Toward Interactive Navigation in Unknown Dynamic Environments

Guofei Chen

CMU-RI-TR-25-50

June 6, 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Ji Zhang, *co-chair*
Wenshan Wang, *co-chair*
Guanya Shi
Chao Cao

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my family.

Abstract

There is a growing demand for mobile robots to act not only as passive observers but also to actively interact with their environment, especially in cluttered and social settings. Meeting this demand requires navigation systems that can both understand interaction-relevant properties of the environment and make plans accordingly. This thesis presents a modular approach to building such an interactive navigation system by addressing the challenges in the perception and planning components.

For perception, we propose a spatio-temporal SLAM module that constructs and maintains a semantic, instance-level map in real-time. By projecting 2D semantic features into 3D space and performing joint tracking and validation using both geometric and semantic cues from sensor data, the system produces a detailed 3D semantic map. This map includes geometric and semantic object features and continuously updates to reflect dynamic changes in the environment. Compared with baseline methods, the instance-level segmentation average precision in the static scene has improved by around 3x, and has the ability to model dynamic objects. The resulting 4D (3D + time) semantic representation enables the identification of candidate objects for interaction, along with their relevant properties.

For planning, we introduce a real-time interactive planning module based on an enhanced Directed Visibility Graph (DV-graph). This graph enables fast global path planning and supports real-time interaction planning that adapts to new sensory inputs. The planning framework is designed to handle complex, unknown, or partially known environments, allowing the robot to reason about and exploit the dynamics of objects to achieve navigation objectives. The average search time was improved by more than 100 times in large-scale environments, while maintaining the highest success rate and path quality.

The complete interactive navigation system, including both perception and planning modules, has been thoroughly validated through extensive experiments in both simulation and real-world scenarios. All experiments were conducted using onboard computation and in real time. Results demonstrate that the proposed system outperforms previous approaches that rely on offline processing and higher computational budgets. Fur-

thermore, the system's modular design makes it suitable for extension to broader applications beyond interactive navigation, such as supporting general-purpose interactions and mobile manipulation tasks.

Acknowledgments

I would like to express my deepest gratitude to my committee co-chairs, Dr. Ji Zhang and Dr. Wenshan Wang, whose unwavering support and insightful feedback have been invaluable throughout the preparation of this thesis and the research behind it. More than advisors, they have profoundly shaped my understanding of what it means to build robust and impactful robotic systems. Their mentorship has taken me on an intellectual journey far beyond what I ever imagined at the start of my Master's.

I am also sincerely thankful to my committee members, Prof. Guanya Shi and Chao Cao. Our discussions were always stimulating and inspiring, often extending beyond the boundaries of robotics and into broader, thought-provoking territory.

My heartfelt thanks go to my parents and Hanxi, whose love, encouragement, and constant presence have sustained me through every step of this pursuit. Their warmth and belief in me have been my greatest source of strength.

I am deeply grateful to my colleagues and collaborators — Botao He, Shibo Zhao, Honghao Zhu, Nader Zantout, Dr. Zhongqiang Ren, and Yu Chen. Working with you has been a true joy. Many of the best ideas in this work were sparked and realized through your enthusiasm, creativity, and dedication.

Lastly, I extend my sincere thanks to all who supported me throughout this journey. Your encouragement, insights, and generosity have had a lasting impact on this work and me!

Funding

This work was partially supported by Chiang Chen Overseas Fellowship.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In many real-world applications, mobile robots are expected not only to perceive their environment but also to actively interact with it, particularly when operating in cluttered or social settings. The ability to interact with the environment becomes especially useful when navigation goals are not directly reachable or when attempting interactions may significantly improve path efficiency. For example, in a home environment cluttered with furniture, finding a collision-free path may be infeasible without physically moving obstacles. In such cases, the robot must engage with its surroundings, such as pushing objects, to create a navigable path. This class of tasks, where purposeful physical interaction is integrated into navigation, is termed interactive navigation.

Designing a robust interactive navigation system that can both track changes in the environment and plan interactions in real time is a nontrivial task. Some existing works attempt to solve the problem using end-to-end neural networks that map observations directly to actions. However, these data-driven methods often fall short in moderate or larger environments and struggle to generalize well to unseen environments. In contrast, a modular design following the "sense-plan-act" paradigm provides several advantages: it decomposes the complex problem into manageable subtasks and benefits from strong inductive biases, thereby reducing the need for extensive data to reach satisfactory performance.

This thesis focuses on designing and improving individual modules in a modularized interactive navigation pipeline. Two primary challenges are addressed:

- Semantic Mapping: Building a dynamic, instance-level semantic representation of the environment. This representation enables the robot to identify candidate objects for interaction, track, and update object states based on motion or new observations. Existing semantic mapping methods often fail to support instance-level reasoning in dynamic environments or track evolving object states effectively.
- Interaction Planning: Developing efficient planning strategies in unknown environments with many potential interaction targets. Traditional planners that rely on dense environment representations are computationally expensive, with complexity increasing exponentially as the number of objects grows. A scalable, real-time planner that adapts to object properties and handles hundreds of dynamic elements is essential for practical deployment.

These modules are designed with the goal of maximizing real system deployment. They can run in real-time with onboard computing resources, without relying on unrealistic information such as segmentation ground truth or terrain information. In the meantime, each module is extensible to broader applications, such as mobile manipulation and language-conditioned navigation.

1.2 Contributions and Document Outline

The primary contributions of the thesis are:

- A real-time, open-vocabulary, spatio-temporal semantic SLAM module that maintains an instance-level semantic map. The system tracks object identities over time and incorporates new observations to reflect dynamic changes in the environment.
- An efficient real-time interaction planning module in large-scale unknown environments that adapts online to object-specific physical properties. The search time was improved by more than 100 times while maintaining the best quality of the path.

- A comprehensive framework for interactive navigation in complex, unknown, or partially known environments.

The structure of the thesis is as follows. In Chapter 2, we address the challenges in instance-level semantic mapping for dynamic environments. In Chapter 3, we address the challenges in efficient topological planning for interactive navigation and present interactive navigation system evaluations. In Chapter 4, we summarize the completed work and discuss future directions

Chapter 2

Semantic Mapping

2.1 Introduction

As robots navigate dynamic, real-world social environments, they should not only recognize the objects around them but also track where these objects were in the past. Crucially, this process should operate in real time with efficient memory use, as most robots are equipped with limited computational resources and have to process sensor information on the fly.

Existing semantic SLAM methods enable robots to build semantically annotated geometric maps in real time, but they have several limitations. Most approaches [19, 37] build closed-vocabulary, class-level mapping instead of open-vocabulary, instance-level mapping, limiting their ability to identify and track individual objects over time. Also, challenges such as occlusions, lighting changes, and missed detections further hinder robust instance-level semantic mapping in real-world scenarios. Additionally, many systems [6, 30, 40, 50] focus on short-term dynamics (e.g., moving people) while neglecting long-term changes such as furniture relocation that happens out of the robot's view. Even advanced methods like Khronos [38] lack open-vocabulary and instance-level capabilities.

Recent scene graph approaches [13, 24, 32, 46] incorporate open-vocabulary detectors [35] to support natural language navigation, but they are time-consuming (minutes to hours) and not suitable for real-time mapping. Moreover, they struggle to capture dynamic relationships between objects, limiting their ability to model scene

evolution.

In summary, existing approaches predominantly address either dynamic change detection (e.g., identifying moving humans or vehicles) or static object recognition (e.g., identifying objects and their relationships). However, few methods tackle both challenges simultaneously—understanding which objects change and how they evolve spatially and temporally. Addressing this dual challenge is crucial for robots to effectively perceive and adapt to dynamic real-world environments.

To meet this challenge, we introduce SuperMap: an open-source scene representation framework that constructs open-vocabulary 4D scene graphs. SuperMap not only captures the spatial layout of the environment but also encodes semantic understanding while tracking temporal changes, enabling real-time interaction and reasoning. Our contributions are as follows:

- **Real-Time, Open-Vocabulary, Spatio-Temporal Semantic SLAM:** We introduce the first real-time, open-vocabulary, 4D instance-level semantic SLAM system. SuperMap enables robots to perform ego-motion estimation while simultaneously tracking the evolution of surrounding objects, significantly enhancing their ability to understand dynamic environments.
- **Spatio-Temporal Object Updates:** We propose a unified framework that integrates real-time 2D-3D object tracking, validation, and change detection. Our approach effectively handles occlusions, partial views, open-vocabulary shifts, and dynamic changes, ensuring the 3D scene graph remains accurate and up-to-date with current instances and spatial relationships.
- **Spatio-Temporal Reasoning:** Our 4D scene graph seamlessly incorporates spatial and temporal information for each object, equipping robots with instance-level spatio-temporal reasoning capabilities (e.g., locating moved objects, recalling past scenes).
- **Open-Source Framework:** To foster advancements in robotic perception, we release the complete codebase, datasets, and benchmarks.

Table 2.1: Capability Checklist for Semantic Mapping and Semantic SLAM Methods. ✓indicates support, ✗indicates limited or no support

Capability	Semantic Mapping						Semantic SLAM			
	HOV-SG[46]	ConceptGraphs[13]	RayFronts[1]	OpenScene[28]	OpenMask3D[44]	SeeGround[21]	Kimera[37]	OVO-SLAM[25]	Khronos[38]	SuperMap (Ours)
Real-Time	✗	✗	✓	✗	✗	✗	✓	✓	✗	✓
Open-Vocabulary	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓
Instance-Level	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
Short-Term	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Long-Term	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Scene Graph	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓

2.2 Related Works

Semantic Mapping Equipping mobile robots with the ability to perceive and understand their surroundings is crucial for real-world applications. To this end, researchers have developed semantic maps to capture environmental meaning and context [34]. However, many recent methods [13, 28] rely on offline processing, requiring pre-collected environmental data (images and depth) to build semantically labeled maps shown in 2.1.

For example, OpenScene [28] uses OpenSeg [51] to extract CLIP embeddings [31] from images and then trains a scene-specific 3D model, but it operates offline, requiring a full scene scan. OpenMask3D [44] back-projects 2D object masks from multiple views to form 3D segments, while SeeGround [21] combines 2D-VLM with 3D geometric cues for instance segmentation. Despite improving 3D accuracy, these methods assume complete scene data, limiting online usability.

LERF [18] and LangSplat [29] employ radiance fields to build 3D maps with CLIP-based features, offering flexible semantic queries but lacking real-time capability due to computational demands. RayFronts [1] enables online semantic mapping for outdoor environments but struggles with change detection and fails to achieve instance-level 3D semantics.

Semantic SLAM To jointly estimate poses within the pipeline, we transform the mapping pipeline into a SLAM pipeline. In recent years, numerous SLAM systems have integrated semantic information during the mapping process. One notable example is Kimera [37], which offers a real-time semantic mapping solution by building a 3D scene graph. However, it relies on closed-set CNN segmentation, which limits its ability to generalize to novel classes. LiDAR-based SLAM approaches, such as SuMa++ [5] and LIOM [55], incorporate CNN-based per-point labels into LiDAR

maps. Despite their effectiveness, these methods do not support open-vocabulary segmentation, restricting their adaptability to diverse environments. SlideSLAM [22] and OVO-SLAM [25] employ open-vocabulary detectors for instance segmentation, allowing for more flexible scene understanding. However, these methods face challenges when dealing with **long-term and short-term dynamic objects**.

Open-Vocabulary Scene Graph HOV-SG [46], ConceptGraphs [13] and CLIO [24] segment and label objects from a full point cloud using SAM and CLIP, organizing them into a hierarchical graph with open-ended labels from CLIP. However, HOV-SG and ConceptGraphs require offline 3D reconstruction before applying SAM and CLIP, making it unsuitable for online SLAM or real-time dynamic tasks. CLIO implements real-time open vocabulary scene graph generation but lacks spatio-temporal capabilities.

Spatio-Temporal Semantic SLAM Recently, Khronos [38] addresses both short-term and long-term dynamics in map representation. However, this method is limited to closed-set objects and lacks the capability for instance-level tracking. Additionally, Khronos struggles to operate in real time, making it difficult for robotics operations. In contrast, to the best of our knowledge, our proposed method is the first real-time, spatio-temporal, open-vocabulary, and instance-level object mapping framework that effectively handles both short-term and long-term dynamic objects.

2.3 Spatio-Temporal SLAM Problem

The spatio-temporal SLAM problem aims to provide an explicit scene representation that captures how object-level changes evolve across space and time through incremental data acquisition. We model the scene as a composition of objects, represented as a scene graph where nodes correspond to objects and edges denote spatio-temporal relationships. In our setting, we process either RGB-D video sequence or a point cloud video sequence including RGB image frames $C = \{C_1, C_2, ..., C_t\}$, depth image frames $D = \{D_1, D_2, ..., D_t\}$, and camera poses $P = \{P_1, P_2, ..., P_t\}$ (where t is the timestamp index). It generates a consistent global map M_t consisting of semantic instance-level objects $O_t = \{O_t^j\}$ (where j is the object index).

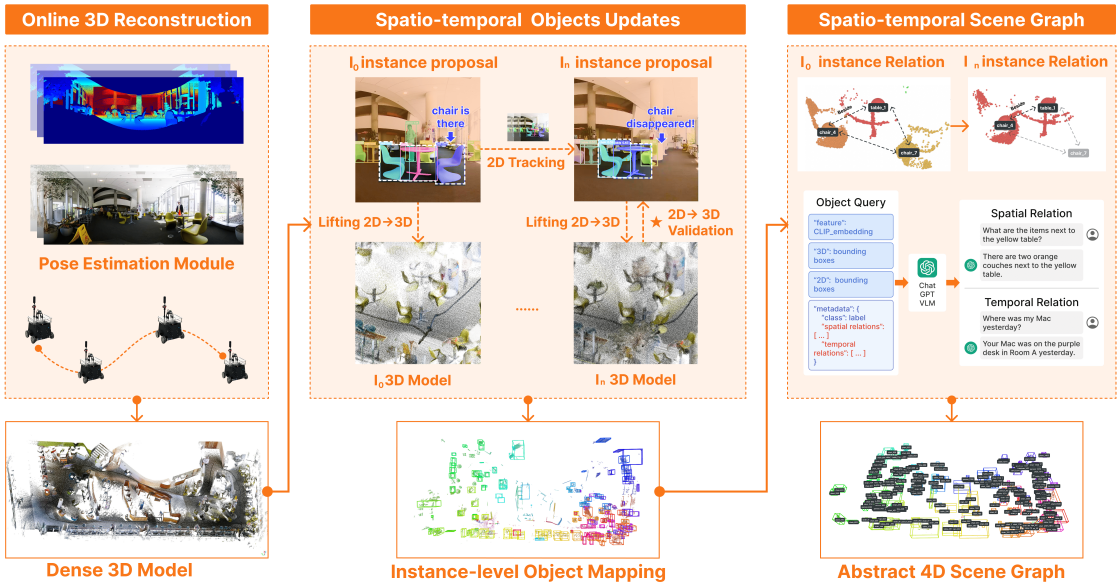

Figure 2.1: **Overview**. The system provides accurate state estimation and rich map representation, ranging from concrete to abstract. It has three modules: an online 3D reconstruction module is based on a LiDAR SLAM system to achieve state estimation and a dense 3D model. Then, the state and 3D model are utilized to detect the short-term and long-term changes in object to achieve instance-level object mapping. Then, the object-level mapping is represented as an abstract 4D scene graph to capture both spatial and temporal relationships between objects.

The spatio-temporal mapping problem is defined as follows: given the current frame observation $Q_t = \{m_t^i, D_t, P_t\}$ and the existing map $M_{t-1} = \{O_{t-1}\}$, determine instance ID I_t and the updated 3D map M_t : $P(I_t, M_t \mid M_{t-1}, Q_t)$. Applying the chain rule, we derive the problem as:

$$P(I_t, M_t \mid M_{t-1}, Q_t) = \underbrace{P(I_t \mid M_{t-1}, Q_t)}_{\text{spatio-temporal object instance association}} \cdot \underbrace{P(M_t \mid M_{t-1}, Q_t, I_t)}_{\text{online map updates}} \quad (2.1)$$

This involves two tasks: spatio-temporal object instance association and the online map update [7].

2.4 Method

The architecture of our spatio-temporal SLAM system, SuperMap, is depicted in Fig. 2.1. Given a sequence of depth measurements, IMU data, and RGB images, SuperMap performs instance-level object mapping while evolving its spatial representation using onboard computing. We provide a detailed explanation of the following components: (1) *Online 3D Reconstruction* for generating dense spatial representations, (2) *Spatio-Temporal Object Updates* to adapt to changing environments, and (3) *Spatio-Temporal Scene Graph construction* for reasoning and decision making.

2.4.1 Online 3D Reconstruction

To reconstruct the object-level map, we utilize SuperOdometry [56] to obtain a colorized dense 3D model and the current robot state estimation using image and LiDAR sensors. These estimated states and 3D models serve as crucial priors for semantic instance tracking, association, and change detection. Due to the limits of space, the details can be found in the supplementary.

2.4.2 Spatio-Temporal Objects Updates

The primary objective of this work is to detect and track how instance-level objects change across space and time. In this section, we address two key questions introduced in Section 2.3:

- How to achieve spatio-temporal object *instance association* while handling occlusions, partial observations, and semantic label shift?
- How to *update the map* to accommodate both short-term and long-term dynamic changes?

Spatio-Temporal Instance Association

Object Tracking in Image Space: Let $\mathbf{B}(t)$ be the set of bounding boxes at time t, where each bounding box $b_i(t)$ corresponds to a detected object. Tracking is performed using ByteTrack [54], which maintains a 2D tracklet state $\mathbf{T}_i(t)$ for each object i. The state are defined as:

$$\mathbf{T}_i(t) = [x_i(t), y_i(t), w_i(t), h_i(t), v_x(t), v_y(t)] \tag{2.2}$$

Here, $[x_i, y_i]$ represents the position, $[w_i, h_i]$ represents the bounding box dimensions, and $[v_x, v_y]$ represents the velocity components.

Motion Prediction with Kalman Filter: The motion prediction model for each tracked object i is based on a Kalman filter:

$$\hat{\mathbf{T}}_i(t+1) = \mathbf{F}\mathbf{T}_i(t) + \mathbf{Q} \tag{2.3}$$

where $\mathbf{F}$ is the state transition matrix, and $\mathbf{Q}$ is the process noise.

3D-aware Instance Tracking: To enhance the tracking robustness, we integrate the 3D map information. Let $\mathbb{M}(t)$ be the 3D map representation. We check the **observability** of each tracker by projecting the 3D object position onto the image plane: $[x_i', y_i'] = \mathbf{P} \cdot \mathbf{M}_i$. Here, $\mathbf{P}$ is the camera projection matrix, and $\mathbf{M}_i$ represents the 3D coordinates of the object.

We then perform a hungarian matching between all the compensated tracklet bounding boxes and the current bounding box inferences. The tracked results are merged into map according to its tracking instance id. Otherwise, the motion compensation is calculated as:

$$\Delta\mathbf{T}_i = \mathbf{R}(t)\mathbf{T}_i(t) + \mathbf{t}(t) \tag{2.4}$$

Where $\mathbf{R}(t)$ and $\mathbf{t}(t)$ are the rotation and translation matrices derived from the camera motion. The motion of the compensated location is then predicted one step using the formula in 2.3.

Online Map Updates

Given a spatial map M_t at time t, we aim to update this map to accommodate both short-term (dynamic) and long-term changes observed through new sensor data. The map at time t can be represented as a set of spatial states:

$$M_t = \{(x_i, s_i) \mid x_i \in \mathbb{R}^3, s_i \in S\} \tag{2.5}$$

Where x_i: Position of the i-th point in the map. s_i: State of the point, including semantic label and occupancy probability. S: State space (e.g., occupancy, semantic label, dynamic/static state).

Geometric Consistency Check: Let the observed depth be a random variable $D(x_i)$ with an associated probability distribution $P(D(x_i))$. The object depth is modeled as a random variable $D_{object}(x_i)$. To determine geometric consistency, we define the difference:

$$\Delta d = E(D_{object}(x_i) - D(x_i)) \tag{2.6}$$

Define $\epsilon > 0$ as the uncertainty of depth measurement. The updated state decision is given by:

$$s_i' = \begin{cases} \text{observable}, & |\Delta d| \leq \epsilon \\ \text{occluded (reduced occupancy)}, & \Delta d < -\epsilon \\ \text{unobservable (ignore)}, & \Delta d > \epsilon \end{cases} \tag{2.7}$$

For observable objects, their observable points are projected onto the image plane: $[x_i', y_i'] = P \cdot M_{obs}$ and used to activate tracklets that are lost. For points that are observable but not detected anymore, the probability of their belonging to the object is reduced.

In summary, the geometric consistency check effectively removes historical points associated with long-term object changes. Short-term object changes are managed through 3D-aware instance tracking, while projection errors arising from modeling short-term dynamic object changes are addressed by reprojection.

2.4.3 Spatio-Temporal Scene Graph

Scene Graph Generation Our instance-level mapping produces a series of time-indexed object-level maps $\{M_1, M_2, \dots\}$. To facilitate downstream reasoning, we transform the time-indexed object-level maps into a 4D spatio-temporal scene graph.

Scene graph G captures object states and relations:

$$G = (V, E_S, E_T) \tag{2.8}$$

Where:

- V : Nodes representing object instances at specific timestamps.
- E_S : Spatial edges between objects in the same frame.
- E_T : Temporal edges linking the same object across frames.

For real-time operation, we first cluster objects based on their centroid distances and then add a spatial edge when an object pair satisfies class-dependent geometric predicates (e.g., on, beside, under). The temporal edges trace an object's trajectory from our object association result, preserving its identity throughout the sequence. The resulting graph compactly captures both instantaneous layout and long-term dynamics, providing an abstract representation of the scene that is easy to reason on. Full construction details and the complete list of spatial predicates appear in the supplementary material.

Spatio-Temporal Reasoning: Our 4D scene graph enhances the ability of Vision-Language Models (VLMs) to perform robust spatio-temporal reasoning over scenes. To achieve this, we optimize VLM interfacing through prompt engineering with two primary objectives: (1) to maximize the use of spatial and temporal information from the scene graph, and (2) to maintain the usability of VLM responses while preserving the model's reasoning capabilities.

To facilitate effective scene graph utilization, we prepend each task question with a brief description of the scene graph structure. Additionally, we embed spatial and/or temporal cues within the question itself to encourage VLMs to reference spatial relationships and temporal states during reasoning.

To balance flexibility in reasoning with practical output, we append each query with clear instructions: while VLMs may include their thought process in the response, they must provide the final answer as one or more object IDs enclosed within ¡answer¿¡/answer¿ tags. A lightweight parser then extracts these IDs from the response, ensuring that the output is both grounded in spatio-temporal context and readily usable for downstream tasks.

2.5 Experiments

2.5.1 Experiment Settings

We evaluate SuperMap on a mecanum wheel robot equipped with a Livox Mid-360 LiDAR and a panoramic camera without hardware time syncronization. All the experiments on the robot run onboard in real time on an Intel i9-14900H CPU and an RTX 4090 laptop GPU.

We aim to address the following research questions through our experiments:

- **Quantitative Results for Semantic and Instance Segmentation:** How does the semantic mapping class-level and instance-level segmentation performance compare to state-of-the-art incremental semantic mapping approaches on the ScanNet benchmark (Sec. 2.5.2)?
- **Spatio-Temporal Object Update:** Over long horizon, real-world scenarios, how well does the 4D instance-level mapping updates dynamically moved objects and preserves the static object identities and spatial relations across time (Sec. 2.5.3)?
- **Spatio-Temporal Reasoning with the 4D Scene Graph:** To what extent does the 4D scene graph improve downstream reasoning about dynamic environments (Sect. 2.5.4)?
- **Real-time Visual-language Navigation:** Can the full pipeline support zero-shot, language-guided navigation with **only onboard compute** for mapping(Sec. 2.5.5)?

These questions guide our evaluation, demonstrating the practical effectiveness and robustness of our proposed method in diverse scenarios.

2.5.2 Quantitative Results for Semantic and Instance Segmentation on Static Scene

Class-level Segmentation: Tab. 2.2 presents a comparison of class-level segmentation precision between our method and other open-vocabulary approaches. Our method achieves an accuracy of 55.48%, only 1.32% lower than the state-of-the-art

Table 2.2: **Offline 3D Semantic Segmentation Benchmarking on ScanNet. Best results are highlighted as first and second.**

Method	Mapping Approach	Without Background			With Background		
		mIoU (%)	f-mIoU (%)	Acc (%)	mIoU (%)	f-mIoU (%)	Acc (%)
ConceptFusion [17]	point-feature	21.76	26.71	34.10	18.57	23.06	28.77
NACLIP-3D [14]		22.32	24.32	33.46	22.32	24.32	33.46
Trident-3D [39]		29.97	37.62	51.06	24.80	27.77	38.43
RayFronts [1]		**41.29**	**46.42**	**56.76**	**32.29**	**39.04**	**49.15**
ConceptGraphs [13]	object-level	21.62	24.32	31.05	20.83	23.61	35.80
HOV-SG [46]		26.79	36.05	35.17	23.48	28.92	38.52
SuperMap (Ours)		27.42	43.50	55.48	22.61	29.10	33.00

Table 2.3: **3D Instance Segmentation Scores on ScanNet.** Best results are highlighted as **first** and second.

Method	Chair		Window		Refrigerator		Sofa		Door	
	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}
HOV-SG [46]	4.58	4.73	0.00	0.00	0.00	0.00	30.00	31.25	9.70	10.40
ConceptGraphs [13]	0.00	2.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SuperMap (Ours)	**63.76**	**74.72**	**42.20**	**67.92**	**62.50**	**62.50**	**33.35**	**83.35**	**10.00**	**25.00**

point feature fusion-based mapping method RayFronts [1]. Our method also significantly outperforms ConceptFusion [17] and ConceptGraphs [13], demonstrating its effectiveness in open-vocabulary segmentation among works on object-centric mapping.

Instance-level Segmentation: Unlike offline methods that rely on complete scene data for prediction, our approach focuses on real-time robotic applications, where **incremental** data fusion and zero-shot adaptability are essential. Incremental prediction poses a greater challenge as it must operate on partial information from the current scan and image while addressing fusion problems. Tab. 2.3 presents a comparison between our approach, HOV-SG [46], and ConceptGraphs [13], all of which support incremental 3D semantic instance prediction.

The results show that our method significantly outperforms both HOV-SG [46] and ConceptGraphs [13], especially for smaller objects that fit within a single camera frame. This highlights the advantage of fusing and tracking full object detections over relying on individual point features or over-segmented instances in instance-level mapping tasks. However, for larger objects like tables where successful tracking requires fusing detections across multiple non-consecutive frames, our method is less effective. Detailed results and further discussion are provided in the appendix.

Computation Resources: To verify real-time performance, we measured memory usage and time per frame (TPF) on ScanNet sequences. SuperMap averages around 0.20 s per frame, supporting on-board operation, whereas HOV-SG [46] and ConceptGraphs [13] need 7–8 s per frame. Comprehensive runtime and memory profiles are provided in the supplementary material.

2.5.3 Spatio-Temporal Object Updates

Our mapping system detects long-term scene changes while preserving the integrity of the object-centric map over extended durations, maintaining both the geometric shape and the unique instance ID of each persistent object. In this experiment, the robot navigated a 30 m $\times$ 20 m indoor environment for approximately 10 minutes, building its map online. After the robot initially traversed certain areas, three objects (a chair, a plant, and a cart) were manually removed, and three new objects (two trash cans and a safety sign) were added.

Fig. 2.2 summarizes the results. The top row illustrates objects that were removed, and the bottom row shows newly added ones. When the robot revisited the modified areas, the dynamic object update module correctly identified the changes: the disappeared objects were removed from the dense reconstruction, the object-centric map, and the scene graph without affecting the identifier of the surrounding objects. Newly added objects were inserted with the correct geometric shape. Throughout the 10-minute trajectory, the identifiers of all unaffected static objects remained consistent, confirming that our spatio-temporal object update module keeps the consistency of the map.

2.5.4 Reasoning with Spatio-Temporal 4D Scene Graph

Our 4D scene graph integrates spatial and temporal relationships, enabling robust reasoning and accurate instance-level object retrieval. We evaluate this using the Google Gemini 2.0 flash multimodal model [45] for object retrieval based on natural language instructions. All inferences are run with Gemini's default generation configuration. For spatial reasoning, Gemini positions itself at a specified location to identify an object. For temporal reasoning, it infers an object's past state or trajectory from current status. We compare Gemini reasoning performance using two inputs:

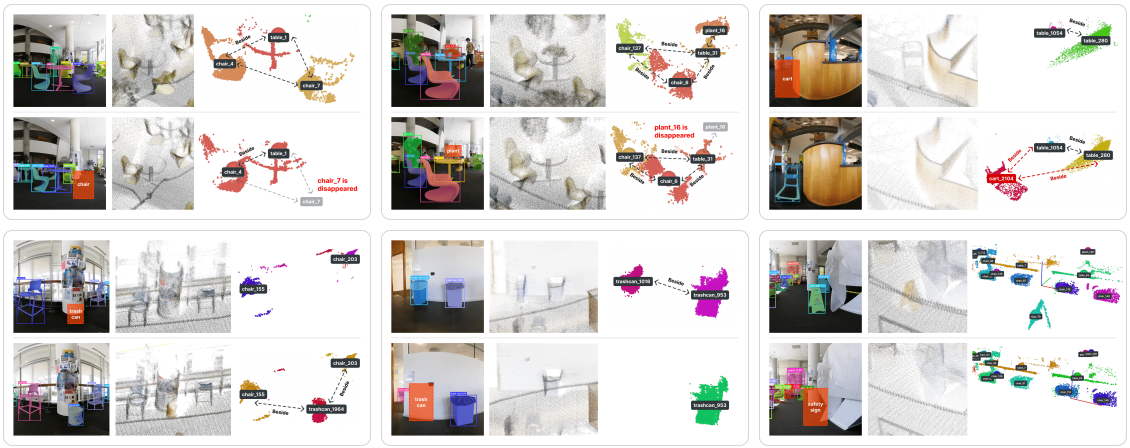

Figure 2.2: **Spatio-temporal object update experiment.** For each scene we show, from left to right, the 2-D detections, the dense 3-D reconstruction, and the object-centric map before and after the change. The top row illustrates the disappearance of a chair, a plant, and a cart, and the bottom row captures the appearance of two trash cans and a safety sign.

- **Scene Graph Input:** Gemini directly leverages spatial and temporal relations, yielding accurate responses.
- **Video Input:** Gemini processes raw frames, often leading to hallucination, object confusion, and misidentification as video length increases.

Prompt 1 in Fig. 2.3 compares spatial reasoning results. We prompt Gemini to position itself at the "cone" and choose the "extinguisher" to use if a nearby "plant" is on fire. With scene graph input, the Gemini accurately interprets object classes, retrieves spatial relations, calculates distances, and identifies the correct object ID. In contrast, video input often leads to confusion between similar objects and misinterpretation of spatial relationships across frames.

Fig. 2.3 prompt 2 compares temporal reasoning results. We prompt Gemini to trace the past trajectory of a "bag" to locate a dropped item after it was carried around. With scene graph input, the Gemini accurately interprets object classes, retrieves spatial and dynamic histories, filters irrelevant records, and returns relevant object IDs and timestamps. In contrast, video input responses often rely on common sense reasoning rather than leveraging spatial and dynamic details, leading to less precise answers. Additionally, video processing takes longer due to the computational cost of encoding high-dimensional visual data [12].

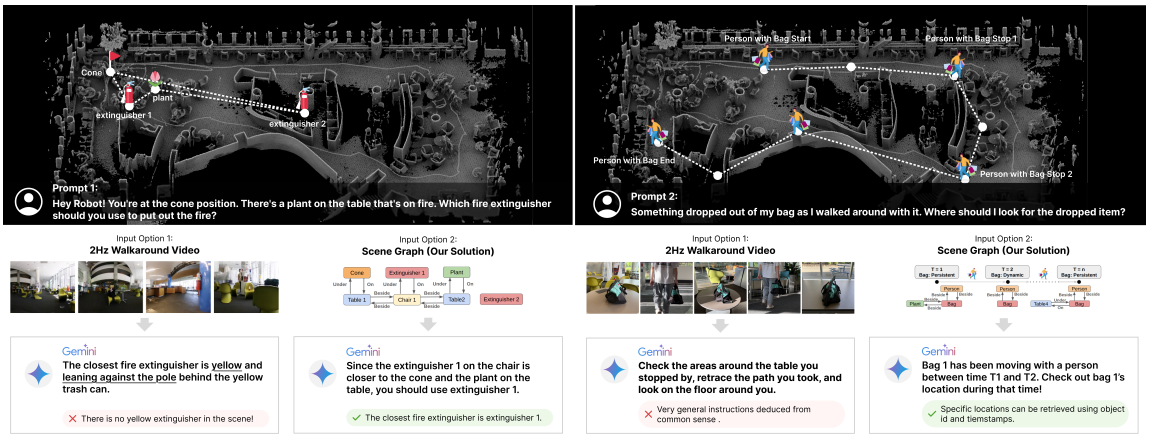

Figure 2.3: **Comparison of reasoning results with video input and scene graph input.** Scene graph input facilitates accurate spatial understanding, flexible temporal understanding, and instance level object retrieval.

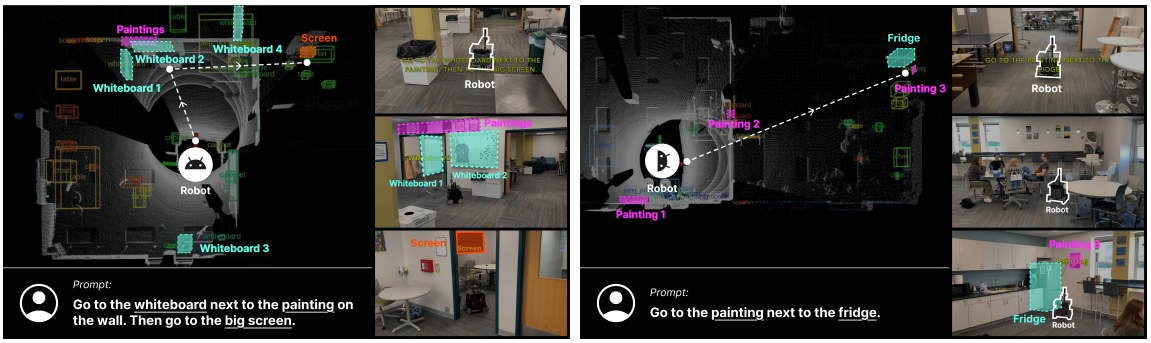

Figure 2.4: **Real-time visual language navigation experiments using our scene graph.** Our system enables realtime instance-level query and navigation.

2.5.5 Real-time Visual-Language Navigation

Visual–language navigation (VLN) challenges agents to follow natural language instructions in complex environments. Most prior work relies on coarse scene representations, limiting reasoning about object identities and spatial relations. We bridge this gap by coupling a language foundation model with our real-time 4D object-level scene graph. The robot begins in an unfamiliar indoor space, explores it, and incrementally constructs an instance-level semantic map. When a user issues an instruction, e.g., "Go to the screen next to the left-most whiteboard", the language model grounds the referents in the scene graph, resolves any ambiguities, and outputs the goal's 3D coordinates. The navigation stack then plans a path and drives the

robot to the correct location.

Fig. 2.4 illustrates the workflow. In the first task (Fig. 2.4 left), even in a room containing four visually similar whiteboards, the robot correctly identifies the target whiteboard and reaches the specified screen. In the second task (Fig. 2.4 right), the robot again successfully identifies and navigates to the specified painting among three similar paintings and arrives at the fridge. These experiments show that our system is memory-efficient enough to support realtime instance-level query.

2.6 Conclusion

This chapter presents SuperMap, a real-time, open-vocabulary SLAM system that builds and maintains an instance-level 4D semantic map. By combining geometric cues from SLAM with semantic cues from 2D foundation models, SuperMap performs joint 2D / 3D object tracking, validation, and change detection on the fly. This unified pipeline keeps the map consistent across both space and time, even in the presence of occlusion, partial views, category shifts, or dynamic object motions. The resulting spatio-temporal scene graph gives robots an explicit, continually updated model of their surroundings, enabling long-horizon memory, rich spatial–temporal reasoning, and language-guided navigation.

Chapter 3

Interaction Planning

3.1 Introduction

Interactive navigation is about robots moving autonomously through their environment dynamically and adaptively, and solutions require a level of intelligence and adaptability that allows the robots to make decisions and adjust their course based on feedback from their surroundings. This paper proposes a solution for interactive navigation in cluttered, unknown environments, focusing on fast and attemptable navigation with environmental interactions. The problem remains challenging as it needs to consider multiple sub-tasks simultaneously, which include: 1) Dynamically updating the environment representation with new sensor data. 2) Executing global path planning to avoid local minima, which also requires considering environmental interactions during the path search. 3) Strategizing interaction planning to manipulate objects. In cases where the environment is large-scale with numerous movable obstacles, the problem's computational complexity increases significantly. Consequently, ensuring the entire system operates in real-time becomes a challenge.

Our work proposes a real-time framework in which mapping, path planning, and manipulation are tightly coupled. Two key ideas achieve this.

The first idea is a hybrid sparse map representation called a Directed Visibility Graph (DV-graph). This graph's edges represent roads and the traversability between the connected vertices. Some of the edges also encode strategies for manipulating movable obstacles away from the path. DV-graph is updated hierarchically at each

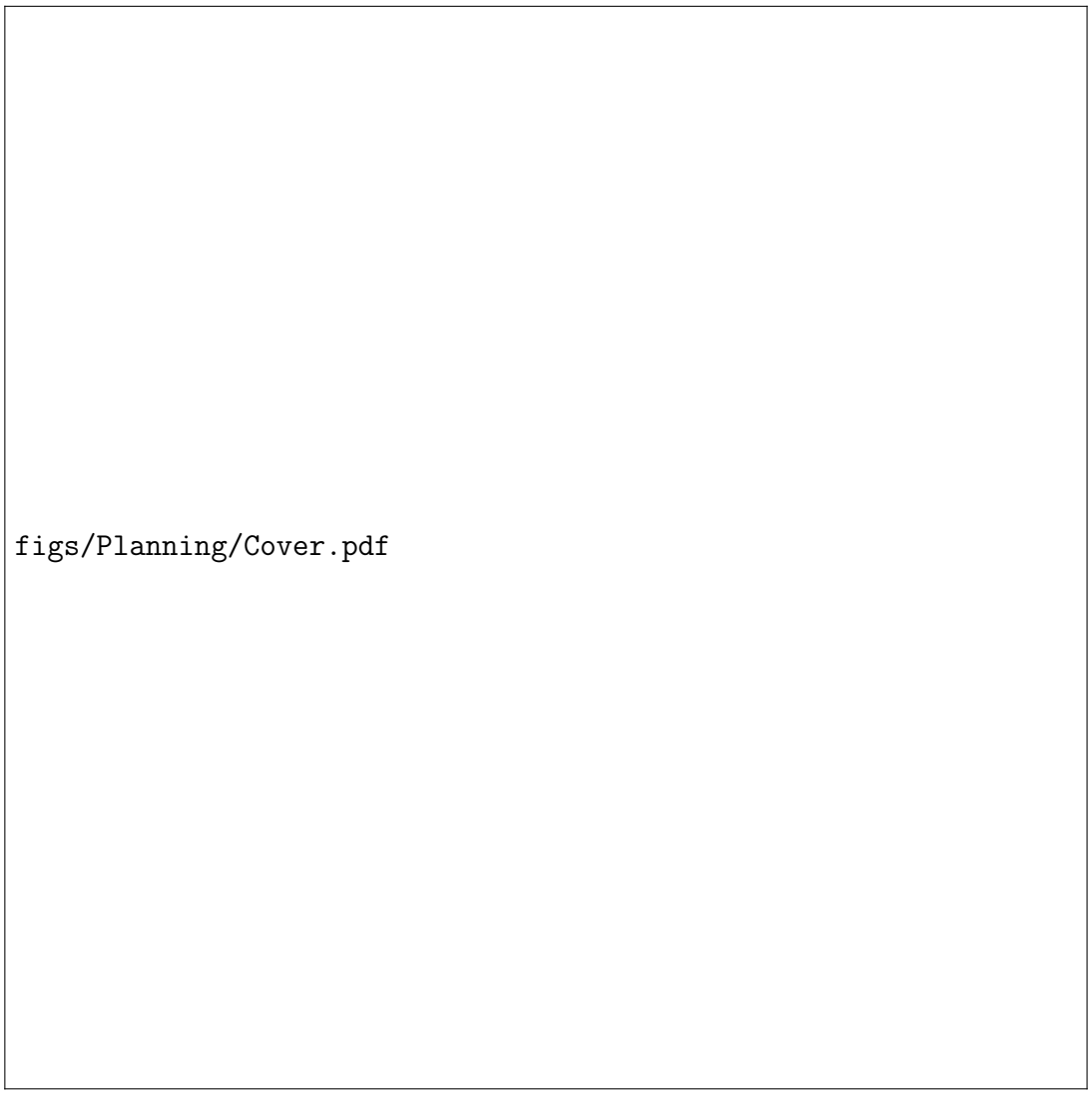

Figure 3.1: Illustration of the interactive navigation through an unknown environment. The colorful curve is the vehicle's trajectory. The obstacles are extracted as red polygons, and the movable objects (orange pixels) are registered as green polygons. The cyan dots represent topological waypoints. The Directed Visibility Graph is marked with cyan and yellow lines. (a) Overview. (b-c) The robot can manipulate the movable obstacles during the navigation and can switch the contact points during manipulation. (d) Object ① is heavy, the robot attempts to move it, update its affordance, and replan an alternative strategy to execute.

data frame. It consists of two layers. The local layer converts the sensor data into a polygon map, on which interaction planning for each movable obstacle is performed. The prospective strategies are encoded into visibility edges and connected between polygons to form the DV-graph. The global layer is a larger DV-graph that is merged and updated with the local layer at each frame. The hierarchical and incremental graph construction framework ensures real-time performance during navigation. Because of the sparse nature of the DV-Graph and its embedded interaction strategies, the global path search can achieve low latency (within $\tilde{1}$0ms) while also considering environmental interactions. This framework also supports manipulation actions such as push, pull, and pick-up. These can be encoded into the same map representation as a multi-level DV-graph for rapid global path search.

The second key idea is adaptive interaction planning via dynamic path search. Given a local region that is divided into multiple components by a movable obstacle, as shown in Fig. 3.3, the planner needs to generate a series of manipulation strategies that re-connect any two dis-connected components so that the robot can clear the movable obstacle from its path and continue the navigation. During the manipulation, the robot infers the obstacle's physical properties through tactile sensing, adaptively adjusts its strategy based on the newly estimated properties, and stores the new strategy in the DV-graph for later purposes.

The proposed system is validated in complex environments of different scales and with movable obstacles. Extensive experiments show that our system is faster than most benchmarks by orders of magnitude while maintaining up to 49% higher path efficiency.

The main contributions can be summarized as:

- An efficient real-time interaction planning method with online adaptation ability for real-world object physics.
- A hybrid map representation that encodes the interaction strategies.
- A complete system for interactive navigation in unknown or partially known environments.

3.2 Related Work

Navigation in Unknown Environments

Navigation in unknown environments has been studied and applied in many field applications, such as for exploration[3], inspection [11], and swarm formation [57]. However, these works regard all obstacles as static and fixed, which limits the robot's ability to move in cluttered environments.

Navigation Among Movable Obstacles

While works on manipulation and navigation for manipulation are prolific, there is less attention on using manipulation for better navigation, i.e., the robot reasons a sequence of manipulation actions to change the pose of movable objects on the map to reach the goal, which isn't accessible or costly to reach without manipulating them. Wilflong[47] proved that navigation among movable objects (NAMO) is NP-hard. Stilman et al.[41, 42] derived optimal solutions for 2D geometric planning for a subset of NAMO problems called $LP1$ with complete knowledge of the environment and objects. Levihn et al. [20] proposed a locally optimal algorithm for $LP1$ NAMO problems in partially known environments, and Muguira-Iturral et al. [27] takes visibility into account in NAMO problem. However, solving a local NAMO problem that is moderately difficult in [41], it takes around one second due to the large search space of manipulations in a long horizon with a dense grid map. In addition, the manipulation actions have been restricted to pushing or pulling at a fixed contact point and direction, and the observation of the shape and state of objects is assumed to be perfect.

Interactive Navigation

There's a recent trend for mapping visual input to agent's actions using reinforcement learning. Xia et al. [48] proposed a simulation environment for interactive robotic tasks. They compared different reinforcement learning algorithms to solve interactive navigation problems, where the robot can push the objects to shorten the path to the goal. However, the agent relies on a precomputed shortest path to operate,

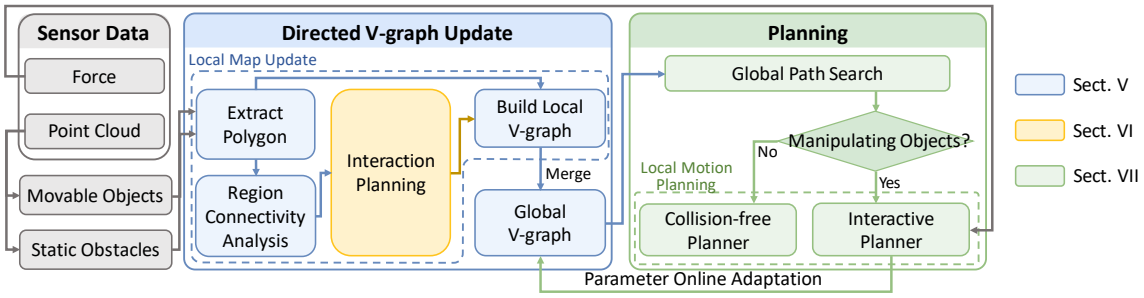

Figure 3.2: A diagram of our system architecture.

which requires accurate knowledge of the map. An end-to-end visual interaction navigation framework is proposed in [53]. The manipulation setting is simplified to a few predefined actions, and the robot only needs to move one object to reach a goal in its view.

This work focuses on a real-time systematic, interactive navigation framework. The introduced hybrid and sparse DV-graph facilitates rapid planning in large-scale, unknown environments. The adaptive interaction planning mechanism endows our system with the capability for online adaptation to the physics of movable obstacles during manipulation.

3.3 Problem Definition

Our problem is divided into two sub-problems: interactive motion planning and adaptive manipulation.

For the interactive motion planning problem, define $Q \subset \mathbb{R}^3$ as the workspace for the robot to navigate. Let $\mathcal{S} \subset Q$ be the perceived sensor data. Here, $\mathcal{S}$ is classified as the object-level point clouds, which contains the points of movable objects $\mathcal{S}_{mov} \subset \mathcal{S}$ and points of the background obstacles $\mathcal{S}_{bg} = \mathcal{S} \backslash \mathcal{S}_{mov}$. We proposed a directed visibility graph (DV-graph) denoted as $\mathcal{G} = (V, \Pi)$, from $\mathcal{S}$, where $\mathbf{v_i} \in V$ represents one vertex and $\langle \mathbf{v_p}, \mathbf{v_q} \rangle \in \Pi$ means an ordered vertex pair in $\mathcal{G}$. The problem can be expressed as:

Problem 1 *Given the robot position* $\mathbf{p}_{robot} \in Q$ *and goal point* $\mathbf{p}_{goal} \in Q$*, find the most energy-efficient path between* $\mathbf{p}_{robot}$ *and* $\mathbf{p}_{goal}$ *on* $\mathcal{G}$*. The energy efficiency considers both the travel distance and effort efficiency when manipulating objects to*

move them out of the way.

Problem 1 is solved repetitively in each planning cycle. During navigation, we online update $\mathcal{G}$ with new sensor data, mapping the newly perceived environment and updating the maintained global map. Then, we re-plan the path on the updated $\mathcal{G}$ in the next planning cycle until arriving at $\mathbf{p}_{goal}$.

For adaptive manipulation, define $Q_{local} \subset Q$ as the local workspace for a single manipulation. Let $\{\mathcal{C}^i_{local} \subset \mathcal{Q}_{local} | i \in \mathbb{Z}^+\}$ be locally disconnected components that are divided by a movable object, as shown in Fig. 3.3. The manipulation strategies are defined as motion primitives π. The problem is described as:

Problem 2 *Given Q_{local} that is divided into n components $\{\mathcal{C}_{local}\}$ by a movable object, find $n(n-1)$ manipulation primitives π that re-connect any two components.*

Problem 2 is solved once for each object that blocks a local space. The manipulation primitives are stored in the map with the object and utilized in two ways. First, it helps the DV-graph determine the travel cost from one component to another to trade off between manipulating or taking a collision-free path. Second, the searched primitives guide the robot's actual manipulation motion controller.

3.4 System Overview

The architecture of the proposed system is shown in Fig. 3.2. The system comprises two primary components: DV-graph construction and updating (Sects. V and VI) and motion planning (Sect. VII). In the DV-graph construction and update phase, the process begins with pre-processed sensor data $\mathcal{S}_{mov}$ and $\mathcal{S}_{bg}$. Then, a polygon extraction procedure transforms $\mathcal{S}_{mov}$ and $\mathcal{S}_{bg}$ into two sets of polygons. These sets form the map representation for the DV-graph.

The topological space connectivity analysis algorithm is designed to determine whether a single movable object divides a local space into several disconnected components. The mobile manipulation simulation module is employed for each movable object if it blocks the way. This module is responsible for generating manipulation strategies to reconnect these locally disconnected spaces and calculating the associated manipulation costs. The output of the simulation module is a comprehensive set of manipulation primitives, encompassing all feasible scenarios considering both

the starting and ending states. Finally, the local DV-graph is constructed with the extracted polygons and the manipulation primitives in order to encode both the geometry and interaction information. The local DV-graph is merged with the global V-graph in each running cycle, facilitating its dynamic update.

In the planning phase, the process initially searches a global path from $\mathbf{p}_{robot}$ to $\mathbf{p}_{goal}$.

During the path execution, the choice of local planner depends on the robot's current navigation context: a collision-free planner is employed when the robot navigates between two visible waypoints. An interactive planner is utilized for mobile manipulation if the current local path segment intersects with movable objects. During manipulation, the object's parameters, e.g., mass, and manipulation cost, are dynamically updated. The interactive planner then adaptively revises the manipulation policy based on these updated parameters. Subsequently, these revised parameters are incorporated to update the corresponding edges in the global DV-Graph.

3.5 DV-graph Construction and Update

3.5.1 Polygon Extraction

To differentiate between static and movable objects in the DV-graph, this module takes in $\mathcal{S}_{mov}$ and $\mathcal{S}_{bg}$ separately and transforms them into two sets of polygons, denoted as $\{\mathcal{P}^i_{mov} \subset \mathcal{Q} | i \in \mathbb{Z}^+\}$ and $\{\mathcal{P}^i_{bg} \subset \mathcal{Q} | i \in \mathbb{Z}^+\}$. To achieve that, we first inflate $\mathcal{S}_{mov}$ and $\mathcal{S}_{bg}$ based on the robot dimension and register them to 2-D image planes. Enclosed polygons are extracted and simplified using the methods in [43] and [10]. This polygon extraction step is developed from [49], detailed explanation is available therein. Finally, the two sets of polygons are combined into a complete local polygon graph, denoted as $\{\mathcal{P}^i_{local} \subset \mathcal{Q} | i \in \mathbb{Z}^+\}$, as shown in Fig. 3.1(a). The $\mathcal{P}_{bg}$ are marked as red polygons, and $\mathcal{P}_{mov}$ are marked as green. Notably, polygons from different classes can intersect, but no intersections occur within the same class due to the topological structure analysis based on [43].

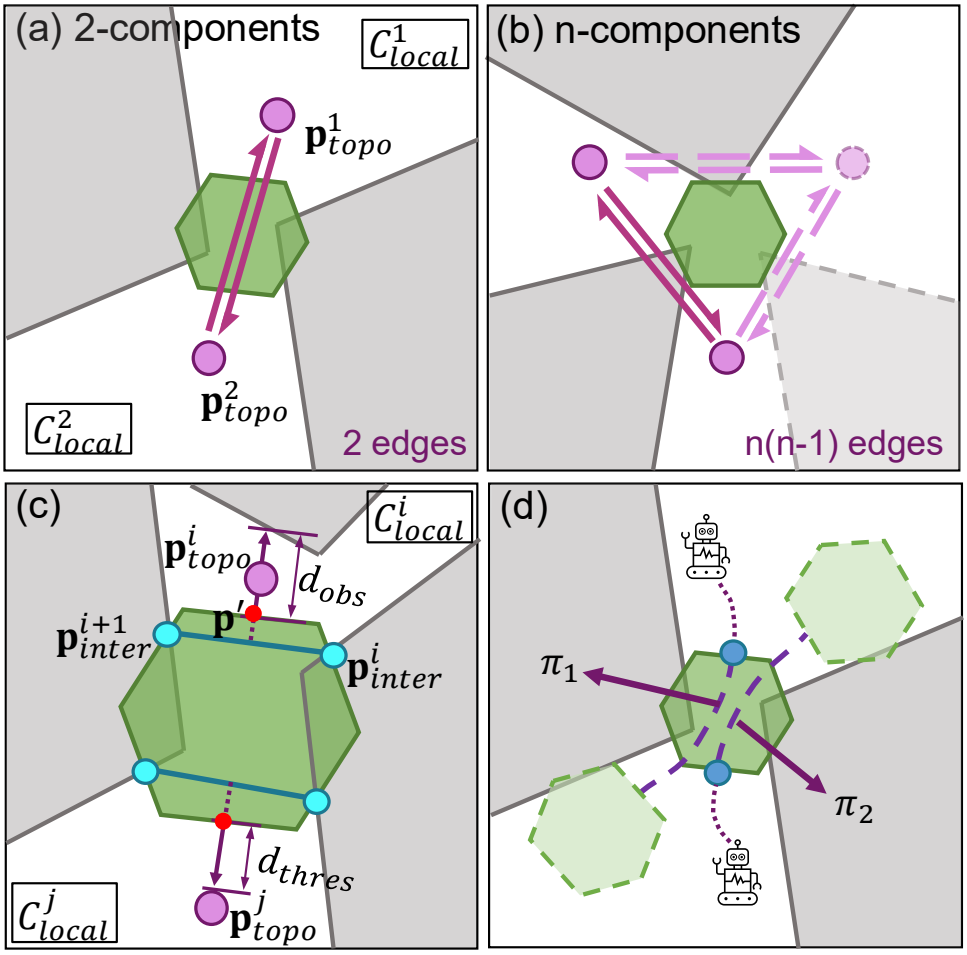

Figure 3.3: (a-b) Path-disconnected components $\{\mathcal{C}^i_{local}\}$ of a local region $\{\mathcal{P}^i_{local}\}$. Green polygon represents $\mathcal{P}_{mov}$ and grey ones are $\mathcal{P}_{bg}$. Topological waypoints $\{\mathbf{p}_{topo}\}$ are marked as purple, potential manipulation strategies are represented as directed edges in purple. (c) Demonstration of the process to choose $\{\mathbf{p}^i_{topo}\}$. Intersected points $\mathbf{p}_{inter}$ are marked in cyan. (d) Illustration of the interaction planning module. Contact point are marked as blue.

3.5.2 Topological Space Connectivity Analysis

The free space of the static background polygon map, denoted as $\mathcal{Q}\backslash\{\mathcal{P}_{bg}^i\}$, is guaranteed to be a path-connected space [36] because of the topological structure analysis. This implies that for any point pairs $\langle \mathbf{p}_i, \mathbf{p}_j \rangle \in \mathcal{Q}\backslash\{\mathcal{P}_{bg}^i\}$, there always exist a connected path. However, when $\{\mathcal{P}_{bg}^i\}$ and $\{\mathcal{P}_{mov}^i\}$ are combined, certain regions become locally path-disconnected, as shown in Fig. 3.3. The space is divided into several locally disconnected components, $\{\mathcal{C}_{local}^i \subset \mathcal{Q} | i \in \mathbb{Z}^+\}$. To navigate through these disconnected spaces, the robot needs to actively manipulate the movable object to re-establish connectivity between regions. As illustrated in Fig. 3.3(a)-(b), the number of disconnected components leads to $n(n-1)$ potential manipulation strategies based on the robot's start and goal regions.

As shown in Fig. 3.3(c), For each $\mathcal{C}_{local}^i$, define $\mathbf{p}_{topo}^i \in \mathcal{C}_{local}^i$ as the representative point of $\mathcal{C}_{local}^i$, which is called topological waypoints because different $\mathbf{p}_{local}^i$ belongs to different topological spaces. To make topological waypoints have good visibility with other vertices in $\mathcal{C}_{local}^i$, the following strategy is adopted. Given a pair of consecutive intersection points $\langle \mathbf{p}_{inter}^i, \mathbf{p}_{inter}^{i+1} \rangle$, marked as cyan in Fig. 3.3(c), calculate its midpoint $\mathbf{p}_{mid}$ and normal vector $\vec{\mathbf{n}} = \overrightarrow{\mathbf{p}_{inter}^i \mathbf{p}_{inter}^{i+1}} \times (0,0,1)$. Then, create its normal vector starting from $\mathbf{p}_{mid}$, and calculate its intersection point with the polygon, denoted as $\mathbf{p}'$ and marked as a red point in the figure. Finally, $\mathbf{p}_{topo}^i$ can be calculated as:

$$\mathbf{p}_{topo}^i = \mathbf{p}' + \vec{\mathbf{n}} \cdot \min(d_{thres}, \frac{d_{obs}}{2}), \tag{3.1}$$

where d_{obs} represents the minimal distance from $\mathbf{p}'$ to its closest obstacle along $\vec{\mathbf{n}}$, d_{thres} is a pre-defined threshold value with unit meters, here we choose 0.5 as the d_{thres}.

Theorem 1 *$\mathbf{p}_{topo}^i$ has at least 3 visible vertices in $\mathcal{C}_{local}^i$.*

Proof: To prove it, two axioms are introduced: i) Any polygon can be decomposed into triangles. ii) Any point in a triangle is visible to all three vertices. Given $\mathcal{C}_{local}^i$, it can be decomposed to a set of triangles $\{\Delta_k \subset \mathcal{C}_{local}^i | k \in \mathbb{Z}^+\}$, because $\mathbf{p}_{topo}^i$ is within $\mathcal{C}_{local}^i$, it must be in one of $\{\Delta_k\}$, thus connecting at least three vertices in $\mathcal{C}_{local}^i$.

If the robot can manipulate the object from $\mathcal{C}_{local}^i$, and achieves $\mathcal{C}_{local}^j$, then the point pair $\langle \mathbf{p}_{topo}^i, \mathbf{p}_{topo}^j \rangle$ is connected added into the DV-graph as a travelsable edge. The

Algorithm 1 DV-graph Update

Input : Segmented Sensor Data: $\mathcal{S}_{bg}, \mathcal{S}_{mov}$, DV-graph: $\mathcal{G}$
Output : Updated DV-graph: $\mathcal{G}$

1 $\{\mathcal{P}_{bg}, \mathcal{P}_{mov}\} \leftarrow$ PolygonExtraction$(\mathcal{S}_{bg}, \mathcal{S}_{mov})$ ▷ Local DV-graph update
2 Construct local DV-graph on $\{\mathcal{P}^i_{bg}\}$
3 **for** *each* $\{\mathcal{P}_{mov}\}$ **do**
4 **for** *each pair* $\langle \mathbf{p}^i_{topo}, \mathbf{p}^j_{topo} \rangle$ **do**
5 $\mathcal{J}, \pi \leftarrow \Gamma(\langle \mathbf{p}^i_{topo}, \mathbf{p}^j_{topo} \rangle, \mu)$
6 **end**
7 Add edge connections between $\{\mathcal{P}^i_{mov}\}$ and $\{\mathcal{P}_{bg}\}$
8 **end**
9 ▷ Local-global DV-graph merge
10 **for** *each vertex* $\mathbf{v}_p \in \{\mathcal{P}^i_{local}\} \cup \{\mathcal{P}^j_{global}\}$ **do**
11 **if** *an association exists* **then**
12 Update the vertex in $\{\mathcal{P}^j_{global}\}$
13 **else if** $\mathbf{v}_p$ *only exists in* $\{\mathcal{P}^j_{local}\}$ **then**
14 Add $\mathbf{v}_p$ to $\{\mathcal{P}^j_{global}\}$ as a new vertex
15 **else**
16 Remove $\mathbf{v}_p$ from $\{\mathcal{P}^j_{global}\}$ based on voting
17 **end**
18 **end**
19 **return** Merged DV-graph $\mathcal{G}$

traversability of the edge is assigned based on the estimated cost of the manipulation.

3.5.3 Interaction Planning

Given a locally path-disconnected region with n-components, as shown in Fig. 3.3(a)-(b), the module is designed to generate $n(n-1)$ local manipulation policies that make each $\langle \mathbf{p}^i_{topo}, \mathbf{p}^j_{topo} \rangle$ re-connected. The input of this module is the local contour graph $\{\mathcal{P}_{local}\}$. For each movable object $\{\mathcal{P}^i_{mov} \subset \mathcal{P}_{local}\}$, the simulation process is described in Algorithm 1, Line 5-9. Given an arbitrary pair of topological waypoints $\langle \mathbf{p}^i_{topo}, \mathbf{p}^j_{topo} \rangle$ with the estimated push affordance μ of the object, the strategy generation function $\Gamma : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R} \times \mathbf{SE}(\mathbf{2})$ outputs the interaction motion primitive π along with the estimated manipulation cost $\mathcal{J}$. Because the output does not restrict manipulation actions, general real-time mobile manipulation solutions can be conveniently adopted into this framework with minor changes. The detailed design of the interaction

primitive generation is discussed in Section 3.6.

3.5.4 Local-Global DV-graph Update

After constructing $\{\mathcal{P}_{local}\}$, we merge the overlapped area of $\{\mathcal{P}_{local}\}$ and $\{\mathcal{P}_{global}\}$ to update the global DV-graph $\mathcal{G}$. The process is described in Algorithm 1, Line 11-19. For each vertex in $\{\mathcal{P}_{local}\}$, we check if there's a close match in $\{\mathcal{P}_{global}\}$ within a certain distance threshold. If so, we update the existing vertex in $\{\mathcal{P}_{global}\}$. If the vertex only exists in $\{\mathcal{P}_{local}\}$, we regard it as a new vertex and add it into the $\{\mathcal{P}_{global}\}$. Conversely, vertices only in $\{\mathcal{P}_{global}\}$, which means they are not observed in the current frame, are not immediately removed. To mitigate sensor noise, we develop a voting mechanism where a vertex is removed from $\{\mathcal{P}_{global}\}$ only after being continuously unobserved for several frames.

3.6 Interaction Primitive Generation

In this section, we propose our solution for Problem 2. We first decompose the Problem 2 into a series of sub-problems (Problem 3). We conduct a kinodynamic path search for a feasible interaction strategy for each sub-problem by expanding motion primitives with a discretized control input space (Sect. 3.6). Our novel design of heuristics is proven to be admissible in general cases and fits well with our sparse polygon-based map representation.

3.6.1 Problem Decomposition

As discussed in Sect. 3.5.3, there are $n(n-1)$ local policies to be generated. It is hard to solve via a single search because the state space is large and with multiple sub-goals, and it is hard to design heuristics with an admissibility guarantee. Therefore, we decompose the original problem into $n(n-1)$ sub-problems and solve them separately.

The process of the problem decomposition is illustrated in Fig. 3.4. Each sub-problem is defined as:

Problem 3 *Given the start position* $\mathbf{p}^i_{topo}$ *and goal position* $\mathbf{p}^j_{topo}$*, find the most energy-efficient manipulation policy that connects* $\mathbf{p}^i_{topo}$ *and* $\mathbf{p}^j_{topo}$ *via a local collision-*

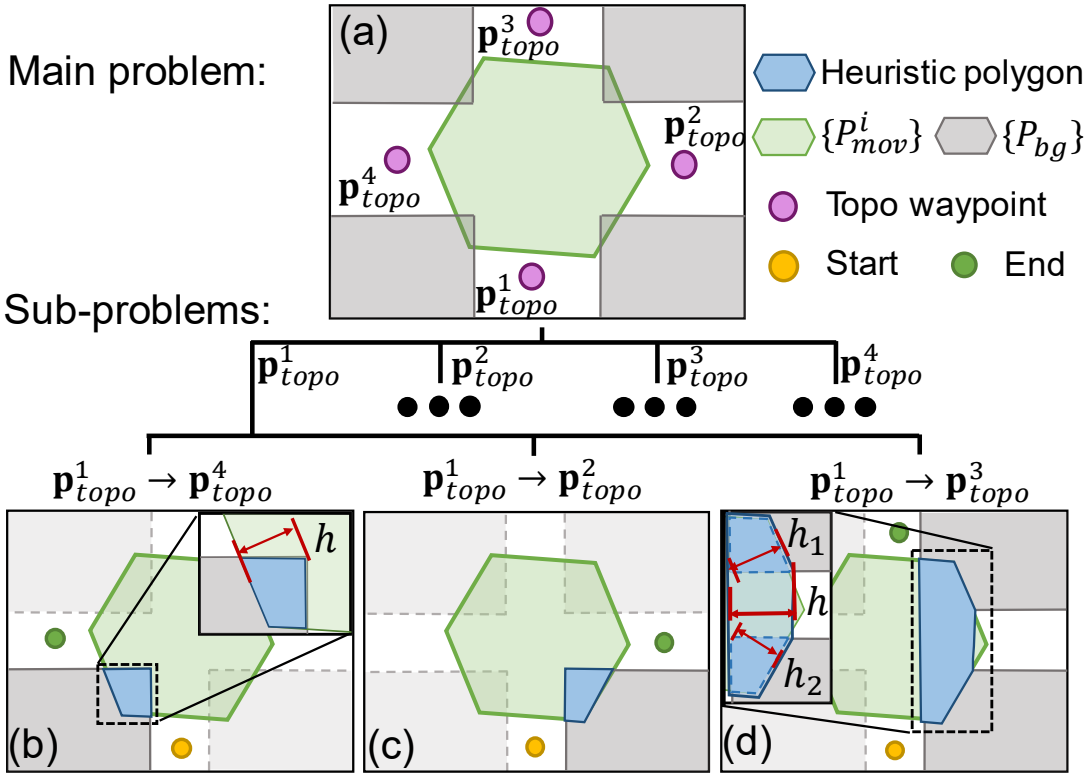

Figure 3.4: Illustration of the problem decomposition. (a) The main problem contains $n(n-1)$ sub-problems. (b-d) Three typical sub-problems. The upper-right corner of (b) explains the calculation of the heuristics. The left-upper corner of (d) demonstrates the case when there are multiple heuristic polygons.

free path.

The problem is solved by a hybrid A* path search method with a novel heuristic design. It has two advantages: 1) the found motion primitive, or path, is kinodynamically feasible for the robot to execute, and 2) the designed heuristics are admissible in general cases, which improves the optimality of the path. Technical details can be found in Sect. 3.6.3.

3.6.2 Push Primitive Generation

The full state space $\mathbb{S}'$ for problem 2 is the combination of the robot's and object's states: $\mathbb{S}' \subset \mathbf{SE}(\mathbf{2}) \times \mathbf{SE}(\mathbf{2})$. Because of the bijection between the robot state and the object's contact points. The object's state can be simplified as a set of available contact points $\mathbf{C} \subset \mathbb{R}^2$, shown as blue points in Fig. 3.3(d). The state space is

then simplified as $\mathbb{S} \subset \mathbf{C} \times \mathbf{SE(2)}$. The state vector $s \in \mathbb{R}^{3+2\times n}$ of a movable object is comprised of object position and orientation (yaw) $x = [p_x, p_y, \psi]^T$ and available contact points $c = [c^1, c^2, ..., c^n]^T$, where $c^i \in \mathbf{C}$, which can be represented as $s := \{x, c\}$. For x dimension, we use velocity as the control input and discretize the input space as $u_x = [-v_{max}, v_{max}]$.

Assume we use a differential vehicle: $v = [v_x, 0], \omega \in \mathbb{R}$. The goal is to find a kinodynamically feasible motion primitive for the state defined above. We use push manipulation as our interacting method to showcase the state transition model for its simplicity and effectiveness in many real-world problems[23][15]. Although planar push is one of the simplest manipulation tasks, two factors make the push control difficult: (a) the model $p(s_{t+1}|s_t, u_t)$ is complicated by many factors, e.g., friction distribution, contact mode, etc.[26]. (b) Both the robot and the push-slider system are underactuated, so it's difficult to find a motion primitive that is kinodynamically feasible for a robot to execute, especially when the robot and the object are sliding relatively[9]. Thus, we favor stable push as defined in [23], aiming to move the object without relative sliding. It solves many problems and simplifies the mapping between the robot and the object to a single transformation in $\mathbf{SE(2)}$.

We identify a subset of potential stable rotation centers based on the object's shape, the friction coefficient k between the robot and the object, and the center of friction. We start with an initial value k_{init} for each object, assuming the object is uniform (the center of friction aligns with its geometric center). This assumption identifies a subset P_s of instant stable rotation centers. If an initial push does not achieve stability, we adjust k_{init} until a stable push is guaranteed. A detailed discussion can be found in Section VII.B.

For a certain contact edge, we can select the robot's motion primitives that will only result in stable pushes and map them to the state of objects. In practice, we apply the differential vehicle model to the movable object and let it head to the normal of the stable push edge. Then we restrict $\frac{\omega}{v_x}$ to let the instantaneous rotation center p fall into P_s.

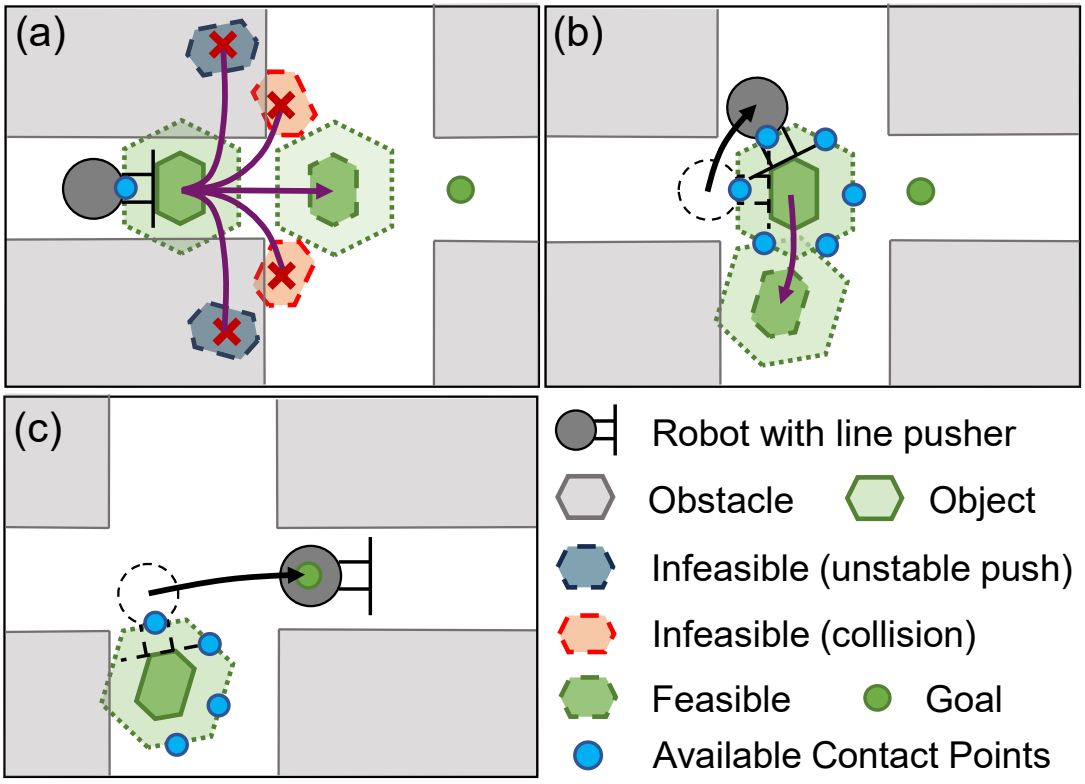

Figure 3.5: Hybrid A* push search with contact point switch.

3.6.3 Hybrid A-Star Path Search

Given a pair of topological waypoints, the problem 3 is solved in two steps. First, find the polygon with the smallest area that intrigues the connection of these topological waypoints, demonstrated as blue polygons in Fig. 3.4(b-d). Here, we call it a heuristic polygon because it provides heuristics to guide the path search. We represent it as $\{\mathcal{P}_{heu}\} = \{\mathcal{P}^i_{mov}\} \cap \{\mathcal{P}_{bg}\}$. If there are multiple heuristic polygons between the two regions, $\{\mathcal{P}_{heu}\}$ is chosen as the union of the vertex set of all these polygons, as shown in Fig. 3.4(d). The heuristic (or the negative reward) of the path search is the minimum distance between parallel lines of the polygon, which is mathematically defined to be the width of the polygon[16]. A hybrid-state astar is then performed with the state and transition model discussed in Sect. 3.6.

3.6.4 Completeness and Admissibility

The completeness cannot be theoretically guaranteed because the state space is discredited. However, our method supports variable-resolution state space to improve the completeness guarantee [8], and the practical performance is satisfactory. For optimality, we can prove the admissibility when the heuristic polygon $\{\mathcal{P}^i_{heu}\}$ or the union of all vertices of multiple heuristic polygons $\{\mathcal{P}^{union}_{heu}\}$ are convex.

Lemma 1 *h is admissible if $\{\mathcal{P}^i_{heu}\}$ is convex.*

According to the definition stated in [16], h is the minimum distance that disengages two convex polygons from contact. Therefore, h is admissible.

Theorem 2 *h is admissible if $\{\mathcal{P}^{union}_{heu}\}$ is convex.*

Proof: As shown in the upper-left part of Fig. 3.4(d), if the union is convex, h must satisfy the condition that $h \leq h_1 + h_2$. Because h_1 and h_2 are both admissible for their heuristic polygons, therefore h is admissible for the union.

3.7 Interactive Motion Planning and Online Adaptation

3.7.1 Global Path Search on DV-graph

To identify the DV-graph $\mathcal{G}$ for the shortest path from $\mathbf{p}_{robot}$ to $\mathbf{p}_{goal}$, the planner first adds $\mathbf{p}_{robot}$ and $\mathbf{p}_{goal}$ as two vertices, connecting them to other visible vertices in P^i_{global}. A breadth-first search is then executed on $\mathcal{G}$ to find the shortest path, denoted as a set of waypoints $\mathbf{P}_{path} = \{\mathbf{p}^i_{wp} \in \mathcal{Q} | i \in \mathbb{Z}^+\}$, if it exists. During the navigation among the unknown environment, vertices and directed visibility edges are dynamically updated and stored in $\mathcal{G}$. In the subsequent operations, the DV-graph $\mathcal{G}$ can be loaded as a prior map. When the environment is fully explored, the global planner is capable of efficiently conducting real-time, globally optimal path searches in large-scale environments ($455m$ path with 1570 vertices in $3ms$, as tested).

Algorithm 2 Path Execution and Adaptation

Input : Next waypoint: $\mathbf{p}_{path}^{next} \in \mathbf{P}_{path}$,
start: $\mathbf{p}_{robot}$, goal: $\mathbf{p}_{goal}$, cost: $\mathcal{J}$,
affordance: μ

```
while p_robot ≠ p_goal do
    if p_wp^next ∈ p_topo and ||p_wp^next − p_robot|| < r then
        Ĵ, π̂, μ̂ ← Γ(⟨p_robot, p_wp^next⟩, μ);
        if ||Ĵ − J|| > τ_thres then
            J ← Ĵ; π ← π̂; μ ← μ̂;
            Update correlated edges in G based on Ĵ;
            Replan the global path;
        else
            InteractivePlanner(π);
        end
    else
        CollisionFreePlanner(p_wp^next);
end
```

3.7.2 Path Execution and Adaptive Replan

During the navigation, the robot switches the local planner based on the current path segment it executes. The working principle is demonstrated in Algorithm 2. The robot employs a collision-free planner [4] for regular navigation. Upon nearing a topological waypoint, it switches to the interactive planner to implement the interaction primitive discussed in Sect. 3.6. The interactive planner considers the following constraints during the runtime: push stability, collision avoidance, and push affordance μ of the robot (e.g., how much weight it can push, the friction coefficient between the pusher and slider). We utilize the same method discussed in Sect. 3.6 for path replan with two differences: 1) a denser grid is utilized for better execution, and 2) only one primitive needs to be generated.

With each sensor frame, the robot utilizes tactile sensing to update the physical properties of the movable obstacle and, therefore, the push affordance. The design of our tactile feedback module is based on three principles:

1. If the object's state doesn't change after maximum push effort, we mark the object as not pushable.
2. If $||\mathcal{J} - \hat{\mathcal{J}}||$ exceeds a threshold, update $\mathcal{J}$ and u_x.

Based on the updated $\mathcal{J}$ and u_x, the planner adaptively re-plans the interaction primitive or the global path as necessary. After the push execution, the updated $\mathcal{J}$ and π are encoded into the DV-graph $\mathcal{G}$ accordingly.

3.8 Experiments

To demonstrate the effectiveness of our interactive navigation system, we conduct 10 navigation tasks in simulated environments with different scales, as shown in Fig. 3.6. For each task, The start and goal positions are chosen based on the criteria that the optimal path length is larger than a threshold. The threshold is set as $15m$ for the $32m \times 32m$ environment and $80m$ for the $330m \times 270m$ one. The simulated ground vehicle is equipped with a Velodyne Puck Lidar used as the range sensor for perception and a 1-D force sensor for tactile sensing. The framework runs on a laptop with an i7-12700H CPU. We configure the system to update the DV-graph at 4.0Hz and perform a path search at each graph update. The spatial resolution is set as $0.15m$. The local layer is a $60 \times 60m$ area with the vehicle in the center.

3.8.1 Computational Efficiency Comparison

To demonstrate the computational efficiency of the proposed framework, we compare our system with three classical path planning methods: **A***, **RRT*** and **BIT***, two NAMO methods: **R-NAMO** and **LAMB**, and our previous work **FAR** that uses a DV-graph without considering interaction. Here, A* is a representative search-based method, BIT* is considered as the state-of-the-art in sampling-based method, R-NAMO and LAMB are two representative methods for NAMO in known and unknown environments, and FAR is considered as the latest global path planning method in unknown environments. The experiment results are demonstrated in Table. 3.1. The italic font indicates that the system is implemented in Python, otherwise implemented in C++. The ground truth (GT) optimal path is noted as the length of the path search by A* on the dense grid map without inflation. The average time consumption is recorded in Table. 3.1 if the path search is successful. The time recorded for sampling-based methods represents the duration until the method initially finds a sub-optimal path, the length of which is no more than 1.5 times the

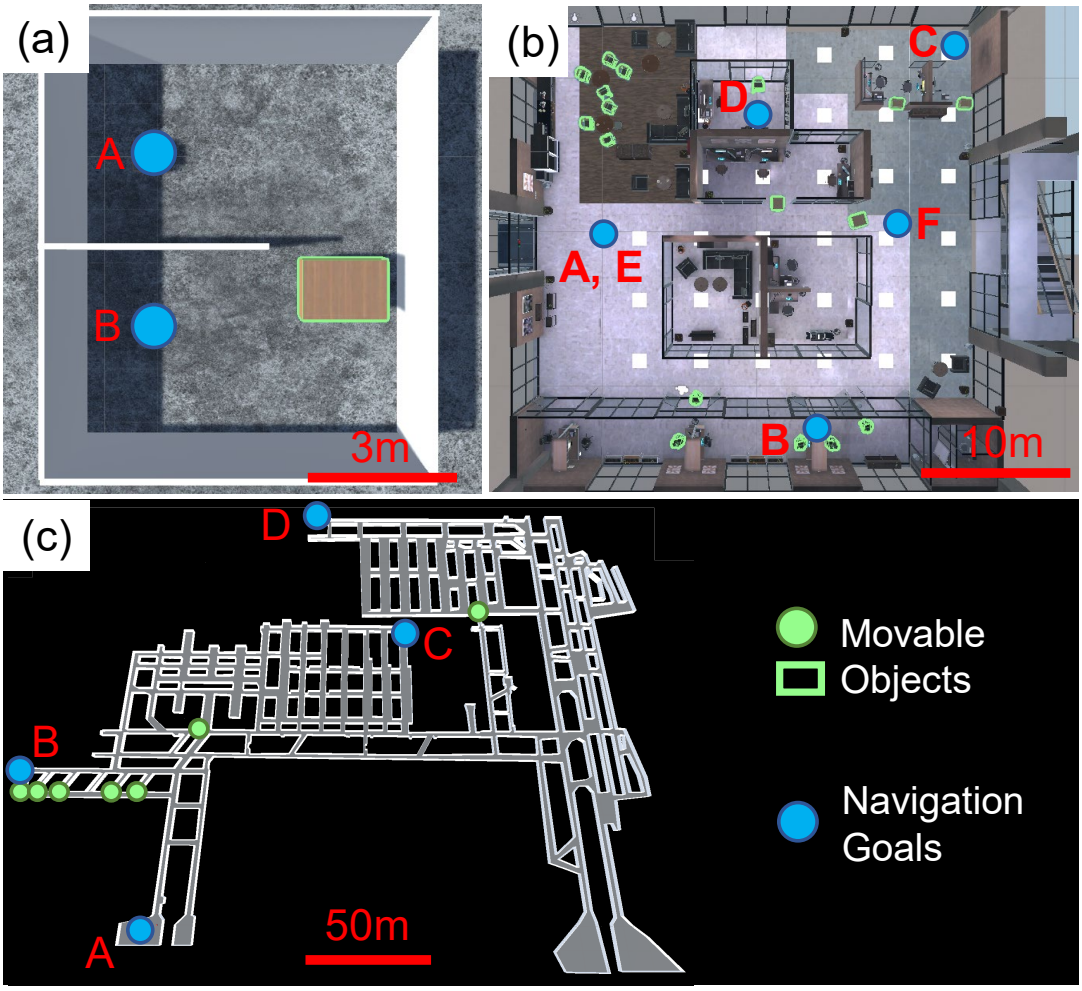

Figure 3.6: Environments overview.

length of the GT optimal path.

As Table. 3.1 shows that considering that the speed of Python can be up to 96.3 times slower than C++ [52], all methods have impressive performance in the room environment. In the office scenario, search-based method runs slower due to the increase of the environment scale, while all NAMO methods fail in this scale. Only FAR and our method can run in real-time and maintain similar speeds in the tunnel scene.

Table 3.1: Average Search Time in [ms]

Environment	Room (no objects)	Room (with objects)	Office	Tunnel
Optimal Path Length (m)	8.6	8.6	47	455
A*	2.2/*179.0(py)*	-	16.0	363.5
RRT*	0.4	-	0.4	325.2
BIT*	**0.17**	-	0.7	1.1e3
R-NAMO	*1.1e3(py)*	*1.2e3(py)*	-	-
LAMB	*7.0e3(py)*	*34.8e3(py)*	-	-
FAR	0.3	-	**0.4**	**1.27**
Ours	0.3	**0.3**	**0.4**	1.32

3.8.2 Path Efficiency Comparison

To demonstrate the path efficiency of the proposed system, we conducted this experiment using the same experiment settings as Sect. 3.8.1. Table. 3.2 presents the results of the experiments. The performance of path efficiency is evaluated by SPL[2]. As shown in the table, A* performs best in the room scenario because of its mapping density and optimality guarantee. In the room with movable obstacles, R-NAMO and LAMB have lower SPL because their manipulation policies take redundant actions. However, our method can plan high quality manipulation policies because it is admissible in general cases. In Office and Tunnel, with the increasing scale of the environments, the path length of manipulation actions has a lower ratio than the total path length; therefore, our method achieves higher SPL in these scenarios. However, R-NAMO and LAMB fail due to their low scalability, and other methods have to take alternate paths or even fail in some sub-tasks as they cannot conduct interactive navigation, thus resulting in lower SPL.

3.8.3 System-level Comparison

To illustrate the potentiality and reliability of our system in field application scenarios, we compare the systematic navigation performance with FAR. The reasons for comparing these two systems are: 1) the superiority of FAR over other path search methods has been demonstrated in [49], and 2) the offline implementations of R-NAMO and LAMB make them fail to be applicable navigation systems for comparison.

Table 3.2: SPL - Success weighted by Path Length

Environment	Room (no object)	Room (with object)	Office	Tunnel
A*	**0.97**	-	0.41	0.81
RRT*	0.84	-	0.37	0.77
BIT*	0.83	-	0.39	0.80
R-NAMO	0.96	0.58	-	-
LAMB	0.96	0.61	-	-
FAR	0.96	-	0.43	0.79
Ours	0.96	**0.70**	**0.86**	**0.97**

We chose Office and Tunnel for the test because they are closer to the field applications. The evaluation metrics included travel distance, navigation time, and success rate for various sub-tasks, with both systems configured identically in terms of local map size, resolution, and update frequency.

The results are shown in Fig. 3.7, Fig. 3.8, and Table. 3.3. Both system achieves satisfying performance in terms of success rate. However, due to the lack of interactive navigation ability, FAR planner makes more attempts to finish each sub-task and sometimes even fails to achieve the goal, e.g., waypoint D in Fig. 3.6(b). The results in Table. 3.3 proves that our system has higher efficiency in different environments.

Table 3.3: System-level Comparison

Environment	Office			Tunnel		
Metrics	Travel Distance (m)	Time (s)	Success Rate	Travel Distance (m)	Time (s)	Success Rate
FAR	232.56	360.34	4/5	1489.63	1911.45	3/3
InteractiveFAR	**109.72**	**242.89**	**5/5**	**1120.91**	**1486.78**	3/3

3.9 Conclusion

This chapter presents InteractiveFAR, an interactive navigation system for large-scale unknown environments cluttered with movable obstacles. Utilizing a dynamic DV-graph that integrates interactions during mapping, the system outperforms benchmarks in path searching. The proposed interactive motion planning and adaptive replan framework help our system manipulate movable obstacles and adjust strategies

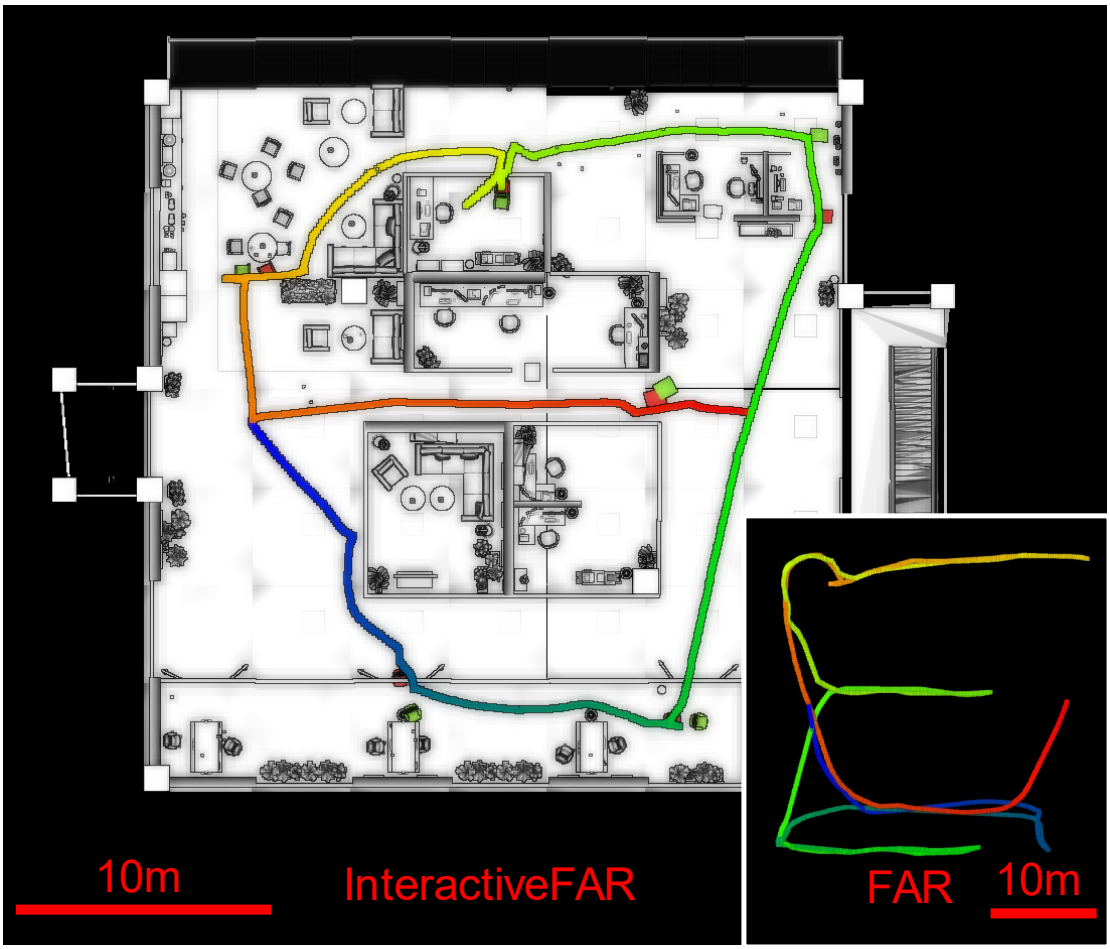

Figure 3.7: The resulting map and trajectories of system-level experiment in Office.

in real time based on new sensor data. Comprehensive experiments and benchmark comparisons validate the efficiency and potential of our system in field applications.

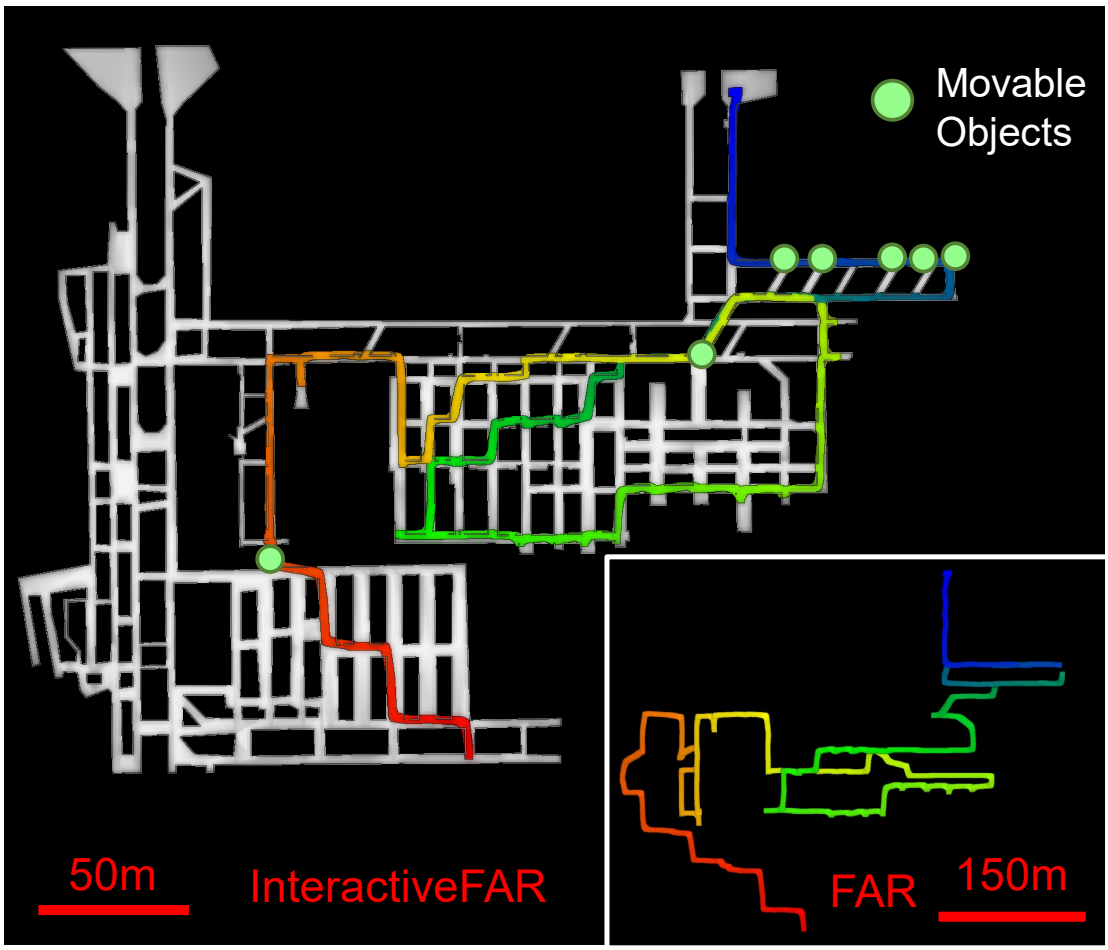

Figure 3.8: The resulting map and trajectories of system-level experiment in Tunnel.

Chapter 4

Conclusions

This thesis presented a modular framework for interactive navigation, addressing the perception and planning challenges that arise in dynamic and unstructured large scale environments. The proposed system achieves real-time performance using only onboard computation and is validated both in simulation and on real robotic platforms. The work is divided into two main components: spatio-temporal instance-level semantic mapping and interactive motion planning.

On the perception side, we proposed a real-time, open-vocabulary, spatio-temporal SLAM system capable of constructing and maintaining a 4D semantic map at the instance level. Unlike prior works limited to static, closed-vocabulary, or offline processing, the proposed perception module performs joint object tracking, validation, and change detection on-the-fly. The system integrates geometric cues from LiDAR SLAM with semantic information from foundation models to build a robust, evolving scene representation. It effectively handles occlusions, semantic label shifts, and long-term environmental changes such as object appearance, disappearance, and relocation. It supports interactive navigation by providing the instance-level map with objects' geometric and semantic features, as well as other downstream tasks such as visual language navigation and long-horizon reasoning in a dynamic world. Quantitative and qualitative experiments demonstrated superior instance segmentation accuracy and temporal consistency compared to existing methods, while achieving real-time processing speeds of 0.2s per frame, which is orders of magnitude faster than offline alternatives.

On the planning side, we introduced an efficient, real-time interactive planning system based on a hybrid sparse map representation called the Directed Visibility Graph (DV-graph). The DV-graph supports topological path planning while encoding possible manipulation strategies for movable obstacles. The planner decomposes the space into static and dynamic regions, detects disconnected components due to obstacles, and generates motion primitives to reconnect the space through manipulation. An adaptive interaction planning mechanism allows the robot to estimate physical properties of obstacles (e.g., mass, friction) and dynamically update the planned strategies based on tactile feedback during execution. This results in robust path planning that trades off between manipulation cost and traversal effort, and supports complex object interactions like push actions to clear a navigable path. Experimental results demonstrated superior planning speed (under 10ms per query) and path efficiency, while maintaining system-level coherence and stability in cluttered environments.

Together, the proposed perception and planning modules form a tightly integrated interactive navigation framework. The entire system is implemented as an open-source software stack. The results affirm the system's capability to build an instance-level semantic map, operate autonomously, and interact adaptively in unknown or partially known, cluttered environments.

This thesis provides a solution for robust real-time robotic navigation that goes beyond passive mapping and planning, enabling agents to actively engage and adapt to their environments. The demonstrated capabilities in mapping, reasoning, and manipulation mark a significant step toward the broader vision of intelligent mobile robots operating autonomously in dynamic human-centric spaces.

4.1 Possible Extensions in Future Work

The interactive navigation system presented in this thesis serves as a minimum viable, real-time, online framework for interactive navigation. While it demonstrates core functionalities, several auxiliary components could further enhance its generality, robustness, and applicability to more complex environments and tasks. Potential directions for future research include:

- **Affordance Estimation.** The current approach to estimating object movability is relatively coarse. Future work could focus on developing more fine-grained affordance models, enabling the system to learn and store physical interaction properties—such as mass, friction, or compliance—at the instance level within the semantic map.
- **Richer Instance Feature Representation.** At present, the planner selects candidate objects for interaction based primarily on semantic class labels. Extending the system to incorporate additional instance-level features, such as texture, geometry, or historical manipulation outcomes, could enable more informed interaction decisions and improve both planning efficiency and success rates.
- **Expanded Interaction Capabilities.** While the system currently supports basic interactions such as pushing, future extensions may include more diverse manipulation behaviors, such as opening doors, pulling handles, or reorienting objects. These capabilities would broaden the range of scenarios in which the system can operate effectively and safely.

Appendix A

Supplementary material for mapping

This supplementary document provides extended experimental results, qualitative analysis, implementation details, and clarifications to support the claims made in the main paper. While our method may not *surpass all existing approaches in isolated segmentation metrics*, it offers unique advantages in dynamic environments and long-term scene understanding.

A.1 Beyond Static Metrics: Evaluating Long-Horizon Scene Consistency

While existing semantic SLAM methods often rely on static metrics such as mIoU or mAP for evaluation, these metrics fail to capture a critical aspect of real-world robotic performance: *long-horizon consistency* in dynamic environments. In practice, autonomous systems must not only detect and segment objects, but also maintain consistent object identities across occlusions, removals, and reappearances. This requires robust data association, temporal memory, and adaptive scene updates—capabilities that are not reflected in traditional segmentation scores.

SuperMap fills this gap by introducing a spatio-temporal object tracking and mapping system that preserves object-level identities over time and across scene

changes. As illustrated in our video demonstrations, SuperMap is able to correctly remove disappeared objects, insert newly observed ones, and re-associate persistent objects despite partial views and semantic drifts. These behaviors are essential for downstream tasks such as language-guided navigation, spatio-temporal reasoning, and long-term planning.

We believe these capabilities call for broader evaluation criteria that go beyond static instance segmentation. To this end, we provide extensive qualitative analysis and real-world robot deployment videos, showcasing how SuperMap maintains scene-level integrity and enables downstream interactions. We encourage future benchmarks to include consistency-aware metrics such as object identity retention, change-aware tracking accuracy, and scene graph stability.

In summary, SuperMap represents a step toward embodied, real-time, open-vocabulary SLAM with reasoning capability. We hope our system and open-source release will catalyze future work on temporally consistent, semantically rich maps for long-term robot autonomy.

A.2 Experiments

A.2.1 Runtime Details for Each Components

We provide additional runtime details information for reproducibility, including:

- Pose estimation: 10hz
- 3D dense mapping and bounding box: 3hz
- 2D instance-level segmentation: 1hz
- 4D scene graph:5hz

Our memory usage and mapping time statistics is shown in Tab. A.1.

A.2.2 Ablation Study: Tracking and Consistency Check

2D-3D joint tracking and consistency check are key techniques of Supermap. We conduct experiments to show their effectiveness in keeping the object ID consistent, removing disappeared objects, and mitigate projection or other outliers.

Table A.1: **3D Instance Segmentation Scores on ScanNet with Memory Usage and Runtime Statistics.** TPF represents mapping time per frame. Best results are highlighted as **first** and second.

Method	Chair		Window		Refrigerator		Sofa		Door		Avg. Mem (MB)	Peak Mem (MB)	TPF (s)
	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}	mAP_{50}	mAP_{25}			
HOV-SG [46]	4.58	4.73	0.00	0.00	0.00	0.00	30.00	31.25	9.70	10.40	8755.86	10226.66	8.623
ConceptGraphs [13]	0.00	2.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**859.86**	**886.18**	**0.092**
SuperMap (Ours)	**63.76**	**74.72**	**42.20**	**67.92**	**62.50**	**62.50**	**33.35**	**83.35**	**10.00**	**25.00**	2030.09	2184.86	0.3604

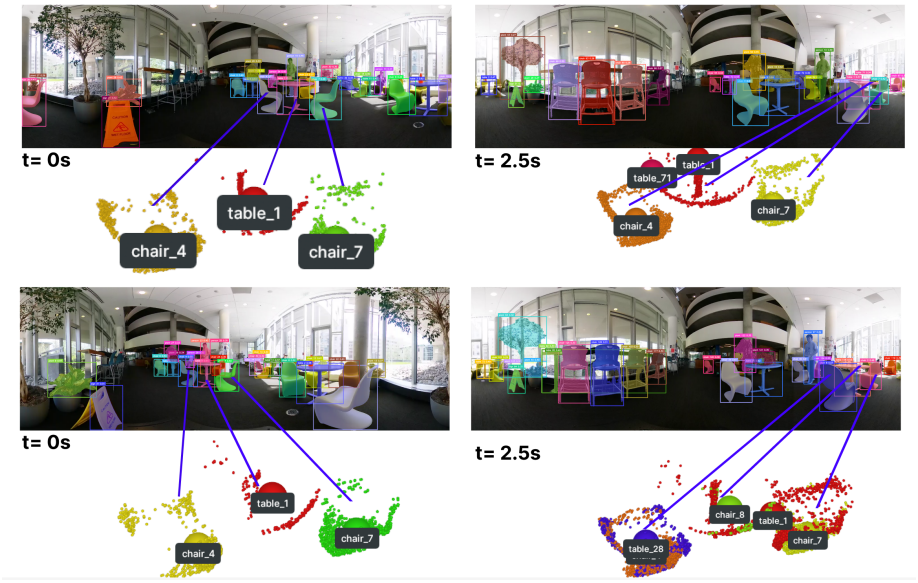

Figure A.1: Ablation of 2D-3D tracking (first row) vs. 2D only tracking (second row) under large viewpoint change. The 2D only case lost the track and degraded mapping.

2D only Tracking vs. 2D-3D Tracking

Figures A.1 and A.2 compare a vanilla ByteTrack baseline (2D-only) with our 2D–3D variant, which compensates the bounding box states using visibility and 3D centroid projections.

- **Large viewpoint change.** In Fig. A.1, rapid camera motion causes the 2D tracker to lose nearby chairs and tables. Our 3D-aware tracker preserves these identities, producing coherent instance ids.
- **Dynamic objects.** Fig. A.2 shows a person walking in front of the robot. The 2D baseline repeatedly initializes new tracks, whereas the 2D–3D tracker maintains a single continuous ID, correctly reflected in the semantic map.

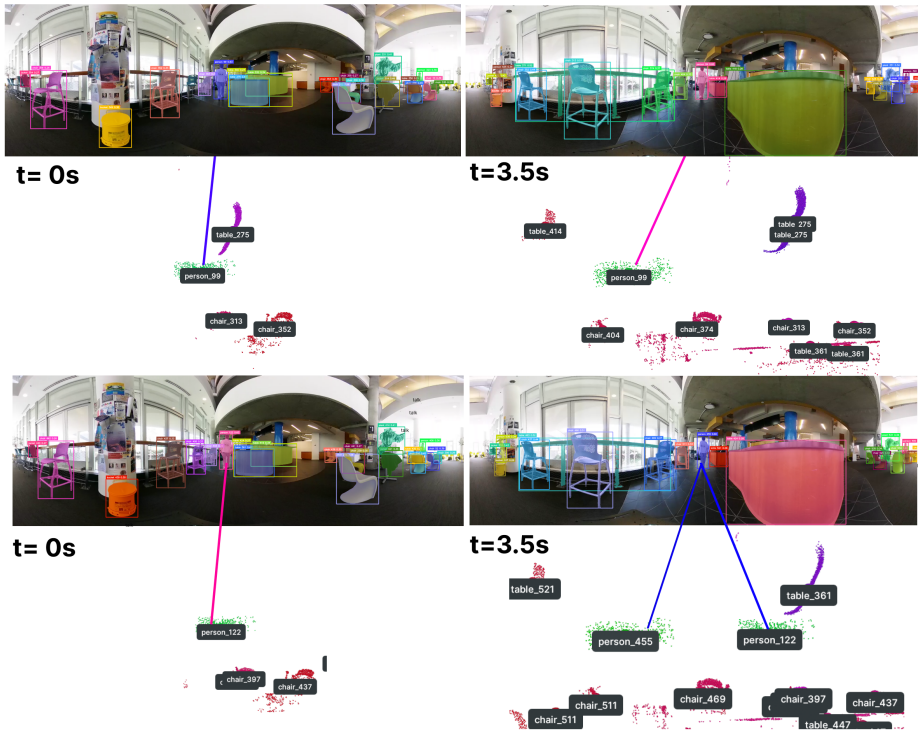

Figure A.2: Ablation of 2D-3D tracking (first row) vs. 2D only tracking (second row) for dynamic objects. 2D only case lost the track and causes the same person modeled as multiple objects in the map.

Impact of Consistency Check

Figure A.3 evaluates the geometric and semantic consistency checks used to prune stale or spurious points as a person walks in front of a static cabinet:

1. *No check.* Without either geometric or semantic consistency checks, history points of the person remain after human moved.
2. *Geometric only.* Enabling the geometric check removes unsupported points but leaves projection outliers on the cabinet.
3. *Geometric + semantic (ours).* Combining both checks eliminates all outliers, yielding an up-to-date map.

The experiment highlights the necessity of joint consistency checking for reliably modeling dynamic scenes.

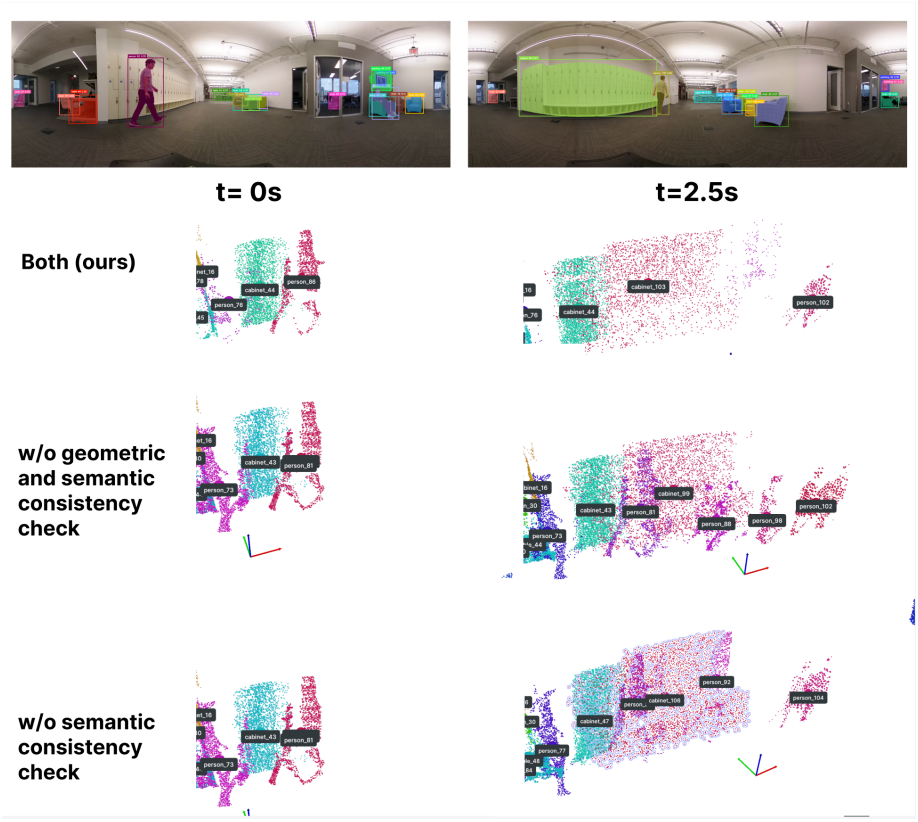

Figure A.3: Ablation of consistency check.

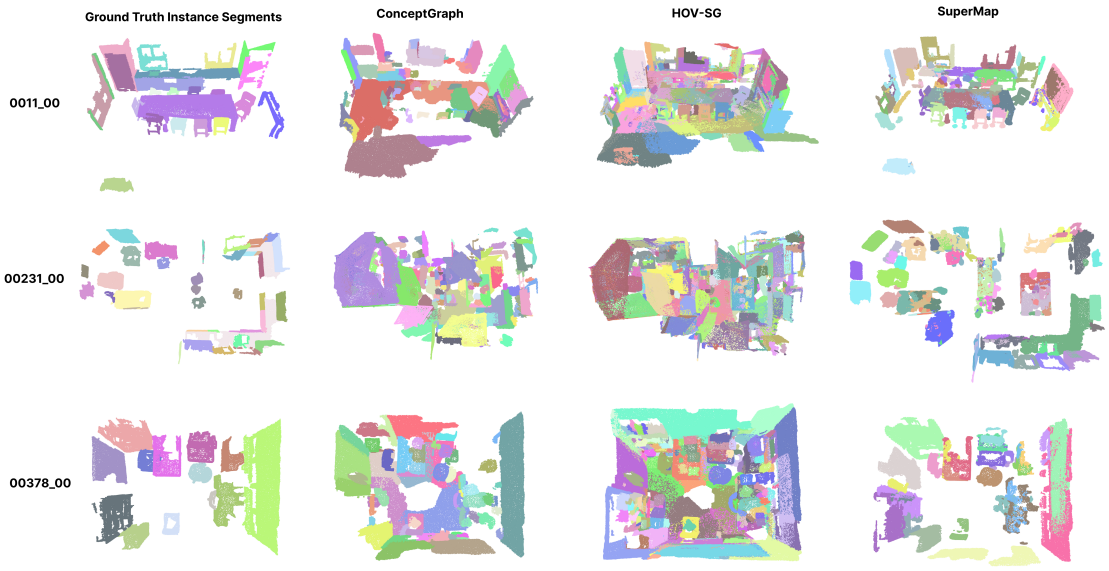

Figure A.4: Comparison of instance segmentation results on sequences with other SOTA methods. Background points are removed. The examples show that our method keeps the identity of the object better and does not ambiguate objects with backgrounds.

A.2.3 Qualitative Instance-level Segmentation Result

We show examples of our instance segmentation results, compared with other SOTA methods. Our method preserves the identity of the object better, while ConceptGraph and HOV-SG tend to fragment a single object. In the meantime, we observe that the CLIP features for instances in ConceptGraph and HOV-SG tend be ambiguous after processing the entire sequence. Background segments cannot be effectively distinguished from foreground objects.

A.2.4 Object-level Mapping for Long-term Dynamic Environments

From Figure A.5 to Figure A.10, we illustrate three newly appeared objects, including a yellow bucket, a cart, and a safety sign; three disappeared objects, including a plant on the table, a blue trash can, and a chair. We compare the first and second traversals of the environment. These examples demonstrate that our object-level mapping system can not only detect which objects have changed, but also update

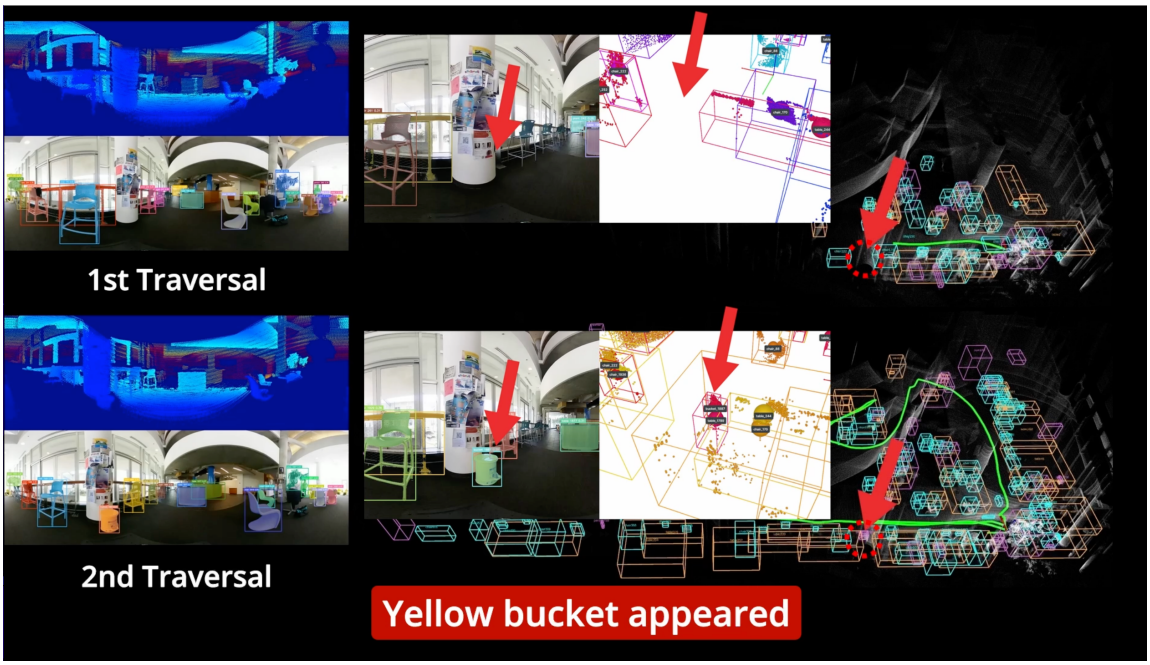

Figure A.5: Newly appeared object: yellow bucket
the existing objects map in time. For more detail, please check out our video.

A.3 Method

We present the spatio-temporal object update algorithm used in our semantic SLAM system. The algorithm takes as input a sequence of observations, instance IDs, and point clouds. Specifically, $\mathbb{B}_{det}$ denotes the set of detected bounding boxes, I is the RGB image, D_I is the depth map, and $Odom$ represents either odometry or the estimated camera pose. A multi-object tracker $\mathcal{T}$, based on motion estimation using a Kalman Filter [54], also to maintain temporal consistency across frames.

A.3.1 2D-3D Consistency Check

To update objects in the map, we perform two key checks:

- Whether the object's voxels remain occupied.
- Whether the occupied and visible voxels are supported by the latest semantic inference from the image.

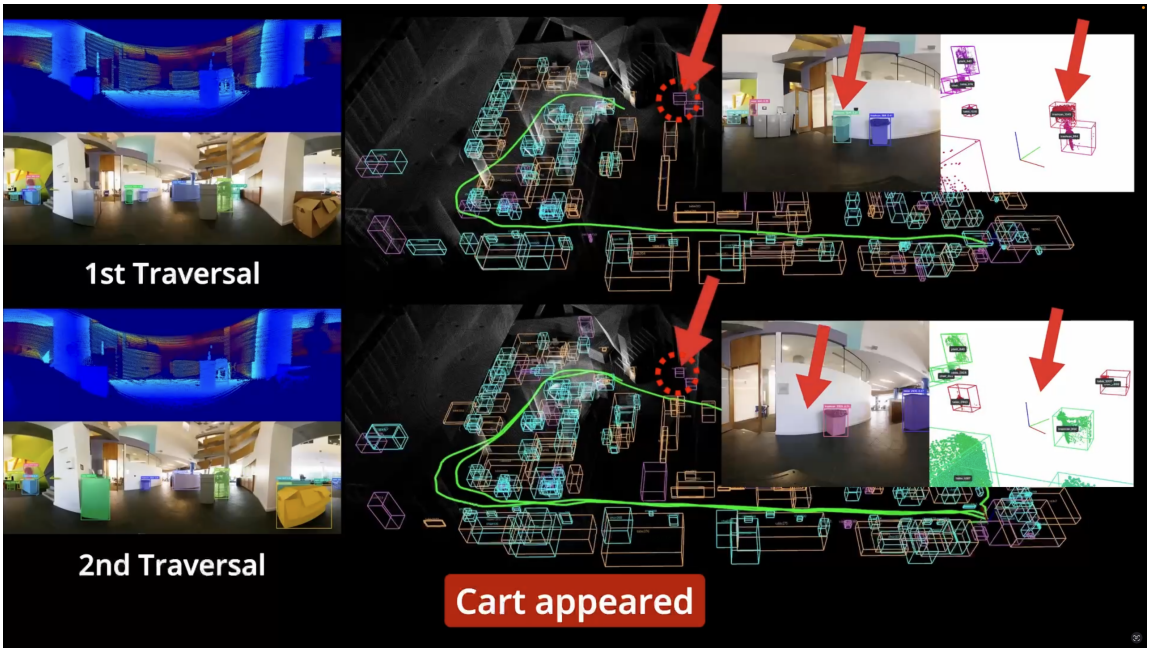

Figure A.6: Newly appeared object: cart

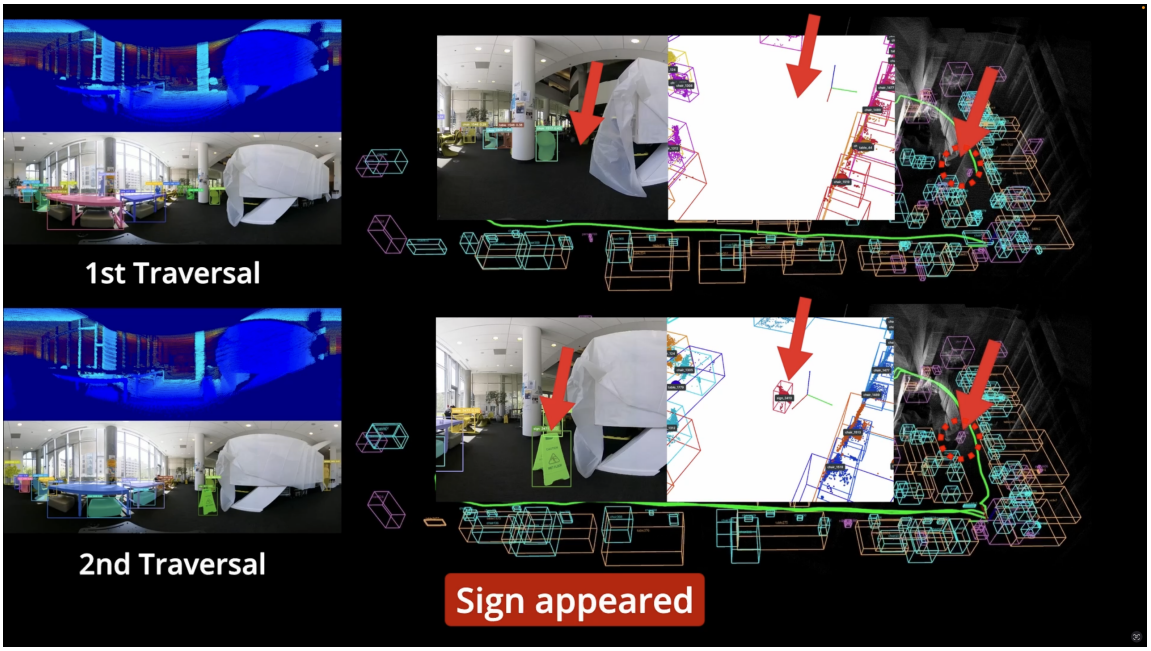

Figure A.7: Newly appeared object: sign

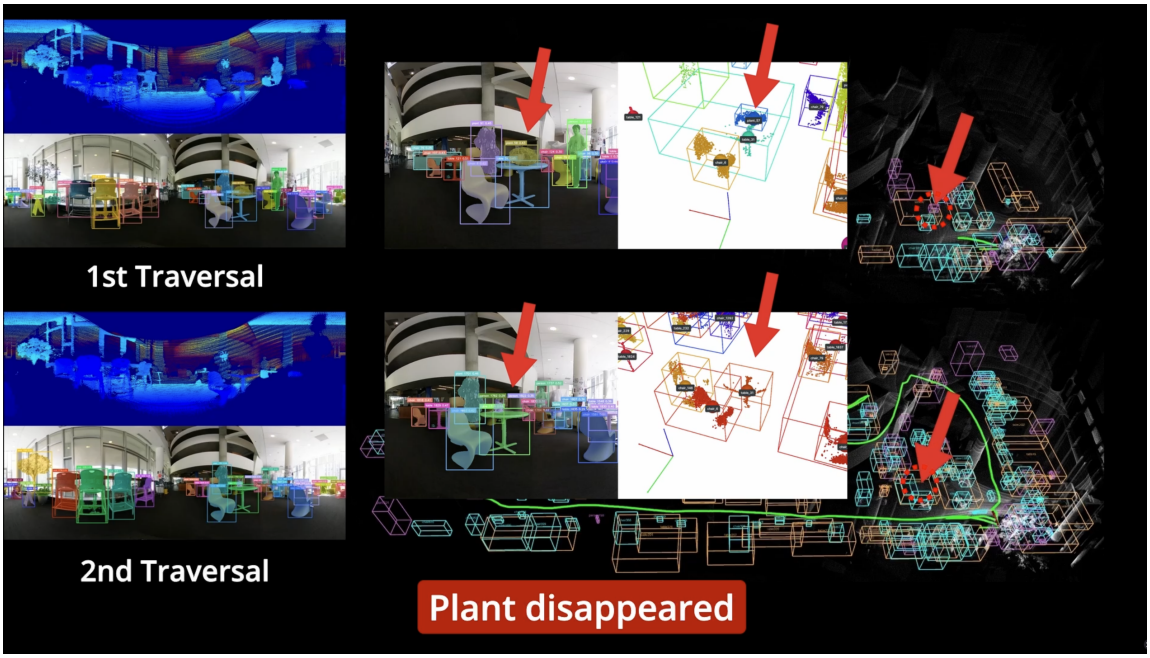

Figure A.8: Disappeared object: plant on table

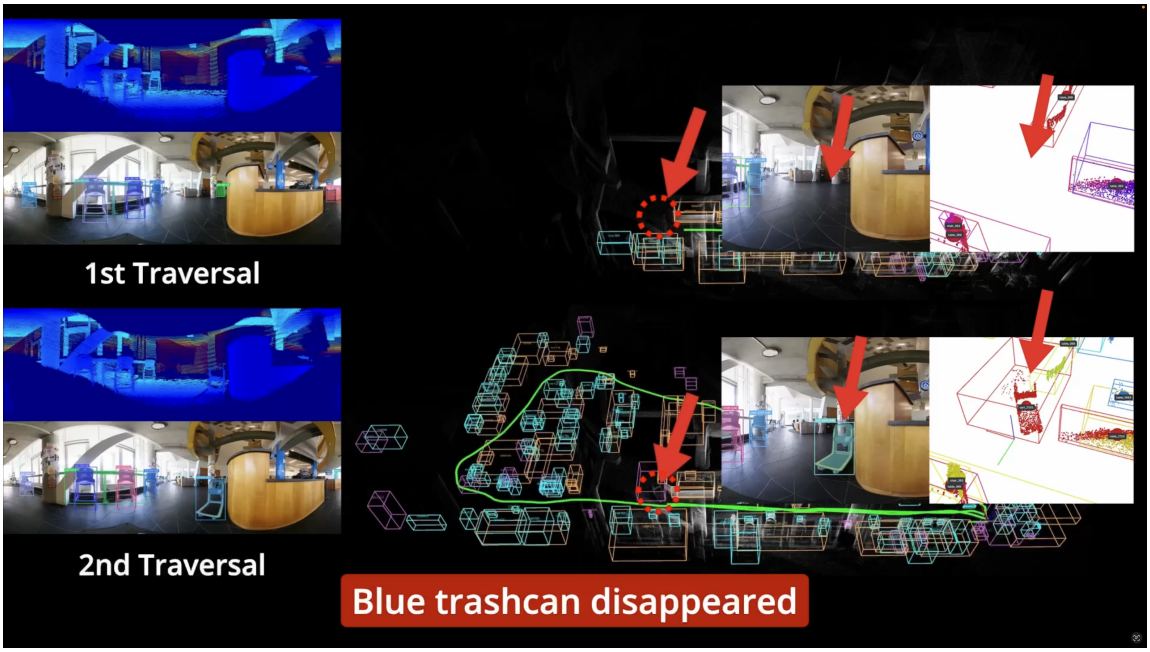

Figure A.9: Disappeared object: blue trash can

Algorithm 3 Spatio-Temporal Map Update

Require: A sequence of observations $\mathbf{S} = \{\mathbb{B}_{\text{det}}, I, D_I, \text{Odom}\}$, and a multi-object tracker $\mathcal{T}$

Initialization: $\mathbf{M} \leftarrow$ `InitMap()`, $\mathcal{T}$`.init()`

forall $\{\mathbb{O}_{neighbor}, \mathbb{B}_{det}, D_I, Odom\} \in \mathbf{S}$ **do**

$\mathbb{O}_{\text{neighbor}} \leftarrow$ `GetNeighborObjects`$(\mathbf{M}, \text{Odom})$

$\mathbb{O}_{\text{front}}, \mathbb{O}_{\text{on}}, \mathbb{O}_{\text{behind}} \leftarrow$ `CompareDepth`$(D_I, \mathbb{O}_{\text{neighbor}}, \text{Odom})$ ▷ Geometric consistency, Eq. 2.7

$\mathbb{O}_{\text{negate}} \leftarrow \mathbb{O}_{\text{front}}$ ▷ Case 2: inconsistent depth

$\mathbb{T}_{\text{alive}} \leftarrow \mathcal{T}$`.getAlive()`

forall $O \in \mathbb{O}_{on}$ **do**

Case 1: consistent observations **if** $O.\texttt{id} \in \mathbb{T}_{alive}.\texttt{id}$ **then**

$\mathcal{T}$`.UpdateTracks`$(O, \mathbb{T}_{\text{alive}})$ ▷ Compensate tracked objects

$\mathbb{T}_{\text{tracked}} \leftarrow \mathcal{T}$`.TrackObjects`$(\mathbb{T}_{\text{alive}}, \mathbb{B}_{\text{det}})$ ▷ Tracking via ByteTrack [54]

$\mathbb{M}_{\text{tracked}} \leftarrow$ `GenerateMasks`$(I, \mathbb{T}_{\text{tracked}})$ ▷ Using SAM2 [33]

$\mathbb{O}_{\text{add}} \leftarrow$ `Unproject`$(D_I, \mathbb{M}_{\text{tracked}}, \text{Odom})$

forall $O \in \mathbb{O}_{on}$ **do**

$O.\texttt{id} \notin \mathbb{T}_{\text{tracked}}$

$\mathbb{O}_{\text{negate}} \leftarrow \mathbb{O}_{\text{negate}} \cup \{O\}$ ▷ Semantic inconsistency

$\mathbb{O}_{\text{neighbor}} \leftarrow$ `ObjNegate`$(\mathbb{O}_{\text{neighbor}}, \mathbb{O}_{\text{negate}})$ ▷ Remove inconsistent objects

$\mathbb{O}_{\text{neighbor}} \leftarrow$ `ObjMerge`$(\mathbb{O}_{\text{neighbor}}, \mathbb{O}_{\text{add}})$

$\mathbf{M} \leftarrow$ `MapMerge`$(\mathbf{M}, \mathbb{O}_{\text{neighbor}})$

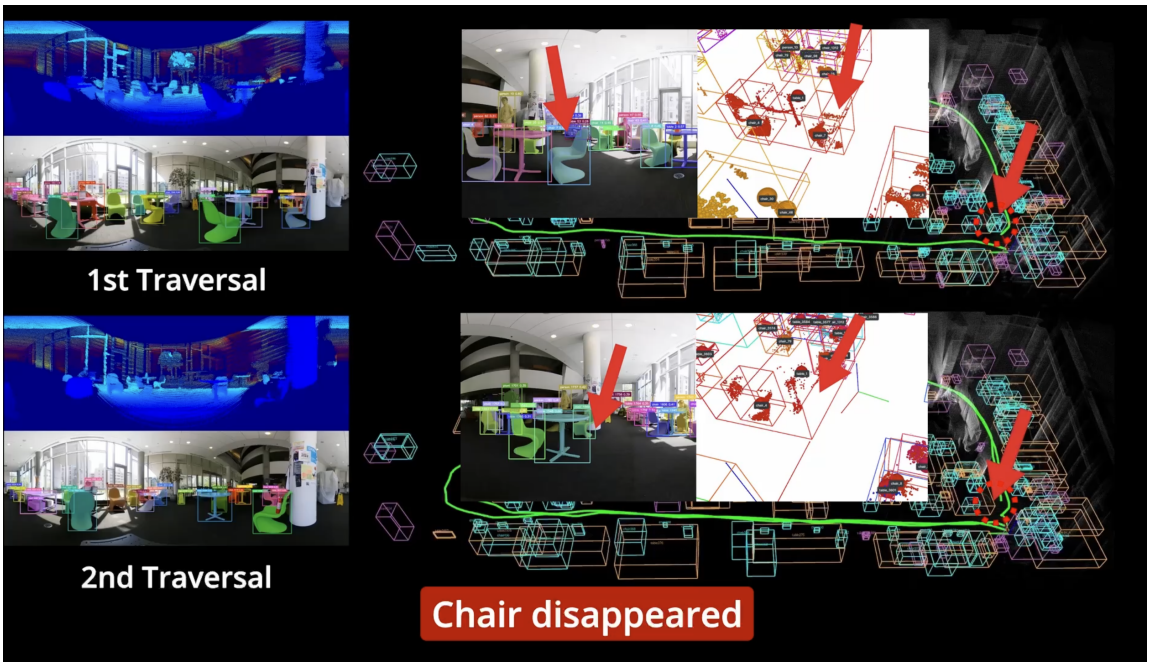

Figure A.10: Disappeared object: one chair
The geometric consistency check is implemented in lines 2–5, while the semantic consistency check is handled in lines 6–19.

A.3.2 3D-aware Instance Tracking

To enhance the matching between new bounding boxes and existing map objects, we introduce a 3D-aware instance tracking mechanism.

Our tracking algorithm builds on Algorithm 1 from ByteTrack [54], preserving the original 2D state representation and motion model for tracklets. In addition, we incorporate geometric cues to update the internal states of the tracklets, enabling more robust and spatially consistent associations.

Bibliography

[1] Omar Alama, Avigyan Bhattacharya, Haoyang He, Seungchan Kim, Yuheng Qiu, Wenshan Wang, Cherie Ho, Nikhil Keetha, and Sebastian Scherer. Rayfronts: Open-set semantic ray frontiers for online scene understanding and exploration. *arXiv preprint arXiv:2504.06994*, 2025. 2.1, 2.2, 2.2, 2.5.2

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 3.8.2

[3] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Robotics: Science and Systems*, volume 5, 2021. 3.2

[4] Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. Autonomous exploration development environment and the planning algorithms. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8921–8928, 2022. doi: 10.1109/ICRA46639.2022.9812330. 3.7.2

[5] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019. 2.2

[6] Linyan Cui and Chaowei Ma. Sof-slam: A semantic visual slam for dynamic environments. *IEEE access*, 7:166528–166539, 2019. 2.1

[7] Yinan Deng, Bicheng Yao, Yihang Tang, Yi Yang, and Yufeng Yue. Openvox: Real-time instance-level open-vocabulary probabilistic voxel representation. *arXiv preprint arXiv:2502.16528*, 2025. 2.3

[8] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The international journal of robotics research*, 29(5):485–501, 2010. 3.6.4

[9] Neel Doshi, Francois R Hogan, and Alberto Rodriguez. Hybrid differential dynamic programming for planar manipulation primitives. In *2020 IEEE In-*

ternational Conference on Robotics and Automation (ICRA), pages 6759–6765. IEEE, 2020. 3.6.2

[10] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973. 3.5.1

[11] Fahimeh Farahnakian, Lauri Koivunen, Tuomas Mäkilä, and Jukka Heikkonen. Towards autonomous industrial warehouse inspection. In *2021 26th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2021. 3.2

[12] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 203–213, 2020. 2.5.4

[13] Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5021–5028. IEEE, 2024. 2.1, 2.1, 2.2, 2.2, 2.2, 2.3, 2.5.2, 2.5.2, 2.5.2, A.1

[14] Sina Hajimiri, Ismail Ben Ayed, and Jose Dolz. Pay attention to your neighbours: Training-free open-vocabulary semantic segmentation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 5061–5071. IEEE, 2025. 2.2

[15] Francois Robert Hogan, Eudald Romo Grau, and Alberto Rodriguez. Reactive planar manipulation with convex hybrid mpc. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 247–253. IEEE, 2018. 3.6.2

[16] Michael E Houle and Godfried T Toussaint. Computing the width of a set. In *Proceedings of the first annual symposium on Computational geometry*, pages 1–7, 1985. 3.6.3, 3.6.4

[17] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023. 2.2, 2.5.2

[18] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 2.2

[19] Ghanta Sai Krishna, Kundrapu Supriya, and Sabur Baidya. 3ds-slam: A 3d

object detection based semantic slam towards dynamic indoor environments. *arXiv preprint arXiv:2310.06385*, 2023. 2.1

[20] Martin Levihn, Mike Stilman, and Henrik Christensen. Locally optimal navigation among movable obstacles in unknown environments. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 86–91. IEEE, 2014. 3.2

[21] Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2.1, 2.2

[22] Xu Liu, Jiuzhou Lei, Ankit Prabhu, Yuezhan Tao, Igor Spasojevic, Pratik Chaudhari, Nikolay Atanasov, and Vijay Kumar. Slideslam: Sparse, lightweight, decentralized metric-semantic slam for multi-robot navigation. *arXiv preprint arXiv:2406.17249*, 2024. 2.2

[23] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The international journal of robotics research*, 15(6):533–556, 1996. 3.6.2

[24] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters*, 2024. 2.1, 2.2

[25] Tomas Berriel Martins, Martin R Oswald, and Javier Civera. Ovo-slam: Open-vocabulary online simultaneous localization and mapping. *arXiv preprint arXiv:2411.15043*, 2024. 2.1, 2.2

[26] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986. 3.6.2

[27] Jose Muguira-Iturralde, Aidan Curtis, Yilun Du, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Visibility-aware navigation among movable obstacles. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10083–10089. IEEE, 2023. 3.2

[28] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023. 2.1, 2.2

[29] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20051–20060, 2024. 2.2

[30] Yuheng Qiu, Chen Wang, Wenshan Wang, Mina Henein, and Sebastian Scherer. Airdos: Dynamic slam benefits from articulated objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8047–8053. IEEE, 2022. 2.1

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2.2

[32] 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. *arXiv preprint arXiv:2307.06135*, 2023. 2.1

[33] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 10

[34] Sonia Raychaudhuri and Angel X Chang. Semantic mapping in indoor embodied ai–a comprehensive survey and future directions. *arXiv preprint arXiv:2501.05750*, 2025. 2.2

[35] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024. 2.1

[36] Alberto Rodriguez and Matthew T Mason. Path connectivity of the free space. *IEEE Transactions on Robotics*, 28(5):1177–1180, 2012. 3.5.2

[37] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020. 2.1, 2.1, 2.2

[38] Lukas Schmid, Marcus Abate, Yun Chang, and Luca Carlone. Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments. *arXiv preprint arXiv:2402.13817*, 2024. 2.1, 2.1, 2.2

[39] Yuheng Shi, Minjing Dong, and Chang Xu. Harnessing vision foundation models for high-performance, training-free open vocabulary segmentation. *arXiv preprint arXiv:2411.09219*, 2024. 2.2

[40] Seungwon Song, Hyungtae Lim, Alex Junho Lee, and Hyun Myung. Dynavins: A visual-inertial slam for dynamic environments. *IEEE Robotics and Automation*

Letters, 7(4):11523–11530, 2022. 2.1

[41] Mike Stilman and James Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12): 1295–1307, 2008. 3.2

[42] Mike Stilman and James J Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2(04):479–503, 2005. 3.2

[43] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1): 32–46, 1985. 3.5.1

[44] Ayça Takmaz, Elisabetta Fedele, Robert W Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023. 2.1, 2.2

[45] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 2.5.4

[46] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. 2.1, 2.1, 2.2, 2.2, 2.3, 2.5.2, 2.5.2, A.1

[47] Gordon Wilfong. Motion planning in the presence of movable obstacles. In *Proceedings of the fourth annual symposium on Computational geometry*, pages 279–288, 1988. 3.2

[48] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020. 3.2

[49] Fan Yang, Chao Cao, Hongbiao Zhu, Jean Oh, and Ji Zhang. Far planner: Fast, attemptable route planner using dynamic visibility update. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9–16. IEEE, 2022. 3.5.1, 3.8.3

[50] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1168–1174. IEEE, 2018. 2.1

[51] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. Hrformer: High-resolution vision transformer for dense predict. *Advances in neural information processing systems*, 34:7281–7293, 2021. 2.2

[52] Farzeen Zehra, Maha Javed, Darakhshan Khan, and Maria Pasha. Comparative analysis of c++ and python in terms of memory and time. 2020. *Preprints.[Google Scholar]*, 2020. 3.8.1

[53] Kuo-Hao Zeng, Luca Weihs, Ali Farhadi, and Roozbeh Mottaghi. Pushing it out of the way: Interactive visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9868–9877, 2021. 3.2

[54] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer, 2022. 2.4.2, A.3, 9, A.3.2

[55] Shibo Zhao, Zheng Fang, HaoLai Li, and Sebastian Scherer. A robust laser-inertial odometry and mapping method for large-scale highway environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1285–1292. IEEE, 2019. 2.2

[56] Shibo Zhao, Hengrui Zhang, Peng Wang, Lucas Nogueira, and Sebastian Scherer. Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8729–8736. IEEE, 2021. 2.4.1

[57] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, et al. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022. 3.2