Intelligent LiDAR Navigation: Leveraging External Information and Semantic Maps with LLM as Copilot

Fujing Xie
Key Laboratory of Intelligent Perception and Human-Machine Collaboration, ShanghaiTech University
Shanghai, China
xiefj@shanghaitech.edu.cn

Jiajie Zhang
Key Laboratory of Intelligent Perception and Human-Machine Collaboration, ShanghaiTech University
Shanghai, China
zhangjj2023@shanghaitech.edu.cn

Sören Schwertfeger
Key Laboratory of Intelligent Perception and Human-Machine Collaboration, ShanghaiTech University
Shanghai, China
soerensch@shanghaitech.edu.cn

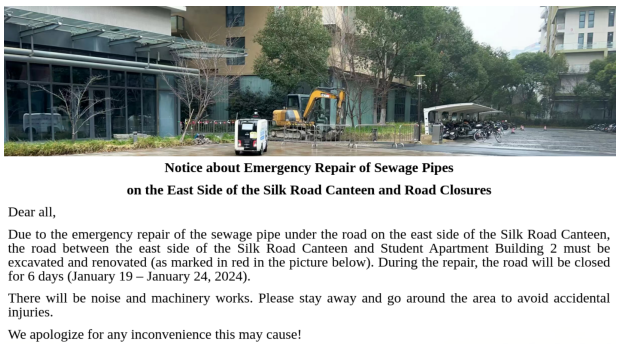
***Abstract*—Traditional robot navigation systems primarily utilize occupancy grid maps and laser-based sensing technologies, as demonstrated by the popular move_base package in ROS. Unlike robots, humans navigate not only through spatial awareness and physical distances but also by integrating external information, such as elevator maintenance updates from public notification boards and experiential knowledge, like the need for special access through certain doors. With the development of Large Language Models (LLMs), which posses text understanding and intelligence close to human performance, there is now an opportunity to infuse robot navigation systems with a level of understanding akin to human cognition. In this study, we propose using osmAG (Area Graph in OpensStreetMap textual format), a semantic map representation to bridge the gap between the capabilities of move_base and the contextual understanding offered by LLMs. Our methodology employs LLMs as actual copilot in robot navigation, enabling the integration of a broader range of informational inputs while maintaining the robustness of traditional robotic navigation systems. Our code, demo, map, experiment results can be accessed at https://github.com/xiexiexiaoxiexie/Intelligent-LiDAR-Navigation-LLM-as-Copilot.**

***Index Terms*—robot navigation, semantic map, large language model.**

I. Introduction

Robots in dynamic environments face challenges when surroundings can't be instantly updated. As depicted in Fig. 1, a campus delivery robot might encounter unexpected obstacles, like a pipe repair closure. Despite prior notice on public websites, the robot's navigation algorithm remains unaware of such obstacles. Most LLM-guided navigation research focuses on using cameras as sensors, as visual data is easier to integrate with language models for semantic information. For instance, [1] achieves basic navigation by asking an LLM to generate code that calls high-level functions like 'turn_left' to control robot movement. [2] navigates robots based on natural language instructions, such as "After passing a white building, take a right next to a white truck." [3] builds an occupancy grid map and generates a language-grounded value map to guide exploration in unfamiliar environments.

In this work, we focus on the navigation task rather than mission planning using LLMs. Traditional navigation algorithms like move_base have been refined over years, but relying solely on occupancy grid maps and laser scans limits their effectiveness in large, dynamic environments, where changes are frequent.



Notice about Emergency Repair of Sewage Pipes
on the East Side of the Silk Road Canteen and Road Closures

Dear all,

Due to the emergency repair of the sewage pipe under the road on the east side of the Silk Road Canteen, the road between the east side of the Silk Road Canteen and Student Apartment Building 2 must be excavated and renovated (as marked in red in the picture below). During the repair, the road will be closed for 6 days (January 19 – January 24, 2024).

There will be noise and machinery works. Please stay away and go around the area to avoid accidental injuries.

We apologize for any inconvenience this may cause!

Fig. 1. The figure above depicts a real-life situation encountered by a 3rd-party delivery robot on our University campus, where it is blocked by an intersection closure. Below the e-mail sent by Office of General Services announcing this closure is shown.

For navigation purposes, the primary concern is whether the path is navigable, rather than the specifics of what may be obstructing it. Therefore, we prefer using laser-based sensors for their direct measurement capabilities and environmental robustness over camera-based approaches.

Navigating complex indoor environments, like large campus buildings, is challenging due to frequent changes such as door operations, furniture shifts, and disruptions like elevator maintenance. Instead of generating new maps for each task, robots should use maps of permanent infrastructure while adapting to real-time conditions, much like humans who rely on both building structure and external information, such as campus notifications .

To support these behaviors, we need a map that is compact and textual, ensuring compatibility with LLMs for token efficiency and ease of human editing to incorporate preferences, compatible with traditional robotic algorithms, and focused on permanent structural information to minimize update frequency. In this work, we utilize the osmAG [4] for these navigation tasks, a hierarchical, topometric, semantic textual map that is utilizing areas defined by polygons which represents physical spaces in the environment, such as rooms and corridors. The line segments of the area polygons that connect two neighboring areas for example doors are called passages. The osmAG can be easily edited by humans to

This work has been partially funded by the Shanghai Frontiers Science Center of Human-centered Artificial Intelligence. This work was also supported by the Science and Technology Commission of Shanghai Municipality (STCSM), project 22JC1410700 "Evaluation of real-time localization and mapping algorithms for intelligent robots". The experiments of this work were supported by the core facility Platform of Computer Science and Communication, SIST, ShanghaiTech University

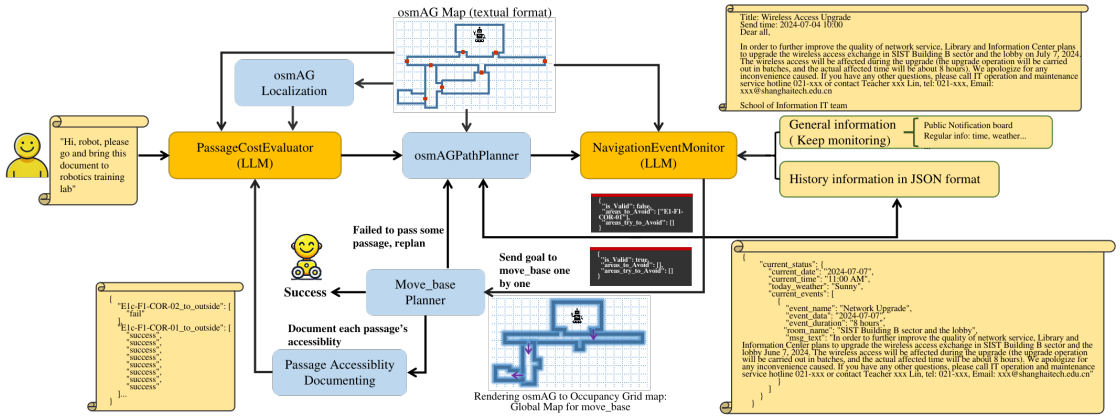


Fig. 2. Pipeline of our algorithm: The *NavigationEventMonitor* keeps tracking events from public notifications and stores information that could impact navigation. When receiving a human instruction, the *PassageCostEvaluator* identifies the destination and assess passage costs based on instructions, accessibility document, and osmAG. This data is used by *osmAGPathPlanner* to plan a path, which is then sent to the *NavigationEventMonitor* for approval. If approved, the path (a list of passages) is sent to move_base to move the robot. If not, *NavigationEventMonitor* suggests areas to avoid, prompting *osmAGPathPlanner* to generate an alternative path until approval is granted.

add semantic information, such as labeling an area as the 'Robotics Training Lab,' making it well-suited for adapting to human environments. The Area Graph was introduced in [5], and the LiDAR-based localization method using osmAG is presented in [6]. Details on the Area Graph and osmAG are omitted here for brevity.

Our contributions are as follows:

- Our system leverages LLMs to comprehend the robot's current capabilities, map data, and combines this with historical data and external information sources like public emails, web announcements, or tweets.
- We integrates LLMs into traditional laser-based navigation systems using osmAG as the map representation, enabling simple implementation on real-world robots equipped with LiDAR and an osmAG map.
- We make our code, simulation environments, and experimental results open-source, allowing others to test different LLMs or navigation techniques.

II. Approach

When robots navigate large buildings, their paths depend on selecting passages between areas, with traversal costs varying by entry and exit points as illustrated in Fig. 3. Humans, however, often refer to areas like "Sector B" rather than specific passages, requiring different treatment of passages and areas when using LLMs for navigation.

We use ChatGPT-4o as our LLM for its advanced capabilities, with the *PassageCostEvaluator* module (Section II-A) assessing passage costs pc_i, and the *NavigationEventMonitor* (Section II-B) validating paths using external information. The areas identified as not accessible based on external information are denoted as A_{inacc}. The pipeline of our algorithm is shown in Fig. 2. Given an osmAG map denoted as $AG = (V, E)$, where V represents vertices (passages) and E represents edges (distance between passages inside one area), the navigation task involves determining the destination area, incorporating extra passage costs, and identifying certain areas to avoid. The goal is to find the optimal sequence of passages from the robot position P_{robot} to the goal area, denoted as $\mathbf{P} = \{v_1, v_2, \ldots, v_k\}$ where $v_1, v_2, \ldots, v_k$ are the intermediate nodes (passages). For $AG = (V, E)$, we use A* to pre-calculate the distance from each passage to every other passage within one area d_{ij} and store this information in a file. This precomputed information is map-specific and does not change over time, as shown by the green paths in Fig. 3. How *osmAGPathPlanner* plan a path is described in Section. II-C, and the integration with ROS move_base is in Section. II-D.

A. Human Instruction Processing & Passage Cost Evaluation

As shown in Fig. 2, the system starts by receiving a human instruction, such as "Please take this document to the robotics training lab." Using the osmAG map, the LLM identifies the destination area and evaluates the cost of each passage based on the type (automatic or handle doors) and the robot's past experiences. Robots that can open doors or rely on automatic doors have lower costs, while consistently problematic passages, like frequently closed fireproof doors, incur higher costs. The module outputs the destination area and passage costs.

B. Navigation Event Tracking & Path Validation

As shown in Fig. 2, the *NavigationEventMonitor* tracks events that could impact navigation, such as elevator maintenance. When such information is received, the LLM assesses its relevance to the robot's tasks. If relevant, a JSON string is generated and stored for later path approval. This module also validates paths from *osmAGPathPlanner*, ensuring they avoid invalid areas based on event data and the robot's capabilities, such as recognizing that a wheeled robot can only use elevators in multi-floor navigation as illustrated in III-E.The module's output includes the validation of the shortest path and identifies areas to avoid if the path is invalid.

C. osmAG Path Planner

Besides the precomputed distance between passages, we introduce external passage costs pc_i as mentioned in Section II-A. Therefore, the total cost of edge (v_i, v_j) is given by:

$$t_{ij} = d_{ij} + pc_i + pc_j$$

The passages in A_{inacc} are denoted as $E_{\text{inacc}} \subset E$. By removing E_{inacc} from E, we obtain $E' = E \setminus E_{\text{inacc}}$.

To incorporate the start position and the destination into the graph, we add vertices and edges accordingly. For the robot's starting position P_{robot}, we add edges between P_{robot} and all passages in the start area (V_{start}). Similarly, for the destination, we include the centroid of the destination area and create edges connecting it to all passages within the destination area ($V_{\text{destination}}$). Let v_s represent the robot's starting position P_{robot}, and v_d represent the centroid of the destination area. The updated set of vertices V' and edges E'' can be expressed as:

$$V' = V \cup \{v_s, v_d\}$$

$$E'' = E' \cup \{(v_s, v_i) \mid v_i \in V_{\text{start}}\} \cup \{(v_d, v_j) \mid v_j \in V_{\text{destination}}\}$$

The final graph for this navigation task, incorporating the start position and destination centroid, is defined as $AG' = (V', E'')$. The path $\mathbf{P}$ is then determined as $\mathbf{P} = A^*(AG')$.

As shown in Fig. 3, the A* method would select the path in (a) only utilizing precomputed passage lengths. However, with insights from the *PassageCostEvaluator*, the left path is assigned a higher cost, leading to the path in (b). The *NavigationEventMonitor* then excludes the yellow area, resulting in the final approved path in (c). Based on the selected areas, we set the polygons of chosen areas as 'occupied' and designate the chosen passages as 'free' (with unselected passages remaining 'occupied'). This ensures the path planned by move_base follows the desired route, as shown in (d).

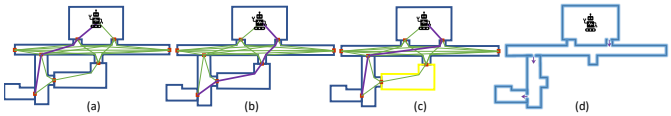


Fig. 3. *osmAGPathPlanning* process: Blue polygons represent areas, red lines denote passages, green paths indicate distances between passages within an area (map-specific and stored in a file), and purple paths show the current shortest path. (a) Shortest path based on distance alone; (b) Experience-based adjustment favors the right path due to consistently open passages; (c) Event monitor advises avoiding the yellow-marked area, likely occupied due to an event; (d) Final path approved by the *NavigationEventMonitor*, with each passage sent as a goal to move_base. The occupancy grid map is rendered as the move_base global map, with other passages closed to enforce path adherence.

D. Integration osmAGPathPlanner with ROS move_base

After the *osmAGPathPlanner* selects the path $\mathbf{P}$ and receives approval from the *NavigationEventMonitor*, it sends each passage as a goal to the move_base and renders a occupancy grid map that is restricted to the chosen areas and the selected passages as the global map in move_base. This ensures the move_base global planner strictly follows the *osmAGPathPlanner*'s path (see Fig. 3(d)). If the robot encounters an inaccessible passage, despite using past experiences and notifications, the failure is recorded, and the *osmAGPathPlanner* re-plans the path, repeating until the robot reaches its destination.

E. Documenting Passages Accessibility

As the robot navigates, this module records the success or failure of each passage traversal, storing the data in a file. The *PassageCostEvaluator* module then sends this data to the LLM to evaluate the cost of each passage (pc_i). If a passage is found inaccessible, it is marked as 'infeasible' and removed from AG' during re-planning of this trial.

III. Experiments

Overall, our approach consistently outperformed move_base by maintaining accurate and context-aware path planning, demonstrating the effectiveness of integrating external information through *PassageCostEvaluator* and *NavigationEventMonitor*. In Section III-A, we outline our experiment setup. Section III-B details how we assess the LLM's ability to track navigation-related information and validate a proposed shortest path. Section III-C compares our method with ROS move_base, while Section III-D evaluates the contributions of the two LLM modules to our approach. Finally, Section III-E demonstrates that our system can operate across multiple floors and avoid elevators under maintenance.

A. Experiments Setup

1) Robot and World: In our experiments, we simulate the "turtlebot3 waffle" model with a 6-meter range laser sensor. Both our method's robots and those using the move_base package share identical parameters for a fair comparison. The environment is simulated in Gazebo, based on an osmAG of a real campus building with over 6,200 square meters and 70+ rooms per floor. We use ROS 'Navfn' as the global planner and 'DWA' (Dynamic Window Approach) as the local planner for move_base. The robot's operations are mostly confined to the first floor for two reasons: move_base lacks multi-floor navigation, and multi-floor access could simplify obstacle avoidance by allowing vertical rerouting. Limiting the tests to the first floor better challenges the algorithm to navigate longer detours outside the building.

2) Cases: To assess the system's ability, we designed 5 experimental cases (4 on the first floor, 1 spanning 2 floors), each involving a simulated instruction and a real event notification sent by the campus administration. For example, an instruction might be, "Please bring these tools to the same room on the second floor," combined with event details. Several starting areas, or trials, are established for each case, along with same destination. In each trial, the robot must recognize restricted areas based on prior notifications and plan a detour around them. The trials are designed so the shortest paths typically cross restricted areas, requiring the robot to adapt intelligently. The experience database begins empty for each case, and as the system navigates, *PassageCostEvaluator* documents each passage's accessibility, gradually building a comprehensive understanding of the environment to optimize performance in subsequent trials.

3) Metric: In some cases, we close the doors of restricted areas, while in others, we leave them open. Therefore, we compare the system based on both the path length and whether the robot enters restricted areas with move_base.

B. Event Tracking & Path Approval Experiment

To ensure the effective operation of the *NavigationEventMonitor* module, it must first assess the relevance of each notification to the robot's navigation task and then determine if the proposed path is valid or needs adjustment based on the tracked events. The module demonstrated high accuracy in determining notification relevance. Out of 20 notifications (10 navigation relevant, 10 irrelevant), the LLM correctly identified whether each notification was relevant to robot navigation. Across 37 trials, 95 path approval requests were sent: 57 of 61 valid paths were approved (with is_valid set to 'true' or the areas to avoid not in the path), and all 34 invalid paths were correctly flagged by setting is_valid to false. However, in 14 of the 95 requests, the responses were semantically correct but deviated from the specified output format, such as returning 'B sector' or a room number instead of the exact area name.

C. Comprehensive System Experiment

We conducted tests across 4 first-floor cases and 37 trials to evaluate our system, comparing it to the ROS move_base package with and without clearing the obstacle layer between trials. Note that the results from move_base without obstacle layer clearing closely approximated the actual shortest path in later trials in one case. But in real life implementation, the global map needs periodic clearing to prevent sensor noise,

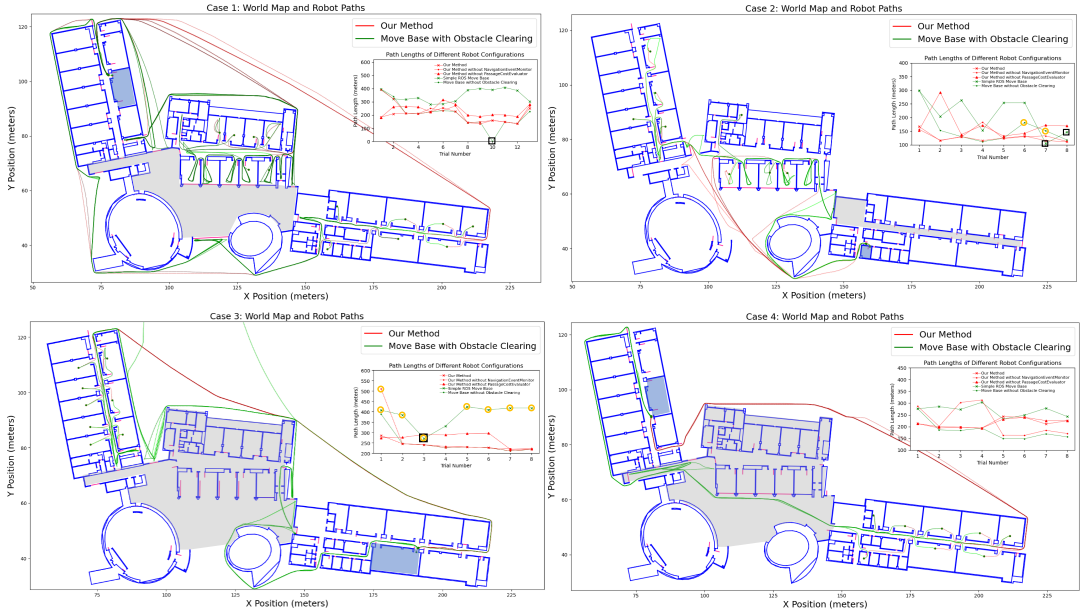


Fig. 4. This figure presents the paths from four experimental cases. For clarity, only the paths from our method and move_base with clearing between trials are displayed on the map. In the map, blue lines represent areas in the osmAG, while red lines indicate passages between them. Each case was tested in a world with different door configurations (open or closed). Grey shaded zones represent restricted areas caused by events in cases: a graduation party in the lobby, classroom renovation in the D sector, a fire drill in the lobby and B sector, and a wireless access upgrade in the lobby and B sector. Blue shaded zones represent destination areas of each case. The inset figures show the path lengths for each trial, comparing our method, our method with ablation testing, move_base with and without obstacle layer clearing. The black box highlights where move_base failed, and the data within the yellow circle shows where the path intruded into restricted areas. In Case 4, only the path with *NavigationEventMonitor* avoids intruding into restricted areas.

environmental changes, or dynamic objects from being treated as permanent, which can hinder navigation.

The results in Fig. 4 and Table I clearly demonstrate that our system significantly outperforms the baseline move_base package in navigating large, complex environments. Across 37 trials, our system consistently avoided restricted areas and successfully reached the designated destinations, reducing the total path length by 58% compared to move_base with obstacle layer clearing between trials. Notably, in Fig. 4 (b), (c) and (d), where the doors to restricted areas are not fully closed, move_base entered restricted areas 25 times and failed 4 times across 74 runs in 37 trials, highlighting the clear advantage of our approach.

TABLE I
COMPARISON OF PATH LENGTHS (M) ACROSS DIFFERENT NAVIGATION CONFIGURATIONS FOR VARIOUS CASES

Navigation configuration	Case 1	Case 2	Case 3	Case 4
Our method	192.5	126.5	236.3	214.7
Our method w/o *NavigationEventMonitor*	221.2	135.4	264.3	223.3
Our method w/o *PassageCostEvaluator*	234.6	171.4	270.0	215.9
Move_base	347.3	202.2	398.5	267.5
Move_base w/o clearing	220.1	158.7	248.1	183.1

D. Ablation Experiment

To assess the impact of individual components in our system, we conducted an ablation study on two key modules: the *PassageCostEvaluator* and the *NavigationEventMonitor*. In the first experiment, we removed the *PassageCostEvaluator*, leaving only the *NavigationEventMonitor* to track events. This allowed the system to avoid specific areas without using prior experience, isolating the *PassageCostEvaluator*'s impact. The results showed that while the system avoided restricted areas, it frequently attempted to pass through infeasible passages, leading to longer detours. This underscored the *PassageCostEvaluator*'s role in optimizing navigation using historical data. In the second experiment, we excluded the *NavigationEventMonitor*, allowing the system to rely solely on past experience. Without the *NavigationEventMonitor*, the system struggled to avoid restricted areas. In cases with closed doors in restricted areas, the system eventually performed similarly to the full method after several trials. However, when doors remained open in restricted areas, the system violated restrictions, highlighting the *NavigationEventMonitor*'s critical role in ensuring compliance with external information during navigation. The ablation studies, detailed in Fig. 4 and Table I, demonstrate the essential contributions of both components to the system's overall performance.

E. Multi-floor Experiment

We tested our system's ability to plan path across multiple floors, focusing on avoiding obstacles like a broken elevator and stairs inaccessible to a wheeled robot. In this test, our system successfully navigated between floors, avoiding maintenance elevators and unreachable stairs while still finding the shortest path. The video on website demonstrates this capability, showcasing a clear advantage over move_base, which lacks multi-floor path planning and doesn't account for the robot's limitations.

REFERENCES

[1] S. H. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities," *IEEE Access*, 2024.

[2] D. Shah, B. Osiński, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on Robot Learning*. PMLR, 2023, pp. 492–504.

[3] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "Vlfm: Vision-language frontier maps for zero-shot semantic navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 42–48.

[4] D. Feng, C. Li, Y. Zhang, C. Yu, and S. Schwertfeger, "Osmag: Hierarchical semantic topometric area graph maps in the osm format for mobile robotics," in *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2023, pp. 1–7.

[5] J. Hou, Y. Yuan, and S. Schwertfeger, "Area graph: Generation of topological maps using the voronoi diagram," in *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019, pp. 509–515.

[6] F. Xie and S. Schwertfeger, "Robust lifelong indoor lidar localization using the area graph," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 531–538, 2023.