.

Unlike previous approaches that focus exclusively on either geometric frontiers or static semantic maps, the proposed framework continuously fuses information from multiple semantic sources to maintain a unified, confidence-based semantic world representation. Adaptive weighting enables the robot to adjust its behavior between exploration and exploitation according to the reliability of recent observations and the stability of stored semantic memory.

The proposed method is evaluated with respect to how the quality and granularity of the underlying semantic information influence task success, navigation efficiency, and robustness. By systematically varying the trust between exploration and memory components, this thesis provides new insights into how semantic reasoning and persistent mapping can be effectively

combined for open-vocabulary, multi-object search in dynamic environments.

To evaluate the contribution of the proposed method, the following research questions are formulated:

- 1. How does integrating zero-shot semantic exploration and persistent 3D semantic mapping affect multi-object search performance and navigation efficiency compared to existing methods?**

Performance is quantified with respect to task success and path efficiency, measured through Success Rate (SR), Success weighted by Path Length (SPL), and Multi-Object Success Rate (MSR) relative to representative state-of-the-art systems such as OneMap [16], VLFM [9], and Pigeon [19].

- 2. How does the interaction between live exploration and accumulated semantic memory influence overall system performance?**

The weighting factor between exploration and memory is varied during graph node fusion to assess impacts on SR and SPL, identifying optimal trade-offs between reactivity and exploitation.

- 3. How robust is semantic navigation to variations in semantic memory granularity, and can exploration reduce the negative effects of coarse semantic representations?**

Navigation performance is evaluated across different levels of semantic retrieval granularity and compared balanced exploration against pure exploitation to assess robustness using SR and SPL.

- 4. How does multi-source fusion of detection confidence, semantic similarity, and memory confidence impact detection robustness and false-positive suppression during exploration?**

This question is evaluated by analyzing Precision, Recall, F1-Score, the Confusion Matrix, and Success Rate (SR) under different fusion weight configurations across COCO, open-vocabulary, and zero-shot object classes.

- 5. What is the computational footprint and real-world robustness of the hybrid framework?**

This aspect is assessed using Frames Per Second (FPS), GPU and CPU utilization, inference latency, and detection stability under sensor noise during physical deployment on a mobile robot.

1.4 Thesis Structure

This work is structured as follows. Chapter 2 describes the state-of-the-art in frontier-based exploration, Reinforcement Learning (RL)-based exploration, foundation-model-based exploration and persistent semantic mapping approaches. Chapter 3 describes the methods used, for hybrid semantic exploration, persistent 3D mapping, promptable zero-shot detection, and

multi-source fusion strategies. Chapter 4 details the practical implementation of the proposed approach, covering the simulation and real-world setup, datasets, software stack, and hardware configuration. Chapter 5 presents the experimental evaluation, including ablation studies, comparative benchmarks, and real-world validation. Finally, Chapter 6 concludes the thesis with a summary of findings, discussion of limitations, and suggestions for future research directions.

2 State of the Art

This work introduces a hybrid semantic exploration framework that combines open-vocabulary semantic perception with autonomous exploration and persistent spatial memory, which combines the fields of geometric exploration, vision-language-guided exploration, and semantic mapping based on the successes encountered by incorporating semantic information from large pretrained foundation models into traditional exploration and mapping pipelines. However, these novel semantic methods still rely on design and data representation patterns derived from classic geometric exploration and mapping methods.

Semantic exploration approaches differ fundamentally in how exploration behavior is generated. Some methods learn end-to-end navigation policies using reinforcement learning or imitation learning, where exploration strategies are implicitly encoded in a trained policy optimized via task-specific reward functions [11, 26]. Other approaches adopt modular architectures that integrate pretrained vision-language models with classical mapping and planning techniques [9, 14, 16].

Reinforcement learning and imitation learning approaches have demonstrated promising results in semantic object search tasks by training agents to navigate toward target objects based on high-level semantic cues [11], primarily within simulated environments. Their main advantage lies in avoiding hand-designed exploration heuristics, as complex behaviors can be learned directly from data through reward optimization [39]. However, such policies typically require extensive training on large datasets (e.g. Replica [40], Habitat [41]), exhibit limited generalization to unseen environments or object [11, 26] categories, and lack interpretability due to their black-box nature [39].

In contrast, modular approaches leverage semantic representations provided by large-scale pretrained VLMs to guide exploration decisions without task-specific retraining [15]. By combining explicit geometric mapping with semantic reasoning derived from foundation models, these systems can achieve zero-shot generalization to novel objects and environments while retaining interpretability and adaptability [9]. Several works extend this paradigm by constructing persistent semantic maps that retain knowledge of previously explored areas [16, 18], enabling more efficient multi-object search and the integration of high-level natural language instructions [17].

Finally, object detection and promptable vision-language models play a crucial role in enabling open-vocabulary semantic understanding for exploration tasks. Recent advances in grounding-capable detectors and segmentation models facilitate zero-shot object recognition based on text prompts, allowing robots to identify and localize previously unseen objects [28, 32, 42]. These models form the semantic foundation upon which hybrid exploration systems are built, enabling flexible object search in diverse real-world environments.

2.1 Frontier-Based Exploration

Frontier-based exploration is a classical and commonly used approach for autonomous mapping and navigation in unknown environments [12]. A *frontier* is defined as the boundary between known free space and unknown regions of the environment [12]. Frontier-based exploration relies on the principle that unexplored areas adjacent to known free regions provide the highest potential information gain. To identify such frontiers, the robot must maintain a global representation of the environment, typically an occupancy grid or voxel map, where each cell is classified as *free*, *occupied*, or *unknown*, based on sensor observations from Light Detection and Ranging (LiDAR) or RGB-D cameras. The robot then iteratively selects and navigates to frontiers to expand its knowledge of the environment.

Quin *et al.* [20] evaluated three commonly used frontier extraction methods that differ primarily in computational efficiency and scalability. The first approach, known as the *Naïve Active Area (NaïveAA)* method, evaluates every cell in the occupancy grid to determine whether it is free and has at least one unknown neighbor. Although this approach is conceptually simple and accurate for small-scale maps, it becomes computationally expensive for larger environments and often produces small, fragmented frontier clusters.

The second approach, the *Wavefront Frontier Detector (WFD)* [2, 20], improves efficiency by using a breadth-first search (BFS) to identify connected frontier regions without exhaustively scanning the entire map. Unlike the NaïveAA method, WFD directly extracts continuous frontier clusters rather than treating each frontier cell individually, significantly reducing redundant computations.

The third method, the *Frontier-Tracing Frontier Detection (FTFD)* [20], further enhances performance by incrementally updating frontier information using only the most recent sensor observations. Instead of re-evaluating the full map, FTFD initiates a BFS from previously known frontier cells that remain within the active area and from the endpoints of the latest sensor rays. Newly visible free-space cells along the scan boundary are evaluated as potential frontiers, while outdated frontier cells that are now occupied or re-observed are removed. By restricting computation to the local scan perimeter, FTFD achieves significantly faster update rates than NaïveAA and WFD, supporting real-time frontier detection even in large-scale environments.

After frontiers have been extracted, a selection strategy determines which frontier the robot should explore next. Simple heuristics such as *nearest-frontier selection* minimize travel distance but can lead to oscillatory behavior between nearby frontiers. Alternatively, selecting the *largest frontier* favors unexplored regions of higher spatial extent, reducing dead-end visits but increasing traversal cost. To address these trade-offs, Bourgault *et al.* [13] introduced a *utility-based frontier selection* framework that combines multiple criteria, such as distance, frontier size, and expected information gain, into a unified objective function. This approach enables more balanced decision-making, improving overall exploration efficiency and map completeness. Many subsequent works have built upon this foundation, incorporating additional factors such as energy consumption, obstacle density, and dynamic environment considerations into the utility function [12].

However, all these methods are primarily designed for geometric exploration without incorporating semantic understanding. As a result, they are optimized for complete map coverage

rather than goal-directed exploration tasks. In scenarios where a robot must locate specific objects or regions based on semantic cues, purely geometric frontier selection often leads to inefficient search behavior and unnecessary traversal. This motivates the integration of semantic reasoning into the frontier-based exploration process, where frontiers can be prioritized not only by geometric utility but also by their semantic relevance to the task objective.

2.2 RL-based Semantic Exploration

In contrast to geometric exploration, which aims to maximize map coverage using the shortest possible path and time, semantic exploration focuses on efficiently locating specific objects or regions of interest described in high-level semantic terms. The objective is to minimize path length and exploration time while prioritizing areas likely to contain relevant targets rather than achieving complete spatial coverage.

Semantic exploration approaches differ fundamentally in how exploration behavior is generated and optimized. RL and imitation learning methods have demonstrated promising results by training agents to navigate toward target objects based on high-level semantic cues [11, 25, 26]. In these approaches, an agent learns a policy that maps high-dimensional sensory observations to low-level control actions by optimizing a task-specific reward function, rather than relying on explicitly designed exploration heuristics [39].

In the context of semantic exploration, RL-based methods are typically formulated as Markov Decision Processes or, more commonly in embodied navigation, as partially observable Markov Decision Processes (Partially Observable Markov Decision Processs (POMDPs)) [21]. At an abstract level, a POMDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where \mathcal{S} denotes the latent environment states, \mathcal{A} the set of possible actions (e.g., move forward, turn left/right), $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ the state transition probability function, $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, and $\gamma \in [0, 1]$ the discount factor. The agent does not observe the full state directly but instead receives high-dimensional, partial observations (e.g., RGB images, depth measurements, or semantic representations), from which it must infer an internal belief about the environment [39].

The reward function encodes task objectives such as reaching a target object, minimizing path length, or maintaining correct orientation, thereby implicitly shaping the agent's exploration behavior [39]. This paradigm has been successfully applied to semantic object-goal navigation, where agents are trained to locate objects specified by high-level semantic descriptions (e.g., object categories or language embeddings) [11, 25, 26].

A key advantage of RL-based semantic exploration lies in its flexibility. Complex navigation behaviors can be learned directly from interaction data without manually designing exploration strategies. However, this flexibility comes at the cost of extensive training requirements, limited interpretability, and reduced robustness to domain shifts, as policies often overfit to the visual statistics and dynamics of the training environments [11, 39]. As a result, many such approaches are primarily evaluated in simulation and struggle to generalize to previously seen scenes, object appearances, or sensor configurations. Commonly, RL algorithms are optimized via policy gradient methods [43], which directly adjust the policy parameters θ to maximize the

expected cumulative reward $J(\theta)$:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | o_t) R(\tau) \right], \quad (1)$$

where τ denotes trajectories sampled from the policy π_{θ} , a_t and o_t are the action and observation at time t , respectively, and $R(\tau)$ is the cumulative reward obtained over trajectory τ .

Majumdar et al. [11] proposed Zero-Shot Object Navigation (ZSON), a zero-shot object navigation framework that leverages the shared embedding space of CLIP [4] to guide navigation policies trained via RL. During training, the agent observes RGB images and previous actions, while the navigation goal is specified by the CLIP embedding of an image containing the target object. The policy is optimized using a shaped reward function:

$$r_t = r_{\text{success}} + r_{\text{angle-success}} - \Delta d_{tg} - \Delta a_{tg} + r_{\text{slack}}, \quad (2)$$

where r_{success} rewards successful target localization, $r_{\text{angle-success}}$ encourages correct orientation, Δd_{tg} and Δa_{tg} penalize distance and angular deviation, respectively, and r_{slack} promotes efficient navigation. At inference time, the image-based goal embedding is replaced by the CLIP embedding of a textual object description, enabling zero-shot generalization to unseen object categories. The action space is discrete and consists of *move forward*, *turn left*, *turn right*, and *stop*. The reward function is used exclusively during training to learn the navigation policy during inference, the agent selects actions solely based on the learned policy without access to the reward signal.

While ZSON demonstrates strong zero-shot object generalization, exploration behavior remains implicitly encoded in the learned policy, which makes it difficult to interpret or adapt to new scenarios. As a consequence, the agent tends to revisit visually familiar regions and lacks explicit mechanisms for systematic exploration of unknown space. Furthermore, the end-to-end RL formulation reduces interpretability and necessitates retraining when visual conditions or environment layouts deviate from the training distribution. Nevertheless, ZSON represents a significant step toward flexible and generalizable semantic exploration by integrating vision-language models with reinforcement learning.

Ramrakhy et al. [26] introduced Pretraining with Imitation and RL Finetuning (PIRLNav), a two-stage framework that combines behavior cloning (Behavioral Cloning (BC)) with RL to improve generalization in semantic navigation tasks. In the first stage, a navigation policy is pretrained via imitation learning on large-scale human demonstrations, learning to reproduce expert actions from observations that include RGB images, pose information, and a categorical goal representation. This pretraining stage provides a strong initialization, which reduces the training time and sample complexity required for subsequent reinforcement learning. Furthermore, with respect to interpretability, the use of BC allows for some insight into the learned behavior, as it is directly derived from human demonstrations. The pre-trained ObjectNav human-demonstrated policy is trained using supervised learning to minimize the cross-entropy loss between predicted and expert actions:

$$\mathcal{L}_{\text{BC}} = - \sum_{t=1}^T \log \pi(a_t^* | s_t), \quad (3)$$

where a_t^* denotes the expert action at time t and s_t represents the agent’s observation. The ObjectNav policy is trained with a CNN+RNN Network, in which a pre-trained vision backbone, Self-Distillation with No Labels (DINO), extracts visual features from RGB images and feeds into the nav policy network along with pose and goal information.

In the second stage, the pretrained policy is fine-tuned using RL, where the objective is to maximize the expected discounted reward:

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=1}^T \gamma^{t-1} r_t \right]. \quad (4)$$

where τ denotes trajectories sampled from the policy π , r_t is the reward at time t , and γ is the discount factor. Although this hybrid training strategy improves sample efficiency and navigation robustness in simulation, the resulting policy remains a black-box model. It does not maintain an explicit semantic memory and requires retraining to adapt to new visual domains or sensor modalities.

To address the limited interpretability of end-to-end policies, Ramakrishnan et al. [25] proposed Potential Functions for ObjectGoal Navigation with Interaction-free Learning (PONI), a supervised, map-based semantic exploration framework. Rather than learning low-level actions, PONI predicts high-level exploration objectives in the form of potential fields defined over a partial semantic map. Two complementary potentials are estimated: an area potential that encourages exploration of unknown space, and an object potential that estimates proximity to the target category. Exploration decisions are derived by scoring geometric frontiers according to a weighted combination of these potentials:

$$U_t(f) = \alpha U_t^a(f) + (1 - \alpha) U_t^o(f), \quad (5)$$

where α explicitly controls the trade-off between exploration and exploitation. This formulation yields interpretable and stable exploration behavior and decouples high-level decision-making from low-level navigation, which is handled by a classical navigation planner.

However, PONI relies on dense semantic annotations and assumes a fixed, closed set of object categories encountered during training. [25] used Mask R-CNN [30] to generate semantic maps with 80 object classes from the COCO dataset [6], which was finetuned within the Gibson dataset [44]. As a result, it does not support open-vocabulary or zero-shot object search and remains sensitive to annotation noise and domain shifts.

RL-based and supervised semantic exploration methods demonstrate the feasibility of learning navigation behavior from semantic cues, but they exhibit recurring limitations that motivate alternative approaches. These include extensive training requirements, limited interpretability, closed-set semantics, lack of persistent semantic memory, and reduced robustness to domain changes. These structural shortcomings have motivated recent work toward modular exploration frameworks that integrate pretrained vision-language models, which are discussed in the following section.

2.3 Foundation-Model-Based Semantic Exploration

In contrast to reinforcement learning-based approaches, which derive navigation behavior through task-specific training, a second line of work leverages large-scale pretrained VLMs as semantic priors within modular exploration systems. These approaches shift the learning burden away from navigation policy optimization toward semantic perception and reasoning, enabling zero-shot generalization to novel objects and environments without task-specific retraining.

VLMs are large pretrained image-text models that learn joint representations of visual and linguistic data from massive web-scale datasets [4, 32]. Their core principle is to embed images and text into a shared latent space, in which semantically related visual and linguistic concepts are mapped to nearby representations. A VLM defines two embedding functions, $f_I(\cdot)$ and $f_T(\cdot)$, which map an image I and a text prompt T to a common embedding space \mathbb{R}^d :

$$\mathbf{e}_I = f_I(I), \quad \mathbf{e}_T = f_T(T), \quad (6)$$

where $\mathbf{e}_I, \mathbf{e}_T \in \mathbb{R}^d$ are high-dimensional feature vectors encoding semantic information. Text inputs are tokenized and processed using transformer-based language encoders, while visual inputs are decomposed into patches or regions and encoded by a vision backbone (e.g., Convolutional Neural Network (CNN) or transformer-based architectures), depending on the model design [1, 27, 32]. Semantic alignment between image and text embeddings is commonly quantified using cosine similarity, which measures the angular similarity between vectors in the shared embedding space (Equation 7):

$$\text{sim}(I, T) = \frac{\mathbf{e}_I \cdot \mathbf{e}_T}{\|\mathbf{e}_I\| \|\mathbf{e}_T\|}. \quad (7)$$

A higher similarity score indicates stronger semantic correspondence between the visual observation and the textual query. This representation enables open-vocabulary reasoning, as arbitrary object descriptions can be matched against visual observations without retraining, forming the foundation for zero-shot semantic perception in exploration tasks.

The defining characteristic of foundation-model-based exploration is the explicit separation between geometric navigation and semantic understanding. Geometric structure is typically handled by classical mapping and planning components (e.g., frontier-based exploration), while semantic relevance is inferred from pretrained models such as CLIP, BLIP-2, or grounding-capable detectors [9, 14, 28]. This modularity improves interpretability and adaptability, but introduces new challenges related to uncertainty handling, semantic consistency, and long-term memory.

Table 4 summarizes representative foundation-model-based exploration frameworks in five categories: the source of semantic signals, the mechanism used to fuse semantics with geometric exploration, the handling of detection confidence and uncertainty, the underlying semantic data representation, and the dominant failure causalities observed in practice.

Method	Semantic signal source	Fusion with geometry	Detection confidence handling	Semantic representation	Primary failure causality
ESC [10]	Object detections + LLM priors	Probabilistic frontier scoring (PSL)	Single-step confidence, no revision	Local symbolic semantic map	False positives amplified by reasoning priors
CoW [14]	Image-text similarity CLIP	No explicit geometric fusion	Threshold-based similarity	No explicit map	Oscillation and local minima near false positives
SemUtil [8]	Closed-set detections + CLIP similarity	Utility map over geometric frontiers	No confidence decay or belief update	Semantic point cloud + scene graph	Persistent corruption from misdetections
VLFM [9]	Dense image-text similarity BLIP	Semantic value-map fused with frontier map	Weighted averaging over observations	Episodic semantic value map	False positives persist across episode

Table 4: Design patterns and limitations of foundation-model-based semantic exploration frameworks.

The table compares how different methods obtain semantic signals, integrate them with geometric exploration, handle uncertainty over time, and represent semantic information, revealing recurring failure modes that motivate the need for persistent semantic memory and belief revision.

A representative example of this paradigm is the Exploration with Semantic Cues (ESC) framework proposed by Zhou et al. [10], which augments traditional frontier-based exploration with semantic cues derived from pretrained VLMs. Specifically, ESC combines a grounded object detector, Grounded Language-Image Pretraining (GLIP) [32], with a LLM (either ChatGPT or DeBERTa) to generate semantic priors that guide exploration decisions. Frontiers are scored based on both geometric utility and semantic relevance to the target object (see Equation 8).

$$P(F) = P(F \mid d_i^t, o^t, r^t) \quad (8)$$

$P(F)$ denotes the probability of selecting frontier F , conditioned on detected objects d_i^t , current image observations o^t , and the robot pose r^t at time t . This formulation is implemented using Probabilistic Soft Logic (PSL), which fuses visual detections with language-derived priors about object co-occurrences and spatial relationships. The robot then selects the frontier with the highest combined score, balancing geometric and semantic information to improve search efficiency. For navigating toward target objects, a classical A-star (A^*) planner is employed to compute collision-free paths based on the occupancy map.

Zhou et al. employs GLIP [32] as the detection backbone to compute 2D bounding boxes, class labels, and confidence scores for objects within the robot’s Field of View (FOV). While effective in simulation, the approach introduces notable computational overhead due to repeated LLM inference and PSL optimization. As a consequence of relying on single-step detections and static commonsense priors, errors introduced by false positives are not attenuated over time. Once a misleading semantic hypothesis is introduced, the probabilistic reasoning layer tends to reinforce rather than correct it, leading to persistent semantic bias during exploration.

Through ablation studies, Zhou et al. observed that object-object and object-room relational

priors can occasionally degrade performance, as commonsense relationships are inherently probabilistic rather than deterministic. Additionally, while ESC maintains a local semantic map during navigation, it lacks mechanisms for long-term memory or belief revision. Consequently, once an incorrect detection or prior is introduced, the framework has no learned means of down-weighting or correcting it over time, which can lead to persistent semantic inconsistencies during extended exploration.

In contrast to such LLM reasoning-based systems, Gadre et al. [14] proposed Clip on Wheels (CoW), a lightweight vision-language exploration framework that relies purely on image-text alignment from CLIP [4] without requiring explicit frontier detection, semantic mapping, or object segmentation. The method guides the robot toward directions with the highest cosine similarity (see Equation 7) between the current visual observation and the target object description.

By eliminating explicit detectors and handcrafted mapping, CoW offers a computationally efficient and conceptually simple baseline for open-vocabulary navigation. However, this simplicity comes at the cost of robustness. The system is highly sensitive to viewpoint variations and clutter, as cosine similarity does not always correlate with true object presence. Without spatial memory or geometric reasoning, the robot may oscillate near false positives or become trapped in local minima. Moreover, because similarity scores vary across object categories, no universal threshold can be established for all targets, resulting in inconsistent stopping behavior and reduced reliability during multi-object search.

Building upon this idea of integrating semantics into classical exploration, Chen et al. [8] introduced Semantic Utility Maps for Object Goal Navigation (SemUtil), a fully modular and training-free framework for object-goal navigation that combines classical SLAM-based mapping with pretrained perception and language models. In contrast to reinforcement learning or imitation learning approaches, SemUtil leverages explicit geometric and semantic reasoning through three core components: a 2D occupancy map for frontier extraction, a semantic point cloud generated by projecting Mask R-CNN detections into 3D space, and a spatial scene graph for high-level semantic reasoning. These three representations collectively form a structured scene model that supports geometric planning, semantic propagation, and reasoning about unexplored regions [8].

The central element of SemUtil is the *utility module*, which fuses geometric frontiers with semantic priors to determine the most promising frontier to explore next. For each map cell, a *utility score* is computed by combining the geometric frontier characteristics, the CLIP-based cosine similarity between the current observation and the target object description, and the semantic cues from the 3D point cloud (e.g., class IDs from Mask R-CNN). This results in a utility map that prioritizes frontiers both spatially and semantically, as illustrated in Figure 3 of the original paper (showing the interaction between geometric and semantic utilities) [8]. SemUtil solves the oscillation issues observed in CoW by explicitly extracting frontiers from the occupancy map and scoring them based on their semantic utility, rather than relying solely on raw similarity scores. This structured approach enables more stable exploration behavior and reduces the likelihood of becoming trapped near false positives, which also applies to other works, which combine VLMs with frontier-based exploration [9].

Importantly, the utility map in SemUtil is not persistent, it is recomputed at every timestep based solely on the current observation and semantic point cloud, without maintaining a long-term memory of past detections or map updates. While this design simplifies computation and eliminates the need for training, it also limits the system’s ability to reason over time or correct previous errors. The reliance on a closed-set detector (Mask R-CNN) restricts open-vocabulary generalization, and any incorrect detection directly corrupts the semantic point cloud, thereby distorting the frontier scoring and leading to suboptimal exploration decisions. Furthermore, since the framework lacks belief revision or memory-based fusion, false detections persist until they leave the robot’s current field of view, reducing consistency and efficiency in long-term navigation.

Unlike Chen et al. [8], who construct a utility map from class-based detections and semantic projections that are recomputed each step, thereby risking information loss or semantic inconsistency, VLFM [9] directly leverages pretrained vision-language models to compute semantic value maps from raw RGB observations. Rather than relying on symbolic object classes, VLFM computes a continuous image-text similarity score between the robot’s current observation and the target text prompt using BLIP-2 [27], as formulated through the cosine similarity function in Equation 7 [14].

The resulting similarity values are spatially projected onto a top-down occupancy grid according to the robot’s FOV, forming a *value map* that quantifies the semantic likelihood of each region leading toward the target object. To account for reduced reliability near the image periphery, a Gaussian weighting function attenuates confidence values based on angular distance from the optical axis (see Fig. 3 in the original paper [9]). This value map is continuously updated through a weighted averaging scheme that fuses new and previous similarity scores according to their confidence weights, enabling smooth map updates and spatial consistency across frames. BLIP-2 is not used for caption generation, but rather for retrieving text-image embeddings to compute similarity scores [9, 27].

During exploration, VLFM fuses this value map with a geometrically extracted frontier map to select the next exploration goal, the frontier with the highest semantic value is chosen as the next waypoint. Target object detection is performed using YOLOv7 [45] for COCO [6] categories and GroundingDINO [28] for open-vocabulary detection. Once an object matching the target query is detected, SAM [29] is applied to generate an accurate mask, and the system transitions from exploration to goal navigation.

This modular framework achieves state-of-the-art performance on the Gibson [44], HM3D, and Matterport3D [46] benchmarks, outperforming prior zero-shot approaches such as ESC, SemUtil, and CoW in both SR and SPL [9]. Despite its efficiency and interpretability, several limitations remain. VLFM relies on a single-source detection pipeline, either GroundingDINO or YOLOv7, making it prone to false positives in open-vocabulary scenarios, which can result in premature stopping behavior. Furthermore, the value map is episodic rather than persistent: it resets after each navigation episode and does not maintain long-term semantic memory, leading to redundant revisits during multi-object search tasks. While the system operates in real time, the use of multiple large-scale pretrained models (BLIP-2 [27], GroundingDINO/YOLOv7 [28, 45], and SAM [29]) demands substantial computational resources, con-

suming approximately 16 GB of VRAM on an NVIDIA RTX 4090 GPU during deployment, which limits scalability on embedded robotic platforms.

Foundation-model-based exploration approaches enable zero-shot semantic navigation without the need for task-specific training and offer improved interpretability compared to learned policies. However, across all reviewed methods, semantic information is either transient, episodic, or locally scoped. None of the approaches maintain a persistent semantic belief that can be revised over time as new evidence is accumulated. This lack of long-term semantic memory leads to repeated exploration, sensitivity to false detections, and inconsistent behavior in multi-object search scenarios.

These recurring limitations motivate the development of exploration frameworks that combine open-vocabulary semantic perception with persistent, revisable semantic memory, which is the focus of this work.

2.4 Map Reconstruction and Persistent Semantic Mapping

This section reviews state-of-the-art methods for persistent semantic mapping both exploration-driven and mapping-centric settings. Existing methods of saving semantic information within different spatial representations are discussed, along with techniques for updating and revising semantic beliefs over time, with the aim of improving efficiency in multi-object exploration tasks.

Persistent Semantic Mapping for Exploration

Table 5 summarizes recent approaches to persistent semantic mapping for exploration tasks. The table highlights how different methods store semantic information, update it over time, and whether they support belief revision, revealing a common lack of mechanisms for correcting erroneous semantic memories.

Method	Semantic Type	Memory	Spatial Representation	Open-Vocab	Update Mechanism	Belief Revision	Primary Limitation
OneMap [16]	Open-Vocabulary Belief Map (CLIP features)	2.5D top-down grid map		✓	Uncertainty-weighted accumulative fusion	✗	Irreversible belief fusion leads to semantic drift
VLMaps [18]	Dense per-cell language embeddings (CLIP-based [4])	2.5D top-down grid map		✓	Multi-view feature averaging (accumulative fusion)	✗	Irreversible feature averaging causes semantic noise and ambiguity
PIGEON [19]	Language-conditioned episodic visual memory (Point of Interest (PoI) snapshots)	2D geometric exploration map + episodic PoI memory		✓	Episodic accumulation of visual evidence (per episode)	✗	No persistent semantic belief state or belief revision mechanism
DualMap [47]	Dual semantic maps (short-term + long-term)	2D grid maps		✓	Dual-stream accumulative fusion	✗	Long-term map cannot correct early semantic errors

Table 5: Comparison of semantic memory representations for exploration. The table highlights how different methods store semantic information, update it over time, and whether they support belief revision.

A representative approach to building persistent open-vocabulary semantic maps during exploration is presented by Busch et al. [16] and Huang et al. [18]. Both methods construct a 2.5D top-down grid map in which semantic information is stored at the cell level in the form of language-aligned visual embeddings, enabling open-vocabulary querying via image-text similarity.

In both frameworks, incoming RGB-D observations are processed by a vision-language model to extract dense semantic features. In OneMap, global CLIP image embeddings are projected into the map using camera intrinsics and extrinsics, while VLMaps employs language-driven semantic segmentation (LSeg) to obtain dense per-pixel language embeddings [16, 18]. These features are associated with corresponding grid cells in the top-down map by back-projecting depth pixels into 3D space and discretizing them onto the 2.5D grid representation.

As the robot explores, newly observed embeddings are fused with existing map entries in an accumulative manner. Busch et al. perform uncertainty-aware recursive fusion, in which observations with higher confidence exert greater influence on the stored semantic representation, whereas Huang et al. apply multi-view feature averaging without explicit uncertainty modeling.

In both cases, once semantic features are integrated into the map, they are not selectively down-weighted or removed at later time steps.

Open-vocabulary object querying is performed by comparing stored map embeddings against the embedding of a text prompt using cosine similarity (see Equation 7). An object is considered detected if the similarity score in any map cell exceeds a predefined threshold. Because this decision relies solely on similarity values rather than explicit object detection or instance verification, both approaches are sensitive to threshold selection and may produce false positives in visually cluttered or ambiguous scenes.

Importantly, neither OneMap nor VLMaps incorporates mechanisms for belief revision or error correction. Once incorrect or noisy observations are fused into the map, they persist indefinitely and can bias future exploration decisions, leading to semantic drift over time [16, 18]. Furthermore, the projection of semantic information onto a 2.5D grid discards vertical structure and instance-level geometry, limiting semantic fidelity in complex environments and preventing accurate 3D object localization for downstream tasks such as manipulation or grasp planning [17].

[19] introduced *PIGEON*, a VLM-driven exploration framework that replaces persistent dense semantic maps with an episodic, object-centric memory abstraction. Instead of maintaining a globally consistent semantic map, PIGEON represents the environment as a set of semantically meaningful *Pols*, corresponding to geometrically salient observation locations enriched with visual context.

At each Pol, the robot stores a small set of Red Green Blue (RGB) observations captured from multiple viewpoints [19]. Semantic evaluation is deferred to query time, where a vision-language model jointly reasons over the language query and the stored RGB observations to assess the semantic relevance of each Pol. These relevance assessments are used to guide Pol selection, while low-level navigation between Pols is handled by a classical planner. Reinforcement learning is employed to fine-tune the VLM’s Pol selection behavior, rather than to directly select navigation actions.

Formally, each Pol p_i is defined by a spatial location x_i and an associated set of RGB observations $\mathcal{I}_i = \{I_{i,1}, \dots, I_{i,K}\}$ captured from different viewpoints at that location [19]. No semantic labels, object identities, or belief states are stored; semantic interpretation is performed on demand and is not consolidated into a persistent semantic world model.

In contrast to dense mapping approaches such as OneMap [16] and VLMaps [18], PIGEON does not perform accumulative fusion of semantic features into a persistent spatial representation. As a result, it is less susceptible to semantic drift caused by irreversible belief fusion, but does not support belief revision or long-term semantic consistency across multiple object-search tasks [19].

Methodologically, PIGEON occupies an intermediate position between dense semantic mapping and object-centric persistent representations such as DualMap [47] and Concept-Graphs [17]. It combines episodic visual memory with query-conditioned semantic scoring and reinforcement learning-based navigation, while deliberately avoiding persistent semantic state estimation.

Jiang et al. [47] introduced *DualMap*, an object-centric framework for online open-vocabulary

exploration that explicitly separates short-term perceptual observations from long-term semantic memory. Unlike dense feature-map approaches such as OneMap [16] and VLMaps [18], which store pixel-wise or cell-wise language embeddings, DualMap reasons over discrete object instances and their spatial relations.

At each timestep, objects are detected and segmented from the RGB image using YOLO-World [48]. For each segmented object, a visual embedding is computed from the cropped image using CLIP’s image encoder [4]. If a textual label is available, an additional text embedding is obtained from CLIP’s text encoder. The final object-level semantic representation is computed as a weighted fusion of image and text embeddings:

$$\mathbf{f}_t = \alpha \mathbf{f}_t^{\text{img}} + (1 - \alpha) \mathbf{f}_t^{\text{text}}, \quad \alpha = 0.7, \quad (9)$$

where $\mathbf{f}_t^{\text{img}}$ and $\mathbf{f}_t^{\text{text}}$ denote the image-based and text-based embeddings at time t , respectively. This object-centric representation enables open-vocabulary semantic reasoning without requiring dense per-pixel feature storage.

DualMap maintains two complementary semantic maps. The *local concrete map* stores recently observed object instances as 3D point clouds with associated semantic embeddings, enabling rapid adaptation to new observations. In contrast, the *abstract map* serves as a long-term semantic memory and stores only stable object instances, referred to as *anchors* (e.g., tables, desks, counters), which are unlikely to change location over time. Smaller or movable objects are treated as *volatile* and are not permanently stored in the abstract map [16].

Objects are added to or updated in the maps based on a combination of semantic similarity between embeddings and geometric overlap, measured via the 3D Intersection over Union (IoU) between observed point clouds. An object is updated from the local concrete map to the abstract map only when its confidence exceeds a predefined threshold, thereby balancing adaptability with long-term stability. Each anchor in the abstract map maintains a list of associated volatile objects that have been observed in its vicinity, allowing the system to reason about object co-occurrence without permanently storing potentially transient items [16].

During object-goal exploration, DualMap does not directly search the entire map for the target object. Instead, the language query is embedded using CLIP and matched against the semantic representations of anchors and their associated volatile objects. Anchors with high semantic relevance are prioritized as navigation goals, and the robot navigates toward them using a classical A* planner [49] on the occupancy map. Once the target object is detected again in the local concrete map, exploration terminates.

By separating short-term perceptual memory from long-term semantic anchors, DualMap achieves more structured and interpretable language-guided navigation than dense feature-map approaches. However, semantic information in DualMap is primarily used for exploitation of previously observed anchors, while exploration itself remains purely geometry-driven. The framework does not provide a mechanism or hyperparameter to explicitly balance semantic exploration and exploitation, as semantic reasoning is only applied after anchors have been formed [47]. As a result, DualMap favors semantic exploitation over semantic exploration, which can limit its ability to actively search for objects that have not yet been associated with existing anchors.

Semantic Scene Reconstruction

In order to incorporate persistent semantic memory into exploration systems, it is essential to build and maintain accurate 3D reconstructions of the environment enriched with semantic information. This section reviews state-of-the-art methods for semantic scene reconstruction, focusing on how different approaches represent spatial and semantic information, their ability to operate in real time, and whether they support object-centric reasoning and which foundation models they leverage. Table 6 summarizes key characteristics of recent semantic scene reconstruction methods.

Method	Representation	Foundation Model	Zero-Shot	Real-Time capability	Object-Centric
ConceptGraphs [17]	Points	OpenCLIP [50], SAM [29]	✓	✗	✓
	Points (dual map)	CLIP-based	✓	✓	✓
	NeRF	OpenCLIP [50], DINOv2 [52]	✗	✗	✗
	region-based TSDF	SEEM [53]	✓	✓	✗
	Voxel grid + semantic ray field	CLIP [4]	✓	✓	✗
	Points	CLIP [4], SAM2 [55], DINOv2 [52]	✓	✓	✓

Table 6: Comparison of semantic scene reconstruction methods. The table contrasts spatial representations, semantic abstractions, temporal update strategies, and the degree of online integration with SLAM systems. Zero-shot refers to deployment in unseen scenes.

Gu *et al.* [17] proposed *ConceptGraphs*, and Maggio *et al.* [56] introduced *Clio*, two object-centric semantic mapping frameworks that maintain persistent object-level representations enriched with open-vocabulary semantic information. Both methods represent the environment in an object-centric manner by extracting objects using open-vocabulary object detectors (e.g., GroundingDINO [28]) and segmenting them with SAM [29], yielding a bounding box and segmentation mask for each detected object. Semantic information is encoded using CLIP-based embeddings. While ConceptGraphs explicitly reconstructs per-object 3D geometry, Clio represents objects as spatially grounded semantic entities with associated embeddings and metadata. Clio additionally incorporates a detector confidence score into the object representation [56]. From this point onward, the two methods differ in how object-centric representations are stored, associated, and fused into a persistent semantic map.

ConceptGraphs [17] updates its semantic map by associating newly detected objects with existing ones if the DBSCAN-filtered 3D point cloud exhibits both high semantic similarity, measured using OpenCLIP [50], and sufficient 3D spatial overlap, quantified via IoU. If these con-

ditions are met, the new point cloud is aligned to the existing object using Iterative Closest Point (ICP) and merged, while semantic embeddings are aggregated over time. Otherwise, a new object node is added to the map. The resulting representation is a 3D scene graph, where nodes correspond to objects and edges encode spatial relationships (e.g., above, next to) derived from relative geometry. Each object is subsequently captioned using a large VLM to generate a human-readable description of its attributes and context [17]. A LLM then processes the set of graph nodes and their captions under a system prompt to refine object descriptions and infer higher-level relationships between objects, by connecting spatial edges between relevant nodes. This enables a range of downstream tasks, including question answering about the scene, robot manipulation, navigation, and localization.

Clio [56] employs a dual-level object-centric memory consisting of a frontend instance buffer and a backend abstract object memory. The frontend maintains a transient buffer of newly detected object instances, each associated with a spatial estimate, a CLIP-based semantic embedding, and a detector confidence score. These instance-level representations are updated at a high frequency and may contain redundant or noisy observations. The backend maintains a compact set of stable, task-relevant object abstractions that serve as long-term semantic memory. In Clio, promotion from instance-level observations to abstract object anchors is governed by a mathematically defined information bottleneck objective operating in semantic embedding space, which compresses redundant observations while preserving task-relevant information. This enables efficient querying by retaining only a small number of semantically meaningful object anchors [57]. During language-guided navigation, semantic reasoning is performed by matching the CLIP embedding of the language query against the abstract object representations, while low-level navigation is handled by a classical planner operating on the geometric occupancy map. While Clio improves efficiency by retaining only task-relevant object abstractions, this task-specific design limits map reusability, as querying non-task objects requires additional map reconstruction. As noted by Jiang et al. [47], DualMap addresses this limitation by adopting a hybrid segmentation strategy for holistic open-vocabulary mapping and by replacing costly inter-object merging with lightweight intra-object consistency checks, enabling persistent semantic memory and more efficient online operation.

Kerr et al. [51] introduced *LERF*, which builds upon Neural Radiance Field (NeRF) [58] to represent a scene as a continuous volumetric function that maps a 3D position \mathbf{x} and viewing direction \mathbf{d} to color and density, $f(\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c})$, learned from multi-view RGB images with known camera poses. While a standard NeRF reconstructs only geometry and appearance, LERF augments the radiance field with an additional language embedding output, $f(\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c}, \mathbf{e})$, where \mathbf{e} denotes a language-aligned semantic embedding. This embedding field is learned by distilling image-level semantic features into 3D during training. Specifically, CLIP [4] and DINO [33] embeddings are extracted from each training image at multiple spatial scales and projected into the NeRF via multi-view consistency, allowing each 3D location to store a stable, language-aligned semantic representation. This additional supervision enables dense 3D relevancy maps to be generated at query time by computing similarity between a text embedding and the learned language field. However, LERF requires extensive offline, scene-specific NeRF training and does not support incremental updates or online exploration, limiting its ap-

plicability to real-time or long-term robotic mapping scenarios.

Yamazaki *et al.* [35] presented *OpenFusion*, a real-time open-vocabulary semantic mapping framework that integrates volumetric Truncated Signed Distance Field (TSDF)-based reconstruction with region-level semantic perception from foundation models. OpenFusion processes incoming RGB images using Segment Everything Everywhere All at Once (SEEM) [53], a promptable VLM capable of zero-shot semantic segmentation based on text or image prompts. For each frame, SEEM [53] produces soft region confidence maps along with a semantic embedding vector for each region.

Depth images and camera poses are fused into a volumetric TSDF map to reconstruct scene geometry. Rather than directly storing semantic embeddings per voxel, OpenFusion associates voxels with lightweight semantic region identifiers with a dictionary mapping each region ID to its corresponding semantic embedding. To establish temporal consistency, the current TSDF map is raycast from the camera pose to render the accumulated semantic regions into the image plane. The rendered regions are then compared with the newly observed regions using geometric overlap and semantic similarity, and region correspondences are solved via a Jonker-Volgenant assignment algorithm [59]. Matched regions are fused by updating confidence scores, while unmatched regions are added as new semantic entries [35].

Semantic embeddings are stored in a global dictionary indexed by region identifiers, rather than per voxel, reducing memory consumption while enabling efficient semantic queries. At query time, a natural-language prompt is embedded using the same VLM, and cosine similarity \mathcal{J} is computed between the query embedding and the dictionary entries. Regions with the highest similarity scores are retrieved and localized via their associated TSDF geometry. This design enables real-time, open-vocabulary semantic mapping with efficient memory usage. However, OpenFusion does not maintain explicit object instances or support object-centric or instance-level semantic reasoning, which could potentially be addressed by additional voxel clustering strategies. Similar to DualMap [47] and Clio [56], OpenFusion follows a local-global fusion paradigm, where semantic information extracted from individual observations is incrementally associated with a global map representation. However, unlike DualMap and Clio, which explicitly maintain semantic abstractions across time, OpenFusion performs this fusion at the level of region-aligned geometry without constructing persistent object instances or multi-level semantic memory [35, 47, 56].

Alama *et al.* [15] introduced *RayFronts*, a real-time open-set semantic mapping framework designed to support both fine-grained semantic scene understanding within sensor range and semantic reasoning about regions beyond the depth perception limit. RayFronts represents the environment using a hybrid spatial abstraction consisting of a sparse voxel-based semantic map for observed regions and a set of semantic ray frontiers anchored at map boundaries for unobserved space.

Given posed RGB-D observations, RayFronts first extracts dense, language-aligned visual features using an efficient vision-language encoder based on RADIO [60] with a SIGLIP [61] adapter. Within the sensor range, depth measurements are fused into a sparse voxel grid, where each occupied voxel stores a persistent semantic embedding aggregated over time via a weighted averaging scheme. This lightweight fusion strategy prioritizes computational

efficiency and online operation, in contrast to more complex multi-stage or object-level fusion pipelines [17, 47, 56].

To reason about regions beyond the depth sensing horizon, RayFronts maintains a VDB-based occupancy map that encodes free, occupied, and unknown space. Three-dimensional frontiers are extracted as boundary voxels separating observed and unobserved regions. Instead of associating a single semantic descriptor with each frontier, RayFronts introduces semantic ray frontiers, in which multiple rays are attached to each frontier voxel. Each ray is parameterized by its origin, direction, and a language-aligned visual embedding, capturing semantic evidence observed along that direction in image space.

Semantic rays are discretized using angular bins and incrementally fused over time, allowing multiple distinct semantic hypotheses to coexist at the same frontier without feature collisions. This stands in contrast to VLFM [9], which maintains a single-query, episodic value map conditioned on a specific object prompt. This design enables multi-object and multi-query semantic reasoning in unobserved space and supports rough triangulation of distant semantic entities as exploration progresses. Importantly, both voxel and ray representations store task-agnostic visual embeddings rather than query-specific scores, allowing the semantic map to be queried at arbitrary times using text or image prompts via cosine similarity [15].

RayFronts is explicitly planner-agnostic and does not prescribe a specific exploration strategy. Instead, it provides a persistent semantic scene representation that can be consumed by downstream planners for object search, exploration, or navigation in large-scale and unbounded environments. By decoupling semantic mapping from planning, RayFronts enables flexible integration with a wide range of exploration and decision-making frameworks while maintaining real-time performance in outdoor settings [15].

Schwaiger et al. [54] introduced *OTAS* (Open-vocabulary Token Alignment for Outdoor Segmentation), a training-free semantic segmentation and reconstruction framework designed for unstructured outdoor environments. Unlike prior open-vocabulary mapping approaches that rely on object-centric segmentation priors, OTAS extracts semantic structure directly from intermediate token representations of frozen vision and vision-language foundation models.

OTAS uses DINOv2 [52] to generate dense visual embeddings at the image patch level, which capture visual similarity but are not language-aligned. To obtain language grounding, dense patch-level embeddings are extracted from CLIP [4], which are language-aligned but noisy and view-dependent. The core idea of OTAS is to cluster visually similar patches based on DINOv2 embeddings and align these clusters with CLIP [4] embeddings via masked average pooling, yielding language-grounded semantic regions without relying on object-centric segmentation [54].

Semantic queries are performed by embedding text prompts using CLIP’s text encoder and computing cosine similarity against the cluster-level embeddings. Optionally, a frozen mask refinement network such as SAM2 [55] can be used to upsample coarse relevance maps to pixel-level segmentations. For multi-view reconstruction, language-grounded features are projected into 3D using depth and camera poses and fused into a persistent point cloud representation, enabling open-vocabulary querying via cosine similarity. Unlike prior object-centric mapping approaches such as ConceptGraphs [17] and Clio [56], OTAS does not require explicit object

detection or instance segmentation, allowing it to capture amorphous or unstructured semantic entities commonly found in outdoor environments, such as vegetation, terrain types and natural landmarks [54].

2.5 Identified Research Gaps in State of the Art

Table 7 provides a comprehensive overview of state-of-the-art methods for open-vocabulary semantic exploration and mapping. The methods are categorized into four main groups: reinforcement learning-based approaches, foundation-model-based exploration, foundation-model-based exploration with persistent semantic mapping, and mapping-centric methods. The second column indicates the type of exploration strategy employed, while the third column highlights whether each method incorporates an explicit exploration-exploitation tradeoff mechanism, i.e., whether the balance between exploring new areas and exploiting known information can be controlled. The fourth and fifth columns denote whether the method supports zero-shot generalization to unseen environments and real-time operation, respectively. The final column indicates whether the method maintains a persistent semantic memory.

Method	Exploration Type	Exploration-Exploitation Tradeoff	Zero-Shot	Real-Time	Persistent Semantic Memory
ZSON [11]	Reinforcement Learning	✗	✗	✗(scene specific training)	✗
PONI [25]	Reinforcement Learning	✓	✗	✗(scene specific training)	✗
VLFM [9]	Foundation-model	✗	✓	✓	✗
OneMap [16]	Foundation-model	✗	✓	✓	✓
DualMap [47]	Foundation-model	✗	✓	✓	✓
RayFronts [15]	Foundation-model	✗(planner-agnostic)	✓	✓	✓
ConceptGraphs [17]	Mapping-centric	✗	✓	✗	✓
OpenFusion [35]	Mapping-centric	N/A	✓	✓	✓
OTAS [54]	Mapping-centric	N/A	✓	✓	✓
SAGE (this work)	Foundation-model	✓	✓	✓	✓

Table 7: Comparison of open-vocabulary semantic exploration and mapping methods. The table highlights differences in exploration strategy, semantic persistence, zero-shot generalization, and explicit exploration-exploitation control.

The main limitation of reinforcement learning-based approaches lies in their limited adaptability to unseen environments, where structural differences such as room layouts, object appearances, and lighting conditions deviate from the training distribution [11, 26]. These methods typically require extensive retraining or fine-tuning to generalize effectively, which limits their applicability in dynamic or real-world settings. Most importantly, RL-based approaches do not construct persistent semantic memory during exploration, preventing the reuse of acquired knowledge across tasks or environments unless such behavior is explicitly encoded during training [11].

Foundation-model-based exploration methods mitigate this limitation by leveraging pre-trained vision-language models to guide exploration in a zero-shot manner toward semantically relevant regions. VLFM [9] combines classical frontier-based exploration with semantic

value maps derived from image-text similarity, enabling efficient zero-shot navigation. However, the semantic representation in VLFM is episodic and reset after each navigation episode, preventing long-term semantic reasoning or refinement over time.

OneMap [16] addresses this limitation by accumulating semantic embeddings in a probabilistic 2D map, enabling persistent semantic querying across exploration episodes. While OneMap integrates semantic similarity maps with open-vocabulary object detectors through consensus filtering, the detector output acts as a hard gating mechanism. Consequently, object hypotheses are only formed when the detector confidence exceeds a predefined threshold, preventing the accumulation of weak but consistent semantic evidence across views when detector confidence remains low.

DualMap [47] improves robustness by introducing explicit object detection and an object-centric dual-map structure that separates short-term observations from long-term semantic anchors. While this design enhances semantic stability, both OneMap [16] and DualMap [47] primarily exploit previously observed semantic information and do not provide mechanisms to actively balance exploration and exploitation. Semantic reasoning is applied mainly after anchors have been formed, limiting their ability to guide early exploration.

RayFronts [15] extends semantic reasoning beyond observed space by introducing semantic ray frontiers, enabling reasoning about unobserved regions outside the depth sensing range. Although this representation supports persistent semantic querying and multi-object reasoning, the framework is explicitly planner-agnostic and does not define a concrete exploration strategy or an exploration-exploitation control mechanism.

Mapping-centric approaches such as ConceptGraphs [17], OpenFusion [35], and OTAS [54] focus on building rich semantic scene representations but do not address exploration behavior. These methods lack mechanisms for semantic-driven frontier selection or adaptive control of exploration based on uncertainty or memory reliability.

In summary, existing approaches either perform semantic exploration without persistent, revisable semantic memory, or construct persistent semantic maps without explicitly guiding exploration. No prior work jointly addresses zero-shot semantic exploration, persistent semantic memory, and an explicit, adjustable exploration-exploitation tradeoff within a unified framework. Furthermore, existing methods rely predominantly on single-source semantic signals, lacking multi-source confidence fusion to robustly suppress false positives during exploration. These identified research gaps motivate the development of SAGE, which aims to integrate these capabilities into a cohesive system for robust and efficient open-vocabulary semantic exploration.

3 Methods

This chapter presents the methodology of the proposed hybrid semantic exploration system, designed to mitigate the key limitations identified in Chapter 2, namely the absence of persistent semantic memory, the reliance on single-source semantic detections, and the lack of an explicit and controllable trade-off between semantic exploration and exploitation in existing approaches.

The proposed system integrates zero-shot semantic perception with frontier-based exploration and persistent 3D semantic mapping into a unified, closed-loop decision-making framework. Semantic evidence acquired during exploration is continuously fused into a long-term spatial memory, while exploration behavior is adaptively modulated based on the reliability of accumulated semantic beliefs.

An overview of the system architecture is provided in Section 3.1, followed by detailed descriptions of the core components: semantic frontier exploration (Section 3.2), promptable zero-shot detection (Section 3.4), persistent semantic 3D mapping (Section 3.3), the multi-source fusion strategy governing semantic belief updates (Section 3.5), and the behavior tree used to orchestrate semantic-guided exploration and navigation (Section 3.6).

3.1 System Overview

The proposed system follows a modular hybrid architecture that tightly couples semantic perception, geometry-driven exploration, and persistent semantic mapping to enable robust open-vocabulary object-guided exploration. Figure 1 provides a high-level overview of the system components and their data flow. The exploration task is specified by a user-provided natural language prompt, which defines the semantic target and conditions the detection, exploration, and fusion modules throughout system execution.

Robot observations at time t are represented as RGB-D measurements $O_t = \{I_t, D_t\}$, where I_t denotes the RGB image and D_t the depth map. The robot pose in the world frame is denoted by P_t . To reduce computational load, observations are temporally and spatially downsampled prior to further processing.

The pre-processed observations are then fed into three parallel modules. (a) The *Memory Module* takes as input the current observations O_t and pose P_t , obtained from the ROS 2 SLAM Toolbox [64], and updates the persistent semantic 3D map M_t . (b) The *Exploration Module* uses O_t and P_t to generate semantic frontiers F_t from the current exploration occupancy grid. (c) The *Detection Module* processes O_t to produce promptable zero-shot detections D_t^{det} .

Each module outputs a set of graph nodes representing semantic hypotheses. Specifically, the memory module produces memory graph nodes G_t^{mem} derived from the persistent map M_t , the exploration module outputs exploration graph nodes G_t^{exp} corresponding to semantic

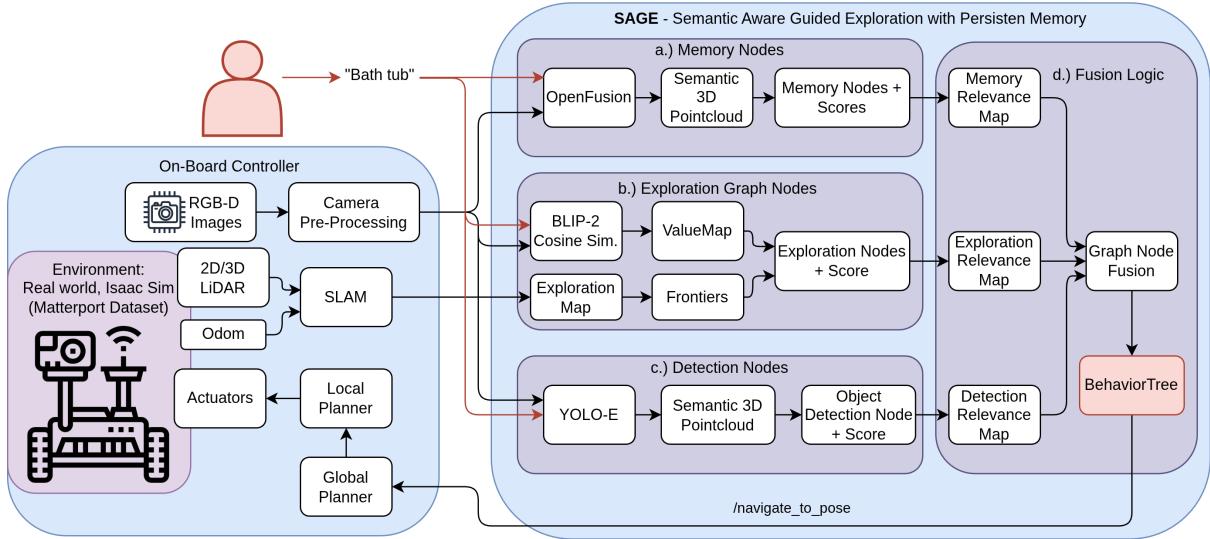


Figure 1: Overview of the SAGE system architecture for open-vocabulary semantic exploration. RGB-D observations and robot poses are processed by three parallel modules: (a) a persistent semantic mapping backend (OpenFusion [35]) that maintains a semantic 3D map, (b) a semantic frontier exploration module that scores geometric frontiers based on language-guided relevance, and (c) a promptable zero-shot detection pipeline for object hypotheses. All semantic hypotheses are represented as graph nodes, filtered by relevance maps, and fused using a multi-source fusion strategy. A behavior tree orchestrates exploration, verification, and navigation actions, while low-level motion planning and execution are handled by the ROS 2 Navigation Stack [62, 63].

frontiers F_t , and the detection module outputs detection graph nodes G_t^{det} obtained from D_t^{det} . This unified graph abstraction enables heterogeneous semantic hypotheses to be compared, filtered, and ranked using a common interface.

Prior to fusion, graph nodes are filtered using a relevance map to suppress nodes located in already explored areas. The remaining nodes are fused using the multi-source fusion strategy described in Section 3.5, resulting in a unified set of weighted graph nodes G_t^{fused} .

Finally, the behavior tree described in Section 3.6 selects the next high-level action based on G_t^{fused} , either navigating toward high-value frontiers for continued exploration or approaching detected objects for verification. Low-level motion planning, obstacle avoidance, and execution are handled by the ROS 2 Navigation Stack [62].

3.2 Semantic Frontier Exploration

Semantic frontier exploration extends classical frontier-based exploration by incorporating semantic relevance derived from vision-language models, enabling task-driven exploration guided by a user-defined semantic prompt. Instead of exploring unknown space uniformly, the robot prioritizes frontiers that are more likely to yield observations relevant to the target concept [2, 9, 15, 65]. Figure 2 illustrates the intermediate map representations and processing stages used to construct the semantic frontier map.

Exploration Occupancy Maps

The system maintains three distinct 2D occupancy grids for navigation and exploration: (a) a SLAM map used for localization and navigation [64], (b) an exploration map used exclusively for frontier detection, and (c) an inflated map that suppresses narrow passages and noisy frontier artifacts. This separation decouples stable navigation from task-specific exploration decisions and prevents semantic exploration logic from modifying the navigation map.

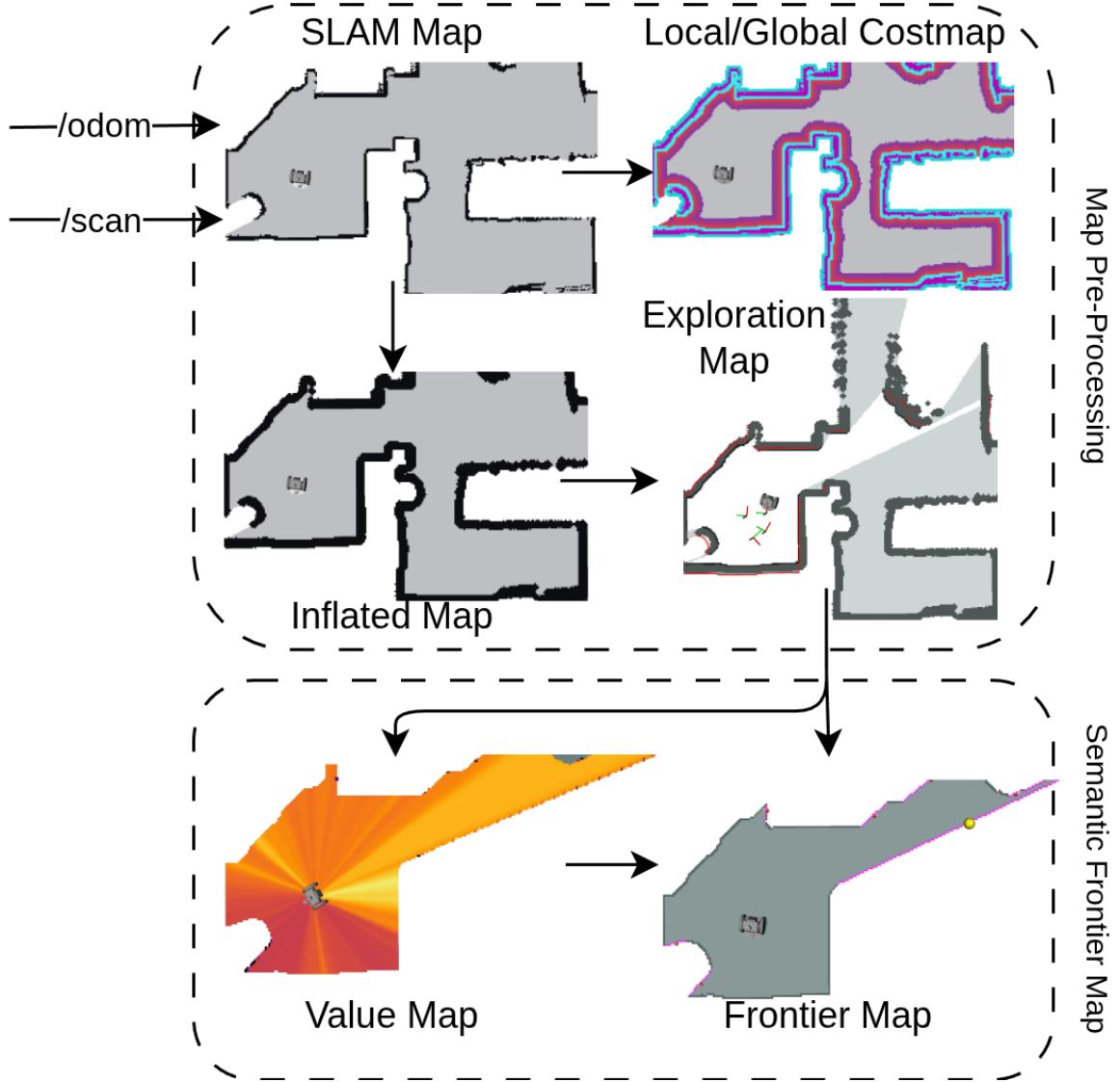


Figure 2: Overview of the map representations used for semantic frontier exploration. The SLAM map is used for localization and navigation, while the exploration map encodes task-specific explored and unexplored regions for frontier detection [64]. An inflated map is used to suppress narrow structures and reduce spurious frontiers. The resulting frontier map is combined with a semantic value map to prioritize exploration toward semantically relevant regions.

Rather than running a separate SLAM instance for exploration, the exploration map is derived directly from the SLAM occupancy grid [64]. Given the robot pose P_t and the SLAM map M_{SLAM} , free space is raycast from the robot into the occupancy grid, using the known sensor model and maximum sensor range, marking traversed cells as explored while preserving un-

known regions beyond sensor reach. This raycasting process is applied to all recorded robot poses accumulated during the current task, yielding an exploration map M_{exp} that reflects the explored workspace.

When a new semantic search task is initiated, all stored poses are cleared and the exploration map is rebuilt from scratch, while the SLAM map remains unchanged. This design ensures that exploration decisions are conditioned solely on task-relevant semantic information and prevents bias from previously explored but semantically irrelevant regions.

Frontier Detection and Calculation

Frontiers are defined as the boundary between known free space and unknown regions in the exploration occupancy grid [2] (see Figure 3). This work uses the algorithm outlined in Algorithm 1 to extract and cluster frontiers from the exploration map.

Let $\mathcal{G} \in \{-1, 0, 100\}^{W \times H}$ denote the exploration occupancy grid, where -1 represents unknown space, 0 free space, and 100 occupied space [64]. The set \mathcal{F}_t denotes the set of detected frontier clusters at time step t . Each frontier cluster \mathcal{P} is a set of spatially connected frontier cells. The set $\{\mathbf{c}_i^{t-1}\}$ contains the centroids of frontier clusters detected at the previous time step and is used to maintain temporal consistency via centroid matching.

Algorithm 1 Geometric frontier extraction and clustering from the exploration occupancy grid

```

1: function EXTRACTFRONTIERS(  $\mathcal{G}, N_{\min}, N_{\max}, d_{\text{match}}, \{\mathbf{c}_i^{t-1}\}$  )
2:    $\mathcal{F}_t \leftarrow \emptyset$                                       $\triangleright$  Output set of clustered frontiers
3:   for all cells  $(x, y)$  with  $\mathcal{G}(x, y) = 0$  do            $\triangleright$  Iterate over free cells
4:     if  $\mathcal{G}(x, y) = 0$  and any 4-neighbor is unknown then
5:       Mark  $(x, y)$  as frontier                          $\triangleright$  Free–unknown boundary
6:     for all unvisited frontier cells  $(x, y)$  do
7:       Grow a cluster  $\mathcal{P}$  using 8-connected BFS       $\triangleright$  Spatially connected frontier region
8:       if  $N_{\min} \leq |\mathcal{P}| \leq N_{\max}$  then
9:         Compute centroid  $\mathbf{c} \leftarrow \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \mathbf{p}$            $\triangleright$  Representative frontier location
10:        Assign frontier ID via nearest centroid match within  $d_{\text{match}} \in \mathbb{R}$ 
11:        if no match found then
12:          Assign new frontier ID                       $\triangleright$  Newly discovered frontier
13:        Add  $(\mathcal{P}, \mathbf{c})$  to  $\mathcal{F}_t$                    $\triangleright$  Store valid frontier
14:    return  $\mathcal{F}_t$                                  $\triangleright$  Set of clustered, tracked frontiers

```

The extracted frontier cells are clustered using an 8-connected Breadth-First Search (BFS) to group spatially contiguous regions. Clusters that fall within predefined size limits (N_{\min} and N_{\max}) are retained, while outliers are discarded. Each valid frontier cluster is represented by its centroid, which serves as the candidate exploration target. Let $\mathcal{P} = \{\mathbf{p}_j \in \mathbb{R}^2 \mid j = 1, \dots, |\mathcal{P}|\}$ denote the set of grid cell coordinates belonging to a frontier cluster. The centroid $\mathbf{c} \in \mathbb{R}^2$ is defined as the arithmetic mean of the spatial coordinates of all cells belonging to the frontier cluster. Frontier identity matching is performed using nearest-neighbor association in

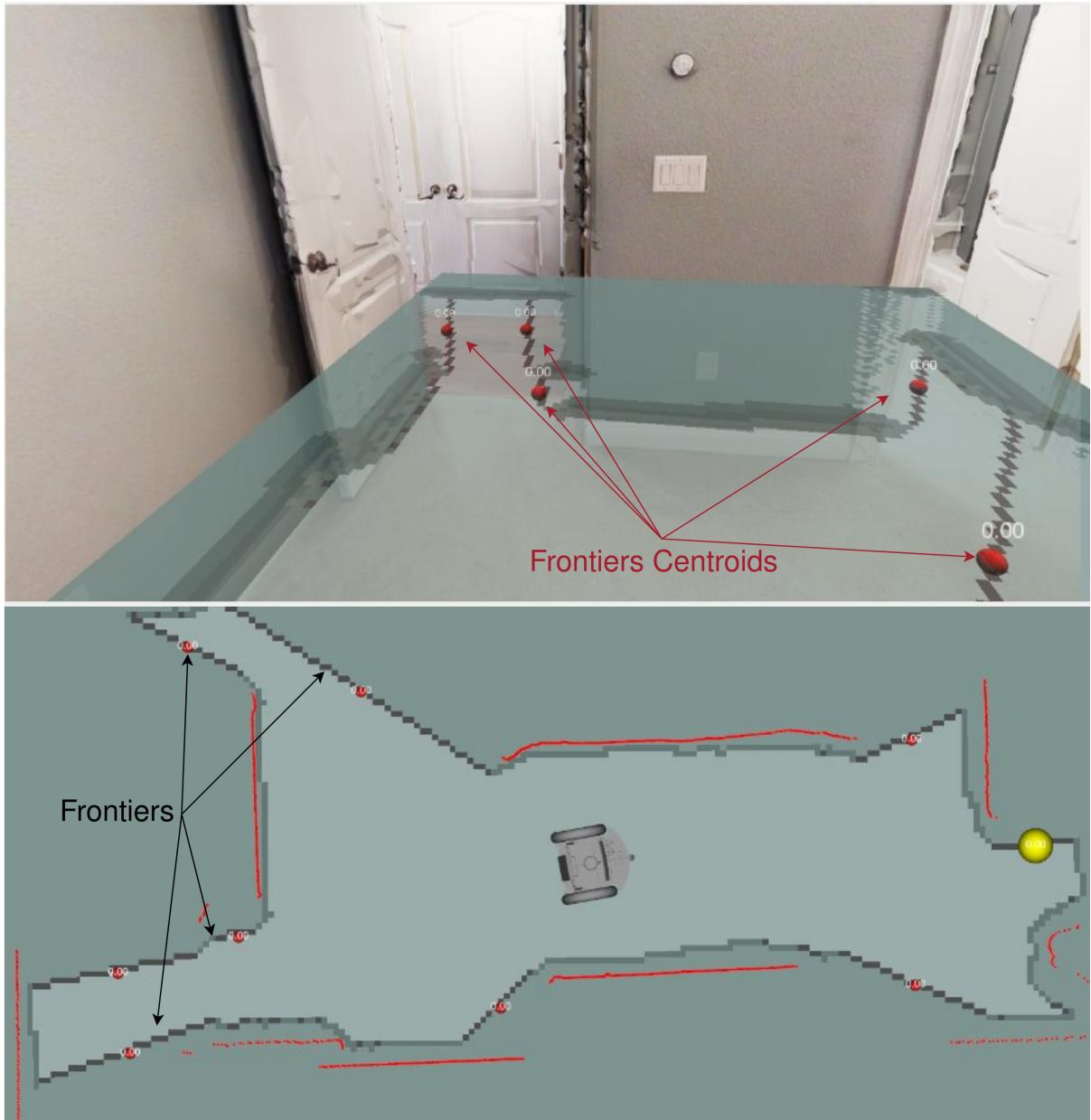


Figure 3: Example of frontier detection on the exploration occupancy grid. Frontiers are identified as free cells adjacent to unknown space and clustered into spatially contiguous regions, with centroids serving as candidate exploration targets.

Euclidean space, where a frontier centroid is assigned the ID of the closest previously observed centroid within distance $d_{\text{match}} \in \mathbb{R}$, otherwise a new frontier ID is created.

$$\mathbf{c} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \mathbf{p} \quad (10)$$

Equation 10 yields a single representative location that approximates the geometric center of the frontier region. This centroid is used as the navigation target for frontier-based exploration and as the reference position for semantic scoring.

The frontier centroids are tracked over time by matching them to previously detected frontiers

based on spatial proximity, allowing for consistent identification of persistent frontiers across time steps. To extract the semantically most relevant frontiers, each frontier is scored using the value map generated by the VLM as described below [9]. The scoring procedure is outlined in Algorithm 2 and Algorithm 3. Let $\mathcal{C} = \{(x_i, y_i, s_i)\}$ denote the semantic value map represented as a set of 2D cells with associated semantic scores s_i , obtained by temporally aggregating cosine similarity values between vision-language model embeddings and the user-defined text prompt (Section 3.2). For a value map cell $q \in \mathcal{C}$, $s(q)$ denotes the semantic similarity score stored at that cell.

Algorithm 2 Value Map Mask for scoring Frontiers

```

1: function GETSCOREFROMVALUemap(  $\mathcal{C}$ ,  $\mathbf{p}$  )
2:    $r \leftarrow 0.3$                                       $\triangleright$  Query radius
3:    $s_{\max} \leftarrow 0$ 
4:   observed  $\leftarrow$  false
5:   for all points  $q \in \mathcal{C}$  do
6:     if  $\|q - \mathbf{p}\|_2 < r$  then
7:        $s_{\max} \leftarrow \max(s_{\max}, s(q))$             $\triangleright$  Max semantic response
8:       observed  $\leftarrow$  true
9:     if observed then
10:      return (observed = true, score =  $s_{\max}$ )
11:    else
12:      return (observed = false)

```

The value map is a 2D grid in which each cell stores temporally aggregated cosine similarity scores (see Section 3.2) between VLM embeddings of scene observations and a user-defined text prompt. Let $\mathcal{V} = \{(x_i, y_i, s_i)\}$ denote the value map, where (x_i, y_i) are grid coordinates and $s_i \in \mathbb{R}$ is the associated semantic similarity score.

To score a frontier, its centroid $\mathbf{c} \in \mathbb{R}^2$ is projected into the value map coordinate frame. The semantic score of the frontier is obtained by querying the maximum similarity score within a fixed-radius neighborhood around \mathbf{c} , as described in Algorithm 2. If no valid score exists within this neighborhood, indicating that the region has not yet been observed, the frontier is marked as unobserved. Using the maximum response emphasizes strong semantic evidence while remaining robust to noise and partial observations, consistent with prior semantic frontier scoring approaches [9].

Algorithm 3 summarizes the construction of semantic frontier graph nodes by combining geometric frontier extraction with semantic scoring.

Algorithm 3 Construction of semantic frontier graph nodes

```
1: function UPDATESEMANTICFRONTIERS(  $\mathcal{G}, \mathcal{V}, \{\mathbf{c}_i^{t-1}\}$  )  
2:    $\mathcal{F}_t \leftarrow \text{EXTRACTFRONTIERS}(\mathcal{G}, N_{\min}, N_{\max}, d_{\text{match}}, \{\mathbf{c}_i^{t-1}\})$             $\triangleright$  Geometric frontier  
   extraction  
3:   for all frontier  $f \in \mathcal{F}_t$  do  
4:      $\mathbf{c} \leftarrow \text{centroid}(f)$                                  $\triangleright$  Representative frontier position  
5:      $(\text{observed}, s) \leftarrow \text{GETSCOREFROMVALUemap}(\mathcal{V}, \mathbf{c})$        $\triangleright$  Semantic value lookup  
6:     Create graph node  $n$   
7:      $n.\text{id} \leftarrow f.\text{id}$   
8:      $n.\text{position} \leftarrow \mathbf{c}$   
9:      $n.\text{score} \leftarrow s$   
10:     $n.\text{observed} \leftarrow \text{observed}$   
11:    Add  $n$  to graph                                      $\triangleright$  Semantic frontier node  
12:   Publish frontier graph                                $\triangleright$  For downstream task and visualization
```

The final step involves creating graph nodes for each frontier, encapsulating their ID, position, semantic score, and observation status. These semantic frontier graph nodes form the primary input to the fusion strategy and behavior tree described in Sections 3.5 and 3.6.

Value Map Generation using Vision-Language Models

The value map can be interpreted as an analogy to gradient ascent in deep reinforcement learning, where the robot seeks to maximize an expected semantic reward by navigating toward regions with high relevance to a target concept [9, 11, 14, 43]. In this work, the value map represents the slope of a semantic reward function. In contrast to classical gradient ascent, movement toward regions of high semantic relevance is constrained by obstacles and unknown space. Consequently, geometrically derived frontiers serve as feasible navigation targets that guide the robot toward high-value regions while ensuring safe traversal, thereby preventing convergence to unreachable local maxima [14]. Figure 4 illustrates this analogy, showing the semantic reward landscape and the role of frontier-based navigation (see Chapter 3.5 for the reward definition).

The value map is generated using a pre-trained vision-language model, specifically BLIP-2 [27], which is queried via a ROS 2 service to compute semantic similarity between visual observations and a user-defined text prompt. The value map node subscribes to RGB images, robot poses, and a global occupancy grid produced by a LiDAR-based SLAM system. This differs from prior semantic frontier approaches such as VLFM [9], which rely on depth camera projections and odometry-based local maps. By leveraging LiDAR-based SLAM, the proposed system maintains a globally consistent exploration map with reduced drift, enabling persistent semantic value accumulation over long trajectories. Figure 5 illustrates the overall value map generation pipeline.

Upon receiving an RGB image and a text prompt, the BLIP-2 service computes an image embedding and a text embedding using its pre-trained visual and textual encoders. The input

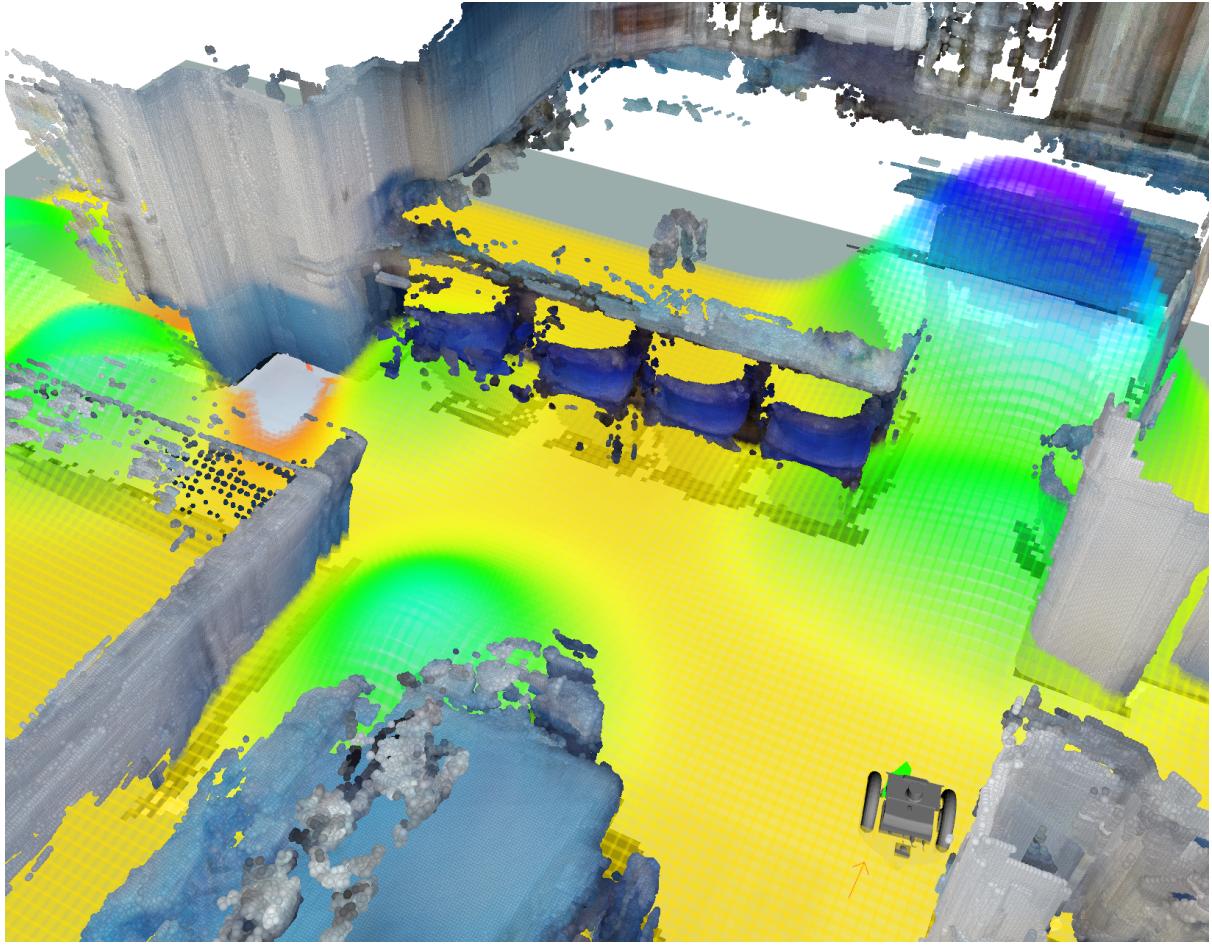


Figure 4: Conceptual visualization of the gradient-ascent analogy for semantic frontier exploration. Semantic relevance is illustrated as a folded reward surface, where frontier and memory graph nodes are elevated according to their semantic scores at position (x, y) . Geometric frontiers constrain feasible ascent directions under spatial obstacles.

image is divided into a grid of fixed-size patches, which are flattened, added with positional embeddings, and linearly projected before being processed by a vision transformer to capture spatial and semantic context [27, 66]. Similarly, the text prompt is tokenized into subword units, embedded, and processed by a language transformer to model semantic and syntactic relationships.

Therefore, both embeddings are projected using learned projection matrices W_I and W_T into a common embedding space (see Equation 11):

$$E_I = W_I f_I(I), \quad E_T = W_T f_T(T) \quad (11)$$

where $f_I(\cdot)$ and $f_T(\cdot)$ denote the visual and textual encoders, respectively. The projection matrices W_I and W_T are learned during pre-training using an Image-Text Contrastive (ITC) objective, which optimizes the cosine similarity between matching image-text pairs while pushing apart non-matching pairs [4, 27]. This training procedure aligns visual and textual representations in a shared semantic embedding space, enabling direct similarity comparison via cosine similarity. The projected embeddings are subsequently L2-normalized to unit length, which is

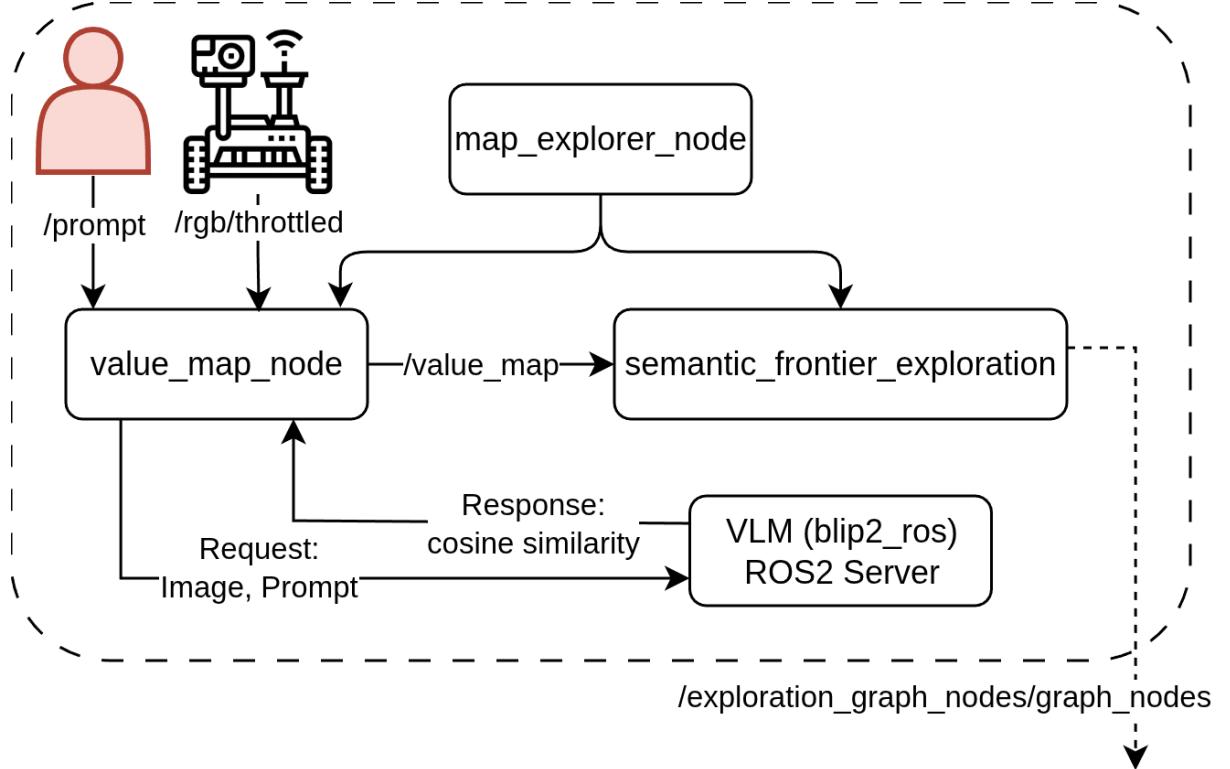


Figure 5: ROS 2 value map generation pipeline using the BLIP-2 vision-language model for computing image-text cosine similarity.

required for cosine similarity computation.

The cosine similarity score S between the normalized image embedding \hat{E}_I and text embedding \hat{E}_T is computed as:

$$S = \hat{E}_I \cdot \hat{E}_T = \frac{\hat{E}_I}{\|\hat{E}_I\|_2} \cdot \frac{\hat{E}_T}{\|\hat{E}_T\|_2} = \cos(\hat{E}_I, \hat{E}_T) \quad (12)$$

Figure 6 illustrates this image-text similarity computation, where visual observations are embedded by the BLIP-2 image encoder and compared against a user-defined text prompt in a shared semantic embedding space. Although cosine similarity is theoretically bounded in the interval $[-1, 1]$, in practice ITC-based VLMs yield similarity scores concentrated in a narrow positive range [4, 27]. This continuous score serves as a measure of semantic relevance between the current visual observation and the target concept.

Cosine Similarity to Value Map Projection

The computed cosine similarity score is integrated into a persistent 2D semantic value map in a pose- and visibility-aware manner. Let V_{t-1} denote the semantic value map and C_{t-1} the associated confidence map at time step $t - 1$, and let s_t be the cosine similarity score obtained from the VLM at the current robot pose p_t .

The update procedure, summarized in Algorithm 4, consists of three conceptual stages: (a) motion-gated temporal decay, (b) visibility-aware observation selection, and (c) confidence-

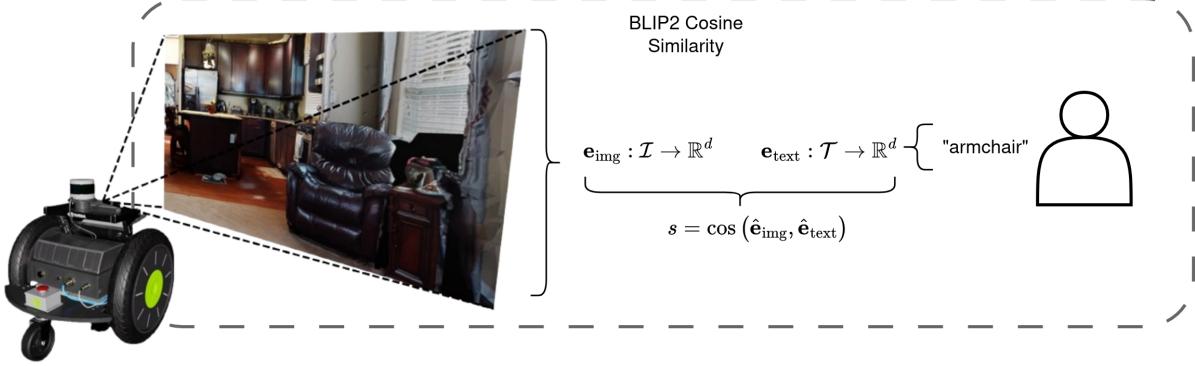


Figure 6: Image-text cosine similarity computation using the BLIP-2 ITC head. An RGB observation is embedded by the visual encoder and compared against a user-defined text prompt in a shared semantic embedding space [27].

weighted fusion. The following update procedure closely follows the semantic value map formulation proposed in VLFM [9], with adaptations for ROS 2 integration.

Algorithm 4 2D Value Map Update using Vision-Language Model Similarity Scores

```

1: function UPDATESEMANTICVALUemap(  $V_{t-1}$ ,  $C_{t-1}$ ,  $s_t$ ,  $\mathbf{p}_t$  )
2:   // (a) Motion-gated temporal decay
3:   if  $\|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2 > \delta_{\text{move}}$  then
4:      $V_{t-1} \leftarrow \lambda V_{t-1}$ 
5:      $C_{t-1} \leftarrow \lambda C_{t-1}$ 
6:   // (b) Visibility and observation confidence
7:   Compute visibility mask  $M_{\text{fov}}(\mathbf{p}_t)$  via raytracing
8:   Compute confidence map  $C_{\text{obs}}(\mathbf{p}_t, M_{\text{fov}})$ 
9:   // (c) Confidence-weighted fusion
10:  for all cells  $(i, j)$  with  $M_{\text{fov}}(i, j) = 1$  do
11:     $v \leftarrow V_{t-1}(i, j)$ 
12:     $c \leftarrow C_{t-1}(i, j)$ 
13:     $c_{\text{new}} \leftarrow C_{\text{obs}}(i, j)$ 
14:    if  $c + c_{\text{new}} = 0$  then
15:      continue
16:     $V_t(i, j) \leftarrow v + \alpha c_{\text{new}}(s_t - v)$ 
17:     $C_t(i, j) \leftarrow \max(\lambda c, c_{\text{new}})$ 
18:  return  $V_t$ ,  $C_t$ 

```

(a) Temporal decay is applied to both the value map and the confidence map only if the robot has translated more than a predefined threshold δ_{move} since the previous update. This motion-gated decay prevents repeated observations from dominating the map when the robot remains stationary or performs pure rotations, while still allowing outdated semantic evidence to fade over time when the robot explores new regions. The decay factor $\lambda \in [0, 1]$ controls the persistence of past observations and prevents oscillation between multiple similar frontiers by gradually reducing the influence of stale semantic evidence when the robot revisits similar

viewpoints [9].

(b) The set of map cells that can be updated at time step t is determined by computing a top-down visibility mask $M_{\text{fov}}(\mathbf{p}_t)$ using raytracing within the current field of view, instead of relying solely on depth camera projections based on odometry [9, 21]. Only cells that are geometrically visible from the robot's pose and not occluded by obstacles are considered for update. For these visible cells, an instantaneous observation confidence map $C_{\text{obs}}(\mathbf{p}_t, M_{\text{fov}})$ is computed, which models the reliability of the current observation as a function of the sensor geometry. Cells near the center of the FOV are assigned higher confidence, while confidence decreases toward the periphery due to reduced resolution and increased distortion. This behavior is modeled using a Gaussian weighting function over the angular deviation from the camera's principal viewing direction. The observation confidence assigned to a visible cell (i, j) is computed as

$$C_{\text{obs}}(i, j) = e^{-\frac{1}{2}\left(\frac{\Delta\theta(i, j)}{\sigma}\right)^2}, \quad (13)$$

where $\Delta\theta(i, j)$ denotes the angular difference between the viewing ray toward cell (i, j) and the robot's forward-facing direction, and σ controls the sharpness of the confidence decay within the FOV. Smaller values of σ result in a narrower high-confidence region centered around the optical axis, while larger values produce a more uniform confidence distribution. This confidence formulation follows the angular weighting strategy used in VLFM to model observation reliability across the FOV [9].

(c) For each visible cell (i, j) , the semantic value stored in the map is updated toward the current similarity score s_t using a confidence-weighted fusion rule. Let $V_{t-1}(i, j)$ and $C_{t-1}(i, j)$ denote the semantic value and confidence stored at cell (i, j) before the update, and let $C_{\text{obs}}(i, j)$ denote the instantaneous observation confidence computed from the current FOV.

The semantic value update is formulated as a weighted interpolation between the previous value and the current similarity score (see Equation (14)):

$$V_t(i, j) = V_{t-1}(i, j) + \alpha C_{\text{obs}}(i, j)(s_t - V_{t-1}(i, j)), \quad (14)$$

where $\alpha \in [0, 1]$ is an update gain that controls how strongly new observations influence the existing value map. Higher observation confidence leads to a stronger correction toward the current similarity score, while low-confidence observations have only a minor effect. In parallel, the confidence map is updated to preserve strong observations over time. Specifically, the confidence assigned to each cell is defined in Equation (15):

$$C_t(i, j) = \max(\lambda C_{t-1}(i, j), C_{\text{obs}}(i, j)), \quad (15)$$

where $\lambda \in [0, 1]$ is the temporal decay factor applied when the robot has translated since the previous update. Using a max operation ensures that regions which have been observed with high confidence remain influential even after decay, preventing repeated low-confidence observations from diluting reliable semantic evidence [9].

Together, Equations (14) and (15) implement a persistent, confidence-aware fusion mechanism that incrementally integrates semantic similarity scores into a spatially consistent value map.

Figure 7 illustrates a generated value map for the prompts “Bed”, “TV”, and the zero-shot prompt “A door leading to a bed”, demonstrating how prompt design can bias exploration toward multiple targets or semantically useful transition regions.

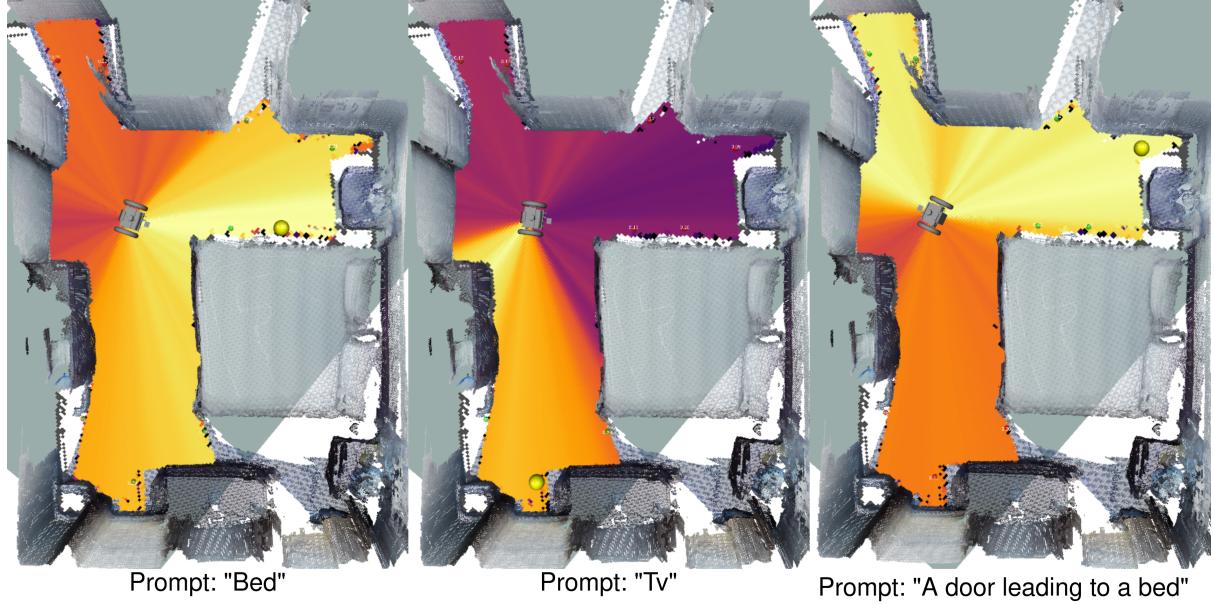


Figure 7: Example value maps generated using BLIP-2 for different text prompts. The value maps highlight regions of high semantic relevance to the prompts “Bed”, “TV”, and “A door leading to a bed”.

These scored frontiers are then encapsulated as graph nodes and passed to the fusion strategy (Section 3.5) for integration with memory and detection modules.

3.3 Persistent Semantic 3D Mapping

The semantic frontier centroids described in Section 3.2 offer short-term exploration targets based on immediate observations. However, to enable effective multi-object search over multiple objects, the system requires a persistent memory that retains information about previously observed objects and their spatial locations throughout the exploration task [16, 17, 47]. The proposed system incorporates a semantic 3D mapping module that constructs a global, persistent semantic representation from RGB-D observations and produces object-level hypotheses in the form of graph nodes. These memory graph nodes are subsequently integrated into the fusion strategy (see Section 3.5) to inform exploration and detection decisions.

At the time of implementation, OpenFusion [35] represented a suitable open-source framework for real-time, open-vocabulary semantic 3D mapping. Its balance between reconstruction fidelity and computational efficiency makes it suitable for onboard deployment on mobile robotic platforms [35]. As discussed in the state-of-the-art analysis (Chapter 2.4), OpenFusion enables zero-shot semantic mapping but does not provide an object-centric abstraction. This work extends OpenFusion with object-level semantic clustering and graph node generation, yielding a persistent semantic memory tailored to multi-object search and long-horizon exploration tasks.

Global Map Construction with OpenFusion

OpenFusion [35] was originally designed for offline semantic reconstruction from pre-recorded RGB-D sequences, such as ScanNet [67] and Replica [40]. In contrast, this work adapts OpenFusion for online operation on a mobile robot by integrating it with a LiDAR-based SLAM system that provides globally consistent pose estimates.

A dedicated ROS 2 wrapper node, referred to as `openfusion_ros`, interfaces OpenFusion with the exploration system (see Figure 8).

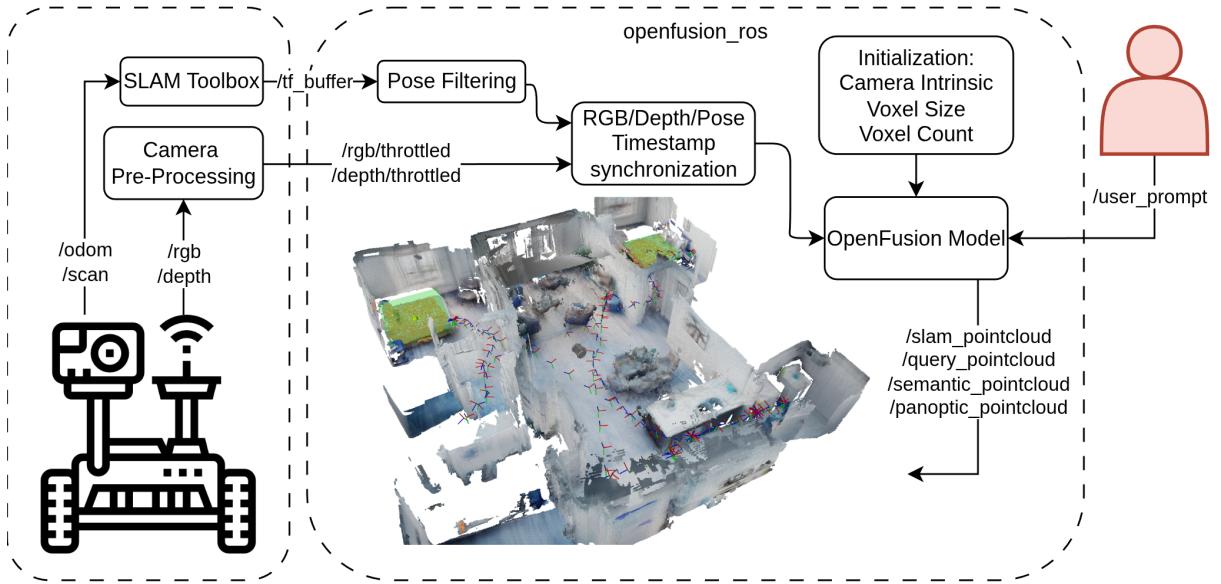


Figure 8: ROS 2 integration of OpenFusion for persistent semantic 3D mapping. RGB images, depth images, and LiDAR-based SLAM poses are synchronized and filtered before integration into a global TSDF volume. On-demand semantic queries generate semantic and panoptic point clouds, which are further processed to extract object-level hypotheses for the fusion strategy.

The wrapper subscribes to RGB images I_t , depth images D_t , camera intrinsics K , and SLAM-based robot poses P_t . To limit computational overhead and avoid redundant observations, RGB-D (I_t, D_t) frames are forwarded to OpenFusion only if the robot pose P_t differs sufficiently from previously integrated viewpoints. This pose filtering reduces redundant TSDF updates from near-identical viewpoints, mitigates over-integration artifacts, and ensures efficient use of the fixed voxel budget. Specifically, an input tuple $\{(I_t, D_t, P_t)\}$ is accepted only if the robot has translated or rotated more than a predefined threshold δ_{move} and represents a novel viewpoint relative to the existing map.

Upon initialization, OpenFusion is configured using the current camera intrinsics and a fixed voxel resolution and voxel budget, which together define the available memory for semantic mapping [35]. For each accepted RGB-D observation (I_t, D_t, P_t) , the data are fused into a global TSDF volume using the SLAM-based pose estimate, yielding a globally consistent 3D reconstruction with reduced drift compared to odometry-only approaches [21].

Each point in the semantic point cloud is associated with a cosine similarity score between the user-defined text prompt and the corresponding visual observation. The visual embeddings are obtained using SEEM [53], a pre-trained VLM optimized for dense, pixel-level representa-

tions. SEEM produces a high-dimensional embedding for each pixel in the RGB image, which is then compared to the text prompt embedding using cosine similarity. By projecting these pixel-wise similarity scores into 3D space via the aligned RGB-D observations, a semantic point cloud is obtained in which each point encodes its semantic relevance with respect to the query.

In practice, only the most semantically relevant predictions are retained by selecting a subset of high-confidence regions, controlled by a top- k selection criterion. This selection acts as a filtering mechanism rather than a strict object count, limiting the number of candidate regions considered for further processing.

Based on these high-confidence regions, SEEM performs instance-aware grouping, yielding a panoptic point cloud in which each 3D point is assigned both a semantic label and an instance identifier [53]. This panoptic representation allows spatially distinct object instances of the same semantic class to be differentiated. While OpenFusion provides the underlying 3D reconstruction and voxel management, semantic embeddings are computed externally using SEEM and fused into the 3D map via aligned RGB-D projections [35, 53].

These instance-level clusters are subsequently aggregated into object-centric memory graph nodes, which are consumed by the fusion strategy for long-horizon semantic exploration and object search.

Semantic Clustering and Graph Node Generation

The semantic point cloud produced by OpenFusion encodes dense geometric structure together with a per-point semantic relevance score derived from vision-language similarity [35, 53]. To extract object-centric hypotheses suitable for long-horizon exploration and persistent memory reasoning, a dedicated clustering and aggregation stage is applied [17].

Prior to clustering, the semantic point cloud is spatially downsampled using a voxel grid filter with a fixed leaf size. This operation reduces sensor noise and computational complexity while preserving the coarse geometry of potential object instances [68, 69].

Let the semantic point cloud be defined as $\mathcal{P} = \{(\mathbf{x}_i, s_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^3$ denotes the 3D position of point i in the global map frame, $s_i \in [0, 1]$ its associated semantic similarity score, and N the total number of points. The objective of clustering is to partition \mathcal{P} into a set of spatially coherent clusters

$$\mathcal{C} = \{C_1, \dots, C_M\}, \quad C_k \subset \mathcal{P}, \quad (16)$$

where each cluster C_k corresponds to a single object hypothesis and M denotes the number of extracted clusters. Clustering is performed using Euclidean distance in 3D space [21, 69]. Two points \mathbf{x}_i and \mathbf{x}_j are assigned to the same cluster if $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \varepsilon_{\text{cluster}}$, where $\varepsilon_{\text{cluster}} > 0$ is a fixed spatial distance threshold controlling the maximum cluster extent. To suppress spurious noise clusters and overly large, diffuse regions, only clusters satisfying $N_{\min} \leq |C_k| \leq N_{\max}$ are retained, where $|C_k|$ denotes the number of points in cluster C_k and N_{\min}, N_{\max} are user-defined bounds.

Each retained cluster C_k is represented geometrically by its centroid (see Equation 17), which serves as the spatial location of the corresponding object hypothesis.

$$\mathbf{c}_k = \frac{1}{|C_k|} \sum_{(\mathbf{x}_i, s_i) \in C_k} \mathbf{x}_i, \quad (17)$$

To assign a semantic confidence score to each cluster, the per-point similarity scores must be aggregated into a single representative value. Although the input point clouds already exclude background geometry due to semantic filtering by OpenFusion [35] and SEEM [53], the distribution of similarity scores within a semantic cluster remains highly non-uniform. Even within a single object instance, confidence varies significantly due to partial observations, occlusions, viewing angle effects, depth noise, and heterogeneous visual evidence across object surfaces. As a result, only a subset of points typically exhibits strong semantic alignment with the query, while the remaining points carry weaker but still relevant signals.

Therefore, a naïve aggregation using the arithmetic mean is insufficient for this setting. Within a semantic cluster, the mean is dominated by weakly informative points and systematically underestimates objects that are only partially visible or observed from suboptimal viewpoints. To obtain a robust estimate of semantic relevance, a percentile-based aggregation is employed instead [70]. Let $S_k = \{s_i \mid (\mathbf{x}_i, s_i) \in C_k\}$ denote the set of similarity scores within cluster C_k . The cluster-level semantic confidence is defined as the 75-th percentile of this set (see Equation 18):

$$\tilde{s}_k = \text{percentile}_p(S_k), \quad (18)$$

Selecting the 75th percentile emphasizes clusters for which a substantial fraction of points exhibits consistently high semantic similarity, while remaining robust to isolated spurious activations. Lower percentiles (e.g., the median) are overly conservative in the presence of localized but strong semantic evidence, whereas higher percentiles (e.g., the maximum) are sensitive to noise [70].

To further favor spatially consistent object hypotheses without allowing large clusters to dominate purely by size, the percentile score is modulated by a logarithmic cluster size factor (see Equation 19):

$$s_k = \tilde{s}_k \cdot \log(|C_k| + 1), \quad (19)$$

where the logarithmic term grows sublinearly with cluster size. In the semantic-only setting considered here, this factor reflects the spatial support of the semantic hypothesis rather than compensating for background clutter, thereby favoring compact objects supported by a coherent extent of evidence.

Finally, each cluster C_k is mapped to a memory graph node

$$n_k = (\mathbf{c}_k, s_k), \quad (20)$$

where \mathbf{c}_k denotes the spatial location of the object hypothesis and s_k its aggregated semantic relevance score.

3.4 Promptable Zero-Shot Detection

To enable goal-directed object search without prior knowledge of object categories, the exploration system integrates a promptable open-vocabulary object detection model. Such models leverage large-scale vision–language pre-training to localize objects specified by arbitrary user-defined prompts, enabling zero-shot generalization beyond a fixed training vocabulary [4].

Table 8 compares representative state-of-the-art detection and segmentation models with respect to their zero-shot capability, supported prompt modalities, output representations, foundation architectures, and suitability for real-time robotic operation.

Method	Zero-Shot	Prompt Type	Instance Segmentation	Bounding Boxes	Foundation Model	Real-Time
Grounding Dino [28]	✓	Text	✗	✓	DINO + BERT	✗
SAM [29]	✓	Visual	✓	✗	ViT	✗
SEEM [53]	✓	Text, Visual, Spatial	✓	✗	ViT + Text Encoder	✗
GLIP [32]	✓	Text	✗	✓	Swin Transformer + BERT	✗
OWL-ViT [71]	✓	Text	✗	✓	ViT + CLIP	✗
Mask R-CNN [30]	✗	None	✓	✓	ResNet + FPN	✗
YOLO-E [42]	✓	Text, Image	✓	✓	YOLO + CLIP [4]	✓

Table 8: Comparison of object detection and segmentation models with respect to zero-shot capability, prompt modality, output representation, and real-time suitability. Real-time capability is defined as achieving at least 10 Hz inference on a single GPU, which is sufficient for low-dynamic robotic exploration tasks.

For integration into a 3D semantic fusion pipeline, 2D detections must be lifted into 3D space using depth measurements and camera intrinsics. A common approach consists of projecting instance-segmented pixels into 3D and clustering the resulting point sets to obtain robust object hypotheses [72, 73]. Consequently, the detection model must provide instance-level segmentation masks, confidence scores, and zero-shot generalization.

Object detection approaches can be categorized into three classes. (a) Closed-vocabulary methods rely on a fixed set of object classes defined during training and cannot generalize to unseen categories, as exemplified by classical detectors such as Mask R-CNN [30] or YOLOv7 [74]. (b) Open-vocabulary methods extend detection to arbitrary categories speci-

fied via text prompts, enabling zero-shot recognition but typically lacking fine-grained instance segmentation [28, 32, 71]. (c) Promptable zero-shot models further generalize this paradigm by supporting multiple prompt modalities, such as textual descriptions and visual reference images, allowing context-aware and flexible object specification [42, 53].

Promptable segmentation models such as SAM [29] and SEEM [53] provide powerful instance mask generation capabilities from visual or textual prompts. However, these approaches rely on heavy transformer-based architectures with prompt-conditioned decoding, leading to inference latency that prevents bounded real-time execution within robotic control loops.

Related to the proposed object search system, VLFM [9] addresses zero-shot object search by integrating multiple perception modules rather than introducing a single unified detection model. Specifically, VLFM combines a closed-vocabulary detector for fast and reliable detections, an open-vocabulary grounding model that operates at a lower frequency to enable zero-shot detection, and a separate segmentation network to extract object centroids used for navigation toward the target. While this modular design facilitates flexible semantic reasoning, it requires multiple sequential inference stages, resulting in substantial computational overhead and limiting its suitability for real-time deployment.

In contrast, YOLO-E [42] unifies promptable zero-shot detection and instance segmentation within a single, efficient architecture. Building upon the YOLO framework [74] and incorporating CLIP-based vision-language alignment, YOLO-E achieves competitive detection accuracy while maintaining real-time inference. On the LVIS benchmark [75], YOLO-E reports average precision values of 35.2 and 33.7 for text and image prompts, respectively, surpassing GLIPv2 and GroundingDINO, which achieve 29.0 and 27.4 AP@. Moreover, the YOLOE-11-L variant employs only 26 M parameters for text prompting and 32 M parameters for image prompting, compared to over 170 M parameters for GroundingDINO and more than 230 M parameters for GLIPv2. With reported inference speeds of 130 Hz on an NVIDIA T4 GPU and 39.2 Hz on mobile phone hardware, YOLO-E satisfies the bounded-latency requirements of online robotic exploration.

These properties make YOLO-E particularly well suited for semantic exploration scenarios that require flexible object specification, instance-level spatial reasoning, and real-time operation under limited computational resources.

Open-Vocabulary Object Detection with YOLO-E

In this work, YOLO-E [42] is integrated as a promptable zero-shot object detection module within the exploration system. Incoming Red Green Blue-Depth (RGB-D) observations are preprocessed by throttling the input frame rate and resizing images to the required network resolution in order to reduce computational load while maintaining sufficient perceptual fidelity.

Given a user-defined text prompt and the current RGB image, YOLO-E [42] performs a single forward pass to produce a set of 2D bounding boxes, confidence scores, and corresponding instance segmentation masks for all objects matching the prompt (see Figure 9). For visualization purposes, an overlay image containing bounding boxes, class labels, and confidence

scores is generated. In addition, an instance ID mask is published to differentiate between multiple detected objects of the same category, and a per-pixel score mask is produced to assign semantic confidence values to segmented image regions.

Using the scored 2D segmentation mask, the instance ID mask, the aligned depth image, and the camera intrinsic parameters, detected object pixels are projected into 3D space (see Section 3.4). This projection yields a semantic point cloud in which each point $\mathbf{p}_i = (x_i, y_i, z_i)$ is associated with a semantic confidence value and an instance identifier. A subsequent clustering stage aggregates the semantic point cloud into object-level hypotheses by grouping spatially coherent points belonging to the same instance. For each cluster, a 3D centroid and a mean confidence score are computed.

The resulting 3D object hypotheses are encapsulated as detection graph nodes and forwarded to the fusion strategy for integration with the persistent semantic memory (see Section 3.5).

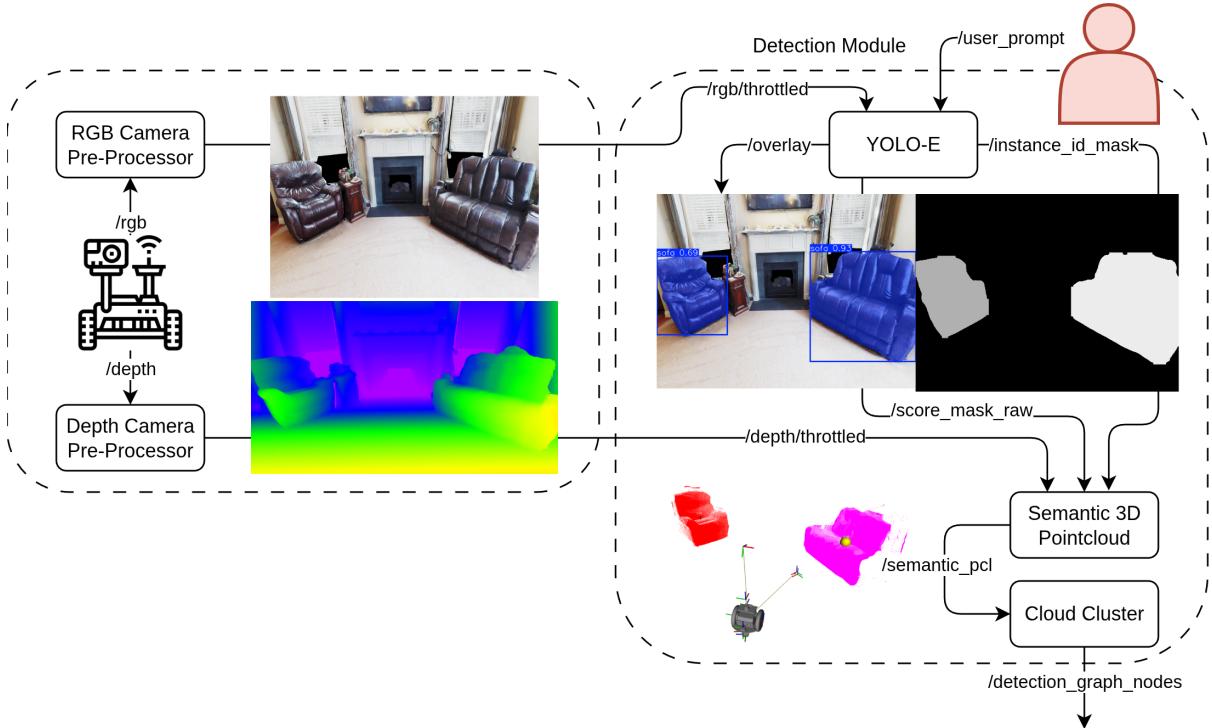


Figure 9: Promptable open-vocabulary object detection and 3D hypothesis generation pipeline using YOLO-E [42].

Depth-Based 3D Localization

Let a pixel (u, v) in the image plane be represented in homogeneous coordinates as $\tilde{\mathbf{p}} = [u, v, 1]^\top$, where u and v denote the horizontal and vertical pixel indices, respectively. Equation (21) defines the camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3 \times 3}$, parameterized by the focal lengths f_x, f_y and the principal point coordinates c_x, c_y [76]. These parameters are obtained by calibrating the RGB-D camera prior to deployment with standard calibration techniques [77].

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (21)$$

Given a metric depth value $z \in \mathbb{R}^+$ at pixel (u, v) , the corresponding 3D point in the camera coordinate frame is obtained via Equation (22), which is illustrated in Figure 10 [76].

$$\mathbf{P}_{\text{cam}} = z \mathbf{K}^{-1} \tilde{\mathbf{p}} = z \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{(u-c_x)}{f_x} z \\ \frac{(v-c_y)}{f_y} z \\ z \end{bmatrix} \quad (22)$$

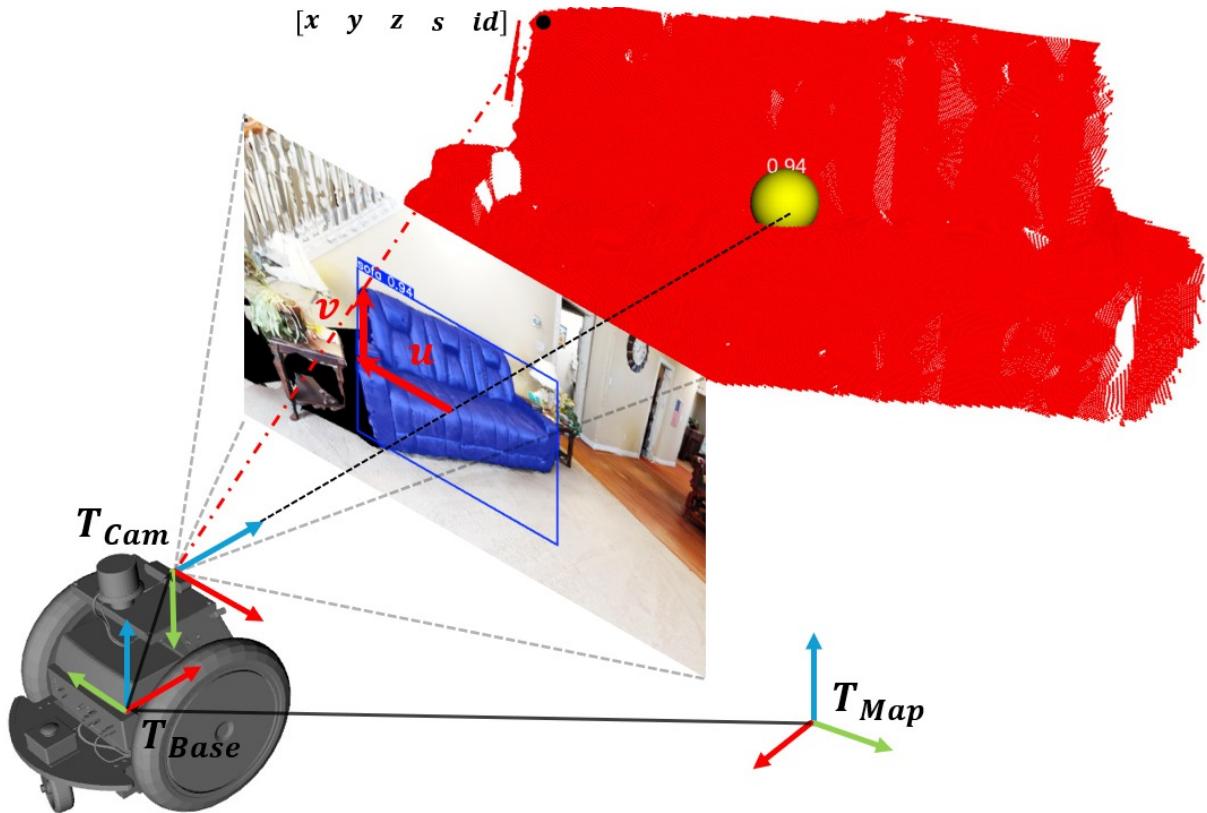


Figure 10: YOLO-E detection to graph node 3D localization

To express the 3D point in the global map frame, the homogeneous transformation $\mathbf{T}_{\text{map} \leftarrow \text{cam}} \in SE(3)$ is applied (see Equation (23)).

$$\mathbf{T}_{\text{map} \leftarrow \text{cam}} = \begin{bmatrix} \mathbf{R}_{\text{map} \leftarrow \text{cam}} & \mathbf{t}_{\text{map} \leftarrow \text{cam}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (23)$$

where $\mathbf{R} \in SO(3)$ denotes the rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ the translation vector. The transformation $\mathbf{T}_{\text{map} \leftarrow \text{cam}}$ is obtained from the robot pose estimate provided by the localization and mapping system.

Using homogeneous coordinates $\tilde{\mathbf{P}}_{\text{cam}} = [\mathbf{P}_{\text{cam}}^\top, 1]^\top$, the 3D point in the global map frame is obtained in Equation (24).

$$\tilde{\mathbf{P}}_{\text{map}} = \mathbf{T}_{\text{map} \leftarrow \text{cam}} \tilde{\mathbf{P}}_{\text{cam}}. \quad (24)$$

Dropping the homogeneous component yields the final 3D point $\mathbf{P}_{\text{map}} \in \mathbb{R}^3$ expressed in the global map coordinate system.

Each pixel (u, v) is additionally associated with a semantic confidence score $s(u, v) \in [0, 1]$ and an instance identifier $\text{id}(u, v) \in \mathbb{N}$, obtained from the instance segmentation output of the visual detector.

While the geometric back-projection and coordinate transformation affect only the spatial components of the point, the semantic attributes are propagated unchanged from the image domain to the 3D representation. Consequently, each 3D point in the global map frame is represented as an augmented semantic point. Each pixel (u, v) is associated with a semantic confidence score and an instance identifier produced by the detector at the current camera pose (see Equation (25)).

$$\mathbf{q}_{\text{map}} = \begin{bmatrix} \mathbf{P}_{\text{map}} \\ s \\ \text{id} \end{bmatrix} = \begin{bmatrix} x_{\text{map}} \\ y_{\text{map}} \\ z_{\text{map}} \\ s(u, v) \\ \text{id}(u, v) \end{bmatrix} \quad (25)$$

This formulation preserves object-level information across the 2D-3D lifting process, enabling instance-consistent aggregation, clustering, and persistent semantic reasoning directly in the global map frame.

Similar to the semantic clustering process described in Section 3.3, the augmented 3D points are grouped into object-level clusters based on spatial proximity and shared instance identifiers. Each cluster is then summarized as a detection graph node, which encapsulates the 3D centroid and aggregated confidence score of the detected object hypothesis. However, instead of using the percentile-based aggregation employed for memory nodes, the mean confidence score is computed for detection nodes, reflecting the typically higher reliability of promptable detection outputs [42, 74] compared to open-vocabulary semantic mapping [35].

3.5 Fusion Strategy

Analogous to the gradient ascent formulation introduced in Section 3.2, the fusion strategy defines the reward function that guides the robot's exploration and exploitation behavior. Its purpose is to combine heterogeneous sources of semantic information into a unified decision space that prioritizes promising regions for object search while maintaining navigational feasibility.

The fusion strategy integrates three complementary types of graph nodes: (a) exploration frontier nodes, representing geometrically reachable but semantically unexplored regions, (b) persistent memory nodes, encoding accumulated semantic evidence from past observations, and (c) detection graph nodes, derived from promptable zero-shot object detections. By jointly

reasoning over these sources, the system balances exploration of unknown space with exploitation of semantically relevant hypotheses.

Prior to fusion, graph nodes from all sources are filtered using a relevance map (Section 3.5). The relevance map is updated continuously during exploration and suppresses graph nodes located in regions that have already been observed and deemed irrelevant to the current semantic query. This mechanism prevents oscillatory behavior and repeated visitation of low-value areas.

After relevance filtering, the remaining graph nodes are fused in two stages, as illustrated in Figure 11. First, exploration frontier nodes and memory nodes are combined using an exploration-memory weighting scheme (Section 3.5), which explicitly trades off short-term exploration incentives against long-term semantic beliefs. Second, detection graph nodes are fused with semantic evidence from the value map and persistent memory via a multi-source detection fusion mechanism (Section 3.5), yielding robust object hypotheses with aggregated confidence estimates.

The fusion process produces two sets of graph nodes: an exploration-memory node set, which guides exploratory navigation, and a detection node set, which represents high-confidence object hypotheses. Both sets are forwarded to the behavior-tree-based navigation policy (Section 3.6), which orchestrates low-level path planning and robot control to navigate toward high-reward regions identified by the fusion strategy.

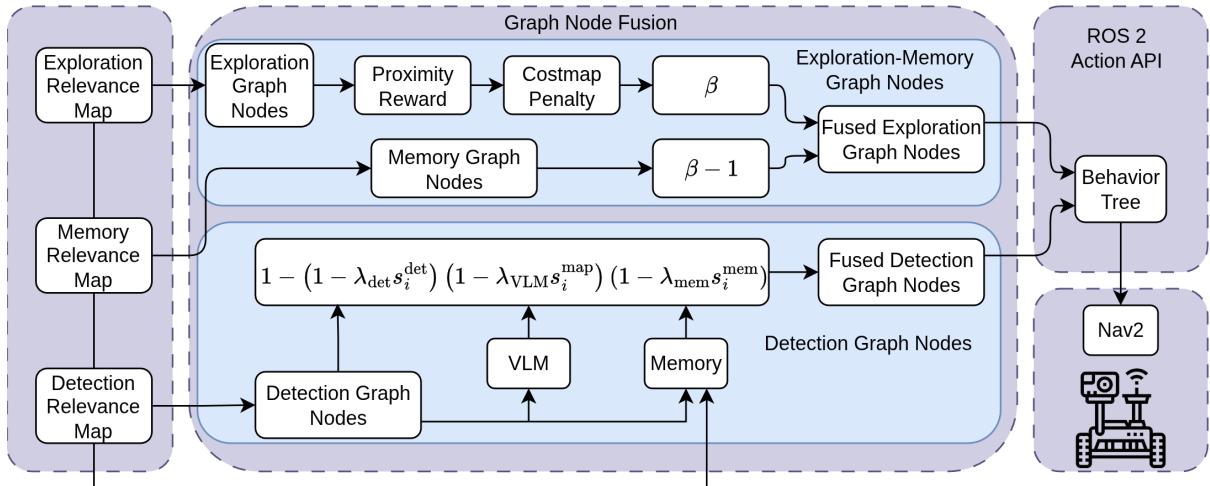


Figure 11: Overview of the fusion pipeline, illustrating how exploration frontiers, persistent memory nodes, and promptable detection hypotheses are filtered, weighted, and fused into a unified graph representation used for navigation decisions.

Exploration-Memory Weighting

A graph node is defined as $n_i = (\mathbf{x}_i, s_i)$, where $\mathbf{x}_i \in \mathbb{R}^3$ denotes the spatial position of the node in the global map frame and $s_i \in [0, 1]$ represents its associated semantic relevance score. For the purpose of exploration-memory weighting, only the planar component $\mathbf{x}_i = (x_i, y_i) \in \mathbb{R}^2$ is considered, since navigation and motion planning are performed on a 2D planar

representation. Height variations are handled implicitly by the mapping and localization system and are therefore omitted at this stage.

Following classical frontier-based exploration strategies [2, 65], graph nodes that are spatially closer to the robot are favored to encourage efficient and stable exploration behavior. To this end, the Euclidean distance between a graph node n_i and the current robot position $\mathbf{x}_r = (x_r, y_r) \in \mathbb{R}^2$ is computed in Equation (26).

$$d_i = \|\mathbf{x}_i - \mathbf{x}_r\|_2, \quad (26)$$

This distance is used to define a proximity persistence weight $w_{\text{prox}}(n_i)$, which increases the influence of nearby nodes while smoothly decaying with distance. The weighting function is formulated as a Gaussian kernel in Equation (27):

$$w_{\text{prox}}(n_i) = 1 + \alpha_{\text{prox}} \exp\left(-\frac{1}{2} \left(\frac{d_i}{r_{\text{prox}}}\right)^2\right), \quad (27)$$

where $\alpha_{\text{prox}} > 0$ controls the maximum amplification of nearby nodes and r_{prox} defines the spatial decay radius of the Gaussian kernel. This formulation softly biases the exploration policy toward frontiers close to the robot while preserving the relative ordering induced by the semantic score s_i .

In addition to proximity, navigation feasibility is taken into account by penalizing graph nodes located in regions with high traversal cost. Similar to the approach proposed in [13], a local neighborhood around each node is defined in Equation (28):

$$\Omega_i = \{\mathbf{x}_j \mid \|\mathbf{x}_j - \mathbf{x}_i\|_2 \leq r_{\text{cost}}\}, \quad (28)$$

where r_{cost} denotes the radius within which costmap values are aggregated and \mathbf{x}_j corresponds to individual costmap cell locations [62]. The locally aggregated cost \bar{c}_i is computed as a Gaussian-weighted average in Equation (29):

$$\bar{c}_i = \frac{\sum_{j \in \Omega_i} c_j \exp\left(-\frac{\|\mathbf{x}_j - \mathbf{x}_i\|_2^2}{2\sigma_c^2}\right)}{\sum_{j \in \Omega_i} \exp\left(-\frac{\|\mathbf{x}_j - \mathbf{x}_i\|_2^2}{2\sigma_c^2}\right)}, \quad (29)$$

where c_j denotes the raw costmap value at location \mathbf{x}_j and σ_c controls the spatial smoothing of the aggregation. The Gaussian-weighted average is used, because it emphasizes costmap values closer to the node location while still incorporating information from the broader neighborhood. This aggregation mirrors the spatial smoothing and inflation mechanisms commonly used in navigation costmaps [62].

To ensure numerical stability and comparability, the aggregated cost is normalized to the interval $[0, 1]$. The normalized cost is then transformed into a penalty weight in Equation (30):

$$w_{\text{cost}}(n_i) = \max(0.3, 1 - \lambda_{\text{cost}} \hat{c}_i), \quad (30)$$

where λ_{cost} determines the influence of the costmap on the final score. A minimum weight of 0.3 is enforced to prevent complete suppression of nodes in high-cost regions, thereby maintaining robustness in cluttered environments.

After defining the proximity and cost-based weighting terms, exploration frontier nodes are reweighted according to Equation (31):

$$s_i^{\text{explore}} = s_i \cdot \beta \cdot w_{\text{prox}}(n_i) \cdot w_{\text{cost}}(n_i), \quad (31)$$

where $\beta \in [0, 1]$ controls the trade-off between exploration and memory exploitation. Such explicit exploration-exploitation trade-offs are widely used in frontier-based and potential-based exploration methods [25]. Larger values of β prioritize the discovery of new areas, while smaller values emphasize revisiting previously observed high-relevance regions. In all experiments, β is treated as a global hyperparameter that can be tuned to adjust the exploration-exploitation balance. Conversely, memory graph nodes are downscaled during exploration in Equation (32).

$$s_i^{\text{memory}} = (1 - \beta) s_i, \quad (32)$$

Multi-Source Detection Fusion

The aim of multi-source detection fusion is to robustly estimate the confidence of detection graph nodes by combining evidence from three complementary sources: (a) the raw confidence score provided by the promptable detection vision model, (b) the semantic value map generated from the cosine similarity of the VLM, and (c) the persistent memory graph encoding accumulated semantic beliefs. A detection hypothesis must be present for fusion to occur: value map and memory cues can only reinforce an existing detection and cannot generate object hypotheses on their own. This prevents false positives caused by dense semantic similarity alone while still allowing consistent contextual and memory-based evidence to increase confidence. In order to effectively combine these heterogeneous sources, a weighted Noisy-OR model is employed. It is well suited for fusing independent probabilistic evidence [78] (see Equation (33)).

$$s_i^{\text{fused}} = 1 - \prod_{k \in \{\text{det, map, mem}\}} \left(1 - \lambda_k s_i^{(k)}\right) \quad (33)$$

where $s_i^{(k)}$ denotes the confidence score from source k and $\lambda_k \in [0, 1]$ represents the corresponding source weight. The weights λ_k allow tuning the influence of each source on the final fused confidence score. In this work, the weights are treated as fixed hyperparameters and tuned empirically. Equation (34) shows the explicit form of the weighted Noisy-OR fusion for the three considered sources: detection confidence, value map score, and memory relevance.

$$s_i^{\text{fused}} = 1 - \left(1 - \lambda_{\text{det}} s_i^{\text{det}}\right) \left(1 - \lambda_{\text{map}} s_i^{\text{map}}\right) \left(1 - \lambda_{\text{mem}} s_i^{\text{mem}}\right) \quad (34)$$

The value map score s_i^{map} is computed by aggregating the semantic similarity values from the value map within a local neighborhood around the detection node (see Equation (35) and Equation (36)):

$$\mathcal{V}_i = \{\mathbf{v}_j \mid \|\mathbf{v}_j - \mathbf{x}_i\|_2 < r_v\} \quad (35)$$

r_v defines the radius of the neighborhood around the detection node location \mathbf{x}_i , and \mathbf{v}_j denotes individual value map cell locations. The value map score is then computed as the mean similarity within this neighborhood in Equation (36):

$$s_i^{\text{map}} = \frac{1}{|\mathcal{V}_i|} \sum_{\mathbf{v}_j \in \mathcal{V}_i} v_j \quad (36)$$

where v_j denotes the semantic similarity value at location \mathbf{v}_j and $|\mathcal{V}_i|$ is the cardinality of the neighborhood set. Intuitively, if a memory node is near to the detection node, then the detection is more likely to be correct, as the memory node encodes prior evidence for the object being present in that location. Consequently, the memory relevance score s_i^{mem} is computed by aggregating the confidence scores of nearby memory nodes using a Gaussian kernel in Equation (37):

$$s_i^{\text{mem}} = \frac{\sum_{m_k \in \mathcal{N}_{\text{memory}}} s_k e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|_2^2}{2\sigma_m^2}}}{\sum_{m_k \in \mathcal{N}_{\text{memory}}} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|_2^2}{2\sigma_m^2}}} \quad (37)$$

where $\mathcal{N}_{\text{memory}}$ denotes the set of memory graph nodes, \mathbf{x}_k and s_k are the spatial position and confidence score of memory node m_k , and σ_m controls the spatial decay of influence. The Gaussian kernel ensures that nearby memory evidence contributes more strongly than distant or outdated observations, while maintaining smoothness and robustness. The memory relevance term acts as a soft spatial prior and does not perform explicit belief updating.

The final graph node representation after multi-source detection fusion is given in Equation (38):

$$\mathbf{n}_i = (\mathbf{x}_i, s_i^{\text{det}}, s_i^{\text{map}}, s_i^{\text{mem}}, s_i^{\text{fused}}) \quad (38)$$

where \mathbf{x}_i denotes the spatial position of the detection node, s_i^{det} is the raw detection confidence from the vision model, s_i^{map} is the aggregated value map score, s_i^{mem} is the aggregated memory relevance score, and s_i^{fused} is the final fused confidence score after multi-source detection fusion.

Instead of relying solely on the raw detection confidence from a single vision model [9], or solely relying on thresholding the cosine similarity or memory [14], the proposed multi-source detection fusion approach combines complementary evidence from multiple independent sources. This leads to higher confidence in true positive detections while reducing the likelihood of false positives, thereby enhancing the robustness and reliability of the object search system by enabling higher confidence thresholds.

Relevance Filtering and Node Suppression

To prevent repeated visitation of low-value areas and oscillatory behavior, a relevance map is maintained during exploration to suppress graph nodes located in regions that have already been observed and deemed irrelevant to the current semantic query. The relevance map is

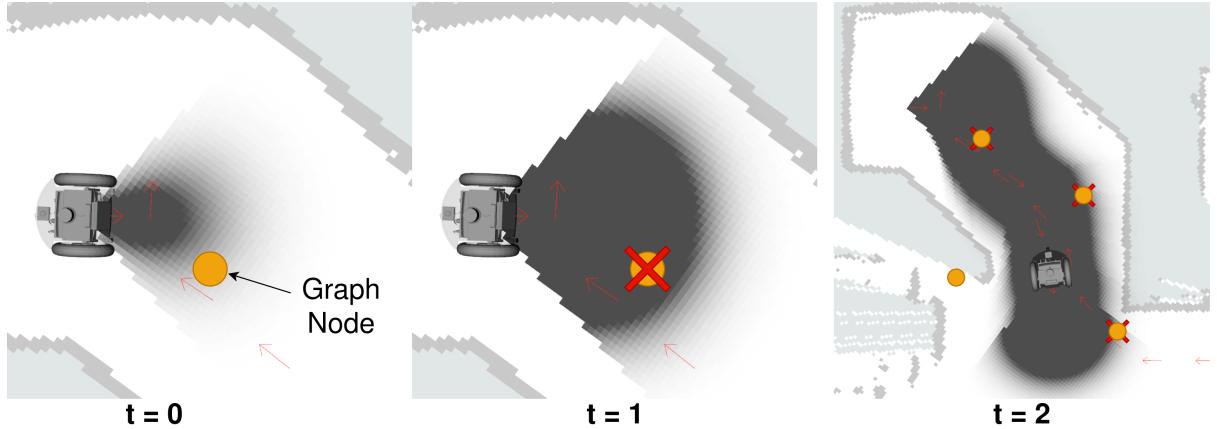


Figure 12: Illustration of relevance-based graph node suppression over time. At $t=0$, a graph node is generated within the robot’s current field of view. As the robot observes the region ($t=1$), the relevance map accumulates visibility evidence, causing the node to be suppressed once the area is deemed sufficiently observed. At $t=2$, only graph nodes outside previously observed regions or associated with high-confidence detections remain active, preventing oscillatory exploration while preserving strong semantic hypotheses.

constructed analogously to the value map described in Section 3.2, but instead of accumulating cosine similarity scores, it encodes spatial visibility and observation history.

While the semantic value map relies exclusively on an angular confidence weighting to model the camera FOV (see Equation (13)), the relevance map extends this formulation by additionally incorporating a radial weighting term. This allows the relevance map to suppress regions that have already been observed within both the angular FOV and a bounded spatial range.

The radial weighting component is defined in Equation (39):

$$w_{\text{rad}}(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{x}_r\|_2^2}{2\sigma_r^2}}, \quad (39)$$

where \mathbf{x}_r denotes the robot position and σ_r controls the spatial decay of relevance with increasing distance. The angular weighting term is defined in Equation (13) in Section 3.2.

The combined relevance weighting is obtained as the product of both components, resulting in high relevance values for regions that are close to the robot and aligned with the camera’s optical axis.

Three instances of the relevance map are maintained with different parameterizations for exploration, detection, and memory graph nodes, respectively. Graph nodes whose relevance value exceeds a predefined threshold are suppressed, indicating that the corresponding region has already been sufficiently observed.

To preserve robustness against missed detections, graph nodes whose detection confidence exceeds a high-confidence threshold (80% in all experiments) are exempt from relevance-based suppression and are directly republished. This mechanism enables the robot to revisit locations with strong object evidence while avoiding repeated exploration of low-value regions.

Figure 12 illustrates the relevance filtering process over three consecutive timestamps.

3.6 Behavior Tree for Semantic-Guided Exploration

A Behavior Tree (BT) [79] is employed to manage the high-level task structure for semantic-guided exploration. The BT orchestrates the robot's actions based on the fused graph nodes generated by the fusion strategy (Section 3.5), integrating initialization, detection, exploration, and termination behaviors into a unified decision-making framework. Low-level motion execution and collision avoidance are delegated to the Robot Operating System 2 (ROS 2) navigation stack, allowing the BT to focus exclusively on task sequencing and semantic reasoning.

High-Level Task Structure

At startup, the BT initializes the system by publishing the user-defined object query and clearing previously accumulated semantic and relevance maps. The BT then enters a detection branch, where it continuously monitors the fused detection graph nodes. If an object hypothesis exceeds a predefined confidence threshold for a sustained duration of one second, the robot navigates to the corresponding graph node, realigns its pose for optimal viewpoint alignment, and performs a verification step before capturing an image of the object.

If no object is detected, the BT transitions into the exploration branch. In this branch, the robot observes all available exploration-memory graph nodes, consisting of unexplored frontiers and persistent memory nodes, by performing in-place rotations to gather additional observations. Following this observation phase, the robot navigates towards the highest-valued exploration-memory node as determined by the fusion strategy. This closed-loop process repeats until either a target object is detected or a predefined time limit is reached.

An overview of this decision logic is illustrated as a flowchart in Figure 13, while the full BT structure is provided in Figure 27 in the Appendix.

Integration with the Navigation Stack

The multi-object search behavior implemented by the BT is encapsulated within a high-level action server, enabling a user or external agent to initiate a search task with a specified object prompt. Optional parameters include a zero-shot prompt for value map generation, a time limit, and soft and hard confidence thresholds for object verification.

The BT interfaces with the ROS 2 [62] navigation stack to execute motion-related actions. Specifically, the `GoToGraphNode` behavior uses the `/navigate_to_pose` action interface to issue navigation goals, while the `ObserveGraphNodes` and `RealignToObject` behavior nodes utilize the `/spin` action interface to perform in-place rotations for environmental observation and viewpoint refinement. The navigation stack is responsible for path planning, obstacle avoidance, and low-level control, thereby decoupling motion execution from high-level decision-making [62, 79].

To increase robustness in cluttered environments, an `ApproachPoseAdjustor` decorator node computes an alternative, costmap-feasible approach pose based on the target graph node location, the robot's current pose, and the global costmap. If no valid approach pose can be found, the original graph node position is used as the navigation goal.

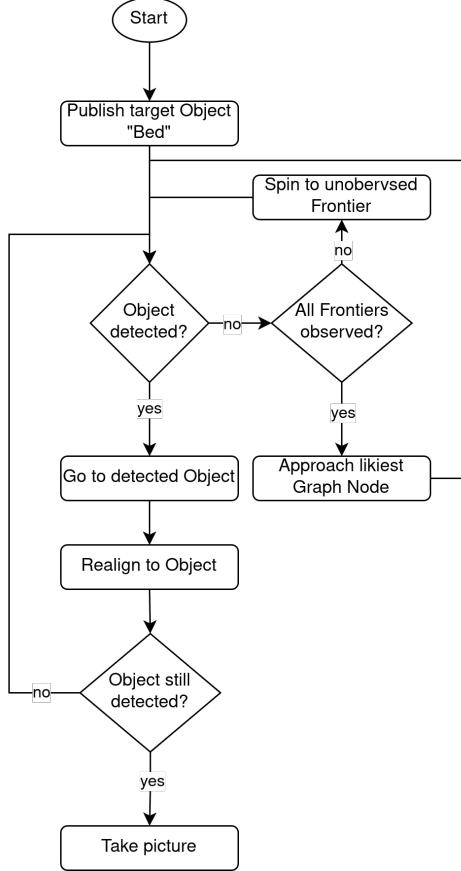


Figure 13: High-level behavior tree flowchart for BT-based semantic-guided exploration. The robot alternates between detection and exploration, navigating to detected objects when confident and otherwise exploring unobserved frontiers until the search terminates.

4 Implementation

This chapter details the practical implementation of the proposed approach, including the simulation setup, datasets, software stack, hardware configuration, and the evaluation metrics used in the experiments.

4.1 Simulation Environment

HabitatSim [41] has been widely adopted in embodied AI research due to its efficient rendering pipeline and support for large-scale datasets such as Matterport3D and HM3D [46]. However, for this work, NVIDIA Isaac Simulation Platform (IsaacSim) was selected as the primary simulation platform because it provides high-fidelity photorealistic rendering, accurate physics simulation, and realistic sensor models. These properties are essential for reducing the sim-to-

real gap and for ensuring that the proposed semantic exploration and fusion strategies transfer reliably to real robotic systems [80].

All experiments are conducted in IsaacSim version 5.0.0 by NVIDIA, leveraging its GPU-accelerated rendering, physically based simulation, and tight integration with ROS 2 [81]. The pre-installed *Carter* mobile robot is used as the navigation platform and is equipped with an RGB-D camera, a 2D LiDAR sensor, and wheel odometry. The robot is controlled through ROS 2 nodes communicating with IsaacSim via Fast DDS [82], enabling direct reuse of the navigation and perception stack employed in real-world deployment.

IsaacSim further supports importing complex indoor environments derived from Matterport3D [46], which are used as evaluation scenes in this work. To accommodate the physical footprint of the mobile robot and to ensure stable navigation behavior, all Matterport3D scenes are uniformly scaled by 50% in each spatial dimension to accommodate the physical footprint of the mobile robot and ensure stable navigation behavior.

Alternative simulators such as HabitatSim [41], MuJoCo [83], and Ignition Gazebo [84] were considered. However, these platforms either lack full physics-based mobile robot simulation, photorealistic sensor rendering, or seamless ROS 2 integration, which are required by the proposed system.

4.2 Dataset

Prior work such as VLFM [9], CoW [14], and OneMap [16] has demonstrated the effectiveness of Matterport3D-derived indoor environments for semantic exploration and object-goal navigation tasks. This experimental protocol closely follows the Habitat Navigation Challenge 2023 [85], which evaluates semantic navigation methods on the HM3D dataset [86].

Accordingly, this work employs Matterport3D-based indoor scenes [46] as the primary source of evaluation environments. The dataset provides a diverse set of real-world indoor spaces with high-quality 3D reconstructions and semantic annotations, making it well suited for evaluating open-vocabulary semantic exploration methods.

Due to the custom simulation setup in Isaac Sim, all baseline methods are re-implemented to ensure a fair comparison under consistent conditions, following the evaluation protocol of the Habitat Navigation Challenge 2023 (see Section 4.5) [85]. All evaluations are conducted on the HM3Dv2 validation split, which contains scenes with diverse layouts, object distributions, and levels of complexity.

The following scenes from the Matterport3D dataset are used in this work:

- 00800-TEEsavR23oF
- 00813-svBbv1Pavdk
- 00814-p53SfW6mjZe
- 00824-Dd4bFSTQ8gi
- 00848-ziup5kvttCCR

- 00876-mv2HUXq3B53

These scenes are also used by Yokoyama et al. and Busch et al., enabling direct and reproducible comparison of experimental results.

4.3 Used Software

The proposed semantic exploration and fusion framework is implemented using ROS 2 (Humble Hawksbill) as the middleware for inter-node communication and system integration. ROS 2 provides a modular, distributed architecture that allows perception, mapping, fusion, and exploration components to operate as independent nodes while remaining tightly synchronized [87].

For navigation, localization, and obstacle avoidance, the Navigation2 (Nav2) stack is employed [62]. In combination with Simultaneous Localization and Mapping (SLAM)-Toolbox [64], this enables robust 2D mapping, localization, and path planning within the simulated indoor environments. Inter-process and inter-container communication is handled via FastDDS, enabling reliable distributed communication between the detection, mapping, and exploration workspaces.

Open-vocabulary object detection is realized using YOLO-E [42], implemented via the Ultralytics framework [88]. YOLO-E is integrated as a ROS 2 lifecycle node, allowing controlled initialization, activation, and shutdown during exploration episodes, and supporting real-time inference for promptable object detection.

Semantic similarity scoring between visual observations and user-defined text queries is performed using BLIP-2 [27], accessed through the LAVIS library [89]. This VLM provides cosine similarity scores that are used to populate the semantic value map and guide exploration toward semantically relevant regions.

Persistent 3D semantic mapping and memory generation are handled by OpenFusion [35], which is wrapped as a dedicated ROS 2 node. OpenFusion incrementally fuses RGB-D observations into a global semantic point cloud representation, serving as the long-term memory component of the proposed system.

4.4 Used Hardware

All experiments are conducted on a dedicated workstation designed to support computationally demanding simulation, perception, and vision-language inference workloads. The hardware configuration is summarized as follows:

- **PC Workstation:**
 - **CPU:** AMD Ryzen 9 5950X (16 cores, 32 threads)
 - **Motherboard:** Gigabyte B550 Gaming X V2
 - **GPU:** NVIDIA GeForce RTX 4090 (ASUS TUF Gaming OC, 24 GB VRAM)
 - **RAM Memory:** 64 GB Corsair Vengeance LPX DDR4

The high core count of the CPU enables parallel execution of multiple ROS 2 nodes, while the RTX 4090 GPU provides sufficient compute and memory capacity for photorealistic simulation in IsaacSim [80], real-time open-vocabulary object detection [42], and VLM [27, 35, 53] inference. This setup ensures that all components of the proposed system can operate in real time without artificial bottlenecks introduced by hardware limitations.

4.5 Evaluation Pipeline Overview

To ensure comparability with prior work such as VLFM [9] and OneMap [16], a dedicated evaluation pipeline is implemented that reproduces the standard object-goal navigation metrics used in embodied Artificial Intelligence (AI) benchmarks [85]. The pipeline operates on recorded navigation trajectories and semantic maps generated during each exploration episode and is illustrated in Figure 14.

The primary evaluation metrics considered in this work are SR and SPL [9, 16, 19]. Computing SPL requires knowledge of the geodesic shortest path from the robot’s initial pose to the nearest instance of the target object. Since all experiments are conducted in a custom IsaacSim-based environment with uniformly scaled Matterport3D scenes, the built-in shortest-path planner provided by HabitatSim cannot be directly used.

Instead, the evaluation pipeline is composed of two main stages: (a) semantic environment reconstruction, and (b) geodesic shortest-path computation based on the reconstructed map.

As described in Chapter 3 Section 3.3, OpenFusion is employed to reconstruct a 3D semantic point cloud from the recorded RGB-D observations [35]. The Matterport3D category set is used as the closed set of semantic classes during evaluation [46]. For each episode, the stored semantic point cloud is filtered according to the target object category specified by the evaluation protocol. All clusters corresponding to the target class are extracted, and their 3D centroids are computed as candidate goal locations.

The geodesic shortest path is then computed from the robot’s initial pose to the nearest target centroid using the global planner of the ROS 2 Navigation2 stack [62]. Path planning is performed on the 2D occupancy grid generated by SLAM Toolbox [64], ensuring consistency with the localization and navigation setup used during exploration. The resulting shortest path is compared to the executed trajectory to compute SR and SPL for each episode.

In contrast to the evaluation procedure of the Habitat Navigation Challenge 2023 [85], which relies on HabitatSim’s internal shortest-path computation, the proposed evaluation pipeline is fully compatible with IsaacSim and the employed ROS 2-based navigation stack. This enables a fair comparison against prior methods while preserving identical metric definitions and avoiding bias introduced by simulator-specific planners.

4.6 Evaluation Metrics

This section defines the evaluation metrics used to assess the performance of the proposed semantic exploration framework and to ensure comparability with prior work in object-goal nav-

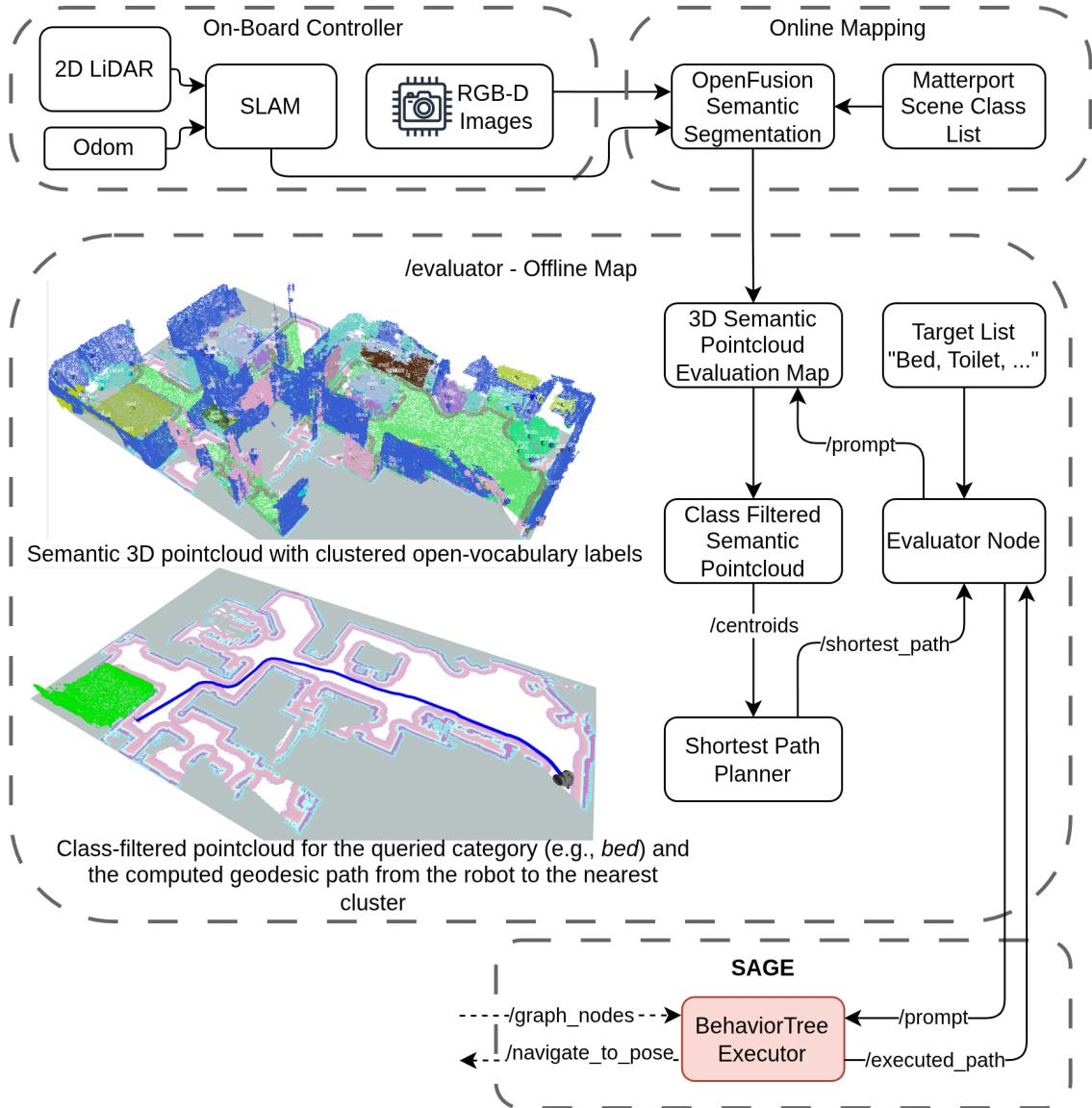


Figure 14: Evaluation pipeline used to compute SR and SPL. Recorded RGB-D observations are converted into a semantic point cloud using OpenFusion, target object clusters are extracted, and the geodesic shortest path is computed via the ROS 2 Navigation2 global planner for metric evaluation [62, 85].

igation. The selected metrics jointly evaluate task success, navigation efficiency, multi-object search capability, and detection robustness, thereby capturing both the quality of semantic reasoning and the reliability of perception-driven decision making. Each metric is explicitly assigned to a corresponding experiment, such that the experimental results directly address and answer the research questions defined in Chapter 1.

Experiment 1: Overall Performance Benchmarking

The first experiment establishes a performance baseline by comparing the proposed hybrid semantic exploration framework against state-of-the-art methods on standard object-goal navigation metrics. This experiment addresses **RQ1** by quantifying the overall effectiveness of the

integrated exploration-memory approach in realistic indoor environments. Results are compared against representative baselines including VLFM [9], VLMaps [18], OneMap [16], and Pigeon [19]. Overall performance is evaluated using SR and SPL, which are widely adopted metrics in embodied AI research [85] and adopted by comparable prior work [9, 16].

In this experiment, an episode is defined as a single object-goal navigation task from the HM3Dv2 validation split, in which the robot is tasked with locating a specified target object within a Matterport3D-derived indoor scene [85].

The SR metric measures the proportion of successful episodes in which the robot reaches the queried target object and is defined in Equation 40,

$$\text{SR} = \frac{1}{N} \sum_{i=1}^N S_i \quad (40)$$

where $S_i = 1$ if the target object is successfully reached in episode i , and 0 otherwise; N denotes the total number of evaluated episodes.

Navigation efficiency is quantified using SPL, which compares the geodesic shortest path to the actual path executed by the robot. The metric penalizes unnecessary detours and is defined only for successful episodes in Equation 41,

$$\text{SPL} = \frac{1}{N} \sum_{i=1}^N S_i \cdot \frac{l_i}{\max(p_i, l_i)} \quad (41)$$

where l_i is the geodesic shortest path length to the target, p_i is the executed path length, and S_i indicates task success for episode i .

An episode is considered successful, if the target object is visible in the RGB image captured by the robot's onboard camera, as verified through post-hoc inspection of the recorded observations. The geodesic shortest path l_i is computed from the robot's initial pose to the nearest instance of the target object. In contrast to HabitatSim-based evaluations, success is not restricted to predefined viewpoint tolerances, but instead includes any robot pose from which the target object is visually observable. All reported results were manually verified to confirm correct target visibility and episode termination. For every episode, the target object class is selected from the HM3Dv2 annotated category set [86].

Experiment 2: Exploration-Memory Fusion Weighting

This experiment investigates the influence of the exploration-memory fusion weighting on navigation performance and behavioral stability, addressing **RQ2**. By systematically varying the trade-off between live semantic exploration and persistent 3D semantic memory, the experiment evaluates how different fusion configurations affect task success, navigation efficiency, and long-horizon search behavior. The primary objective is to identify which weighting configuration yields the best balance between reactivity to newly observed information and stability derived from accumulated semantic memory.

The fusion weighting parameter

$$\lambda_{\text{exploration}} \in [0, 1]$$

controls the relative influence of frontier-based exploration and memory-driven exploitation during graph node fusion. A value of $\lambda_{\text{exploration}} = 0$ corresponds to purely memory-driven behavior, while $\lambda_{\text{exploration}} = 1$ corresponds to purely exploration-driven behavior. The parameter is varied in increments of 0.2. For each configuration, multiple navigation episodes are executed in the HM3Dv2 validation scenes, and performance is evaluated using SR and SPL.

In contrast to Experiment 1, this experiment employs multi-object search episodes in order to explicitly evaluate the benefit of persistent semantic memory across sequential object queries. Within a single episode, the robot is required to locate multiple target objects in sequence without resetting the environment or clearing the semantic map. This setting reflects realistic long-horizon search scenarios in which previously acquired semantic information can be reused to improve subsequent navigation decisions.

For evaluation, SR and SPL are computed per target object and aggregated across all episodes. Success for an individual target is defined identically to Experiment 1, and SPL is computed using the geodesic shortest path from the robot’s pose at the start of each sub-task to the nearest instance of the corresponding target object.

For each episode, five target objects are selected from the HM3Dv2 annotated category set [86] and queried in a fixed random order. A global random seed of 42 is used to ensure reproducibility across different fusion weight configurations. Similar to Experiment 1, six Matterport3D scenes from the HM3Dv2 validation split are used for evaluation, with two distinct starting positions per scene.

Experiment 3: Sensitivity to Semantic Map Granularity

This work employs a 3D semantic mapping approach based on OpenFusion [35], which fuses VLM embeddings into a volumetric representation. The granularity of semantic retrieval directly influences the quality of the resulting semantic map, with coarser retrievals introducing increased noise and ambiguity.

Granularity is controlled via the retrieval depth parameter top-k, which retrieves the top k most similar semantic regions from the SEEM [53] embedding dictionary within OpenFusion [35]. Lower top-k values yield sharper but potentially incomplete semantic representations, while higher top-k values produce denser semantic maps at the cost of increased noise. The top-k parameter affects the semantic retrieval stage prior to fusion and thus indirectly controls the density of the semantic map [35].

To counteract noise introduced by higher top-k values, the exploration weight $\lambda_{\text{exploration}}$ is increased across experimental configurations, shifting trust from memory-based semantic cues toward frontier-driven exploration. The top-k parameter is systematically varied from 5 to 25 in increments of 5. For each configuration, multiple navigation episodes are executed in the HM3Dv2 validation scenes, and performance is evaluated using SR and SPL.

To isolate the effect of semantic granularity, the starting pose is fixed across all runs. For each top-k value, two weighting configurations are evaluated: (a) a memory-dominant setting ($\lambda_{\text{exploration}} = 0$) and (b) an exploration-dominant configuration using the optimal weighting identified in Experiment 2 (RQ2). This design enables a direct assessment of whether increased

reliance on frontier-based exploration can compensate for semantic noise introduced by higher top-k values.

Experiment 4: Evaluation of Multi-Source Detection Fusion

This experiment evaluates the effectiveness of the multi-source semantic fusion strategy proposed in Chapter 3 Section 3.5 in fusing with a Noisy-OR model to improve detection robustness and suppress false positives during exploration, by increasing the detection thresholds, thereby addressing **RQ3**. Following prior work on open-vocabulary semantic perception, detection robustness is evaluated using confusion-matrix-based metrics such as precision, recall and F1-score, which are commonly employed to assess semantic retrieval quality in open-vocabulary settings [42, 54, 74]

Each episode is manually annotated based on the executed navigation trajectory, the recorded YOLO-E detection overlay, and system logs, with a semantic outcome describing whether the robot correctly reached the target object, detected the wrong object, or observed the target but ignored it. Failure modes that are unrelated to the detection process itself, such as exploration timeouts, lack of motion, or insufficient proximity to the target, are excluded from the detection evaluation and treated as neutral cases.

Label	Condition	Interpretation
TP	Successful target reach with $S \geq \tau$	Correct detection and navigation to the target object.
FP	Object mismatch with $S \geq \tau$	Robot selects a visually or semantically similar but incorrect object.
FN	Threshold rejection	Target object is observed but not selected because all detection scores remain below τ .
TN	Exploration-related failure	Episode outcome is determined by exploration or navigation limitations rather than detection quality.

Table 9: Abstracted confusion-matrix mapping for detection robustness evaluation.

Four fusion strategies are evaluated: (a) detector-only (YOLO-E), (b) detector + semantic similarity (BLIP-2), (c) detector + memory confidence, and (d) the full Noisy-OR fusion strategy combining all three sources. All fusion strategies must contain the detector source, as the other components are complementary and cannot independently trigger detections, only reinforce them. Across each fusion strategy, all experiments are executed using a fixed detection threshold of $\tau = 0.8$, which defines the robot’s online decision-making behaviour and serves as the basis for manual annotation. The resulting detection confidences are then reused in a post-hoc evaluation to compute confusion matrices and classification metrics for additional thresholds $\tau = 0.5$ and 0.6 .

Precision in this context measures the proportion of correct detections among all positive detections made by the robot and is defined in Equation 42:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (42)$$

where TP denotes true positives, meaning the object was correctly detected, and FP denotes false positives, meaning an incorrect object was detected. Recall measures the proportion of actual target objects that were correctly detected and is defined in Equation 43,

$$\text{Recall} = \frac{TP}{TP + FN} \quad (43)$$

where FN denotes false negatives, meaning the target object was observed but not selected because all detection confidences remained below the decision threshold τ . Importantly, false negatives in this experiment exclusively arise from conservative thresholding decisions and do not indicate a failure of the perception pipeline to observe the target object. The F1-score is the harmonic mean of precision and recall, providing a balanced measure of detection performance, and is defined in Equation 44,

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (44)$$

The False Positive Rate (FPR) quantifies the proportion of incorrect detections among all negative cases and is defined in Equation 45,

$$\text{FPR} = \frac{FP}{FP + TN} \quad (45)$$

where TN denotes true negatives, corresponding to episodes in which no detection decision is evaluated due to exploration-related failure modes or insufficient observability. TN corresponds to neutral episodes in which no detection decision was made and is included solely for the computation of the false-positive rate.

The confusion matrix for single-source and multi-source fusion is primarily reported for the operational threshold $\tau = 0.8$, which was empirically selected to provide a balanced trade-off between sensitivity and specificity in preliminary tests. In addition, precision–recall curves are generated by sweeping the decision threshold τ from 0 to 1 in increments of 0.01 using the same annotated episodes. This threshold sweep enables an analysis of detection robustness across different operating points without requiring additional experimental runs.

Experiment 5: Real-World System Performance

This experiment defines a deployment-oriented evaluation protocol for assessing the real-world performance of the proposed semantic exploration framework on a physical mobile robot platform. The objective is to characterize system robustness, efficiency, and deployability in realistic indoor environments, thereby addressing **RQ5**.

The evaluation focuses on monitoring computational resource utilization, including CPU, GPU, and RAM usage, as well as inference latency and effective frame rate (FPS) of the

perception and mapping components during live operation. Each component's computational load is logged to enable fine-grained performance analysis.

In addition, the semantic map voxel resolution in OpenFusion is systematically reduced to analyze the trade-off between GPU memory consumption and runtime performance. This analysis is intended to assess the feasibility of deploying the system on resource-constrained robotic platforms.

5 Discussion and Results

Aim of this chapter is to answer the research questions posed in Chapter 1 through a series of experiments designed to evaluate the performance of the proposed multi-object search framework. The first experiment establishes baseline performance against state-of-the-art methods, while subsequent experiments systematically analyze the impact of key design choices on navigation efficiency, mapping robustness, and detection reliability. Each experiment is structured to isolate specific variables, allowing for a clear assessment of their contributions to overall system performance.

5.1 Experiment 1: Benchmarking on Matterport Scenes

Evaluates baseline performance in multi-object search compared to state-of-the-art frameworks (OneMap, VLIM, Pigeon) using SR, SPL, and MSR.

5.2 Experiment 2: Impact of Exploration-Memory Weighting

Analyzes how varying the balance between live exploration and persistent memory influences navigation efficiency and task success. The results in Figures 15 and 16 indicate the best trade off between reactivity to new observations and exploitation of accumulated knowledge occurs at an intermediate weighting factor, between 0.2 and 0.4 exploitation weight.

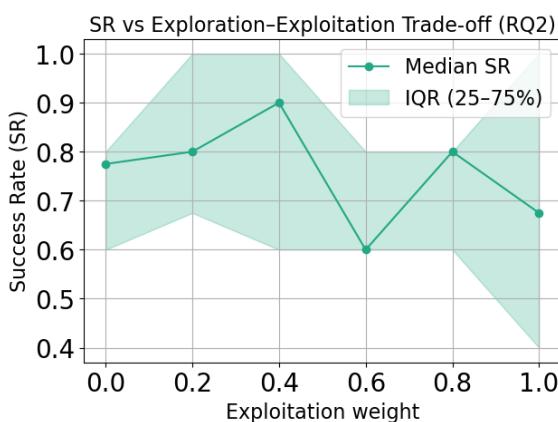


Figure 15: TODO: Replace with relevant figure for Experiment 2

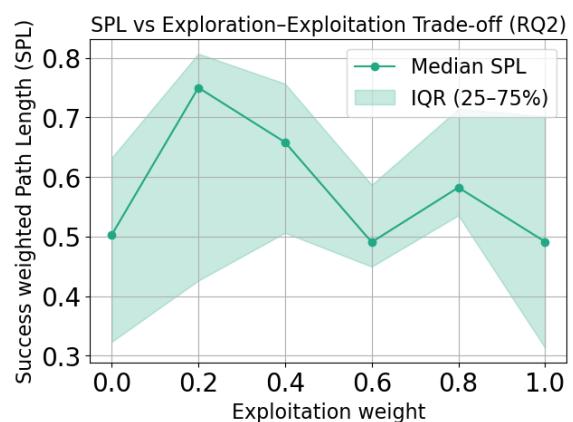


Figure 16: TODO: Replace with relevant figure for Experiment 2

The failure mode breakdowns in Figures 17, 18, and 19 further illustrate how different weighting strategies impact the types of failures encountered, highlighting the importance of balancing exploration and memory for robust performance. Overall, the biggest failure was due to misdetections with 11.6% of all episodes for this experiment, followed by not seeing the target (7.8%) and ignoring the target (4.4%). The failure mode "Stopped at wrong object" was the biggest contributor to misdetections, accounting for 11.6% of all episodes. This is due to the Noisy-OR fusion strategy being too lenient in accepting false positive detections from the object detector (see Section 5.4 for more details). The second biggest failure mode was "Did not see target" with 4.9% of this experiment, which can be attributed to the exploration strategy not adequately covering the environment or the object being occluded. The third biggest failure mode was "Ignored target" with 4.6% of all episodes, likely due to strict detection thresholds or misclassifications by the object detector.

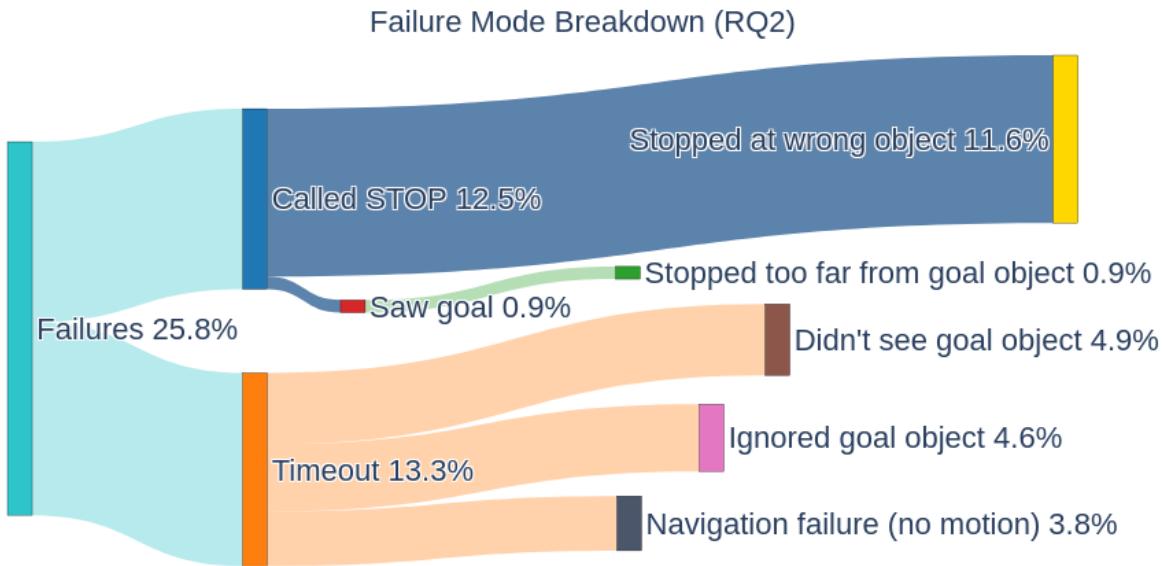


Figure 17: TODO: Replace with relevant figure for Experiment 2

Figure 18 and Figure 19 show the failure mode breakdowns for the 0.6 and 0.4 exploitation weight settings, respectively. At 0.6 exploitation weight, the system tends to rely more on accumulated memory, which may lead to false hints, especially for early exploration stages. OpenFusion [35] uses the Top-k most relevant clusters from the semantic embedded dictionary, for zero-shot capability. However, this can lead to incorrect associations if the memory contains false positives, because the robot did not yet observe the target object. Conversely, at 0.4 exploitation weight, the system is more reactive to new observations, which can help correct false hints from memory, and if the robot's current observations, eg semantic frontier nodes are low in confidence, the robot can still rely on memory to guide exploration, to the most promising areas.

If exploitation is dominant, especially in early exploration stages, the robot may be guided to similar objects that are not the target, leading to misdetections. On the other hand, if exploration is too dominant, the robot may miss opportunities to leverage accumulated knowledge, resulting in inefficient search patterns and potentially overlooking the target object. Interest-

ingly, both have a similar SPL score for pure exploration (0.0 exploitation weight) and pure exploitation (1.0 exploitation weight), indicating that both extremes lead to inefficient navigation paths. SR at pure exploration is slightly better than pure exploitation, likely because the robot can adapt to new observations, whereas pure exploitation may lead to following false hints from memory.

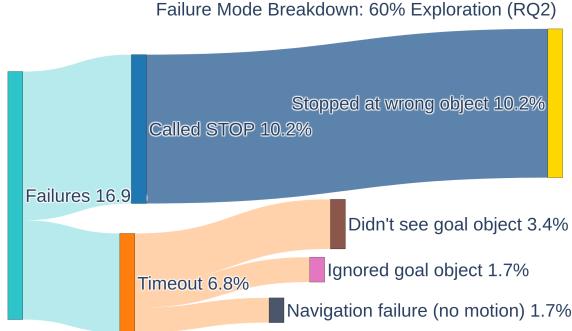


Figure 18: TODO: Replace with relevant figure for Experiment 4

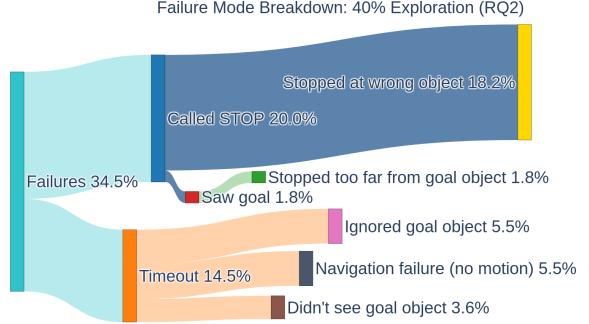


Figure 19: TODO: Replace with relevant figure for Experiment 4

For the best performance, an exploitation weight between 0.2 and 0.4 is recommended, as it provides a good balance between leveraging accumulated knowledge and adapting to new observations, leading to improved navigation efficiency and task success. For the following experiments, an exploitation weight of 0.4 is used as it provided the best overall performance in this experiment.

5.3 Experiment 3: Sensitivity to Semantic Map Granularity

Investigates how varying the semantic retrieval depth affects mapping robustness and overall navigation stability.

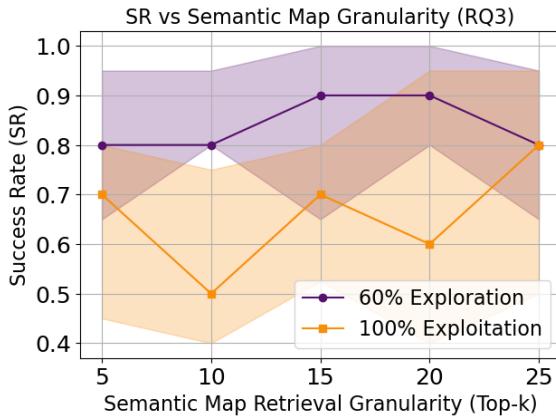


Figure 20: TODO: Replace with relevant figure for Experiment 3

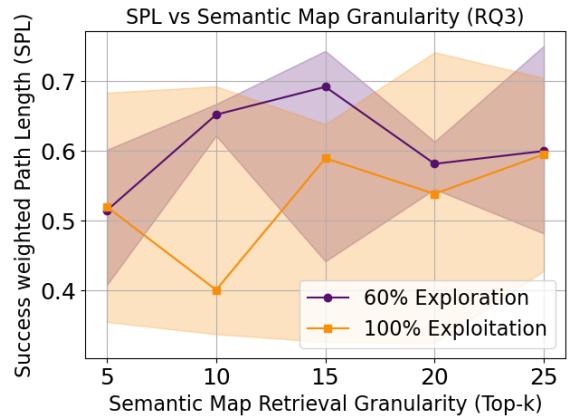


Figure 21: TODO: Replace with relevant figure for Experiment 3

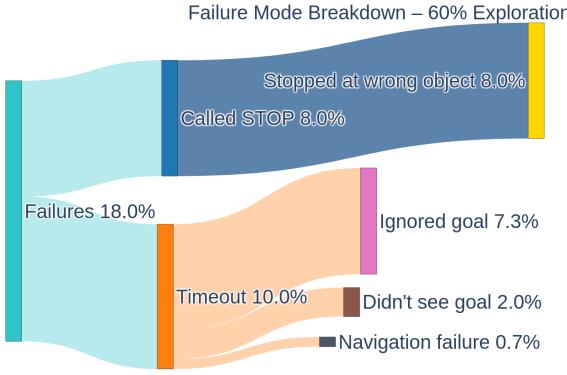


Figure 22: TODO: Replace with relevant figure for Experiment 3

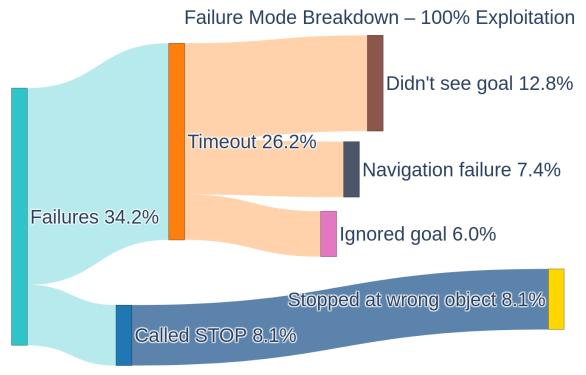


Figure 23: TODO: Replace with relevant figure for Experiment 3

5.4 Experiment 4: Effect of Multi-Source Semantic Fusion

Examines how combining detection confidence, semantic similarity, and memory reliability improves detection robustness and reduces false positives.

Fusion Variant	τ	$P (\uparrow)$	$R (\uparrow)$	$F1 (\uparrow)$	$FPR (\downarrow)$	TP	FP	FN	TN
Single source detection	0.5	0.910	0.909	0.910	0.364	528	52	53	91
Detection + VLM score	0.5	0.903	0.928	0.915	0.406	539	58	42	85
Detection + Memory	0.5	0.909	0.914	0.912	0.371	531	53	50	90
Multi-source Fusion	0.5	0.906	0.917	0.912	0.385	533	55	48	88
Single source detection	0.6	0.914	0.892	0.902	0.343	518	49	63	94
Detection + VLM score	0.6	0.910	0.910	0.910	0.364	529	52	52	91
Detection + Memory	0.6	0.910	0.905	0.908	0.364	526	52	55	91
Multi-source Fusion	0.6	0.910	0.910	0.910	0.364	529	52	52	91
Single source detection	0.8	0.941	0.632	0.756	0.161	367	23	214	120
Detection + VLM score	0.8	0.912	0.876	0.894	0.343	509	49	72	94
Detection + Memory	0.8	0.913	0.799	0.852	0.308	464	44	117	99
Multi-source Fusion	0.8	0.913	0.888	0.901	0.343	516	49	65	94

Table 10: Detection performance metrics across fusion strategies and thresholds. Arrows indicate whether higher (\uparrow) or lower (\downarrow) values are better. Best values per threshold τ are highlighted in bold. TODO: Check numbers. May need to be updated.

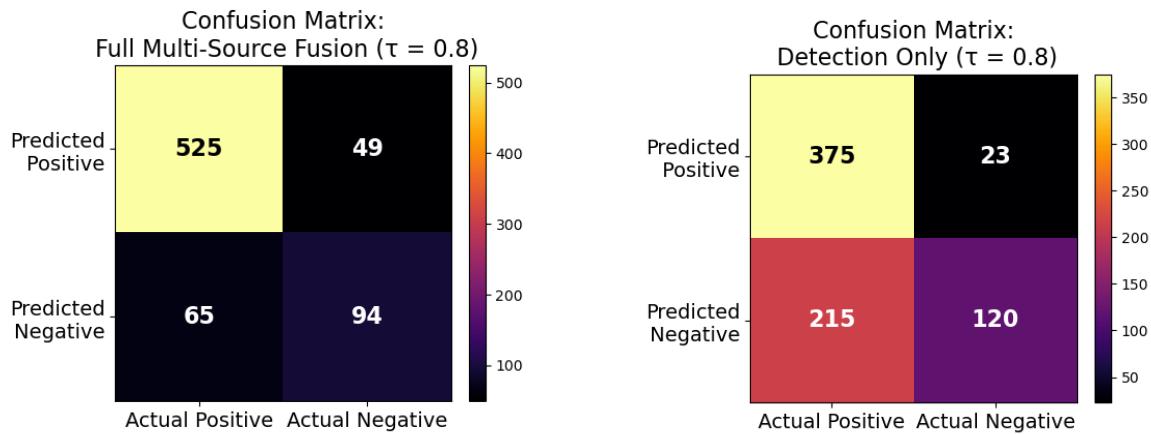


Figure 24: TODO: Replace with relevant figure for Experiment 4

Figure 25: TODO: Replace with relevant figure for Experiment 4

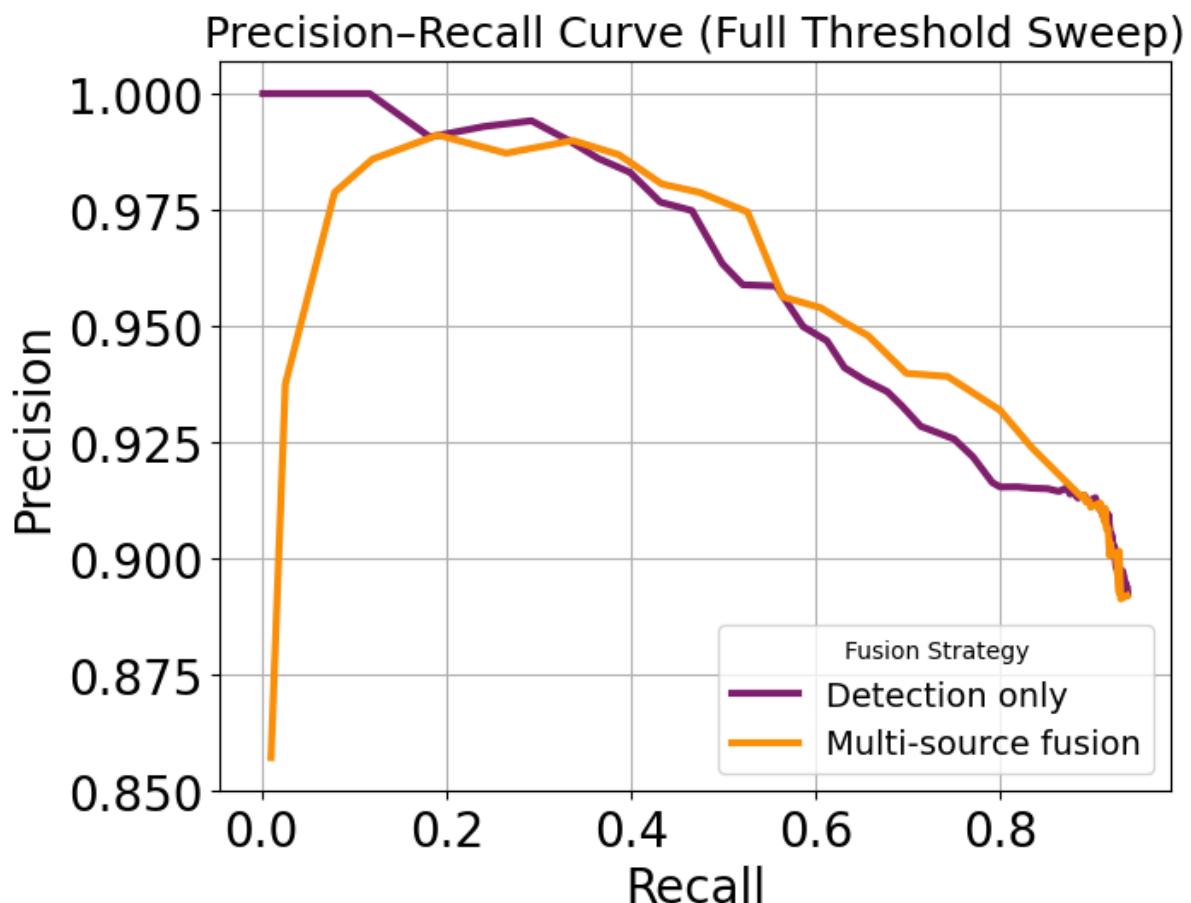


Figure 26: TODO: Replace with relevant figure for Experiment 4

5.5 Experiment 5: System Efficiency and Real-World Validation

Assesses runtime performance, resource utilization, and stability under real-world sensor noise during physical deployment.

6 Summary and Outlook

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Bibliography

- [1] A. Vaswani *et al.* “Attention Is All You Need.” Comment: 15 pages, 5 figures. arXiv: [1706.03762 \[cs\]](https://arxiv.org/abs/1706.03762). (Aug. 2, 2023), [Online]. Available: <http://arxiv.org/abs/1706.03762> (visited on 12/11/2025), pre-published.
- [2] A. Topiwala, P. Inani, and A. Kathpal. “Frontier Based Exploration for Autonomous Robot.” arXiv: [1806.03581 \[cs\]](https://arxiv.org/abs/1806.03581). (Jun. 10, 2018), [Online]. Available: <http://arxiv.org/abs/1806.03581> (visited on 12/11/2025), pre-published.
- [3] T. B. Brown *et al.* “Language Models are Few-Shot Learners.” Comment: 40+32 pages. arXiv: [2005.14165 \[cs\]](https://arxiv.org/abs/2005.14165). (Jul. 22, 2020), [Online]. Available: <http://arxiv.org/abs/2005.14165> (visited on 01/02/2026), pre-published.
- [4] A. Radford *et al.* “Learning Transferable Visual Models From Natural Language Supervision.” arXiv: [2103.00020 \[cs\]](https://arxiv.org/abs/2103.00020). (Feb. 26, 2021), [Online]. Available: <http://arxiv.org/abs/2103.00020> (visited on 12/13/2025), pre-published.
- [5] A. Brohan *et al.* “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control.” Comment: Website: <https://robotics-transformer.github.io/>. arXiv: [2307.15818 \[cs\]](https://arxiv.org/abs/2307.15818). (Jul. 28, 2023), [Online]. Available: <http://arxiv.org/abs/2307.15818> (visited on 01/02/2026), pre-published.
- [6] T.-Y. Lin *et al.* “Microsoft COCO: Common Objects in Context.” Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. arXiv: [1405.0312 \[cs\]](https://arxiv.org/abs/1405.0312). (Feb. 21, 2015), [Online]. Available: <http://arxiv.org/abs/1405.0312> (visited on 12/11/2025), pre-published.
- [7] S. Schwaiger *et al.* “UGV-CBRN: An Unmanned Ground Vehicle for Chemical, Biological, Radiological, and Nuclear Disaster Response.” arXiv: [2406.14385 \[cs\]](https://arxiv.org/abs/2406.14385). (Sep. 20, 2024), [Online]. Available: <http://arxiv.org/abs/2406.14385> (visited on 01/02/2026), pre-published.
- [8] J. Chen, G. Li, S. Kumar, B. Ghanem, and F. Yu. “How To Not Train Your Dragon: Training-free Embodied Object Goal Navigation with Semantic Frontiers.” Comment: Accepted by/To be published in Robotics: Science and Systems (RSS) 2023; 11 pages, 5 figures. arXiv: [2305.16925 \[cs\]](https://arxiv.org/abs/2305.16925). (May 26, 2023), [Online]. Available: <http://arxiv.org/abs/2305.16925> (visited on 12/11/2025), pre-published.
- [9] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher. “VLFM: Vision-Language Frontier Maps for Zero-Shot Semantic Navigation.” version 1. (2023), [Online]. Available: <https://arxiv.org/abs/2312.03275> (visited on 12/11/2025), pre-published.

- [10] K. Zhou *et al.* “ESC: Exploration with Soft Commonsense Constraints for Zero-shot Object Navigation.” arXiv: [2301.13166 \[cs\]](https://arxiv.org/abs/2301.13166). (Jul. 6, 2023), [Online]. Available: <http://arxiv.org/abs/2301.13166> (visited on 12/11/2025), pre-published.
- [11] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra. “ZSON: Zero-Shot Object-Goal Navigation using Multimodal Goal Embeddings.” Comment: code: <https://github.com/gunagg/zson>. arXiv: [2206.12403 \[cs\]](https://arxiv.org/abs/2206.12403). (Oct. 13, 2023), [Online]. Available: <http://arxiv.org/abs/2206.12403> (visited on 12/11/2025), pre-published.
- [12] I. Lluvia, E. Lazkano, A. Ansuategi, I. Lluvia, E. Lazkano, and A. Ansuategi, “Active Mapping and Robot Exploration: A Survey,” *Sensors*, vol. 21, no. 7, Apr. 2, 2021, ISSN: 1424-8220. DOI: [10.3390/s21072445](https://doi.org/10.3390/s21072445). [Online]. Available: <https://www.mdpi.com/1424-8220/21/7/2445> (visited on 12/12/2025).
- [13] F. Bourgault, A. Makarenko, S. Williams, B. Grocholsky, and H. Durrant-Whyte, “Information Based Adaptive Robotic Exploration,” vol. 1, Feb. 1, 2002, 540–545 vol.1, ISBN: 978-0-7803-7398-3. DOI: [10.1109/IRDS.2002.1041446](https://doi.org/10.1109/IRDS.2002.1041446).
- [14] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. “CoWs on Pasture: Baselines and Benchmarks for Language-Driven Zero-Shot Object Navigation.” arXiv: [2203.10421 \[cs\]](https://arxiv.org/abs/2203.10421). (Dec. 14, 2022), [Online]. Available: <http://arxiv.org/abs/2203.10421> (visited on 12/11/2025), pre-published.
- [15] O. Alama *et al.* “RayFronts: Open-Set Semantic Ray Frontiers for Online Scene Understanding and Exploration.” arXiv: [2504.06994 \[cs\]](https://arxiv.org/abs/2504.06994). (Apr. 9, 2025), [Online]. Available: <http://arxiv.org/abs/2504.06994> (visited on 12/11/2025), pre-published.
- [16] F. L. Busch, T. Homberger, J. Ortega-Peimbert, Q. Yang, and O. Andersson. “One Map to Find Them All: Real-time Open-Vocabulary Mapping for Zero-shot Multi-Object Navigation.” arXiv: [2409.11764 \[cs\]](https://arxiv.org/abs/2409.11764). (Mar. 3, 2025), [Online]. Available: <http://arxiv.org/abs/2409.11764> (visited on 12/11/2025), pre-published.
- [17] Q. Gu *et al.* “ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning.” Comment: Project page: <https://concept-graphs.github.io/> Explainer video: <https://youtu.be/mRhNkQwRYnc>. arXiv: [2309.16650 \[cs\]](https://arxiv.org/abs/2309.16650). (Sep. 28, 2023), [Online]. Available: <http://arxiv.org/abs/2309.16650> (visited on 12/11/2025), pre-published.
- [18] C. Huang, O. Mees, A. Zeng, and W. Burgard. “Visual Language Maps for Robot Navigation.” Comment: Accepted at the 2023 IEEE International Conference on Robotics and Automation (ICRA). Project page: <https://vlmaps.github.io>. arXiv: [2210.05714 \[cs\]](https://arxiv.org/abs/2210.05714). (Mar. 8, 2023), [Online]. Available: <http://arxiv.org/abs/2210.05714> (visited on 12/11/2025), pre-published.
- [19] “PIGEON: VLM-Driven Object Navigation via Points of Interest SelectionPreprint. Work in Progress.” (), [Online]. Available: <https://arxiv.org/html/2511.13207v1> (visited on 12/11/2025).
- [20] P. Quin, D. Nguyen, T. Vu, A. Alempijevic, and G. Paul, “Approaches for Efficiently Detecting Frontier Cells in Robotics Exploration,” *Frontiers in Robotics and AI*, vol. 8, p. 616470, Feb. 25, 2021. DOI: [10.3389/frobt.2021.616470](https://doi.org/10.3389/frobt.2021.616470).

- [21] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). Cambridge, Massachusetts London: MIT Press, 2006, 1 p., Description based on publisher supplied metadata and other sources, ISBN: 978-0-262-20162-9 978-0-262-30380-4.
- [22] S. Liu, M. Zhang, P. Kadam, and C.-C. J. Kuo, 3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods. Cham: Springer International Publishing, 2021, ISBN: 978-3-030-89179-4 978-3-030-89180-0. DOI: [10.1007/978-3-030-89180-0](https://doi.org/10.1007/978-3-030-89180-0). [Online]. Available: <https://link.springer.com/10.1007/978-3-030-89180-0> (visited on 01/03/2026).
- [23] M. Tellaroli, M. Luperto, M. Antonazzi, and N. Basilico. "Frontier-Based Exploration for Multi-Robot Rendezvous in Communication-Restricted Unknown Environments." arXiv: [2403.11617 \[cs\]](https://arxiv.org/abs/2403.11617). (Jul. 19, 2024), [Online]. Available: <http://arxiv.org/abs/2403.11617> (visited on 01/03/2026), pre-published.
- [24] V. S. Dorbala, J. F. Mullen, and D. Manocha, "Can an Embodied Agent Find Your "Cat-shaped Mug"? LLM-Guided Exploration for Zero-Shot Object Navigation," IEEE Robotics and Automation Letters, vol. 9, no. 5, pp. 4083–4090, May 2024, Comment: 10 pages, ISSN: 2377-3766, 2377-3774. DOI: [10.1109/LRA.2023.3346800](https://doi.org/10.1109/LRA.2023.3346800). arXiv: [2303.03480 \[cs\]](https://arxiv.org/abs/2303.03480). [Online]. Available: <http://arxiv.org/abs/2303.03480> (visited on 12/11/2025).
- [25] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman. "PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning." Comment: 8 pages + supplementary. Accepted in CVPR 2022. arXiv: [2201.10029 \[cs\]](https://arxiv.org/abs/2201.10029). (Jun. 17, 2022), [Online]. Available: <http://arxiv.org/abs/2201.10029> (visited on 12/11/2025), pre-published.
- [26] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das. "PIRLNav: Pretraining with Imitation and RL Finetuning for ObjectNav." Comment: 8 pages + supplement. arXiv: [2301.07302 \[cs\]](https://arxiv.org/abs/2301.07302). (Mar. 26, 2023), [Online]. Available: <http://arxiv.org/abs/2301.07302> (visited on 12/11/2025), pre-published.
- [27] J. Li, D. Li, S. Savarese, and S. Hoi. "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models." arXiv: [2301.12597 \[cs\]](https://arxiv.org/abs/2301.12597). (Jun. 15, 2023), [Online]. Available: <http://arxiv.org/abs/2301.12597> (visited on 12/15/2025), pre-published.
- [28] S. Liu *et al.* "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection." Comment: Code will be available at <https://github.com/IDEA-Research/GroundingDINO>. arXiv: [2303.05499 \[cs\]](https://arxiv.org/abs/2303.05499). (Jul. 19, 2024), [Online]. Available: <http://arxiv.org/abs/2303.05499> (visited on 12/15/2025), pre-published.
- [29] A. Kirillov *et al.* "Segment Anything." Comment: Project web-page: <https://segmentanything.com>. arXiv: [2304.02643 \[cs\]](https://arxiv.org/abs/2304.02643). (Apr. 5, 2023), [Online]. Available: <http://arxiv.org/abs/2304.02643> (visited on 12/15/2025), pre-published.

- [30] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask R-CNN.” Comment: open source; appendix on more results. arXiv: [1703.06870 \[cs\]](https://arxiv.org/abs/1703.06870). (Jan. 24, 2018), [Online]. Available: <http://arxiv.org/abs/1703.06870> (visited on 01/06/2026), pre-published.
- [31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv.org. (Oct. 11, 2018), [Online]. Available: <https://arxiv.org/abs/1810.04805v2> (visited on 01/08/2026).
- [32] L. H. Li *et al.* “Grounded Language-Image Pre-training.” Comment: CVPR 2022; updated visualizations; fixed hyper-parameters in Appendix C.1. arXiv: [2112.03857 \[cs\]](https://arxiv.org/abs/2112.03857). (Jun. 17, 2022), [Online]. Available: <http://arxiv.org/abs/2112.03857> (visited on 12/15/2025), pre-published.
- [33] D. Jiang *et al.* “From CLIP to DINO: Visual Encoders Shout in Multi-modal Large Language Models.” arXiv: [2310.08825 \[cs\]](https://arxiv.org/abs/2310.08825). (Mar. 8, 2024), [Online]. Available: <http://arxiv.org/abs/2310.08825> (visited on 12/14/2025), pre-published.
- [34] D. S. Chapolat, D. Gandhi, A. Gupta, and R. Salakhutdinov. “Object Goal Navigation using Goal-Oriented Semantic Exploration.” Comment: Winner of the CVPR 2020 AI-Habitat Object Goal Navigation Challenge. See the project webpage at <https://devendrachaplot.github.io/projects/semantic-exploration.html>. arXiv: [2007.00643 \[cs\]](https://arxiv.org/abs/2007.00643). (Jul. 2, 2020), [Online]. Available: <http://arxiv.org/abs/2007.00643> (visited on 12/11/2025), pre-published.
- [35] K. Yamazaki *et al.* “Open-Fusion: Real-time Open-Vocabulary 3D Mapping and Queryable Scene Representation.” arXiv: [2310.03923 \[cs\]](https://arxiv.org/abs/2310.03923). (Oct. 5, 2023), [Online]. Available: <http://arxiv.org/abs/2310.03923> (visited on 12/11/2025), pre-published.
- [36] K. M. Jatavallabhula *et al.* “ConceptFusion: Open-set Multimodal 3D Mapping.” Comment: RSS 2023. Project page: <https://concept-fusion.github.io> Explainer video: <https://www.youtube.com/watch?v=rkXgws8fiDs> Code: <https://github.com/concept-fusion/concept-fusion>. arXiv: [2302.07241 \[cs\]](https://arxiv.org/abs/2302.07241). (Oct. 23, 2023), [Online]. Available: <http://arxiv.org/abs/2302.07241> (visited on 01/03/2026), pre-published.
- [37] R.-Z. Qiu *et al.* “Learning Generalizable Feature Fields for Mobile Manipulation.” Comment: Preprint. Project website is at: <https://geff-b1.github.io>. arXiv: [2403.07563 \[cs\]](https://arxiv.org/abs/2403.07563). (Nov. 26, 2024), [Online]. Available: <http://arxiv.org/abs/2403.07563> (visited on 12/11/2025), pre-published.
- [38] A. Zareian, K. D. Rosa, D. H. Hu, and S.-F. Chang. “Open-Vocabulary Object Detection Using Captions.” Comment: To be presented at CVPR 2021 (oral paper). arXiv: [2011.10678 \[cs\]](https://arxiv.org/abs/2011.10678). (Mar. 14, 2021), [Online]. Available: <http://arxiv.org/abs/2011.10678> (visited on 01/03/2026), pre-published.
- [39] M. Ghasemi, A. H. Moosavi, and D. Ebrahimi. “A Comprehensive Survey of Reinforcement Learning: From Algorithms to Practical Challenges.” Comment: 79 pages. arXiv: [2411.18892 \[cs\]](https://arxiv.org/abs/2411.18892). (Feb. 1, 2025), [Online]. Available: <http://arxiv.org/abs/2411.18892> (visited on 01/05/2026), pre-published.

- [40] J. Straub *et al.* “The Replica Dataset: A Digital Replica of Indoor Spaces.” arXiv: [1906.05797 \[cs\]](https://arxiv.org/abs/1906.05797). (Jun. 13, 2019), [Online]. Available: <http://arxiv.org/abs/1906.05797> (visited on 01/05/2026), pre-published.
- [41] M. Savva *et al.* “Habitat: A Platform for Embodied AI Research.” Comment: ICCV 2019. arXiv: [1904.01201 \[cs\]](https://arxiv.org/abs/1904.01201). (Nov. 25, 2019), [Online]. Available: <http://arxiv.org/abs/1904.01201> (visited on 01/05/2026), pre-published.
- [42] A. Wang, L. Liu, H. Chen, Z. Lin, J. Han, and G. Ding. “YOLOE: Real-Time Seeing Anything.” Comment: ICCV 2025 Camera-ready Version. arXiv: [2503.07465 \[cs\]](https://arxiv.org/abs/2503.07465). (Oct. 17, 2025), [Online]. Available: <http://arxiv.org/abs/2503.07465> (visited on 01/05/2026), pre-published.
- [43] P. S. Thomas and E. Brunskill. “Policy Gradient Methods for Reinforcement Learning with Function Approximation and Action-Dependent Baselines.” arXiv: [1706.06643 \[cs\]](https://arxiv.org/abs/1706.06643). (Jun. 20, 2017), [Online]. Available: <http://arxiv.org/abs/1706.06643> (visited on 01/06/2026), pre-published.
- [44] F. Xia, A. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. “Gibson Env: Real-World Perception for Embodied Agents.” Comment: Access the code, dataset, and project website at <http://gibsonenv.vision/>. CVPR 2018. arXiv: [1808.10654 \[cs\]](https://arxiv.org/abs/1808.10654). (Aug. 31, 2018), [Online]. Available: <http://arxiv.org/abs/1808.10654> (visited on 01/06/2026), pre-published.
- [45] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.” arXiv: [2207.02696 \[cs\]](https://arxiv.org/abs/2207.02696). (Jul. 6, 2022), [Online]. Available: <http://arxiv.org/abs/2207.02696> (visited on 01/06/2026), pre-published.
- [46] A. Chang *et al.* “Matterport3D: Learning from RGB-D Data in Indoor Environments.” arXiv: [1709.06158 \[cs\]](https://arxiv.org/abs/1709.06158). (Sep. 18, 2017), [Online]. Available: <http://arxiv.org/abs/1709.06158> (visited on 01/06/2026), pre-published.
- [47] J. Jiang, Y. Zhu, Z. Wu, and J. Song, “DualMap: Online Open-Vocabulary Semantic Mapping for Natural Language Navigation in Dynamic Changing Scenes,” *IEEE Robotics and Automation Letters*, vol. 10, no. 12, pp. 12612–12619, Dec. 2025, Comment: 14 pages, 14 figures. Published in IEEE Robotics and Automation Letters (RA-L), 2025. Code: <https://github.com/Eku127/DualMap> Project page: <https://eku127.github.io/DualMap/>, ISSN: 2377-3766, 2377-3774. DOI: [10.1109/LRA.2025.3621942](https://doi.org/10.1109/LRA.2025.3621942). arXiv: [2506.01950 \[cs\]](https://arxiv.org/abs/2506.01950). [Online]. Available: <http://arxiv.org/abs/2506.01950> (visited on 01/07/2026).
- [48] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. “YOLO-World: Real-Time Open-Vocabulary Object Detection.” Comment: Work still in progress. Code & models are available at: <https://github.com/AILab-CVC/YOLO-World>. arXiv: [2401.17270 \[cs\]](https://arxiv.org/abs/2401.17270). (Feb. 22, 2024), [Online]. Available: <http://arxiv.org/abs/2401.17270> (visited on 01/08/2026), pre-published.

- [49] R. Kabir, Y. Watanobe, M. R. Islam, and K. Naruse. “Enhanced Robot Motion Block of A-star Algorithm for Robotic Path Planning,” arXiv.org. (Dec. 25, 2023), [Online]. Available: <https://arxiv.org/abs/2312.15738v1> (visited on 01/08/2026).
- [50] M. Cherti *et al.* “Reproducible scaling laws for contrastive language-image learning,” arXiv.org. (Dec. 14, 2022), [Online]. Available: <https://arxiv.org/abs/2212.07143v2> (visited on 01/08/2026).
- [51] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. “LERF: Language Embedded Radiance Fields.” Comment: Project website can be found at <https://lerf.io>. arXiv: [2303.09553 \[cs\]](https://arxiv.org/abs/2303.09553). (Mar. 16, 2023), [Online]. Available: <http://arxiv.org/abs/2303.09553> (visited on 01/08/2026), pre-published.
- [52] M. Oquab *et al.* “DINOv2: Learning Robust Visual Features without Supervision.” arXiv: [2304.07193 \[cs\]](https://arxiv.org/abs/2304.07193). (Feb. 2, 2024), [Online]. Available: <http://arxiv.org/abs/2304.07193> (visited on 01/08/2026), pre-published.
- [53] X. Zou *et al.* “Segment Everything Everywhere All at Once.” arXiv: [2304.06718 \[cs\]](https://arxiv.org/abs/2304.06718). (Jul. 11, 2023), [Online]. Available: <http://arxiv.org/abs/2304.06718> (visited on 12/11/2025), pre-published.
- [54] S. Schwaiger, S. Thalhammer, W. Wöber, and G. Steinbauer-Wagner. “OTAS: Open-vocabulary Token Alignment for Outdoor Segmentation.” arXiv: [2507.08851 \[cs\]](https://arxiv.org/abs/2507.08851). (Sep. 22, 2025), [Online]. Available: <http://arxiv.org/abs/2507.08851> (visited on 01/09/2026), pre-published.
- [55] N. Ravi *et al.* “SAM 2: Segment Anything in Images and Videos.” Comment: Website: <https://ai.meta.com/sam2>. arXiv: [2408.00714 \[cs\]](https://arxiv.org/abs/2408.00714). (Oct. 28, 2024), [Online]. Available: <http://arxiv.org/abs/2408.00714> (visited on 01/09/2026), pre-published.
- [56] D. Maggio *et al.* “Clio: Real-time Task-Driven Open-Set 3D Scene Graphs.” version 3. arXiv: [2404.13696 \[cs\]](https://arxiv.org/abs/2404.13696). (Apr. 29, 2024), [Online]. Available: <http://arxiv.org/abs/2404.13696> (visited on 01/08/2026), pre-published.
- [57] C. Westphal, S. Hailes, and M. Musolesi. “A Generalized Information Bottleneck Theory of Deep Learning.” arXiv: [2509.26327 \[cs\]](https://arxiv.org/abs/2509.26327). (Oct. 14, 2025), [Online]. Available: <http://arxiv.org/abs/2509.26327> (visited on 01/10/2026), pre-published.
- [58] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.” Comment: ECCV 2020 (oral). Project page with videos and code: <http://tancik.com/nerf>. arXiv: [2003.08934 \[cs\]](https://arxiv.org/abs/2003.08934). (Aug. 3, 2020), [Online]. Available: <http://arxiv.org/abs/2003.08934> (visited on 01/11/2026), pre-published.
- [59] D. F. Crouse, “On implementing 2D rectangular assignment algorithms,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1679–1696, Aug. 2016, ISSN: 1557-9603. DOI: [10.1109/TAES.2016.140952](https://doi.org/10.1109/TAES.2016.140952). [Online]. Available: <https://ieeexplore.ieee.org/document/7738348> (visited on 01/11/2026).

- [60] M. Ranzinger, G. Heinrich, J. Kautz, and P. Molchanov. “AM-RADIO: Agglomerative Vision Foundation Model – Reduce All Domains Into One.” Comment: CVPR 2024 Version 3: CVPR Camera Ready, reconfigured full paper, table 1 is now more comprehensive Version 2: Added more acknowledgements and updated table 7 with more recent results. Ensured that the link in the abstract to our code is working properly Version 3: Fix broken hyperlinks. arXiv: [2312.06709 \[cs\]](https://arxiv.org/abs/2312.06709). (Apr. 30, 2024), [Online]. Available: <http://arxiv.org/abs/2312.06709> (visited on 01/11/2026), pre-published.
- [61] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. “Sigmoid Loss for Language Image Pre-Training.” Comment: ICCV’23 Oral. arXiv v2: fix typo in pseudocode; v3: clarify t vs t’ init; v4: add SigLIP Base, Large, Shape-Optimized 400M results. Models released at: https://github.com/google-research/big_vision. Xiaohua and Lucas contributed equally. arXiv: [2303.15343 \[cs\]](https://arxiv.org/abs/2303.15343). (Sep. 27, 2023), [Online]. Available: <http://arxiv.org/abs/2303.15343> (visited on 01/11/2026), pre-published.
- [62] S. Macenski, T. Moore, D. Lu, A. Merzlyakov, and M. Ferguson, “From the Desks of ROS Maintainers: A Survey of Modern & Capable Mobile Robotics Algorithms in the Robot Operating System 2,” *Robotics and Autonomous Systems*, vol. 168, p. 104493, Oct. 2023, ISSN: 09218890. DOI: [10.1016/j.robot.2023.104493](https://doi.org/10.1016/j.robot.2023.104493). arXiv: [2307.15236 \[cs\]](https://arxiv.org/abs/2307.15236). [Online]. Available: <http://arxiv.org/abs/2307.15236> (visited on 01/13/2026).
- [63] V. Makoviychuk et al. “Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning.” Comment: tech report on isaac-gym. arXiv: [2108.10470 \[cs\]](https://arxiv.org/abs/2108.10470). (Aug. 25, 2021), [Online]. Available: <http://arxiv.org/abs/2108.10470> (visited on 01/14/2026), pre-published.
- [64] S. Macenski and I. Jambrecic, “SLAM Toolbox: SLAM for the dynamic world,” *Journal of Open Source Software*, vol. 6, p. 2783, May 13, 2021. DOI: [10.21105/joss.02783](https://doi.org/10.21105/joss.02783).
- [65] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Jul. 1997, pp. 146–151. DOI: [10.1109/CIRA.1997.613851](https://doi.org/10.1109/CIRA.1997.613851). [Online]. Available: <https://ieeexplore.ieee.org/document/613851/authors> (visited on 01/14/2026).
- [66] G. Sharir, A. Noy, and L. Zelnik-Manor. “An Image is Worth 16x16 Words, What is a Video Worth?” arXiv: [2103.13915 \[cs\]](https://arxiv.org/abs/2103.13915). (May 27, 2021), [Online]. Available: <http://arxiv.org/abs/2103.13915> (visited on 01/15/2026), pre-published.
- [67] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes.” arXiv: [1702.04405 \[cs\]](https://arxiv.org/abs/1702.04405). (Apr. 11, 2017), [Online]. Available: <http://arxiv.org/abs/1702.04405> (visited on 01/16/2026), pre-published.
- [68] “Figure 2. Downsampling a point cloud with voxel grid filtering..” ResearchGate. (), [Online]. Available: https://www.researchgate.net/figure/Downsampling-a-point-cloud-with-voxel-grid-filtering_fig2_346331400 (visited on 01/16/2026).

- [69] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3D Point cloud based object maps for household environments,” *Robotics and Autonomous Systems*, Semantic Knowledge in Robotics, vol. 56, no. 11, pp. 927–941, Nov. 30, 2008, ISSN: 0921-8890. DOI: [10.1016/j.robot.2008.08.005](https://doi.org/10.1016/j.robot.2008.08.005). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0921889008001140> (visited on 01/17/2026).
- [70] (“PDF) Robust Statistics: The Approach Based on Influence Functions,” ResearchGate. (), [Online]. Available: https://www.researchgate.net/publication/346630632_Robust_Statistics_The_Approach_Based_on_Influence_Functions (visited on 01/17/2026).
- [71] M. Minderer, A. Gritsenko, and N. Houlsby. “Scaling Open-Vocabulary Object Detection.” arXiv: [2306.09683 \[cs\]](https://arxiv.org/abs/2306.09683). (May 22, 2024), [Online]. Available: <http://arxiv.org/abs/2306.09683> (visited on 01/17/2026), pre-published.
- [72] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander. “Joint 3D Proposal Generation and Object Detection from View Aggregation.” Comment: For any inquiries contact aharakeh(at)uwaterloo(dot)ca. arXiv: [1712.02294 \[cs\]](https://arxiv.org/abs/1712.02294). (Jul. 12, 2018), [Online]. Available: <http://arxiv.org/abs/1712.02294> (visited on 01/17/2026), pre-published.
- [73] J. Meier, L. Scalerandi, O. Dhaouadi, J. Kaiser, N. Araslanov, and D. Cremers. “CARLA Drone: Monocular 3D Object Detection from a Different Perspective.” version 2. arXiv: [2408.11958 \[cs\]](https://arxiv.org/abs/2408.11958). (Oct. 21, 2024), [Online]. Available: <http://arxiv.org/abs/2408.11958> (visited on 01/17/2026), pre-published.
- [74] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You Only Look Once: Unified, Real-Time Object Detection.” arXiv: [1506.02640 \[cs\]](https://arxiv.org/abs/1506.02640). (May 9, 2016), [Online]. Available: <http://arxiv.org/abs/1506.02640> (visited on 01/17/2026), pre-published.
- [75] A. Gupta, P. Dollár, and R. Girshick. “LVIS: A Dataset for Large Vocabulary Instance Segmentation.” Comment: Extension of the CVPR’19 paper describing release v0.5, the LVIS Challenge, and baseline results. arXiv: [1908.03195 \[cs\]](https://arxiv.org/abs/1908.03195). (Sep. 15, 2019), [Online]. Available: <http://arxiv.org/abs/1908.03195> (visited on 01/17/2026), pre-published.
- [76] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in Python* (Springer Tracts in Advanced Robotics). Cham: Springer International Publishing, 2023, vol. 146, ISBN: 978-3-031-06468-5 978-3-031-06469-2. DOI: [10.1007/978-3-031-06469-2](https://doi.org/10.1007/978-3-031-06469-2). [Online]. Available: <https://link.springer.com/10.1007/978-3-031-06469-2> (visited on 01/18/2026).
- [77] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000, ISSN: 1939-3539. DOI: [10.1109/34.888718](https://doi.org/10.1109/34.888718). [Online]. Available: <https://ieeexplore.ieee.org/document/888718/citations> (visited on 01/18/2026).
- [78] A. Dedieu, G. Zhou, D. George, and M. Lazaro-Gredilla. “Learning noisy-OR Bayesian Networks with Max-Product Belief Propagation.” arXiv: [2302.00099 \[cs\]](https://arxiv.org/abs/2302.00099). (Jan. 31, 2023), [Online]. Available: <http://arxiv.org/abs/2302.00099> (visited on 01/19/2026), pre-published.

- [79] M. Colledanchise and L. Natale, “On the Implementation of Behavior Trees in Robotics,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5929–5936, Jul. 2021, ISSN: 2377-3766, 2377-3774. DOI: [10.1109/LRA.2021.3087442](https://doi.org/10.1109/LRA.2021.3087442). arXiv: [2106.15227](https://arxiv.org/abs/2106.15227) [cs]. [Online]. Available: <http://arxiv.org/abs/2106.15227> (visited on 01/19/2026).
- [80] Z. Zhou et al., “Towards Building AI-CPS with NVIDIA Isaac Sim: An Industrial Benchmark and Case Study for Robotics Manipulation,” in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, Apr. 14, 2024, pp. 263–274. DOI: [10.1145/3639477.3639740](https://doi.org/10.1145/3639477.3639740). arXiv: [2308.00055](https://arxiv.org/abs/2308.00055) [cs]. [Online]. Available: <http://arxiv.org/abs/2308.00055> (visited on 01/19/2026).
- [81] S. Salimpour, J. Peña-Queralta, D. Paez-Granados, J. Heikkonen, and T. Westerlund, “Sim-to-Real Transfer for Mobile Robots with Reinforcement Learning: From NVIDIA Isaac Sim to Gazebo and Real ROS 2 Robots.” arXiv: [2501.02902](https://arxiv.org/abs/2501.02902) [cs]. (Jan. 6, 2025), [Online]. Available: <http://arxiv.org/abs/2501.02902> (visited on 01/19/2026), pre-published.
- [82] S. Lee, T. Kim, J. Chae, and K.-J. Park. “Optimizing ROS 2 Communication for Wireless Robotic Systems.” version 1. Comment: 10 pages, 8 figures. arXiv: [2508.11366](https://arxiv.org/abs/2508.11366) [cs]. (Aug. 15, 2025), [Online]. Available: <http://arxiv.org/abs/2508.11366> (visited on 01/21/2026), pre-published.
- [83] K. Zakka et al. “MuJoCo Playground.” arXiv: [2502.08844](https://arxiv.org/abs/2502.08844) [cs]. (Feb. 12, 2025), [Online]. Available: <http://arxiv.org/abs/2502.08844> (visited on 01/19/2026), pre-published.
- [84] N. Koenig and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Sep. 2004, 2149–2154 vol.3. DOI: [10.1109/IROS.2004.1389727](https://doi.org/10.1109/IROS.2004.1389727). [Online]. Available: <https://ieeexplore.ieee.org/document/1389727> (visited on 01/19/2026).
- [85] X. Puig et al. “Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots.” Comment: Project page: <http://aihabitat.org/habitat3>. arXiv: [2310.13724](https://arxiv.org/abs/2310.13724) [cs]. (Oct. 19, 2023), [Online]. Available: <http://arxiv.org/abs/2310.13724> (visited on 01/20/2026), pre-published.
- [86] S. K. Ramakrishnan et al. “Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI.” Comment: 21 pages, 14 figures. arXiv: [2109.08238](https://arxiv.org/abs/2109.08238) [cs]. (Sep. 16, 2021), [Online]. Available: <http://arxiv.org/abs/2109.08238> (visited on 01/20/2026), pre-published.
- [87] S. Macenski, A. Soragna, M. Carroll, and Z. Ge. “Impact of ROS 2 Node Composition in Robotic Systems.” Comment: IEEE Robotics and Automation Letters, 2023. arXiv: [2305.09933](https://arxiv.org/abs/2305.09933) [cs]. (May 17, 2023), [Online]. Available: <http://arxiv.org/abs/2305.09933> (visited on 01/20/2026), pre-published.
- [88] R. Sapkota and M. Karkee. “Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 and YOLOv5 Object Detectors for Computer Vision and Pattern Recognition.” arXiv: [2510.09653](https://arxiv.org/abs/2510.09653) [cs]. (Oct. 15, 2025), [Online]. Available: <http://arxiv.org/abs/2510.09653> (visited on 01/20/2026), pre-published.

- [89] D. Li, J. Li, H. Le, G. Wang, S. Savarese, and S. C. H. Hoi. “LAVIS: A Library for Language-Vision Intelligence.” Comment: Preprint of LAVIS technical report. arXiv: [2209.09019 \[cs\]](https://arxiv.org/abs/2209.09019). (Sep. 15, 2022), [Online]. Available: <http://arxiv.org/abs/2209.09019> (visited on 01/20/2026), pre-published.

List of Figures

Figure 1	Overview of the SAGE system architecture for open-vocabulary semantic exploration. RGB-D observations and robot poses are processed by three parallel modules: (a) a persistent semantic mapping backend (OpenFusion [35]) that maintains a semantic 3D map, (b) a semantic frontier exploration module that scores geometric frontiers based on language-guided relevance, and (c) a promptable zero-shot detection pipeline for object hypotheses. All semantic hypotheses are represented as graph nodes, filtered by relevance maps, and fused using a multi-source fusion strategy. A behavior tree orchestrates exploration, verification, and navigation actions, while low-level motion planning and execution are handled by the ROS 2 Navigation Stack [62, 63].	30
Figure 2	Overview of the map representations used for semantic frontier exploration. The SLAM map is used for localization and navigation, while the exploration map encodes task-specific explored and unexplored regions for frontier detection [64]. An inflated map is used to suppress narrow structures and reduce spurious frontiers. The resulting frontier map is combined with a semantic value map to prioritize exploration toward semantically relevant regions.	31
Figure 3	Example of frontier detection on the exploration occupancy grid. Frontiers are identified as free cells adjacent to unknown space and clustered into spatially contiguous regions, with centroids serving as candidate exploration targets.	33
Figure 4	Conceptual visualization of the gradient-ascent analogy for semantic frontier exploration. Semantic relevance is illustrated as a folded reward surface, where frontier and memory graph nodes are elevated according to their semantic scores at position (x, y) . Geometric frontiers constrain feasible ascent directions under spatial obstacles.	36
Figure 5	ROS 2 value map generation pipeline using the BLIP-2 vision-language model for computing image-text cosine similarity.	37
Figure 6	Image-text cosine similarity computation using the BLIP-2 ITC head. An RGB observation is embedded by the visual encoder and compared against a user-defined text prompt in a shared semantic embedding space [27].	38
Figure 7	Example value maps generated using BLIP-2 for different text prompts. The value maps highlight regions of high semantic relevance to the prompts “Bed”, “TV”, and “A door leading to a bed”.	40

Figure 8	ROS 2 integration of OpenFusion for persistent semantic 3D mapping. RGB images, depth images, and LiDAR-based SLAM poses are synchronized and filtered before integration into a global TSDF volume. On-demand semantic queries generate semantic and panoptic point clouds, which are further processed to extract object-level hypotheses for the fusion strategy.	41
Figure 9	Promptable open-vocabulary object detection and 3D hypothesis generation pipeline using YOLO-E [42].	46
Figure 10	YOLO-E detection to graph node 3D localization	47
Figure 11	Overview of the fusion pipeline, illustrating how exploration frontiers, persistent memory nodes, and promptable detection hypotheses are filtered, weighted, and fused into a unified graph representation used for navigation decisions.	49
Figure 12	Illustration of relevance-based graph node suppression over time. At $t=0$, a graph node is generated within the robot's current field of view. As the robot observes the region ($t=1$), the relevance map accumulates visibility evidence, causing the node to be suppressed once the area is deemed sufficiently observed. At $t=2$, only graph nodes outside previously observed regions or associated with high-confidence detections remain active, preventing oscillatory exploration while preserving strong semantic hypotheses.	53
Figure 13	High-level behavior tree flowchart for BT-based semantic-guided exploration. The robot alternates between detection and exploration, navigating to detected objects when confident and otherwise exploring unobserved frontiers until the search terminates.	55
Figure 14	Evaluation pipeline used to compute SR and SPL. Recorded RGB-D observations are converted into a semantic point cloud using OpenFusion, target object clusters are extracted, and the geodesic shortest path is computed via the ROS 2 Navigation2 global planner for metric evaluation [62, 85].	59
Figure 15	TODO: Replace with relevant figure for Experiment 2	65
Figure 16	TODO: Replace with relevant figure for Experiment 2	65
Figure 17	TODO: Replace with relevant figure for Experiment 2	66
Figure 18	TODO: Replace with relevant figure for Experiment 4	67
Figure 19	TODO: Replace with relevant figure for Experiment 4	67
Figure 20	TODO: Replace with relevant figure for Experiment 3	67
Figure 21	TODO: Replace with relevant figure for Experiment 3	67
Figure 22	TODO: Replace with relevant figure for Experiment 3	67
Figure 23	TODO: Replace with relevant figure for Experiment 3	67
Figure 24	TODO: Replace with relevant figure for Experiment 4	68
Figure 25	TODO: Replace with relevant figure for Experiment 4	68
Figure 26	TODO: Replace with relevant figure for Experiment 4	69
Figure 27	Behavior Tree structure of the proposed hybrid semantic exploration system. The BT alternates between detection and exploration branches based on object detection status, guiding the robot's navigation and observation strategies.	85

List of Tables

Table 1	Overview of semantic zero-shot and trained exploration approaches. All of these methods are capable of navigating to a goal described in natural language, however, they differ in zero-shot applicability to new scenes and real-time capability.	2
Table 2	This table provides an overview of representative persistent or memory-based semantic mapping approaches, comparing their training requirements, real-time capability, underlying memory representations, and the extent to which semantic memory is integrated into exploration or planning.	4
Table 3	This table summarizes key limitations of existing semantic exploration frameworks, providing representative example works for each limitation and outlining their practical implications for autonomous navigation and exploration.	5
Table 4	Design patterns and limitations of foundation-model-based semantic exploration frameworks. The table compares how different methods obtain semantic signals, integrate them with geometric exploration, handle uncertainty over time, and represent semantic information, revealing recurring failure modes that motivate the need for persistent semantic memory and belief revision.	15
Table 5	Comparison of semantic memory representations for exploration. The table highlights how different methods store semantic information, update it over time, and whether they support belief revision.	19
Table 6	Comparison of semantic scene reconstruction methods. The table contrasts spatial representations, semantic abstractions, temporal update strategies, and the degree of online integration with SLAM systems. Zero-shot refers to deployment in unseen scenes.	22
Table 7	Comparison of open-vocabulary semantic exploration and mapping methods. The table highlights differences in exploration strategy, semantic persistence, zero-shot generalization, and explicit exploration-exploitation control.	27
Table 8	Comparison of object detection and segmentation models with respect to zero-shot capability, prompt modality, output representation, and real-time suitability. Real-time capability is defined as achieving at least 10 Hz inference on a single GPU, which is sufficient for low-dynamic robotic exploration tasks.	44
Table 9	Abstracted confusion-matrix mapping for detection robustness evaluation.	62
Table 10	Detection performance metrics across fusion strategies and thresholds. Arrows indicate whether higher (\uparrow) or lower (\downarrow) values are better. Best values per threshold τ are highlighted in bold. TODO: Check numbers. May need to be updated.	68

List of source codes

A Methodology details

A.1 Behavior Tree Structure



Figure 27: Behavior Tree structure of the proposed hybrid semantic exploration system. The BT alternates between detection and exploration branches based on object detection status, guiding the robot's navigation and observation strategies.