



¹Nokia
Bell Labs



²Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
⁴BSc Computer Engineering [CE]



³TopoLynx
Ltd.

Semantic Navigation Agnostic to Map-Robot Variations

László Freund^{1,2,4}
author
BSc Student
Nokia Bell Labs

Márk Czimber^{2,3,4}
author
BSc Student
HUN-REN Sztaki

Domokos Kiss, PhD²
supervisor
Senior Lecturer

Dániel Hadházi²
supervisor
Scientific Assistant

Gábor Sörös, PhD¹
supervisor
Nokia Bell Labs
Senior Research Scientist

Kornél Czimber, PhD³
supervisor
University of Sopron
Associate Professor, Vice Dean

2024, Scientific Student Conference

Abstract

Building autonomous systems that scale in the real world, adapt to human interaction, and evolve with technological progress remains a significant challenge even today. This encouraged us to explore the possibilities. We take on robotic navigation as a multidisciplinary challenge that requires technological solutions from various domains, including spatial computing, control theory, and semantic understanding. More specifically we aim to create a general architecture for commanding robots using natural language.

We developed a semantic navigation framework for robots, independent from spatial representation and robot configuration. Once a map is constructed, semantic features provided by a foundation model are added to the detected objects to handle complex natural language queries, for example: “Identify the object where cold beverages are kept”. We create a sparse, skeleton graph of the map capturing the underlying structure of the scene. We introduce three novel contributions: (a) a post-processing algorithm that enhances sparse skeleton graphs to produce cleaner, more structured representations, (b) a raycasting-based method that enables safe path generation specifically for ground robots, and (c) an algorithm facilitating efficient exploration of unknown environments, while incrementally building the topological graph.

We partition the graph into semantically meaningful regions (e.g., rooms) and assign contextual labels using a Large Language Model. The graph’s nodes serve as strategic waypoints, providing navigational guidance for the agent through the environment. We employ reinforcement learning to develop a local policy for navigating between these points, making our framework invariant to robot configurations. To reduce the iteration cycle during development, we implement and showcase our system in a photorealistic simulation environment with an emphasis on seamless transfer to real-world deployment.

We conduct experiments to evaluate the efficacy of our proposed solution. We demonstrate that our skeleton graph construction approach represents a viable solution for both comprehensive topological representation and efficient ground robot path planning, while achieving competitive performance for aerial vehicle navigation. To comprehensively evaluate our pipeline, we conducted two distinct case studies that assess both its simulated and real-world performance.

Our novelty lies in the unique integration and improved collaboration of distinct, state-of-the-art research achievements. In essence, our architecture equips robots to navigate between discoverable objects in the environment, instructed by human language queries. Our focus is on modularity and further future extensibility, along with developments in the field of robotics.

Kivonat

A technológiával lépést tartó, emberi interakcióra tervezett, valós környezetben működő autonóm rendszerek fejlesztése a mai napig nagy kihívást jelent. Ez ösztönzött minket, hogy feltárjuk a lehetőségeket. A robot navigációt választottuk, mint multidiszciplináris kihívás, mely számos tudományterületről igényel technológiai megoldásokat, beleértve a térmodellezést, a vezérlést és a szemantikus megértést. Konkrétan célunk egy általános architektúra létrehozása, amely természetes nyelv használatával vezérli a robotokat.

Kifejlesztettünk egy általánosított, térábrázolástól és robotkonfigurációtól független navigációs keretrendszert robotok számára. Térképet készítettünk a környezetről, és az azonosított tárgyakat alapmodell által generált szemantikus jellemzőkkel látjuk el. Ez olyan lekérdezéseket tesz lehetővé, mint például: „Találd meg azt a helyet, ahol hideg italokat tarthatnak”. Létrehozunk a térképről egy ritka vázgráfot a környezet szerkezetének leírására. Három új megoldást mutatunk be: (a) egy utófeldolgozó algoritmust, amely javítja a ritka vázgráfok képességét egy tisztább, strukturáltabb reprezentáció érdekében, (b) egy sugaras vetítés alapú módszert, amely biztonságos útvonalak generálását teszi lehetővé földi robotok számára, és (c) egy algoritmust, amely hatékony felfedezést biztosít ismeretlen környezetekben, miközben fokozatosan építi a topológiai gráfot.

A gráfot jelentéssel bíró klaszterekre (például szobákra) bontjuk, és ezeket egy nagy nyelvi modell segítségével címkézzük fel. A gráf csúcsai stratégiaileg pozícionált környezeti pontokat reprezentálnak, amelyek alapot biztosítanak a tájékozódáshoz. A csomópontok közötti navigációhoz megerősítéses tanulást alkalmazunk, ami függetleníti a keretrendszert a robot konfigurációjától. A fejlesztési ciklus rövidítése érdekében rendszerünket fotórealisztikus szimulációban valósítjuk meg, figyelembe véve a valós környezetbe való későbbi adaptálás lehetőségét is.

Kísérleteket végzünk, hogy értékeljük a javasolt megoldás hatékonyságát. Bemutatjuk, hogy a vázgraf-konstrukciós megközelítésünk mind a topológiai reprezentáció, mind a hatékony földi robot útvonaltervezés szempontjából megfelelő megoldást jelent, miközben versenyképes teljesítményt ér el a légi járművek navigációjában is. A teljes körű értékelés érdekében két különálló esettanulmányt készítettünk, amelyek a megoldás szimulált és valós körülmények közötti teljesítményét egyaránt vizsgálják.

Tudományos eredményünk abban rejlik, hogy egyedi módon kombináljuk a legújabb kutatási eredményeket és javítjuk az együttműködésüket. Az architektúránk lehetővé teszi a robotok számára, hogy nyelvi parancsok alapján navigáljanak a felfedezhető tárgyak között. Alapvető célunk a moduláris felépítés és a jövőbeli bővíthetőség biztosítása, igazodva a robotika területén zajló folyamatos fejlődéshez.

Table of Contents

1	Introduction	5
1.1	Motivation	5
1.2	Domain Introduction	6
1.3	Difficulties	6
1.4	Approach	8
2	Background	9
2.1	Simultaneous Localization and Mapping (SLAM)	9
2.2	Map Representations for Navigation	10
2.3	Foundation Models	12
2.4	Semantic Mapping	13
2.5	Scene Graphs	14
2.6	Robot Navigation	15
2.6.1	Global Navigation	16
2.6.2	Local Navigation	18
2.7	Exploration	19
3	Methodology	20
3.1	Simulation Environment	21
3.2	Map representation and spatial queries	22
3.2.1	Collision detection	22
3.2.2	RayWalking	23
3.3	Topologic graph	24
3.3.1	Post-Processing	25
3.3.2	Exploration	25
3.4	Semantic Mapping	26
3.4.1	Clustering	26
3.4.2	Labeling	27
3.5	Navigation	27
3.6	Language Instructions	29
3.7	Implementation	29
4	Results	30
4.1	Navigation graph experiment	30
4.1.1	Baseline methods	30
4.1.2	Evaluation criteria	31
4.1.3	Results	31
4.2	Graph clustering experiment	34
4.2.1	Evaluation	35
4.3	Case study: Semantic Navigation in Simulated Office Scene	36
4.4	Case study: Scene Graph Construction of Real World Scene	37
5	Discussion and Conclusion	38
5.1	Possibilities	38
5.2	Limitations	39
5.3	Outlook	39
6	References	40

1 Introduction

1.1 Motivation

In a rapidly evolving world, autonomous robots gain prominent attention by aiding humans in offices and hospitals, operating flawlessly on production lines, assisting workflow, and enhancing efficiency and productivity across various industries. A substantial portion of these are mobile robots operating in diverse and crowded environments collaboratively addressing complex scenarios alongside humans. This underscores the necessity for them to comprehend natural language in order to perceive and interpret the world as we do. Robotics is an intriguing and challenging research field, focused on creating machines that can sense, act, and interact with the physical world around us. To tackle this challenge, agents need to understand their surrounding environment, highlighting the importance of research into representation methods as a valuable contribution.

From this domain, navigation is particularly challenging, as the robot must account for all obstacles, adapt to dynamic environments, and handle uncertain sensor data, which makes directional decisions less straightforward. Conventional planning and control methods depend on accurately modeling the environment, the robot’s movement dynamics, and predicting the behavior of other agents [12], often by deducing priors before making control decisions [83, 102]. Addressing these challenges requires a robust approach to construct an environment model that the robot can effectively utilize.

We aim to develop a general framework for commanding robots using natural human language illustrated in Fig. 1. A key focus is on ensuring modularity and scalability, accommodating changes in both robot and environment configurations. We design interchangeable building blocks to enable optimization for a range of use cases. Additionally, the system should efficiently handle maps of varying scales and complexities that reflect human environments, maintaining fast processing times even for large scenes. These capabilities serve as a foundation to enable robots to explore unfamiliar environments while maintaining collision-free navigation.



Figure 1: A comprehensive semantic understanding of indoor scenes necessitates both scene-level and object-level semantic information. Semantic navigation allows robots to be guided by complex, indirect natural language commands. The image illustrates how this system could be effectively deployed in an office setting.

In our thesis, we address the following question and later provide a solution:

How can we design an autonomous robot navigation framework, that efficiently solves complex tasks, explores unknown scenes, or collaborates with humans in shared confined spaces, while remaining agnostic to system-level components such as environment representation or robot type?

1.2 Domain Introduction

Developing such a system requires numerous components, which increase in complexity as the system becomes more sophisticated. These components often involve integrating state-of-the-art methods from multiple research areas to effectively overcome navigation challenges. For each component, several solutions are available, making interchangeability a crucial aspect of the system design. Such complex systems can be decomposed into the following building blocks:

Robot configuration is the physical platform that interacts with its environment through sensors and actuators.

Mapping and Localization enable the robot to construct a map of its surroundings and determine its position within the scene.

Map Representations are data structures that effectively store environmental data to support navigation and fast processing.

Object Recognizers identify and classify objects within the environment, to facilitate collision avoidance and general understanding of object relations.

Language Interpreters process natural language commands to allow human-robot interaction and shared understanding of the surroundings.

Text or Image Generators create explanations of visual content based on given sensor data, aiding in complex scene understanding.

Low-level Controllers enable robots to autonomously solve complex tasks based on local observations, such as controlling joint actuators or avoiding obstacles.

Each subcomponent is supported by extensive literature; choosing the best approach is primarily a design decision, dependent on the specific task or the integrated performance of multiple components.

1.3 Difficulties

The development of robots capable of executing complex tasks in human environments remains a significant challenge, as these systems are both rare and exceedingly difficult to construct. This complexity stems from the need to coordinate and optimize the seamless cooperation between multiple subsystems, each contributing to the robot’s overall functionality.

The real-world problem

Training robots directly in real-world environments is not only costly due to the risk of damaging the robots, but it also poses safety concerns with people nearby, introducing unpredictability, and making it an impractical approach for early development stages.



Figure 2: The photorealistic simulation capabilities provided by Nvidia’s Isaac Sim [77] and the robotic development interface provided by Isaac Lab [31] make a powerful framework for developing and testing robotic algorithms. This integration enables the construction of complex pipelines with high-level components with support for multiple training paradigms, including reinforcement learning (trial and error) and imitation learning (human behavior replication). The framework’s flexibility allows for training across various robot embodiments, with successful simulation performance serving as a prerequisite for real-world deployment [120].

Physically accurate simulation of sensors also presents a challenge as direct deployment of robots in real-world settings is often not feasible. These limitations expose numerous difficulties in accurately replicating real-world conditions within simulated environments. Fortunately, numerous physics-based photorealistic simulators have recently emerged [30, 31, 119, 79, 101, 117, 36], Fig. 2, providing a strong foundation to explore their capabilities and focus more on the broader navigation problem.

Computational Burdens

As a fundamental challenge in robotics, the agent must simultaneously determine its position in the environment, plan its subsequent movements, and compute control inputs for actuator joints, all while processing sensor data in real time - tasks that significantly increase computational demands.

While certain computations can be offloaded to a network, many time-critical and safety-critical features still must be computed onboard, creating significant constraints on system performance and scalability. To enable effective collaboration with humans a shared understanding and communication interface is essential. Foundation models have demonstrated remarkable success in comprehensive semantic understanding, offering a promising solution, though challenges persist, particularly regarding computational demands and memory requirements [59].

Scalability

To enable broad applicability and ensure modularity, it is also essential to assess how effectively a system scales as the configuration parameters are adjusted, such as accommodating larger environments or managing a higher number of agents. Scaling laws for multi-agent systems work in theory [22], but the dynamics change significantly if the environment becomes more complex. A greater number of agents typically necessitates a larger operating space, making it crucial to design a framework that works with a variety of scenes. Multiple robots must be able to scan the surroundings simultaneously, while consistently upholding collision-free navigation regardless of the map’s size.

Robotics Community

Despite the extensive research efforts in robotics, the field still suffers from a scarcity of accessible resources. Many papers are published without accompanying source code, and the available open frameworks are frequently hindered by limited documentation, specific version dependencies, or narrow focus on particular problems. As a result, few generalized models are available to be adapted across diverse settings. Furthermore, building compatibility for both ground and aerial robots from the outset can eliminate the need to restart development processes from scratch. Unfortunately, such considerations are frequently overlooked, introducing challenges for independent robotics research. Managing complexity and enabling rapid prototyping often become barriers, leaving major advancements primarily to larger research groups or companies

Semantics

A key difficulty lies in the hard-to-obtain nature of high-quality semantic data, which often requires extensive manual labeling or sophisticated sensor systems. Open vocabulary systems have gained importance due to their ability to recognize and interpret a wide range of concepts displayed in Fig. 3, extending beyond a fixed set of predefined labels. These systems allow robots to adapt to diverse environments and to collaborate with humans without the need for retraining on every new task or object encountered.

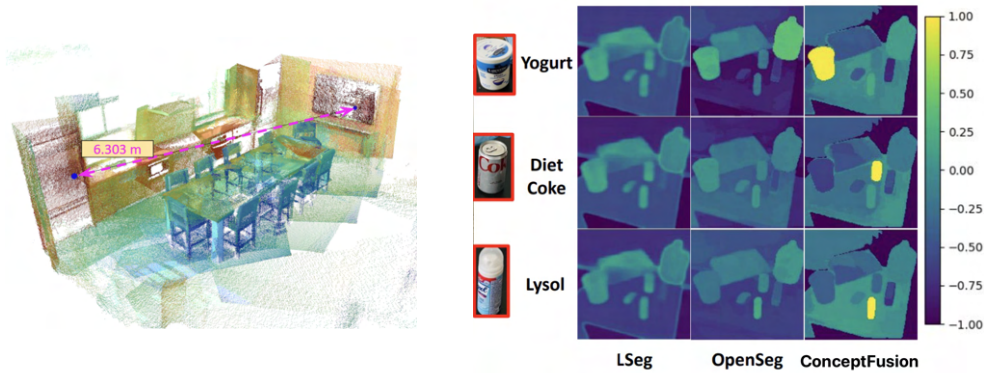


Figure 3: A key benefit of elevating semantic features to 3D is the ability to reason about spatial attributes. This enables semantic spatial queries, such as locating specific objects or measuring distances between items. The figure on the right demonstrates the perception and segmentation of various items, achieved through state-of-the-art foundation models [58], which provide rich semantic data about the surroundings.

1.4 Approach

We aim to integrate these concepts as a sequence of building blocks into a computationally efficient and scalable system that provides a comprehensive, end-to-end framework for semantic robot navigation. Our framework enables seamless interoperability between humans and robots by structuring semantic information hierarchically, facilitating effective collaboration without any loss of contextual understanding. Our focus is on constructing a compact yet detailed structural representation of the scene, allowing for transparent and robust path planning while remaining agnostic to both robot and environment configurations. We demonstrate how our framework can be applied to the online exploration of unknown scenes, autonomously building such representations in real time.

2 Background

How to enable robots to perform tasks based on high-level instructions to achieve navigation in complex environments considering object significance, spatial relationships, and contextual cues?

While robot navigation presents various technical challenges, semantic navigation (navigation based on understanding and interpreting features of the surroundings) stands as a complex interdisciplinary domain, spanning multiple research fields including environmental understanding, human-robot interaction, and spatial reasoning. Research on semantic robot navigation has produced diverse approaches [71, 44, 18, 51, 69], each introducing unique innovations.

The solution typically involves constructing a map representation enriched with semantic features, identifying navigation waypoints, associating these waypoints with relevant semantic entities, conducting waypoint searches, and guiding the robot along the generated path. On the robot side, integrating environmental perception, performing simultaneous localization and mapping, and proactively avoiding potential collisions collectively constitute a highly intricate process. We depend on these foundational components, as they are indispensable due to the inherent complexity of the task, as demonstrated in Fig. 4.

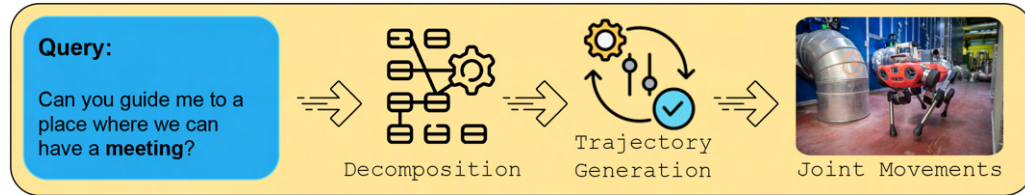


Figure 4: The image explains how difficult is to obtain actuation control from a complex natural language command. It also highlights there are several decomposition steps involved to understand the surroundings and later on plan a trajectory to the goal.

2.1 Simultaneous Localization and Mapping (SLAM)

How does the agent estimate an unknown environment’s map while monitoring its current location?

Localization is a crucial aspect, as the robot must determine its current whereabouts before it can proceed with further planning and task execution. In line with existing literature, this thesis will refer to the robot’s combined position and orientation as its “pose”. In simulation, the need for pose estimation is entirely eliminated by the agent always knowing its exact location from the rendering engine with perfect accuracy. In real-world applications determining the agent’s pose becomes a significant challenge. This is where SLAM comes into play, providing a sophisticated solution to estimate the agent’s pose while simultaneously mapping the environment.

Dense SLAM: Unlike sparse SLAM methods, which are primarily designed for pose estimation [27] and typically yield maps useful for localization, Dense SLAM creates detailed 3D reconstructions beneficial for broader applications, including AR and robotics.

Dense SLAM methods are generally divided into two primary categories: **Frame-centric SLAM** minimizes photometric error across consecutive frames, jointly estimating per-frame depth and frame-to-frame camera motion. Frame-centric approaches [25, 118] are efficient, as individual frames host local rather than global geometry (e.g. depth maps), and are attractive for short sequences, but if a dense global map is needed, it must be constructed on demand by assembling all of these parts which are not necessarily fully consistent over longer trajectories.

In contrast, **Map-centric SLAM** uses a unified 3D representation across the SLAM pipeline, enabling a consistent and integrated system. Compared to purely local frame-to-frame tracking, a map-centric approach leverages global information by tracking against the reconstructed 3D consistent map. One key feature of map-centric SLAM is **loop closure**, which detects when the agent revisits a previously mapped area. This process helps correct drift errors accumulated over time by aligning the current map with past observations, ensuring the consistency of the reconstructed 3D map across large environments.

Classical map-centric approaches often use voxel grids [89, 94] or points [63, 125] as 3D representations. Voxel grids allow fast feature look-up but can suffer from high storage costs and limited adaptability, whereas point-based representations like surfel clouds offer dynamic resolution adjustments and easier loop correction, benefiting online applications like SLAM [125]. Recently, neural network-based maps, such as iMAP [113], demonstrate potential with features like sensible hole filling, though they face challenges with high computational demands [60, 133, 16]. Gaussian splatting SLAM combines advantages of both, using Gaussian distributions to adaptively represent fine details, manage uncertainties, enable efficient map correction and photorealistic representation [80, 132].

2.2 Map Representations for Navigation

How can we effectively construct a digital representation of our physical world?

The chosen representation should ideally allow efficient depth rendering, support color information to enhance scene understanding, outline features, and maintain consistent performance even in large scenes. For navigation, the representation should facilitate streamlined sensor simulation, allowing for fast rendering and rapid collision testing to achieve real-time operation. Various approaches span from volume-based models to advanced techniques [122] with unique advantages and limitations, making the choice of representation highly task-specific and dependent on the application’s requirements (speed, detail, sensor compatibility). Representations discussed by the literature:

- 1. Volume-based models**, such as the voxel model, are commonly derived from video game scene representations and are widely applied in computer vision tasks or 3D object detection due to their ease of implementation [70]. These structures offer a uniform representation of space and are easy to update; however, they inherently pose challenges for extension and scalability. To handle the high memory and computational demands, they require representing and indexing spatial data at different levels of detail with methods such as C-Tree-P [85], Octree [81], or Bounding Volume Hierarchy (BVH) [24]. Downsides arise from choosing the right voxel size as it introduces a hyperparameter; large cells might not provide exact subdivision and small ones increase the search complexity.

2. Point Clouds (Fig. 5 c) can be leveraged to form a large-scale geospatial data structure providing high-quality raw representation of the 3D space with three-dimensional points usually obtained by LiDAR and depth sensors. Point clouds are considered the primary data sources for robot perception in dynamically changing 3D scenes [54]. However, they are sensitive to scale as larger point clouds may require increased memory to store. Point clouds can be obtained by depth estimation techniques, which aim to determine the complete 3D geometry of a static environment solely from a set of provided images [128]. Depth estimation techniques include methods such as **stereo matching** (deriving depth from two or more images) [35] or **structured light** [26] (measuring depth by projecting patterns on surfaces) to create dense point clouds.

3. Mesh-based approaches can achieve photorealism when high-quality textures are applied, as they represent structures using vertices freely positioned in 3D space and faces of varying sizes. Modern graphics pipelines are optimized for triangle mesh processing and offer advantages like easy collision detection, simple partitioning, quality of 3D reconstruction [108], and efficient prefetching to enhance process speed without overwhelming system resources. **Watertight meshes** (continuous surfaces with no gaps or holes) are particularly ideal for rendering, collision detection, and surface analysis [53]. Challenges arise from the non-Euclidean structure, irregularities (varying size, order) [4], and approximating intricate details, such as the geometry of plants.

4. Signed Distance Functions (SDFs) are memory-efficient implicit models, that provide a more flexible approach to represent complex surfaces. They capture 3D geometry and color textures by encoding the distance to the nearest surface at any point in space. It is a widely used approach for 3D reconstruction and can be drift-free for small scenes, which is ideal for navigation [17]. However, fast and robust signed distance field computation is either a performance bottleneck, because of high-resolution fields, or a nearly impossible task because of degeneracies in input meshes [28].

5. Neural Radiance Fields (NeRFs) [82] are novel view synthesis methods that use volume reconstruction with implicit neural scene representations typically via multi-layer perceptrons (MLPs) to learn the geometry and lighting of the 3D scene [40]. They offer highly realistic 3D reconstructions, capturing fine details and complex lighting effects with impressive accuracy. However, they come with several limitations. NeRFs are: **highly specific to each scene**, which limits their adaptability; **difficult to update** dynamically, making real-time modifications challenging; **computationally intensive to render** due to their implicit nature; **non-trivial**, as they involve merging scene-specific neural representations [96].

6. 3D Gaussian Splat (Fig. 5 b) methods represent the scene with a set of anisotropic Gaussians, each containing optical properties: color, opacity, mean and covariance values representing the Gaussian’s position and its ellipsoidal shape [80]. Since they are differentiable volumetric representations, they can be rasterized efficiently by projecting them to 2D. Gaussians are easy to **combine, crop, extend, and modify** because they are separate, localized elements rather than parts of a single continuous function. They allow **fast rendering** without extensive computations like ray marching or neural inference, enabling real-time, high-quality approximation of the physical environment. However, they are still **memory-intensive** in large environments due to the sheer quantity of Gaussians increasing substantially, introducing scalability challenges usually handled by splitting scenes into multiple intersected blocks [23]. Another key limitation is the inability to define splat edges precisely, limiting their use in collision detection.



Figure 5: The images show a simulated scene and its representations using different methods: (left) a photorealistic simulation of the Isaac Sim Office environment; (middle) a 3D Gaussian Splat reconstruction of the scene; (right) a colored point cloud reconstruction of the scene. The reconstructions only approximate the original scene and introduce blurriness (Gaussian Splat) and graininess (point cloud), but objects and structures can still be recognized.

Each of these methods excels in specific scenarios but faces challenges in others, making them highly task-specific. For semantic navigation, a hybrid spatial representation is required, that integrates semantic features enabling intuitive querying through a natural language interface.

2.3 Foundation Models

Can we develop unified, scalable, and adaptable approaches to tackle a broad range of complex tasks through a deep understanding of the environment?

Foundation models are designed to handle **multi-modal input data**—including text, images, audio, and video. By training on such diverse data, they learn broad representations that are adaptable across various tasks with **output** varying depending on the task. For example, they can generate natural language responses, predict labels for image classification, provide dense descriptions for semantic segmentation, or produce 3D object locations for spatial tasks. The flexibility of foundation models allows them to output contextually relevant responses without needing additional retraining, enabling applications across natural language processing, computer vision, and multimodal reasoning [15, 58]. They are mainly used to convert high-dimensional inputs into abstract, structured representations that can be interpreted and manipulated easier [38, 127].

In the era dominated by FMs, image segmentation has undergone significant evolution [131]. Although **YOLO** is not practically a foundation model, it can be leveraged for a wide variety of tasks such as solving the object detection problem framing it as a regression problem to spatially separated bounding boxes and associated class probabilities. Fast YOLO is the fastest general-purpose object detector in the literature and YOLO pushes the state-of-the-art in real-time object detection [99]. Recent advances like **CLIP** (Contrastive Language-Image Pre-Training) obtain queryable semantic information and can be instructed in natural language to predict the most relevant text snippet, given an image, enabling zero-shot (instant) transfer to many existing datasets [95]. Segmentation methods also advanced significantly, for instance, **Segment Anything Model (SAM)** and its subsequent iteration, **SAM 2**, provide highly flexible and efficient segmentation capabilities [65, 98]. These models can generalize well across various datasets and image domains, making them valuable tools for tasks requiring precise pixel-level understanding. They help to bridge the gap between object detection and fine-grained scene understanding by enabling robust and accurate segmentation of individual objects, regions, and parts at multiple granularities.

Large language models, such as **LLama 3** [29] and **GPT-4** [32], demonstrate impressive capabilities in natural language processing. These models can generate coherent, context-aware responses and summarize content. Combined with vision models, they enable multimodal understanding for tasks like image captioning, visual question answering, or visual reasoning, by jointly considering language and scene information. Additionally, **GPT-4o** can process input and provide output in any combination of text, audio, image, and video. Besides their powerful capabilities, we currently lack a clear understanding of how they work when they fail, and what they are even capable of [34].

2.4 Semantic Mapping

How can an autonomous system interpret its surroundings to understand and differentiate between various objects and concepts?

Semantic features are crucial for accurately identifying object characteristics, enabling the clustering of related parts and the separation of unrelated ones. It’s also important to align the robot’s interpretation with how humans perceive the environment, to establish a common understanding of the surroundings. Even if the system can detect and assign labels to objects [66], those labels may not be broad or flexible enough resulting in limitations. The robot may recognize certain objects in specific contexts but may struggle to generalize its understanding to new or more diverse environments. This challenge gives rise to the **open-set multimodal 3D mapping problem**, which states: (1) semantic maps should be open-set, capable of capturing a broad range of concepts at varying levels of detail—for example, recognizing that a “can of soda” also relates to the query “something to drink”; (2) they should be multimodal, allowing queries across as many modalities as the robot or end-users can utilize [58].

Recent advancements in foundation models [127] (large vision-language neural networks) have revolutionized the field by offering robust, generalized solutions for semantic understanding. Their use cases span a variety of applications, including **semantic segmentation** (dividing an image into meaningful parts), **3D scene description** (providing a structured outline of objects within a 3D space), and **zero-shot 3D classification** (identifying 3D objects without prior training in specific examples). Additional applications include **affordance prediction** (estimating how an object might be used or interacted with) and **dynamics prediction** (forecasting changes in object position or state over time) [38]. By leveraging foundational models for object recognition, image segmentation, and feature extraction—capable of interpreting natural language queries—instance-level information and semantic understanding can be extended to 3D by projecting 2D label masks into 3D space [87, 5]. This significantly improved the success rate of executing language query-based navigation tasks [88]. Conceptfusion [58] Fig. 6 utilized these ideas to abstract the scene into a comprehensive language description, allowing it to be fed into large language models (LLMs) to handle complex queries [33].

Inspired by the success of these methods the application of foundational models to extract vision-language features from 2D images and integrate those into a 3D map is quite promising and highly suitable for semantic navigation.

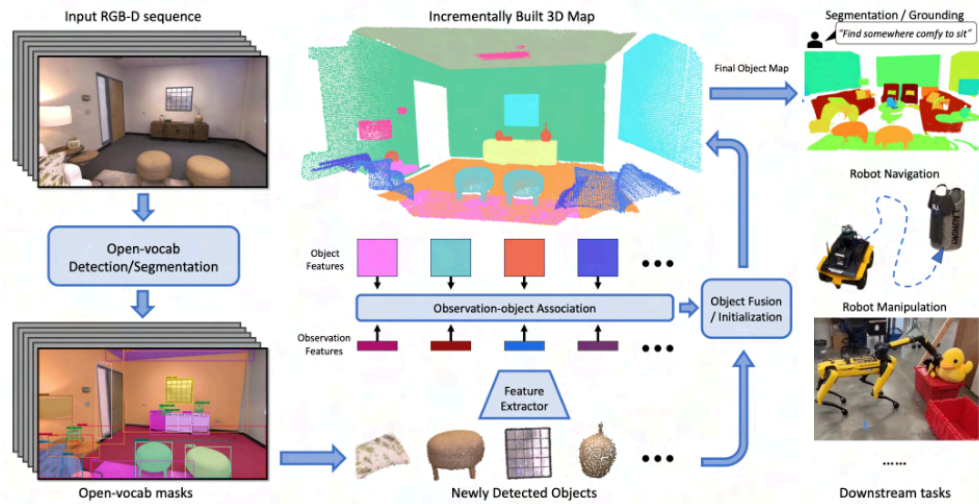


Figure 6: **ConceptFusion** [58] integrates foundation models, combined with SLAM, to generate semantic understanding of environments through open-set, multimodal 3D mapping. First, RGB and depth images are processed, then object masks and feature embeddings aligned at the pixel level are mapped to 3D points to create a dense 3D map. Second, images, text, and audio are encoded into query vectors for language queries without requiring additional training or fine-tuning. With dense 3D mapping and multimodal querying capabilities, it supports zero-shot reasoning across a wide range of concepts.

2.5 Scene Graphs

How can the hierarchical structure of indoor scenes be captured to represent relationships between objects and locations?

This requires higher-level visual understanding and reasoning [19], for example injecting further information into a general map representation, capturing the hierarchical nature of buildings, or assigning semantic features to free-space regions. VLMaps [52] assigns semantic feature embeddings to every point in the point cloud. Aside from redundancy, this approach requires the storage of potentially large feature vectors alongside each voxel, making it storage and memory inefficient.

3D Scene Graphs provide a prominent solution and have recently emerged from computer games as powerful high-level representations of 3D environments. A 3D scene graph is a layered graph where nodes represent spatial concepts at multiple levels of abstraction (from low-level geometry to objects, places, rooms, buildings, etc.) and edges represent relations between concepts. Armeni et al. [7] proposed the use of 3D scene graphs in computer vision by introducing the first algorithms to convert a metric-semantic 3D mesh into a 3D scene graph. Kim et al. [64] reconstructs a 3D scene graph of objects and their relations. Rosinol et al. [103, 104] propose a novel 3D scene graph model that (i) is built directly from sensor data, (ii) includes a subgraph of places (useful for robot navigation), (iii) models objects, rooms, and buildings, and (iv) captures moving entities in the environment. Other works like **RIO** [121] and **SceneGraphFusion** [126] infer objects and relations from point clouds, RGB-D sequences, or object detections.

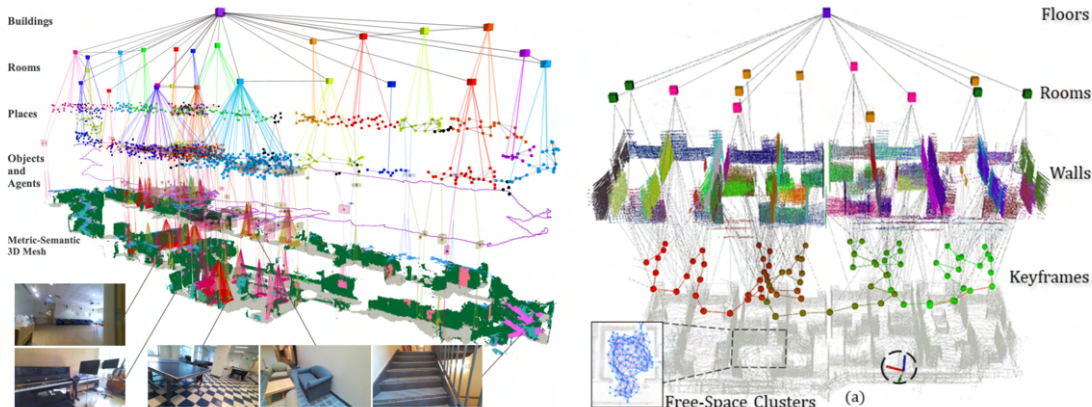


Figure 7: **Hydra** [55] (left) and **S-Graphs** [9, 10] (right) are real-time spatial perception systems that construct the scene graph simultaneously with the mapping process. In their case, leaf nodes in the scene graph depend specifically on the method or map representation, making them less modular and harder to replace.

Hughes et al. construct scene graphs of large-scale environments from sensor data in real time, integrated with the mapping process [55]. Similarly, Bavle et al. introduced an approach to real-time scene graph creation by using keyframes as leaf nodes, clustered into rooms detected through plane segmentation [10]. Their method allows graph construction using only LIDAR measurements. The scene graphs generated by these methods are shown in Fig. 7. Abdelrhman et al. [124] shows that organizing spatial and semantic information in a graph format allows robots to efficiently reason about the layout, object relationships, and potential paths within a scene, making its utilization evident. These structures are further well applicable for large-scale indoor environments and long-horizon task plans instructed with natural language [97].

2.6 Robot Navigation

How can an agent navigate potentially unknown 3D scenes effectively?

Robots typically perceive their environment through sensor fusion and derive an understanding of the scene through specific algorithms [76]. The agent needs to uphold an estimate of its state (pose, velocity, etc) along with a perspective understanding of the surrounding space. To enable scalable navigation, it is often impractical to store the entire map within this state representation. Instead, global waypoints are paired with decomposed, bounded regions allowing local navigation to efficiently manage tasks. Influential methods have introduced the idea of factorizing long-range navigation into two components: long-range collision-free pathfinding and perception-based local robot control [37]. If we consider an efficiently decomposed map, valid routes can be obtained through a global topology search, for instance with A* [50]. Multiple shortest paths can be weighted [73], which offers on-route optimization with real-world constraints. Local navigation generates safe trajectories between waypoints primarily focusing on perception and actuator control.

2.6.1 Global Navigation

How can an agent efficiently plan collision-free trajectories in a mapped scene?

The collision boundaries shift between 2D and 3D based on whether the robot is ground-based or aerial. To enable effective spatial collision checking, it is common practice to decompose the representation into smaller, well-defined regions, using methods that distinguish the local structure or identify the free space within the map [3]. Since the map is often unstructured, determining the appropriate size and spacing of these partitions is rarely straightforward. Another challenge is that the map may be overly complex and filled with potential collision risks, resulting in ambiguous navigation information. This necessitates a clear structure for pathfinding, typically organized as a waypoint graph [3]. However, constructing the graph and establishing edge connections is a non-trivial task. The initial step involves building the node structure by subdividing the free space into discrete regions, with each region representing a distinct segment of the environment.

3D Voxel and 2D Grid methods offer a solution with a uniform representation of the components. The grid map’s primary purpose is to provide a search space for planning [74], thus they are often used for solving tasks like collision checking (Fig. 8 b) or path planning with iterative nature [116]. Their ease of implementation makes them a popular choice in robotics, particularly in environments where quick adaptation and simple data processing are crucial. Current novel approaches implement intersection detection to generate semantic and topological maps on grids [39] or improve the overall mapping performance [100]. However, the use of numerous grid cells can introduce a critical scalability issue by greatly increasing search complexity, especially across large surfaces and multiple 3D levels. This requires sampling from the spatial volume to enable efficient exploration of large, high-dimensional spaces.

Sampling-based planners, such as Probabilistic Roadmaps (PRMs) [62] and Rapidly Exploring Random Trees (RRTs) [67], efficiently solve the scalability problem by approximating the topology of the configuration space by constructing a graph through sampling and connecting points with collision-free local paths [37]. PRM-RL enhances the approach by connecting waypoints where the local agent successfully navigates, rather than relying on straight-line connections. This method allows the reinforcement learning agent to adaptively construct feasible paths within the probabilistic roadmap framework [1]. To weight topology nodes, Brian Ichter et al. propose to learn critical samples and use them more heavily as they offer advantageous choices during navigation [57]. PRM in 3D becomes more complex, as it accounts for an object’s position and orientation [110]. However, a major drawback is that random sampling leads to an unordered structural decomposition and does not guarantee an optimal solution.

Voronoi Diagrams generate points that maximize the distance from obstacles, introducing an intuitive structure to the navigation graph. The Voronoi diagram of a finite set of objects is a fundamental geometric structure that subdivides space into regions, each region consisting of the points that are closer to a given object than to the others [14]. The intersecting boundaries of these regions reveal points that are farthest from the objects at the perimeter. These points can be used as nodes in a waypoint graph, shown in Fig. 8 a. Nico Pietroni et al. generated a grid-shell structure that is based on Voronoi diagrams, which offers better static performances with respect to quad-based grid-shells [92]. A

Voronoi diagram can be defined more generally as the minimization diagram of a finite set of continuous functions (Generalized Voronoi Diagram, GVD). Usually, each of those functions is interpreted as the distance function to an object [42]. Grid-based GVDs are also efficient tools for motion planning in mobile robots [123]. Computation of 3D skeleton diagrams and sparse graphs that maintain the topology and connectivity of the original space can also be achieved by using a combination of 2D Voronoi Diagram-based planning and 3D graphics skeletonization (extracting simplified structure outlines) [90, 111]. The main limitation of this approach is that mapping results often require pre-processing into a specific representation, such as a Euclidean Signed Distance Field (ESDF), to be suitable for GVD generation [21].

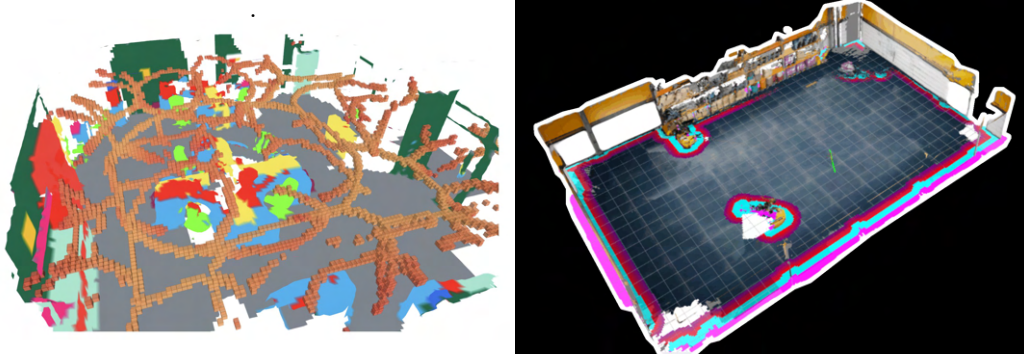


Figure 8: Navigation Graphs generated using two different methods: (a) shows a Generalized Voronoi Diagram (GVD), with orange blocks representing optimal trajectories equidistant from obstacles [55]; (b) shows a grid-based cost map generated by the Nav2 navigation stack, using the Nvblox package. Colored contours around obstacles represent increasing cost levels, which help the robot to plan safer around these obstacles [49].

Raycasting-based methods [21, 43] are suitable for all representations as long as a collision-checking interface is available. They align with the process of exploration in a new dynamic environment, by casting laser beams fired from a point in space along a particular direction. In recent advancements, Xinyi Chen et al. introduced **SkeletonFinder** [21], a method for generating trajectory-independent sparse topological skeleton graphs. Their approach employs an efficient ray sampling and validation mechanism to identify distinctive free space regions, which form the vertices of the skeleton graph. Edges are established based on traversability between adjacent vertices, and a cycle formation scheme is utilized to maintain the graph’s compactness. This method has demonstrated the ability to generate sparse graphs in substantially shorter times compared to state-of-the-art techniques, providing high-quality global planning paths. Experiments conducted in real-world maps further validate its effectiveness in practical scenarios [21], demonstrating that it generates more efficient paths than GVDs and achieves search times up to 20,000 times faster than grid-based methods.

In summary, while grids, PRMs, and Generalized Voronoi Diagrams remain among the most popular approaches, raycasting-based methods stand out due to their efficiency and their inherent alignment with the incremental exploration of indoor scenes, enabling a more general framework.

2.6.2 Local Navigation

How can an agent leverage its perception to navigate safely and efficiently between waypoints?

Local navigation agents use fused local perceptions to estimate their state and make decisions based on these features to generate control signals, allowing for safe and efficient navigation in bounded areas. This often involves avoiding dynamic obstacles, such as humans. Local navigation is approached by different research fields [2]:

- **Traditional computer vision** uses hand-crafted features and algorithms for object detection, requiring domain expertise and extensive manual tuning [61].
- **Traditional control** encompasses a family of controllers, such as Model Predictive Control (MPC) approaches, which directly use an explicit and separately identifiable model of the system to simulate outcomes into the future and learn optimal control derived from learned connections [41, 83].
- **Learning-based techniques**, such as deep and reinforcement learning, obtain features from data or interaction with the environment using neural networks, enabling adaptability, but requiring significant computational resources upon complex tasks. Combining deep- and reinforcement learning addresses critical challenges related to the dimensionality and scalability of perception systems, enabling generalized object detection capabilities [72, 102]

Local navigation must handle collision avoidance and control the agent’s movement on the actuator scale, guided by the global path edges as ground truth directions. Reinforcement learning [114] has become increasingly popular for demonstrating promising performance and success across many complex tasks [47]. However, this remains a highly challenging problem due to the high-dimensional nature of both the observation and action spaces, as well as the underlying complex dynamics [48]. The decision-making process of a robot in reinforcement learning can be discussed from two primary perspectives: model-free and model-based approaches [115].

Model-based reinforcement learning (RL) involves constructing a model of the environment’s dynamics, which allows the agent to predict future states and plan its actions accordingly. This approach is particularly relevant in scenarios where data efficiency is crucial, as it enables the agent to simulate interactions without relying on extensive real-world experiences [84]. On the other hand, **model-free RL** bypasses the need for an explicit model and learns directly from interaction data, often requiring more extensive exploration but allowing for greater flexibility in handling unpredictable or complex environments [93, 8].

Rather than training local navigation agents end-to-end, collision avoidance and actuator control can be organized hierarchically shown in Fig. 9. This decomposition enables the reuse of pre-trained low-level policies across various applications, enhancing versatility and adaptability by abstracting the specific robot configuration at a higher level. Additional benefits include learning over extended temporal transitions, exploring over longer periods, and training and exploring within more semantically meaningful action spaces [86]. With such capabilities, reinforcement learning offers an effective approach for local navigation and control.

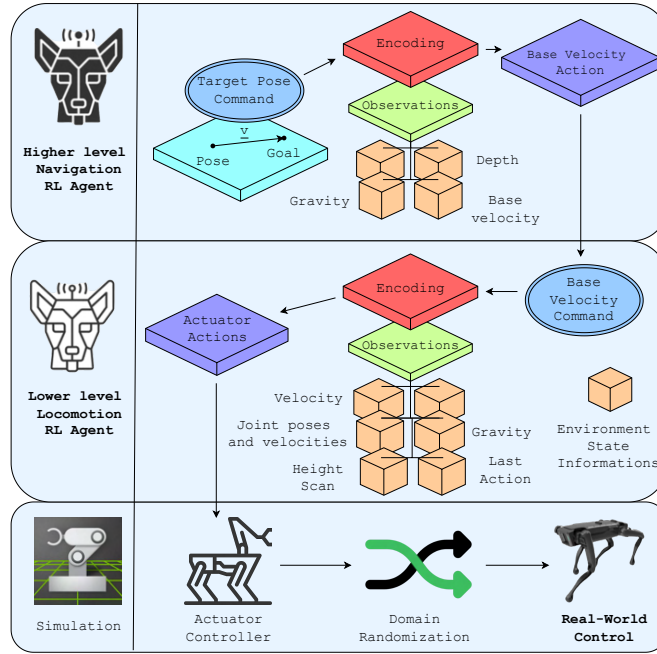


Figure 9: Local navigation organized into a hierarchical decomposition. The high-level agent computes heading velocities based on a target position command, which are then passed to the low-level agent. The low-level outputs joint actions, which are converted to actuator actions through a learned actuator-simulator. Domain randomization supports effective transfer to real-world environments. (icon source: internet).

2.7 Exploration

How can an agent autonomously capture the structure of an unknown environment?

One of the main issues in computer vision is to recover the 3D structure of large obstacle-filled unknown scenes from visual data. A robot actively moves to perceive and incrementally maps its surroundings to gain understanding, thus formulating the exploration problem [107, 78]. Several comprehensive studies have tackled this solution and demonstrated successful results in real-world applications [44, 91]. Traditional 2D LiDAR-based exploration policies [75] aim to find frontiers by parsing their surroundings for uncovered free space. The robot navigates toward these regions, collecting and processing sensor data to incrementally build and refine the map [129]. Frontiers can be prioritized by weighting regions with a higher density of objects [20]. Integrating semantic navigation pipelines with exploration policies enables robots to overtake the mapping of unknown scenes, and significantly reduce the manual effort required to create them. With the anticipated increase in robotic deployments, this capability offers significant potential: mapping multiple scenes and merging them into a shared knowledge base could establish extensive operational environments for robotic systems.

In the next section, we aim to address the demonstrated challenges of semantic navigation through topological scene graph construction and exploration, encompassing the full scope of literature studied in the Section 2 section.

3 Methodology

Constructing a pipeline that enables robots to navigate in an environment based on semantic queries involves several distinct fields, with various specific approaches to choose from. Constructing a complex system without sacrificing generality requires a careful balance of **flexibility** and **modularity** in its design. Each component in the pipeline should be invariant to changing any other component. The vast amount of solutions in each field also encourages us to keep our focus on crucial factors like **computing efficiency**, **scalability**, and **object-oriented design**. With these in mind, we propose a Semantics-Augmented Topologic Inference framework for Navigation, **SatiNav** Fig. 10.

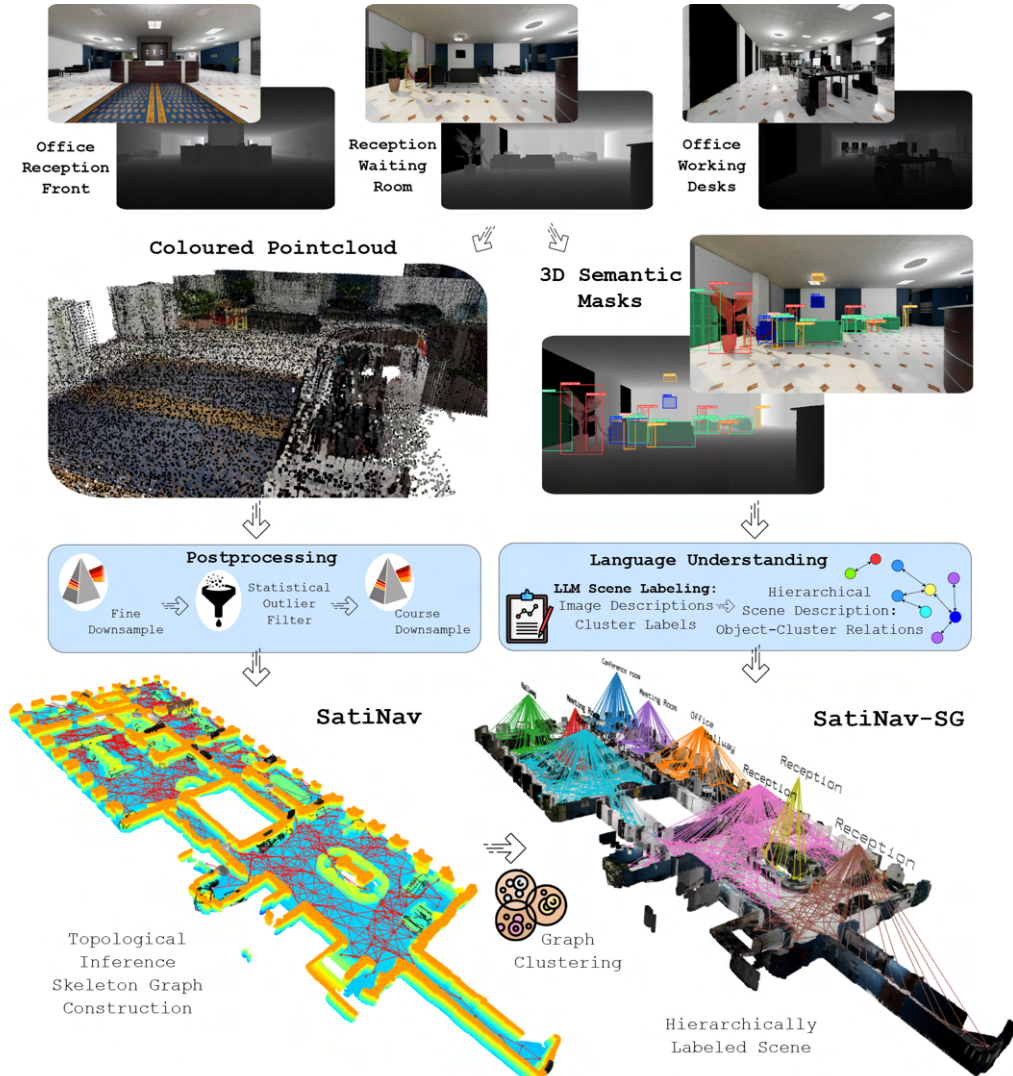


Figure 10: The production pipeline of our proposed framework **SatiNav** (Semantics-Augmented Topologic Inference for Navigation). First, RGB-D frames, along with semantic segmentation masks are projected into 3D space. Next, a topology graph is constructed. The nodes of the graph are organized into clusters, which are then augmented to construct **SatiNav-Scene Graph** for hierarchical description of the scene.

In our workflow (Fig. 10), a map representation of the environment is constructed using ground truth poses provided by a simulation engine. To ensure generality, we capture raw spatial data in a point cloud representation. Semantic features are then integrated using foundation models that project labeled 2D segmentation masks for keyframes onto the point cloud. Semantic labels are assigned based on feature consensus across multiple viewpoints, defining 3D bounding boxes for objects. This setup enables a robot to query object locations and interpret goal destinations in natural language. Next, a comprehensive topological map of free space points is constructed.

We introduce a novel algorithm to generate structurally coherent and efficient topological graphs, supporting efficient online exploration for ground or aerial robots. We partition the graph into clusters, label them based on image descriptions and nearby objects, and adapt it to the robot’s configuration by connecting accessible waypoints. We then apply reinforcement learning to navigate between these points.

3.1 Simulation Environment

We implement our framework in NVIDIA Isaac Sim, a photorealistic simulation engine with a wide variety of tools for robotics projects. Isaac Sim provides realistic physics simulation shown in Fig. 11, high-quality rendering, and AI-based capabilities, making it ideal for developing and testing autonomous systems in virtual environments.



Figure 11: The images show the simulation engine during runtime in the Nvidia Office environment, where (left) multiple ANYmal-C quadrupeds navigate the scene and (right) Nova Carter wheeled robot enjoys an afternoon view.

A manually controlled wheeled robot captures the environment in multiple sessions. RGB-D images are acquired alongside their corresponding position and orientation parameters, allowing for later reconstruction of the environment. For both image extraction and data serialization, we utilize Robot Operating System (ROS) nodes, which we designed to be adaptable for usage in real-world scenes without requiring further modifications. While Isaac Sim offers advanced capabilities, it deliberately incorporates noise in its simulations, adding a layer of realism. These simulated imperfections, especially around transparent surfaces, challenge researchers to develop algorithms and perception models that can handle similar complexities in real-world scenarios. This added difficulty helps improve the robustness and adaptability of systems, preparing them to interpret and respond effectively in diverse and imperfect environments.

We use Nvidia Isaac Lab for robot control, as it provides a comprehensive framework for simulating complex robotic behaviors and handling tasks such as motion planning, sensor data processing, and navigation. Isaac Lab also supports modular development, allowing

us to easily integrate custom algorithms and fine-tune the robot’s performance. Its built-in tools facilitate seamless switching between simulation and real-world deployments, ensuring that the system remains robust and adaptable across different environments.

We conduct our experiments in Isaac Sim’s stock office environment navigating an ANYmal-C [6] quadruped robot, which is one of the many pre-assembled robot configurations available in the simulator. The complexity of our setting amplifies challenges for quadruped robots, such as requiring adaptive locomotion strategies, facing drift in state estimation due to high-impact forces, and limited perception from self-occlusions and sensor field of view. Overcoming these challenges requires planning feasible trajectories based on the robot’s capabilities [51]. If the system can successfully master tasks in simulation, we can begin accurately refining simulation parameters to align with real-world conditions. These parameters are often derived from actuator configurations or from pre-trained neural networks [56].

3.2 Map representation and spatial queries

Our framework is compatible with any map representation as long as it supports collision checking, making it adaptable to a wide range of algorithms, from voxel-based approaches to hyperrealistic Gaussian splatting models. Point clouds serve as our base representation. From a set of posed RGB-D frames, we project depth image values using the pinhole camera model; this method offers an intuitive and robust approach to 3D reconstruction. The poses of the frames are extracted from the simulation, but in reality, a feature-based visual or visual-inertial SLAM method can be utilized. Although the recordings are captured in a simulation environment, the reconstructions exhibit noise due to refractions and simulated camera imperfections. To enable accurate, high-resolution spatial queries, this noise must be filtered out.

We begin by downsampling the point cloud using voxel downsampling with a voxel size of 2.5 cm to remove redundant points and ensure a uniform point density across surfaces. Next, we apply the statistical outlier removal (SOR) algorithm, which filters out points whose averaged distance from their nearest neighbors exceeds the mean by 2.5 standard deviations, effectively eliminating scattered points. To further reduce computational costs in subsequent processing steps, we apply an additional voxel downsampling with a voxel size of 5 cm. This two-stage downsampling approach is justified, as SOR performs better at higher resolutions, but assumes statistically high density near valid points. At this stage, depending on the quality of the initial recordings, we obtain a map that supports efficient spatial queries.

3.2.1 Collision detection

Our algorithms rely on a fundamental collision detection operation for all further map operations. The point cloud data is stored in a KD-Tree [11] data structure, which is optimized for efficient nearest-neighbor search operations, that can be leveraged for ray-casting, the core algorithm for collision detection in point clouds. The KD-Tree is queried to locate the nearest neighbor of a given spatial point within the map. A collision is determined when the distance between the given query point and its nearest neighbor is less than a predetermined threshold value. This mechanism enables efficient raycasting in the point cloud map. By discretizing a ray into sequential points and performing collision checks at each point, we can determine if and where the ray intersects with obstacles in

the environment. This is particularly useful for simulating sensor measurements, checking line-of-sight between positions, and validating trajectories for collision-free navigation. The sampling density along the ray can be adjusted based on the required precision and computational constraints, with denser sampling providing more accurate but computationally intensive results.

3.2.2 RayWalking

Traditional raycasting effectively checks line-of-sight but faces limitations in complex environments for ground robot navigation. We propose RayWalking Fig. 12, a novel algorithm that enables safe-path queries for ground robots, even in challenging multi-floored environments. Unlike conventional raycasting that performs simple line-of-sight checks, RayWalking evaluates whether a traversable path exists between two points that a ground-based robot could safely navigate.

The algorithm operates by first discretizing a ray between two points at a specified resolution, similar to traditional raycasting. However, RayWalking extends this approach by casting additional rays downward from each discretized point to assess ground continuity. For each downward ray, we store the vertical coordinate of its intersection point with the environment. The path is marked as invalid if the elevation difference between consecutive intersection points exceeds a defined threshold, effectively detecting unsurmountable obstacles or drops.

This approach naturally aligns with the physical constraints of walking or ground robot movement and can be easily calibrated by setting the threshold to match the maximum step height of a specific robot configuration. RayWalking successfully handles stairs and surfaces not aligned with the ground plane, as gradual elevation changes within the robot’s climbing capability are considered valid paths. Since the algorithm builds upon the fundamental raycasting operation, it maintains computational efficiency while providing more sophisticated path validation for ground-based navigation.

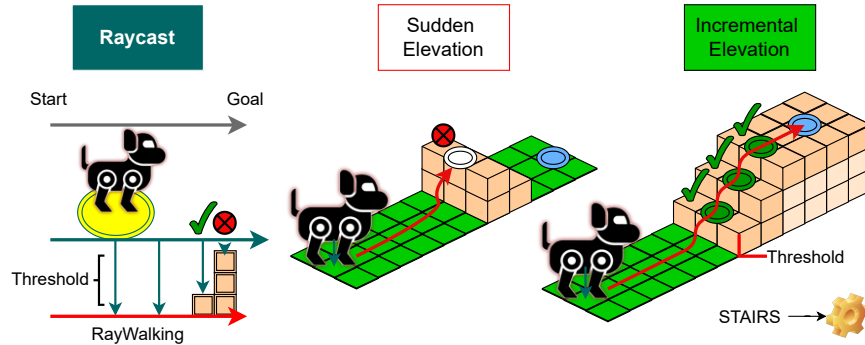


Figure 12: **RayWalking:** A path validation method that evaluates traversability by performing sequential ground-level raycasts along a discretized path, rejecting sudden elevation changes while permitting gradual height adjustments for ground-based navigation. This approach naturally aligns with the physical constraints of walking or ground-based robot movement, enabling straightforward path validation in challenging scenarios such as navigating stairs or handling misaligned maps.

3.3 Topologic graph

Enabling efficient and intuitive spatial querying we construct a sparse, interpretable representation of the environment. This must robustly capture the underlying scene structure to support various downstream tasks. Sparse skeleton graphs, such as Skeleton-Finder, have demonstrated promising results, outperforming the widely-used Generalized Voronoi Diagram-based approaches [21]. The algorithm employs iterative frontier-based exploration of the environment using raycasting, with its 2D implementation detailed in Fig. 13. However, our experiments revealed that the base algorithm alone produced results insufficient for downstream applications. We identified several key limitations:

- Inconsistent node and edge distribution, resulting in regions of both excessive and insufficient density. While sparse edge structures may enhance pathfinding efficiency, they compromise the graph’s interpretability and utility for other applications.
- The topological graph generates paths unsuitable for ground robot navigation
- The method’s potential for unexplored environment exploration remains unaddressed in existing literature.

Addressing these limitations, we present three novel contributions: (a) **SkeletonProcessor**, a post-processing algorithm that enhances sparse skeleton graphs to produce cleaner, more structured representations, (b) a RayWalking-based method that enables safe path generation specifically for ground robots, and (c) **SkeletonExplorer**, an algorithm facilitating efficient exploration of unknown environments, particularly for ground robots.

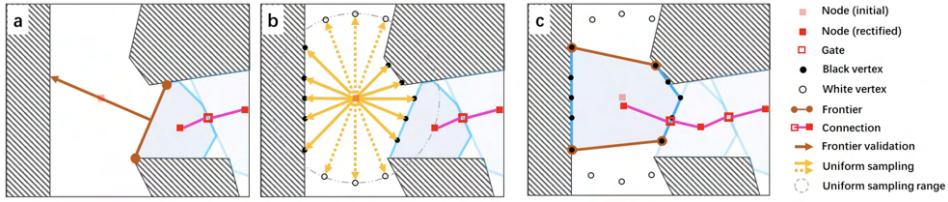


Figure 13: An illustration shows SkeletonFinder’s graph generation algorithm workflow. It shows an example algorithm iteration with frontier validation and node expansion [21]. At the beginning, the input map is set and an initial node is expanded at an arbitrary position within free space. (a) The algorithm is initialized at an arbitrary position and the topologic graph is constructed by conduction **node** expansion operations. During the initial **node** expansion, a few frontiers are identified and pushed into the pending frontiers First-In-First-Out(FIFO) list *Frontiers_{pending}*. Then, the algorithm iteratively grows the skeleton graph in a breadth-first manner by popping out frontier f from *Frontiers_{pending}* and, if possible, expanding a new **node**, by performing raycasting from its center along its normal direction. The figure shows a successful frontier expansion, the raycasting did not detect any obstacles within its range. (b) To expand a **node**, we first perform raycasting from its initial position along uniformly sampled directions up to a truncated distance. For each ray, if no environmental **obstacles** or other **polyhedra** are detected within the truncated distance, a **white vertex** is constructed at the truncated endpoint of the ray, indicating free space not yet reached by any node. Otherwise, a **black vertex** is set at the first detected position on the ray segment. If a ray detects a polyhedron of another node, it indicates that there exists a cycle in the skeleton graph to be closed and the detected polyhedron is stored along with black. Given **black** and **white vertices**, we aim to construct the **polyhedron** of the current node and identify new frontiers f_{new} on the **polyhedron** boundary. The figure shows how **black** and **white vertices** are detected using raycasting. (c) During the **node** expansion, a **polyhedron** is grown while new frontiers are identified and pushed into *Frontiers_{pending}*. Moreover, skeleton graph cycles are formed if necessary.

3.3.1 Post-Processing

SkeletonProcessor aims to enhance the output of **SkeletonFinder** to create a more efficient and consistent graph representation, which can be utilized for ground robot navigation. The algorithm operates in two primary stages addressing distinct challenges: (1) edge optimization and densification and (2) redundant node elimination.

The first stage focuses on edge reconstruction. For each node, the algorithm queries neighboring nodes within a defined “connection” threshold using KD-Tree nearest neighbor search operations. Path validity between node pairs is evaluated using different criteria based on robot type. For aerial robots, traditional raycasting is employed with the robot’s base radius as the collision threshold. For ground robots, the algorithm projects points to 2D at a specified base height and applies RayWalking to verify traversability. Valid queries result in edge connections between nodes. This approach not only enables ground robot navigation in multi-floored environments but also produces a more comprehensive representation by capturing local spatial relationships beyond the basic skeleton structure.

The second stage addresses node redundancy by examining nodes within a “too close” threshold to their neighbors. These nodes are processed in ascending order of proximity. Node merging is performed when either node can functionally replace the other by maintaining all necessary connections. When bidirectional replacement is possible, the merged node position is determined through weighted linear interpolation, with weights based on each node’s neighbor count. This adaptive approach maintains detailed structure where necessary while reducing redundancy in open spaces. The final step selects the largest connected subgraph to eliminate unreachable locations from the representation.

3.3.2 Exploration

When exploring an unseen environment, a key challenge is deducing the topological structure based solely on local observations. Depth and LIDAR measurements are widely used for this purpose, as they intuitively reveal explorable regions by identifying frontiers—open spaces with a high computed information gain. While point clouds offer navigation-friendly data, many exploration methods still rely on additional, often resource-intensive environmental representations. This reliance arises from the need for frontier detection or information gain calculation, which frequently requires memory-heavy occupancy grid maps. Additionally, safe path generation typically requires a dedicated system component to identify secure routes for robot navigation.

To address these issues, we introduce **SkeletonExplorer**, a map-agnostic exploration algorithm for ground robots based on RayWalking. *SkeletonExplorer* can efficiently explore large-scale, unknown environments by progressively constructing a topological graph of the scene. The resulting graph can be processed by *SkeletonProcessor* to create a more structured representation, supporting downstream tasks. The algorithm operates with any map representation that can be built incrementally and supports collision checking. Starting from an initial position, a local map of the surrounding area is created. Obstacles (black vertices) and free-space regions (white vertices) are identified by raycasting within the map representation, similar to the *SkeletonFinder* algorithm (illustrated in Fig. 13). For ground robots, white vertices are further validated with a RayWalking step from the initial position to each white vertex location. If an obstacle is detected along this path, the vertex is marked black and placed on the ray where the RayWalking hit occurred.

Notably, these operations rely solely on local observations, without needing a global map. Next, frontiers are identified and stored in a FIFO queue, as in the *SkeletonFinder* algorithm. Exploration proceeds as the robot navigates to each frontier point retrieved from the FIFO queue, using A* search on the explored topology graph. Since the topology contains only paths validated as safe, each frontier is guaranteed to be reachable. These frontiers are locally mapped and validated by checking for safe paths in their normal direction. If a frontier is deemed valid, a node expansion point is found, and the robot navigates to it. This expansion process continues until the FIFO queue is empty.

3.4 Semantic Mapping

We propose a modified approach to semantic mapping that enhances the projection of semantic segmentation masks onto 3D reconstructions. Our method distinguishes between relevant and irrelevant semantic features, maintaining a dense geometric reconstruction of the environment while selectively projecting only user-specified objects of interest. Structural elements like walls, doors, and ceiling fixtures are treated as irrelevant, resulting in a more focused and practical semantic representation.

To enrich semantic understanding with a more descriptive spatial context, we integrate our representation with structural information from the *SkeletonProcessor* graph. The environment is segmented into meaningful regions using graph clustering, with each cluster assigned semantic labels and language descriptions.

3.4.1 Clustering

When clustering navigation graphs, a variety of algorithms can be employed to identify meaningful groupings or communities within the network. Each algorithm has a distinct approach, offering different perspectives on how to partition the graph:

1. **Edge Betweenness Partition** uses edge betweenness centrality to find clusters by measuring the number of shortest paths passing through each edge. Removing edges with the highest betweenness, directly partitions the graph into the specified number of clusters, efficiently identifying regions of high connectivity. A notable example of this approach is the Girvan-Newman [45] algorithm, which iteratively removes edges with the highest betweenness to reveal a hierarchical structure.
2. The **Louvain Method** [13] focuses on modularity optimization, merging smaller communities iteratively to maximize the modularity score. It is especially suitable for large navigation graphs, where it can quickly identify clusters that maximize the density of edges within communities compared to between them.
3. **Spectral Clustering** [109]: Based on the eigenvalues of the graph Laplacian, spectral clustering leverages linear algebra techniques to partition the graph. It finds clusters by embedding nodes in a lower-dimensional space, making it effective at capturing complex structural relationships within the network. This method is particularly advantageous when the underlying data exhibits intricate patterns that are not easily captured by traditional clustering algorithms.
4. **Infomap Algorithm** [105]: This flow-based approach uses random walks and information theory to detect communities by minimizing the description length of the flow within the network. Infomap is particularly effective at capturing dynamics and flow patterns in navigation graphs, making it well-suited for applications where agent movement or flow is significant in the network structure.

These algorithms provide a variety of techniques for clustering navigation graphs, each offering a distinct approach to identifying structural and functional regions within the network. We conduct experiments to evaluate their effectiveness in clustering the densely connected topological graphs generated by *SkeletonProcessor*.

3.4.2 Labeling

Objects are associated with clusters based on proximity to their nearest graph waypoint. For each cluster, we identify key nodes based on their betweenness centrality scores. To obtain visual context, we retrieve the original camera poses closest to these key nodes, capturing representative images of each spatial region.

These images, along with a global scene description, are processed using the **LLama 3-Vision** vision-language model (VLM). This enables detailed analysis of the scene’s visual and semantic elements. Subsequently, the **LLama 3 8b** language model is prompted to label and briefly describe each cluster by taking into account:

- The presence and labels of **objects** within each cluster
- **Visual descriptions** generated from the retrieved images
- **Predicted labels** derived from these descriptions
- The broader **environmental context**, providing situational awareness

The result is a semantically enriched sparse skeleton graph, **SatiNav-Sg** shown in Fig. 14, that enables both spatial-aware language queries and robot navigation tasks. This enriched graph also enhances robot navigation tasks by providing intuitive and descriptive spatial labels, making it adaptable for complex, goal-driven exploration and interaction. The prompts for image description and cluster labeling can be found in the Appendix.

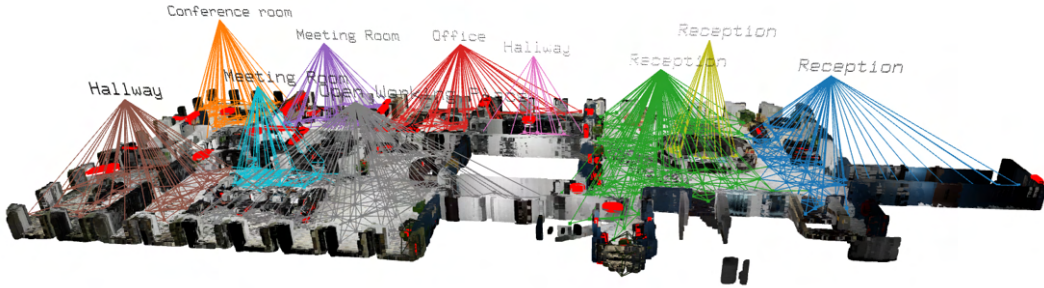


Figure 14: **SatiNav-SG** of Isaac Sim’s Office environment constructed by SatiNav (Semantics-Augmented Topological Inference for Navigation). The topological graph is generated from raw point cloud data using *SkeletonProcessor*, clustered through edge betweenness partitioning, and labeled by an LLM. Clusters are represented by colored pyramids grouping nodes, with detected objects shown in red.

3.5 Navigation

Navigation involves determining a series of actions the robot should take to reach its destination. Although a straight line provides the fastest route between any two points, avoiding static and dynamic obstacles is desirable behavior from an autonomous navigation system. However, the complexity and diversity of indoor scenes make this a

difficult challenge. The computational cost of classical path-finding algorithms can grow exponentially with the number of nodes, while reinforcement learning agents get stuck in local minima because of reward sparsity in large scenes. To remedy this, a partition of space is needed, which accurately models the topology of the traversable environment.

Our **Global Navigation** approach uses *SkeletonProcessor* to automatically identify waypoints within the environment for efficient pathfinding. The connection mechanism of *SkeletonProcessor* assures that for any given waypoint, all neighboring waypoints are reachable via a straight trajectory, disregarding dynamic obstacles. Given a starting and a target point, their corresponding closest waypoints are queried on the topological graph. We implement graph shortest path search using the A* [50] algorithm, as it is easy to implement and widely used in practice. A* finds the shortest path between the starting and target points. This path is further shortened by checking if any waypoints can be skipped with only safe straight edges. Path shortening removes all nodes from the path that can be bypassed through successful RayWalking to the next node. Subsequently, approach radii are computed for each node in the path. The approach radius represents the maximum radius around a node within which the next node remains reachable via RayWalking. For objects, the approach radius is determined by the size of the bounding box. The coordinates of the waypoints, along with their respective approach radii, are then provided to the local navigation agent.

The **Local Navigation** agent’s main working logic is based on a hierarchical structure, which excels in decomposing complexity for separating task difficulty into navigation and locomotion [68]. Lower-level control is simpler for robots with differential drive systems, as control signals can be directly calculated using the differential drive equations. However, the high degrees of freedom in humanoid and quadruped robots, such as the ANYmal-C, introduce significant complexity, necessitating a general control algorithm, for which Reinforcement Learning (RL) offers a promising solution. Our RL agents shown in Fig. 15 can be structured hierarchically, with a lower-level agent trained to follow velocity commands and a higher-level agent trained to provide those velocity commands. The higher-level agent guides the lower-level agent in avoiding obstacles, leveraging depth or LiDAR measurements to navigate toward destinations specified by relative position. The lower-level agent calculates joint movements to track the given velocity command. With no dynamic obstacles present in our scene, the higher-level component is replaced by a heading and velocity vector to the next topological waypoints for navigation.

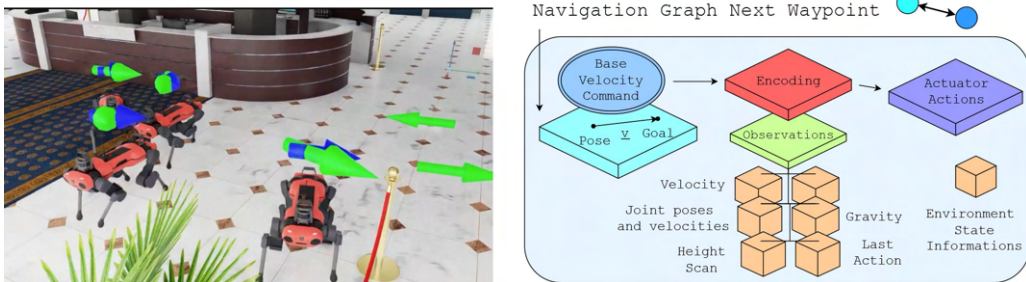


Figure 15: (a) High-level agent trained in parallel with multiple quadruped robots with the green arrows indicating velocity commands and target positions. (b) The architecture of the leveraged low-level agent. The lower-level agent [106] operates by learning a policy that leverages both proprioceptive measurements of the robot and environmental information around its base. [56].

Using a velocity command based on target direction alignment, the robot first rotates toward the direction of the next waypoint. Once aligned, it can move in a straight line following the vector from its current location to the target. We experimented with training the lower-level agent; however, it struggled in challenging scenarios, leading us to adopt a pre-trained policy for controlling joint movements. Fig. 15 illustrates the architecture of the low-level agent.

3.6 Language Instructions

Upon constructing the semantic scene graph with added capabilities for robot navigation, various practical applications become feasible, including robot navigation controlled by complex language instructions. Our implementation, **SatiNav-LI** Fig. 16 uses an LLM-based agent, which supports nuanced and indirect instructions.

We employ **Llama 3 8b** as the core LLM. While more advanced models with superior reasoning capabilities exist, these are unnecessary for our application. Llama is given a system prompt that defines its task, provides guidance for interpreting input queries, and includes instructions on formulating responses (see the Appendix for system prompt details). For scalability, the environment description is divided into chunks, each containing global context and specific details about a cluster, including its label and the labels of objects within it. The LLM interprets the query along with the scene description to identify relevant objects. The shortest paths are then calculated from the robot’s current position to the center of each selected object, with the object having the shortest path chosen. This path is subsequently relayed to the navigation agent.

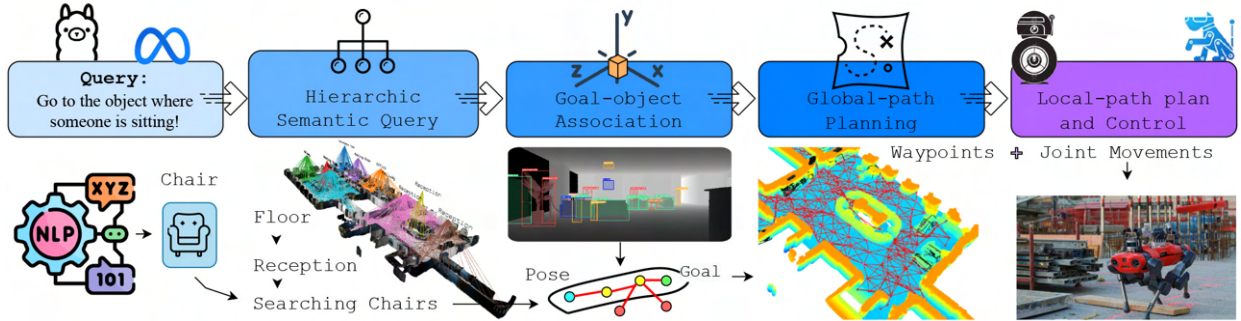


Figure 16: SatiNav-LI (Language Instructions): The figure illustrates the query pipeline, demonstrating a hierarchical search for chairs to identify objects and locate the nearest required object. In this example, the shortest path to the nearest chair is calculated, and local navigation directs the robot toward the target direction.

3.7 Implementation

Given the need for several distinct components to operate seamlessly, our framework is implemented within a Docker image, enabling installation-free, out-of-the-box experiments. Missions are structured using `py_trees`, and communication is facilitated through ROS. For efficiency, point cloud processing tasks are implemented in C++, while ROS nodes and other high-level components concerning semantics are implemented in Python. Code for the pipeline is available at https://github.com/lacykaltgr/semantic_navigation.

4 Results

In the previous sections, we outlined our complex framework for enhancing robot navigation. Below, we present the results of the experiments conducted to evaluate the efficacy of our proposed solution, **SatiNav**.

First, we demonstrate how **SkeletonProcessor** and **SkeletonExplorer** provide a more efficient and comprehensive approach to representing and navigating indoor environments. This experiment highlights the improvements in path-finding efficiency and structural coverage achieved through our framework.

Second, we explore the utility of **SatiNav** in identifying and labeling clusters of free space, offering a comparative analysis of different clustering algorithms. This step emphasizes how various methods contribute to meaningful segmentation and spatial understanding within an environment.

To comprehensively evaluate the SatiNav pipeline, we conducted two distinct case studies that assess both its simulated and real-world performance. The first study examines the framework’s capabilities in a controlled office environment simulation, focusing on semantic navigation accuracy, language model interpretation, and pathfinding robustness. This analysis reveals both the system’s strengths in handling contextual queries and its current limitations in path optimization and obstacle avoidance. Additionally, we conducted an investigation of scene graph construction in a real-world environment, where we encountered and analyzed various practical challenges including sensor noise, data drift, and object recognition complexity. This second study particularly emphasizes the framework’s performance under non-ideal conditions, providing crucial insights into the requirements for real-world deployment and highlighting specific areas where preprocessing and error-handling mechanisms require enhancement.

Together, these experiments illustrate how SatiNav advances the state of the art in semantic robot navigation, showcasing its potential for improved efficiency and intelligent spatial reasoning in real-world scenarios. The case studies provide a comprehensive assessment of the SatiNav framework’s current capabilities and limitations, establishing clear directions for future improvements in both controlled and real-world applications.

4.1 Navigation graph experiment

We compare our navigation graph construction algorithms **SkeletonProcessor** and **SkeletonExplorer** against existing baselines for both ground (2D graph) and aerial (3D graph) robots. Our experiments include three baseline implementations: **grid-based** graph, **probabilistic roadmap** graph, and **sparse SkeletonFinder** graph. The hyperparameters used are listed in the Appendix. Our results demonstrate that **SkeletonProcessor** and **SkeletonExplorer** represent the only viable options for comprehensive and efficient ground robot path planning, while also achieving competitive performance in aerial vehicle path planning.

4.1.1 Baseline methods

The **2D Grid** baseline navigation graph is constructed by uniformly sampling waypoints at a specified cell size on a grid at a set height parallel to the floor plane. Neighboring nodes are connected if the path between them is unobstructed, as determined by RayWalking. The largest connected subcomponent is then selected to ensure a cohesive

navigation graph. Note that this method does not support the automatic construction of multi-floor graphs. We excluded experiments with 3D Voxel Maps due to their infeasible time complexity in our setting.

Constructing a **Probabilistic Roadmap** navigation graph involves taking random samples on a plane or in space until a specified number of valid nodes are found. Connections are established by identifying neighboring nodes within a defined connection radius and creating edges where a clear RayWalking or raycasting path exists between the nodes. As these are statistical methods, for each measurement tests with 5 different seeds are averaged.

We also compare our method with a navigation graph constructed with the plain **SkeletonFinder** algorithm. Note that a graph of this kind does not support safe ground robot navigation as connections are based purely on raycasting, thus the results are only interpretable in the 3D case. However, the proposed **SkeletonExplorer** algorithm provides a good analogy for the 2D setting. To validate **SkeletonProcessor**, post-processing steps such as the close-node removal mechanism and RayWalking-based connections are evaluated through ablation studies.

4.1.2 Evaluation criteria

Detected object locations are used to measure the quality of a navigation graph. First, to achieve more meaningful results, objects are filtered, so that there are no two objects with a distance shorter than 1 m . Next, all possible filtered object pairs are taken and the ones with an object-to-object Euclidean distance of less than 10 m are removed. For both objects in each pair, the 10 nearest waypoints in the navigation graph are queried, with the closest waypoint used for pathfinding. If either object lacks reachable waypoints on the graph, the test for that pair is marked as a failure. For the remaining object pairs, pathfinding statistics are gathered, including pathfinding time, path length, path shortening time, and shortened path length. Path shortening is achieved by identifying waypoints along the path that can be skipped through successful RayWalking to the next node. As path lengths and path-finding times would give unbalanced results, they have to be normalized. We illustrate path length and path-finding time statistics, by normalizing the values with the Euclidean distance between the objects, resulting in relative path-lengths and search given in *millisecund/meter*. Pathfinding is deemed successful if the navigation graph provides a safe trajectory between the objects. Aggregated pathfinding statistics are then used to calculate overall pathfinding success rates. In addition to object pathfinding statistics, basic graph metrics such as the number of nodes, and the average number of neighbors per node are also compared.

4.1.3 Results

For 3D navigation of **aerial robots**, Table 1 and Fig. 17 compares the performance of 3D navigation graphs, specifically focusing on probabilistic roadmaps (PRM), SkeletonFinder (SkeletonFinder3D), and SkeletonProcessor (SkeletonProcessor3D).

Due to their unstructured nature, probabilistic roadmaps require a dense distribution of points to adequately cover the scene. While this dense distribution allows PRM to generate efficient trajectories, the high number of nodes and average neighbor count considerably increase search times.

	Nodes	Neighbors	Success %	Find path		Shorten path	
				Rel.Length	Time ($\frac{ms}{m}$)	Rel.Length	Time ($\frac{ms}{m}$)
PRM 3D (2k points)	1437	11	14.75%	1.62	3,	1.52	1.09
PRM 3D (4k points)	3990	23	90.25%	1.64	2.43	1.59	1.57
SkeletonFinder3D	1203	2	91.90%	1.89	0.29	1.74	1.60
SkeletonProcessor3D	830	12	90.87%	1.63	0.65	1.58	1.18

Table 1: **Comparison of graph construction algorithms in 3D.**

SkeletonFinder3D, in contrast, achieves similar coverage with nearly one-fourth of the nodes used in PRM, though this reduction leads to longer path lengths. SkeletonProcessor3D further optimizes node density by removing redundant, closely positioned nodes where possible, resulting in a more efficient graph structure. This optimization comes with a trade-off, however: the connection mechanism in SkeletonProcessor3D can increase the average neighbor count per node, which may raise path-finding times. Nevertheless, SkeletonProcessor3D’s edge structure provides a comprehensive representation of safe paths, reducing the need for path-shortening steps compared to SkeletonFinder3D.

SkeletonProcessor’s localized connection and node-removal approach inherently limit the neighbor count per node within a specific proximity, independent of the map’s overall size. This design enables SkeletonProcessor to retain scalability benefits as graph size increases. Additionally, being denser than SkeletonFinder, SkeletonProcessor forms a more structured graph, making it well-suited for critical applications and particularly effective for hierarchical search methods based on clustering.

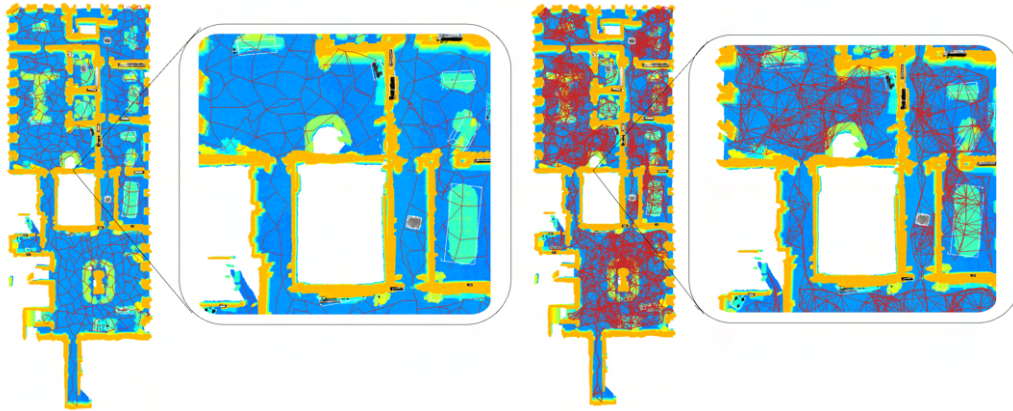


Figure 17: The 3D skeleton graphs generated SkeletonFinder (left), SkeletonProcessor (right). SkeletonFinder’s graphs are much more sparse in terms of edges, overlooking potentially important routes. There is also less redundancy present in terms of nodes in SkeletonProcessor.

For 2D navigation of **ground robots**, Table 2 and Fig. 18 compares the performance of various 2D navigation graph methods, including grid-based approaches (Grid 2D), probabilistic roadmaps (PRMs), SkeletonProcessor (SkeletonProcessor2D), and SkeletonExplorer (SkeletonExplorer2D).

In environments with narrow corridors, grid-based methods must employ a finely tuned leaf size to cover the scene effectively. While this achieves thorough coverage, it results in a large node count, which drives up path-finding times, although producing relatively efficient path lengths. PRMs, on the other hand, achieve adequate coverage with fewer points in 2D, though they still generate overly dense graphs that slow down search algorithms. SkeletonProcessor2D, similar to its 3D version, demonstrates an optimal balance between coverage and node count, achieving highly efficient paths with significantly reduced path-

	Nodes	Neighbors	Success %	Find path		Shorten path	
				Rel.Length	Time ($\frac{ms}{m}$)	Rel.Length	Time ($\frac{ms}{m}$)
Grid 2D (0.5m)	4293	6	85.46%	1.66	7.94	1.59	4.57
Grid 2D (0.75 m)	613	6	6.01%	1.25	0.23	1.17	1.95
PRM 2D (1k points)	584	10	61.4%	2.19	0.34	2.13	1.89
PRM 2D (2k points)	1496	20	85.51%	1.89	0.31	1.75	1.51
SkeletonProcessor2D	820	8	90.72%	1.72	0.14	1.69	2.15
SkeletonExplorer2D	968	2	92.17%	2.13	0.18	1.99	3.16

Table 2: **Comparison of graph construction algorithms in 2D.**

finding times. Although path-shortening steps in SkeletonProcessor2D can be computationally demanding, they offer only slight improvements in path efficiency, indicating that further path-shortening may be unnecessary in most applications. SkeletonExplorer2D, which progressively constructs the navigation graph, provides even greater coverage efficiency; however, it inherits certain challenges from SkeletonFinder3D, such as handling increased path complexity in narrow corridor navigation.

In our next phase, we plan to enhance SkeletonExplorer2D by incorporating the node and connection optimization strategies utilized in SkeletonProcessor, aiming to improve both efficiency and adaptability in constrained environments. This approach is expected to yield a more versatile solution for navigation graph construction in ground robots.

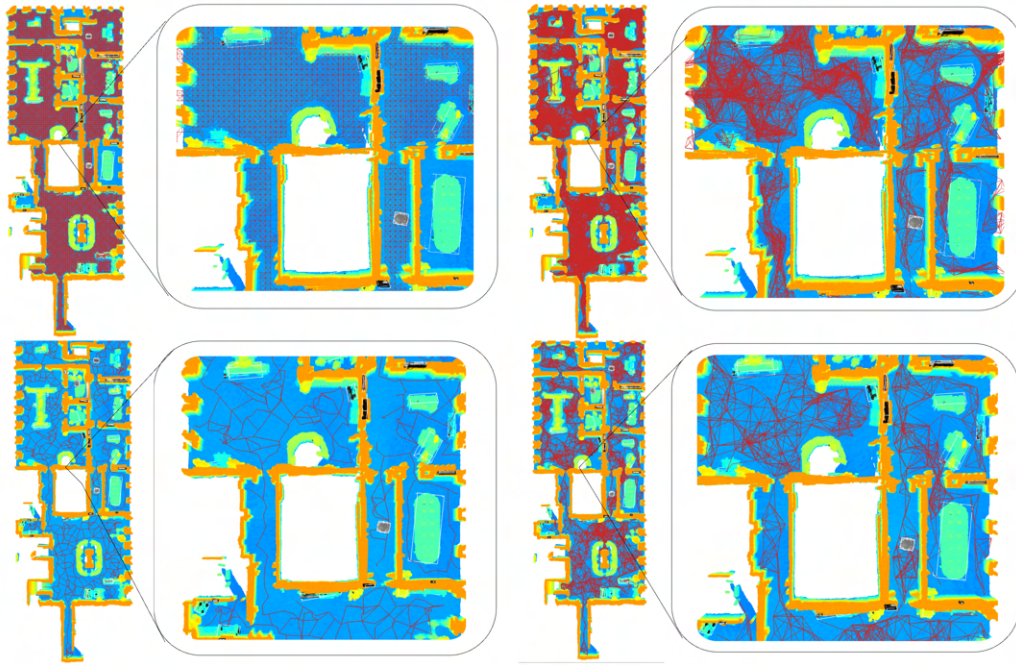


Figure 18: The 2D topological graphs generated by (a) grid-sampling, (b) probabilistic roadmap sampling, (c) SkeletonExplorer, (d) SkeletonProcessor.

The influence of hyperparameters, along with the combined performance of SkeletonExplorer and SkeletonProcessor is illustrated in Table 3 and Fig. 19. Using a connection threshold of $3.0m$ creates a much denser graph structure, bringing both advantages and disadvantages. This dense configuration allows for the discovery of shorter paths but incurs significantly higher time complexity in path-finding. Some of this increased complexity is mitigated during the path-shortening phase. By contrast, applying a smaller connection threshold results in a sparser graph with fewer nodes, yielding more feasible

Connection threshold	Too close threshold	Nodes	Neighbors	Success %	Find path		Shorten path	
					Rel.Length	Time ($\frac{ms}{m}$)	Rel. Length	Time ($\frac{ms}{m}$)
-	-	968	2	92.17%	2.13	0.18	1.99	3.16
3.0	-	968	14	92.17%	1.69	0.99	1.66	1.56
3.0	0.5	827	12	93.67%	1.70	0.55	1.67	1.60
3.0	1.0	453	6	93.67%	2.08	0.12	2.01	1.77
4.0	1.0	569	12	93.72	1.70	0.28	1.69	1.15

Table 3: Evaluating components of SkeletonProcessor with SkeletonExplorer2D as the baseline.

graph search times while maintaining similar path quality. Setting this threshold to 1 meter does not reduce scene coverage, though path lengths become somewhat longer. Raising the connection threshold to 4 meters addresses these issues, striking a balance between path length and search efficiency. This configuration provides an effective compromise, producing both manageable path-finding times and optimized paths. Through our tests, we found this threshold to perform well across various scenes.

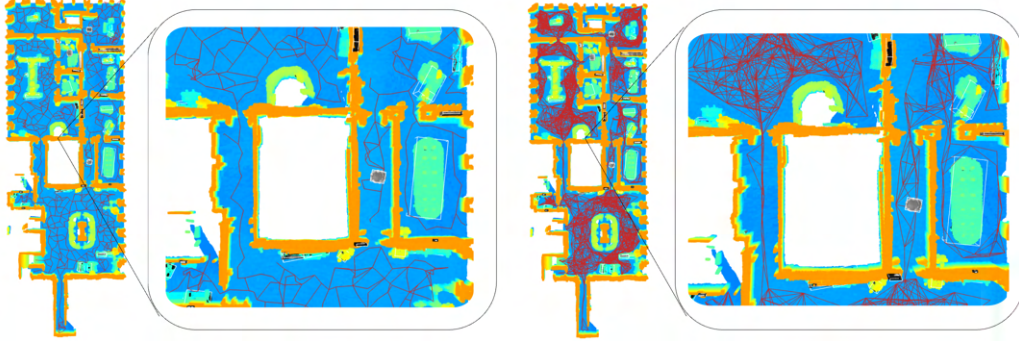


Figure 19: The 2D skeleton graphs generated by (a) SkeletonExplorer, (b) SkeletonExplorer along with post-processing with SkeletonProcessor.

4.2 Graph clustering experiment

Aside from path planning for robot navigation, the underlying structure of a topologic graph is useful for other downstream tasks too. There is a demand for navigation graphs that are able to robustly capture complex concepts like rooms, buildings, and hallways. Comprehensive topologic graphs allow us to extract such features using graph theory or other data mining techniques.

A straightforward benchmark for assessing structural robustness is clustering, where each waypoint in the navigation graph is grouped based on a measure of similarity. Notably, the topology of indoor environments naturally contains such structures, as rooms or buildings. In this section, we assess how effectively different clustering algorithms can identify these entities based solely on the graph structure, demonstrating how the topological graphs generated by our method, shown in Fig. 20 can be utilized for such tasks.

4.2.1 Evaluation

Edge betweenness clustering produced highly intuitive clusters and effectively recognized hallways. However, the need to specify the number of clusters can lead to suboptimal outcomes, such as merging distinct spaces like a meeting room with an adjacent area. Additionally, edge betweenness clustering becomes computationally inefficient for larger graphs due to its complexity.

The **Louvain method** efficiently detected densely connected subgraphs, making it well-suited for dense skeleton graphs produced by SkeletonProcessor. However, it sometimes missed smaller substructures, such as hallways, or introduced unnecessary splits due to its resolution limit. Also, its non-deterministic nature led to variable results, reducing robustness.

Spectral clustering created clean, intuitive clusters and uniquely identified the reception area as a single unit. Nevertheless, it was computationally intensive due to eigenvalue decomposition and highly sensitive to parameter choices, impacting consistency. While it shared hall-detection traits with edge betweenness, it is highly affected by data perturbations making it less robust.

Infomap produced a higher number of clusters compared to other methods and effectively captured small-node substructures, which was beneficial for identifying smaller sections or ones with a small number of nodes. However, the higher number of clusters, may not be suitable for all applications.

The choice of clustering algorithm depends on the context, as "intuitive clustering" can have different interpretations. For example, should a reception area be treated as a single cluster or divided based on visibility and structural boundaries? We found edge betweenness clustering to be the most suitable for our needs due to its intuitive results and alignment with pathfinding. However, its limitations prompt us to explore alternative solutions for larger graph structures. This study demonstrates that our graph representation effectively supports straightforward clustering based on structural properties.

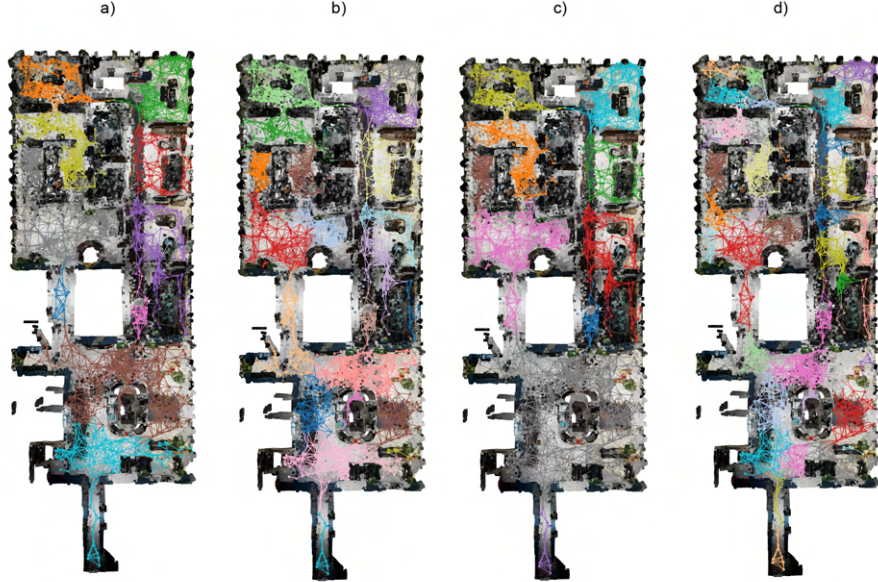


Figure 20: Topologic graphs produced by SkeletonProcessor clustered with different methods: (a) edge betweenness partitioning, (b) Louvain community clustering, (c) spectral clustering, (d) infomap clustering.

4.3 Case study: Semantic Navigation in Simulated Office Scene

The recordings taken in simulation were of high quality, yielding well-aligned results after processing. The scene featured relatively sparse object placement, with elements positioned precisely and detected easily by object detection algorithms. This simplicity made it an accessible task for vision-language models (VLMs) and large language models (LLMs), although many similar-looking areas introduced complexity to spatial reasoning. In terms of navigation, variability was noted in the responses generated by the LLM. While most queries were completed successfully, the LLM sometimes failed to deduce the purpose of a query, which led to incorrect conclusions. Nevertheless, it effectively dismissed irrelevant queries, such as "Where can I buy a pair of shoes?", highlighting its general capability despite the known limitations of smaller LLMs.

Pathfinding showed non-optimal routes, primarily due to the constraints of the simple nature of the path-shortening policy and the discretization of trajectories into straight-line segments. Additionally, paths often ran too close to obstacles due to the positioning of waypoints. Although this placement benefits topological representation, it should be penalized during route selection to avoid potential collisions. This sometimes resulted in unnatural rotations and simulation stalls, necessitating a restart. Implementing a more advanced local planner agent would resolve these issues and significantly improve reliability.

Despite these challenges, pathfinding queries remained stable across different starting positions, and the system effectively handled sequential queries illustrated in Fig. 21. While the current navigation component is not yet robust enough for real-world deployment, it provides a solid, modular base for further development and enhancement.

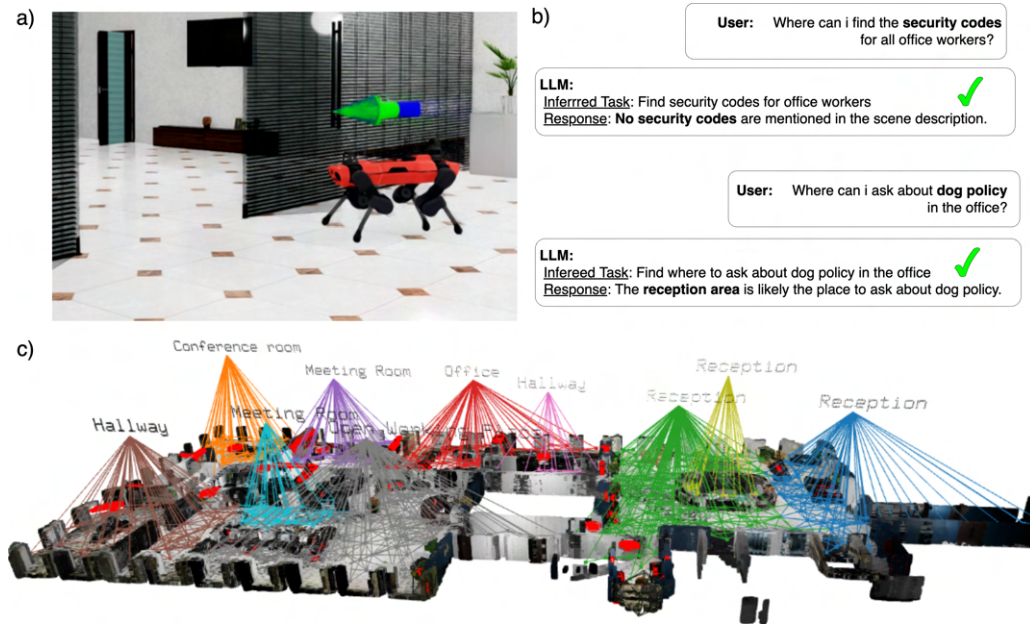


Figure 21: (a) ANYmal-C navigating in the Office environment, with a green arrow indicating its current heading direction (velocity command). (b) Example queries to the LLM: irrelevant queries are understood and denied, while achievable ones are accepted and executed. (c) SatiNav Scene Graph representation of the simulated Office environment.

4.4 Case study: Scene Graph Construction of Real World Scene

The recording of the building level at Nokia Skypark in Budapest, as shown Fig. 22 presented several challenges. It contained noise that persisted despite outlier removal efforts. Data drift was another significant issue, as the ground plane was not perfectly aligned with the coordinate axes. Nonetheless, the ray-walking method proved effective at managing these drifts. Inconsistencies in the scene, such as doors being open in some recordings and closed in others, added further complexity and required manual correction. This issue could be mitigated by integrating object detection to automatically flag and handle such inconsistencies, allowing ray paths to pass through bounding boxes (BBX) of doors as needed.

The scene also featured redundant objects, such as numerous identical chairs in a conference room, each stored as a separate object despite having the same bounding box and shape. This redundancy led to unnecessary memory load. Additionally, misdetections by both vision-language models (VLMs) and large language models (LLMs) contributed to the challenges, impacting the overall reliability of object recognition and interpretation. These data limitations underscore the need for improved preprocessing and handling mechanisms to enhance the robustness and efficiency of future analyses and navigation strategies.

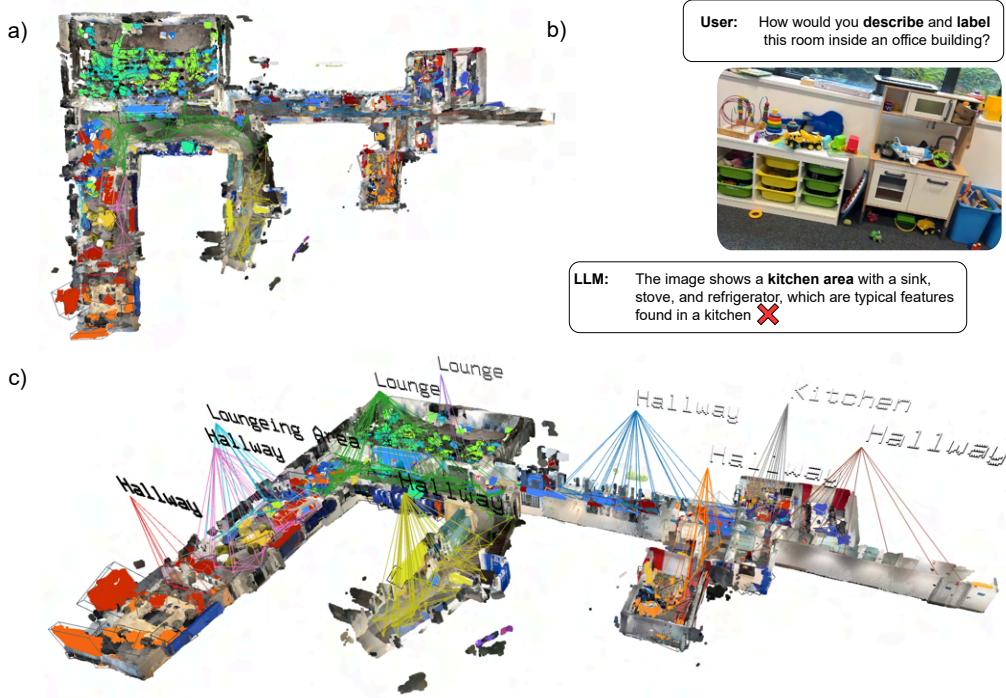


Figure 22: (a) Top-view of the scene graph generated by SatiNav, showing high object redundancy in the top left due to multiple identical green chairs. The topological graph successfully achieves full room coverage, even in real-world scenarios. (b) Image description LLM struggling to recognize a complex scene involving a toy kitchen set. (c) SatiNav Scene Graph representation of the Real World Office environment, visualized.

5 Discussion and Conclusion

5.1 Possibilities

Our proposed framework is capable of efficiently navigating complex environments using natural language commands, achieved through an understanding of semantic labels and topology graph-based route planning. The framework is adaptable to different robot configurations, providing a versatile foundation for robot navigation and supporting the testing and development of reinforcement learning (RL) agents across diverse task specifications, by defining safe waypoints. In future development, we aim to utilize Dreamer-V2 agents [46] for the navigation task organized into a hierarchical structure, potentially replacing pre-trained configurations. This involves integrating dynamic object detection (humans) into the higher-level component replacing the simple target direction alignment. Combined with exploration, the robot could build a map of the environment online to infer safe navigation paths by using front-end depth or a full 360° LiDAR scan to detect obstacles in its local vicinity.

Rich semantic segmentation enables not only navigation but also detailed task descriptions, such as retrieving objects from specified locations and delivering them to target destinations. The ANYmal-C, equipped with a robotic arm, can grasp objects, which could be trained by an RL agent to identify and segment specified items based on semantic information. Beyond navigation, additional components can be incorporated into the hierarchical structure to expand the system’s task capabilities.

Extensive real-world maps could support large-scale navigation, potentially covering entire academic buildings, such as the BME I or Q building. The map could be augmented with semantic labels organized into a hierarchical structure, enabling hierarchical queries for navigation in large buildings. Data storage could be optimized with a hierarchically balanced tree structure with spatial decomposition [85]. The 3D map visualization could also help students and visitors navigate the university building, help locate searched rooms or facilities. In the long term, this approach could be extended to enable an integrated representation of city areas, allowing multiple robots to assist humans with everyday tasks such as load-bearing support or handing over objects. Stonex X120GO SLAM Laser Scanner provided by TopoLynx Ltd can be used to synchronously obtain texture information and produce color point clouds and partial panoramic images [112].

Large real-world maps also open up opportunities for sim-to-real transfer and real-world testing of the proposed framework in concrete scenarios. The actuator network supports this transfer by learning the required control inputs for the robot joints, making the transition feasible in theory. Domain randomization further aids this process by introducing variability in the simulated environment, allowing the robot to generalize its behavior and adapt to diverse real-world conditions more effectively.

Real-world testing could further refine the topology graph by weighting edges based on human dynamics within the environment. In areas with frequent human traffic, the shortest path may become less efficient due to increased complexity in avoiding collisions. Relevant edges could be prioritized based on prior observations of indoor space usage patterns. This approach would enable the construction of a navigation system similar to Google Maps, from which various graph algorithms could be applied to optimize route selection and adapt to dynamic conditions.

5.2 Limitations

Our methods are shown to work very well in synthetic environments, however, we have not had the chance yet to test them with real robots in real environments. The gap between the simulated and real worlds degrades the performance of the policies of agents. Domain randomization, domain adaptation, imitation learning, meta-learning, and knowledge distillation offer promising solutions to bridge this gap [130]. Dynamic obstacles present another limitation of our approach; however, this could be addressed by implementing a more sophisticated local planner.

The quality of the map representation—particularly the clarity and minimal noise of the point cloud—is essential for algorithmic accuracy, as the algorithms depend on well-defined spatial data to operate effectively. High-quality sensor scans can ensure this clarity; however, any noise in the data may distort collision checking by inaccurately defining the boundaries of free space. In large-scale environments, memory limitations become a concern, as the entire map may need to be stored on a hard drive or in the cloud. To operate efficiently, only the relevant portion of the hierarchy would be loaded locally when needed.

Additionally, language models have inherent limitations and may not always respond accurately, as complex scene comprehension can be challenging even for advanced foundation models. Achieving precise object detection from 2D RGB-D images poses further difficulties, as these images often lack the necessary detail to identify complex shapes or distinguish overlapping objects.

5.3 Outlook

Despite the above limitations, our system comprehensively addresses the semantic robot navigation challenge through a highly modular pipeline with well-defined, flexible interfaces, allowing seamless integration of new algorithms as they emerge. The system enables map construction that is independent of specific representations for large, intricate scenes. These maps are created either through manual camera control or autonomous exploration by the robot, producing a robust environmental model that combines topological structure and semantic data to support natural language-based commands. The pipeline’s design facilitates effective clustering of semantic information, producing a hierarchical map representation that captures object relationships and spatial hierarchies. Through a novel topological graph approach, the framework efficiently finds the shortest paths across scalable and modular environments, enabling robust navigation in a wide range of scenarios.

This system is tested in a photorealistic, physics-based simulation, performing local control via reinforcement learning. By embedding a hierarchical navigation structure within reinforcement learning, our framework achieves robot configuration modularity that supports various applications in applied robotics. Furthermore, this architecture promotes a comprehensive understanding of large scenes, enabling robots to respond to different task descriptions and ensuring both human and machine interpretability. We outlined a series of future extensions and potential advancements, positioning our research to be a valuable contribution to the future of the Industrial Metaverse.

References

- [1] Automated aerial suspended cargo delivery through reinforcement learning. *Artificial Intelligence*, 247:381–398, 2017. Special Issue on AI and Robotics.
- [2] *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1*. Springer International Publishing, 2020.
- [3] Zeyad Abd Algfoor, Mohd Shahrizal Sunar, and Hoshang Kolivand. A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, 2015(1):736138, 2015.
- [4] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models, 2023.
- [5] Iñigo Alonso, Luis Riazuelo, Luis Montesano, and Ana C. Murillo. 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation, 2021.
- [6] ANYbotics. ANYmal C – The Next Step in Robotic Industrial Inspection, August 2019. Presented at the World Robot Conference, Beijing, August 20, 2019.
- [7] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera, 2019.
- [8] Somil Bansal, Roberto Calandra, Kurtland Chua, Sergey Levine, and Claire Tomlin. Mbmf: Model-based priors for model-free reinforcement learning, 2017.
- [9] Hriday Bavle, Jose Luis Sanchez-Lopez, Muhammad Shaheer, Javier Civera, and Holger Voos. Situational graphs for robot navigation in structured indoor environments, 2022.
- [10] Hriday Bavle, Jose Luis Sanchez-Lopez, Muhammad Shaheer, Javier Civera, and Holger Voos. S-graphs+: Real-time localization and mapping leveraging hierarchical representations, 2023.
- [11] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
- [12] Luke Shimanuki Beomjoon Kim. Learning value functions with relational state representations for guiding task-and-motion planning. *In Conference on Robot Learning*, page 955–968, 2020.
- [13] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.
- [14] Jean-Daniel Boissonnat, Frank Nielsen, and Richard Nock. Bregman voronoi diagrams. *Discrete and Computational Geometry*, 44(2):281–307, April 2010.
- [15] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- [16] Leonard Bruns, Jun Zhang, and Patric Jensfelt. Neural graph mapping for dense slam with efficient loop closure, 2024.
- [17] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2013.
- [18] Makram Chahine, Alex Quach, Alaa Maalouf, Tsun-Hsuan Wang, and Daniela Rus. Flex: End-to-end text-instructed visual navigation with foundation models, 2024.
- [19] Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. A comprehensive survey of scene graphs: Generation and application.

- IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26, January 2023.
- [20] Juntaing Chen, Guohao Li, Suryansh Kumar, Bernard Ghanem, and Fisher Yu. How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers, 2023.
 - [21] Xinyi Chen, Boyu Zhou, Jiarong Lin, Yichen Zhang, Fu Zhang, and Shaojie Shen. Fast 3d sparse topological skeleton graph generation for mobile robot global planning, 2022.
 - [22] Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas Roy, and Chuchu Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?, 2024.
 - [23] Yu Chen and Gim Hee Lee. Dogaussian: Distributed-oriented gaussian splatting for large-scale 3d reconstruction via gaussian consensus, 2024.
 - [24] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Seminal graphics: pioneering efforts that shaped the field*, 1976.
 - [25] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J. Davison. Deepfactos: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 5(2):721–728, April 2020.
 - [26] A. Dipanda and S. Woo. Towards a real-time 3d shape reconstruction using a structured light system. *Pattern Recognition*, 38(10):1632–1650, 2005.
 - [27] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry, 2016.
 - [28] Kenny Erleben and Henrik Dohlmann. Signed distance fields using single-pass gpu scan conversion of tetrahedra, 2007.
 - [29] Abhimanyu Dubey et al. The llama 3 herd of models, 2024.
 - [30] Coevoet et al. Software toolkit for modeling, simulation and control of soft robots. *Advanced Robotics*, 31:1208–1224, November 2017.
 - [31] Mittal et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.
 - [32] OpenAI et al. Gpt-4 technical report, 2024.
 - [33] Qiao Gu et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023.
 - [34] Rishi Bommasani et al. On the opportunities and risks of foundation models, 2022.
 - [35] Sun et al. Stereo matching using belief propagation. In *Computer Vision — ECCV 2002*, pages 510–524, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
 - [36] Xavier Puig et al. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.
 - [37] Aleksandra Faust, Oscar Ramirez, Marek Fiser, Kenneth Oslund, Anthony Francis, James Davidson, and Lydia Tapia. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning, 2018.
 - [38] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, Brian Ichter, Danny Driess, Jiajun Wu, Cewu Lu, and Mac Schwager. Foundation models in robotics: Applications, challenges, and the future, 2023.
 - [39] Scott Fredriksson, Akshit Saradagi, and George Nikolakopoulos. Grid-fast: A grid-based intersection detection for fast semantic topometric mapping, 2024.
 - [40] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review, 2023.
 - [41] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.
 - [42] M.L. Gavrilova. *Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008.
 - [43] Shuang Geng, Zelin Ning, Fu Zhang, and Boyu Zhou. Epic: A lightweight lidar-

- based uav exploration framework for large-scale scenarios, 2024.
- [44] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Deendra Singh Chaplot. Navigating to objects in the real world, 2022.
 - [45] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
 - [46] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models, 2022.
 - [47] Nathaniel Hamilton, Kyle Dunlap, and Kerianne L. Hobbs. Investigating the impact of choice on deep reinforcement learning for space controls, 2024.
 - [48] Nicklas Hansen, Jyothir S V au2, Vlad Sobal, Yann LeCun, Xiaolong Wang, and Hao Su. Hierarchical world models as visual whole-body humanoid controllers, 2024.
 - [49] Hardbake. Using nvblox with isaac sim, 2022.
 - [50] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
 - [51] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots, 2023.
 - [52] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation, 2023.
 - [53] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups, 2020.
 - [54] Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, and Konrad Schindler. Dynamic 3d scene analysis by point cloud accumulation, 2022.
 - [55] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization, 2022.
 - [56] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), January 2019.
 - [57] Brian Ichter, Edward Schmerling, Tsang-Wei Edward Lee, and Aleksandra Faust. Learned critical probabilistic roadmaps for robotic motion planning, 2019.
 - [58] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. Conceptfusion: Open-set multimodal 3d mapping, 2023.
 - [59] Peiwen Jiang, Chao-Kai Wen, Xinping Yi, Xiao Li, Shi Jin, and Jun Zhang. Semantic communications using foundation models: Design approaches and open issues, 2023.
 - [60] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields, 2023.
 - [61] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb: Performance comparison for distorted images, 2017.
 - [62] Lydia Kavraki, Petr Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12:566 – 580, 09 1996.
 - [63] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 1–8, 2013.
 - [64] Ue-Hwan Kim, Jin-Man Park, Taek-jin Song, and Jong-Hwan Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents.

- IEEE Transactions on Cybernetics*, 50(12):4921–4933, December 2020.
- [65] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.
 - [66] Hema Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.
 - [67] J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, 2000.
 - [68] Joonho Lee, Marko Bjelonic, Alexander Reske, Lorenz Wellhausen, Takahiro Miki, and Marco Hutter. Learning robust autonomous navigation and locomotion for wheeled-legged robots. *Science Robotics*, 9(89), April 2024.
 - [69] Juntong Lin, Xuyun Yang, Peiwei Zheng, and Hui Cheng. End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning, 2019.
 - [70] Feng Liu and Xiaoming Liu. Voxel-based 3d detection and reconstruction of multiple objects from a single image, 2021.
 - [71] Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics, 2024.
 - [72] Rongrong Liu, Florent Nageotte, Philippe Zanne, Michel de Mathelin, and Birgitta Dresp-Langley. Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review. *Robotics*, 10(1):22, January 2021.
 - [73] Carlos Linares López and Ian Herman. Evolving a* to efficiently solve the k shortest-path problem (extended version), 2024.
 - [74] Steve Macenski, Matthew Booker, and Joshua Wallace. Open-source, cost-aware kinematically feasible planning for mobile and surface robotics, 2024.
 - [75] Steve Macenski, Francisco Martin, Ruffin White, and Jonatan Gines Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2020.
 - [76] Steve Macenski, Tom Moore, David V. Lu, Alexey Merzlyakov, and Michael Ferguson. From the desks of ros maintainers: A survey of modern and capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems*, 168:104493, October 2023.
 - [77] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.
 - [78] E. Marchand and François Chaumette. Active vision for complete scene reconstruction and exploration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):65–72, 1999. ffinria00352545f.
 - [79] MathWorks. Matlab and simulink for robotics and autonomous systems. *Robotics and Autonomous Systems*, 2018.
 - [80] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian splatting slam, 2024.
 - [81] Donald Meagher. Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer, 10 1980.
 - [82] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for

- view synthesis, 2020.
- [83] Michal Minarik, Robert Penicka, Vojtech Vonasek, and Martin Saska. Model predictive path integral control for agile unmanned aerial vehicles, 2024.
 - [84] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey, 2022.
 - [85] Czimmer Márk. Development of a real time point cloud processing and self-driving software. 2023.
 - [86] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning?, 2019.
 - [87] Laksh Nanwani, Anmol Agarwal, Kanishk Jain, Raghav Prabhakar, Aaron Monis, Aditya Mathur, Krishna Murthy Jatavallabhula, A. H. Abdul Hafez, Vineet Gandhi, and K. Madhava Krishna. Instance-level semantic maps for vision language navigation. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, August 2023.
 - [88] Laksh Nanwani, Kumaraditya Gupta, Aditya Mathur, Swayam Agrawal, A. H. Abdul Hafez, and K. Madhava Krishna. Open-set 3d semantic instance maps for vision language navigation – o3d-sim. *Advanced Robotics*, page 1–14, August 2024.
 - [89] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking, 2011.
 - [90] Helen Oleynikova, Zachary Taylor, Roland Siegwart, and Juan Nieto. Sparse 3d topological graphs for micro-aerial vehicle planning, 2018.
 - [91] Sotiris Papatheodorou, Nils Funk, Dimos Tzoumanikas, Christopher Choi, Binbin Xu, and Stefan Leutenegger. Finding things in the unknown: Semantic object-centric exploration with an mav. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, page 3339–3345. IEEE, May 2023.
 - [92] Nico Pietroni, Davide Tonelli, Enrico Puppo, Maurizio Froli, Roberto Scopigno, and Paolo Cignoni. Voronoi grid-shell structures, 2014.
 - [93] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control, 2020.
 - [94] Victor Adrian Prisacariu, Olaf Kähler, Ming Ming Cheng, Carl Yuheng Ren, Julien Valentin, Philip H. S. Torr, Ian D. Reid, and David W. Murray. A framework for the volumetric integration of depth images, 2014.
 - [95] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.
 - [96] Ravi Ramamoorthi. Nerfs: The search for the best 3d representation, 2023.
 - [97] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning, 2023.
 - [98] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024.
 - [99] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.
 - [100] Yunfan Ren, Yixi Cai, Fangcheng Zhu, Siqi Liang, and Fu Zhang. Rog-map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-

- based motion planning, 2023.
- [101] Open Robotics. Gazebo and gazebo sim: A robotic simulator. *Robotics and Autonomous Systems*, 2022.
 - [102] Angel Romero, Yunlong Song, and Davide Scaramuzza. Actor-critic model predictive control, 2024.
 - [103] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans, 2020.
 - [104] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: from slam to spatial perception with 3d dynamic scene graphs, 2021.
 - [105] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Physical Review Letters*, 101(11):118701, 2008.
 - [106] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022.
 - [107] Lukas Schmid. *Robust Active Perception and Volumetric Mapping in Unknown Changing Environments*. Doctoral thesis, ETH Zurich, 2022.
 - [108] Simon Schreiberhuber. Depth sensing and mesh-based 3d reconstruction. Master’s thesis, TU Wien, Faculty of Electrical Engineering and Information Technology, Vienna, April 2023.
 - [109] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.
 - [110] Eric Sihite, Benjamin Mottis, Paul Ghanem, Alireza Ramezani, and Morteza Gharib. Efficient path planning and tracking for multi-modal legged-aerial locomotion using integrated probabilistic road maps (prm) and reference governors (rg), 2022.
 - [111] Alexander Sikorski and Martin Heida. Voronoi graph – improved raycasting and integration schemes for high dimensional voronoi diagrams, 2024.
 - [112] Stonex. X120go slam laser scanner: Accurate & versatile, 2023.
 - [113] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. imap: Implicit mapping and positioning in real-time, 2021.
 - [114] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, London, England, 2nd edition, 2018. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 2.0 Generic License.
 - [115] Phillip Swazinna, Steffen Udluft, Daniel Hein, and Thomas Runkler. Comparing model-free and model-based algorithms for offline reinforcement learning, 2022.
 - [116] Richard Szabo. Topological navigation of simulated robots using occupancy grid, 2004.
 - [117] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.
 - [118] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras, 2022.
 - [119] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
 - [120] By Amulya Vishwanath and Khannah Shaltiel. Fast-track robot learning in simulation using nvidia isaac lab, 2024.
 - [121] Johanna Wald, Helisa Dharmo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
 - [122] Zhengren Wang. 3d representation methods: A survey, 2024.

- [123] Jian Wen, Xuebo Zhang, Qingchen Bi, Hui Liu, Jing Yuan, and Yongchun Fang. G²vd planner: Efficient motion planning with grid-based generalized voronoi diagrams, 2024.
- [124] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.
- [125] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015.
- [126] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences, 2021.
- [127] Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Liang He, and Yuan Xie. Recent advances of foundation language models-based continual learning: A survey, 2024.
- [128] Christopher Zach. *Dense Reconstruction*, pages 179–181. Springer US, Boston, MA, 2014.
- [129] Jing Zeng, Yanxu Li, Jiahao Sun, Qi Ye, Yunlong Ran, and Jiming Chen. Autonomous implicit indoor scene reconstruction with frontier exploration, 2024.
- [130] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *CoRR*, abs/2009.13303, 2020.
- [131] Tianfei Zhou, Fei Zhang, Boyu Chang, Wenguan Wang, Ye Yuan, Ender Konukoglu, and Daniel Cremers. Image segmentation in foundation model era: A survey, 2024.
- [132] Liyuan Zhu, Yue Li, Erik Sandström, Shengyu Huang, Konrad Schindler, and Iro Armeni. Loopsplat: Loop closure by registering 3d gaussian splats, 2024.
- [133] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam, 2022.