Journal of Industrial Information Integration 48 (2025) 100941

Contents lists available at ScienceDirect

Journal of Industrial Information Integration

journal homepage: www.elsevier.com/locate/jii

Full length article

From issues to routes: A cooperative costmap with lifelong learning for Multi-AMR navigation

Jiyeong Chae[1], Sanghoon Lee[1], Hyunkyo Seo, Kyung-Joon Park*

Department of Electrical Engineering and Computer Science, and Center for Physical AI, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, 42988, Republic of Korea

ARTICLE INFO

Keywords:
Path planning
Lifelong learning
Multi-AMR

ABSTRACT

In large-scale industrial environments where multi-AMR (Autonomous Mobile Robot) systems are deployed, the unpredictable occurrence of obstacles can significantly disrupt AMR navigation, hindering task execution. To overcome such disruptions, AMRs must frequently replan their routes in real time, often resulting in suboptimal trajectories. This paper proposes a multi-AMR path planning framework based on a Cooperative Costmap with Lifelong Learning, designed to enable efficient navigation even in environments where obstacle patterns are not known a priori. Inspired by issue-propagation models in social-network theory — which describe how public attention rises and fades over time within a network — the proposed approach models the temporal influence of encountered obstacles, allowing predictive path planning that adapts to changing obstacle patterns. The framework incorporates a lifelong learning mechanism to incrementally refine the influence parameter over time, thus ensuring adaptability in dynamic industrial settings. Simulation experiments demonstrate that the proposed approach increases task throughput by up to 18.0% and reduces average travel time by up to 30.1% compared to the standard ROS 2 navigation stack.

1. Introduction

In smart manufacturing architectures where production lines, warehouses, industrial monitoring systems, and cloud-based analytics are tightly integrated [1,2], automation is increasingly achieved by deploying diverse robots, including Autonomous Mobile Robots (AMRs), to enhance operational productivity [3,4]. These AMRs perform tasks while autonomously navigating their surroundings. Furthermore, fleets of AMRs are often coordinated within a multi-AMR system [5] in large-scale facilities where industrial monitoring is implemented [6,7]. However, obstacles such as forklifts and Automated Guided Vehicles (AGVs) may appear intermittently as part of transportation workflows, disrupting the movement of AMRs [8]. In particular, when these obstacles completely block passage along the planned aisle, the AMRs must detour via alternative routes, imposing potentially severe delays on the entire manufacturing workflow. In scenarios where system operators cannot accurately predict these obstacle patterns, AMRs must rely solely on sensor-based real-time path replanning to avoid collisions [9]. Such reactive approaches often lead to inefficient route adjustments, ultimately reducing the overall efficiency of the multi-AMR system.

To address the challenge of unpredictable obstacles, we propose a Cooperative Costmap with Lifelong Learning. This framework enables obstacle-occurrence information detected by one AMR to be shared across the multi-AMR system, influencing other AMRs' path planning. Inspired by issue-propagation models [10,11], we introduce an Obstacle Influence model — a time-decay function that determines how long a detected obstacle should continue to influence robot path planning — which activates when an AMR detects an unexpected obstacle that hinders navigation. In this model, the influence of an obstacle shared by the robots rises sharply when it is first detected and then gradually diminishes over time. Leveraging this principle, we mathematically capture how obstacles modulate navigation decisions in the multi-AMR system over time. The Obstacle Influence model is used to incorporate and later remove obstacle-occurrence information from each AMR's costmap. Under these conditions, the decay rate of the Obstacle Influence model directly affects how soon an AMR may re-enter a previously obstructed region. To adjust the decay rate adaptively, we introduce a lifelong learning mechanism based on historical obstacle patterns in each region. Here, obstacle patterns encompass not only where obstacles emerge but also how long they tend to persist in each aisle, thereby considering both spatial distribution and temporal duration. By learning whether obstacles tend to disappear quickly or persist

* Corresponding author.
E-mail addresses: cowldud3@dgist.ac.kr (J. Chae), leesh2913@dgist.ac.kr (S. Lee), hysun43@dgist.ac.kr (H. Seo), kjp@dgist.ac.kr (K.-J. Park).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.jii.2025.100941
Received 9 July 2025; Received in revised form 6 August 2025; Accepted 1 September 2025
Available online 10 September 2025

for a long time in specific regions, the system intelligently minimizes unnecessary collision risk and improves path efficiency.

This paper introduces the Cooperative Costmap with Lifelong Learning framework, designed to enable efficient global path planning in smart manufacturing facilities characterized by unpredictable obstacle patterns. By enabling obstacle-occurrence information sharing across AMRs, and dynamically adjusting the influence of the information over time, the proposed framework allows AMRs to continuously adapt to dynamic industrial scenarios in a lifelong learning manner. Notably, the proposed approach requires only a 2D LiDAR sensor, with no additional cameras or sensor-fusion pipeline, thus minimizing hardware requirements. It is also fully compatible with the standard ROS 2 navigation stack, integrating as a costmap layer plugin that requires no modifications to existing planners or controllers.

The remainder of this paper is organized as follows. Section 2 reviews related work on multi-AMR navigation. Section 3 details the proposed Cooperative Costmap with Lifelong Learning framework, building on the Obstacle Influence Model. Section 4 validates our framework in simulated environments that closely resemble real-world industrial settings. Finally, Section 5 concludes the paper and outlines directions for future work.

2. Related work

In large-scale industrial environments, it is essential to scale single robot capabilities to multi-AMR deployments [12–14]. Accordingly, several studies propose methods to optimize the paths of multiple robots while avoiding collisions with obstacles. The study in [15] proposes a Deep Q-network-based approach that prevents multiple robots from simultaneously choosing risky paths and learns policies that yield optimal navigation. The authors in [16] propose a reinforcement learning-based navigation approach that leverages a visual transformer and self-attention mechanism, enabling multiple robots to cooperate safely and achieve collision-free navigation. In [17], the authors introduce a novel imitation learning framework based on a modified Bayesian model to design distributed collision avoidance policies for multi-robot coordination. The work in [18] proposes a priority-based cooperative planner, in which robots avoid obstacles according to assigned priorities. However, most of these approaches either depend on computationally intensive machine-learning pipelines — such as reinforcement or deep learning methods [19,20] — or lack modular interfaces for seamless integration with the ROS 2 navigation stack.

In environments where unexpected obstacles may appear, several studies integrate custom layers into the costmap as plugins to extend the functionality of the ROS 2 navigation stack. In [21], the authors incorporate a YOLOv8-based image-segmentation layer into the local costmap, allowing the AMR to accurately distinguish traversable terrain from obstacles. However, this approach does not consider integration with multi-AMR systems. Other studies enable multi-AMR systems to adaptively select optimal paths in dynamic, real-time conditions. For instance, the work in [22,23] proposes cooperative perception mechanisms where robots share local occupancy information to avoid obstacles and generate global paths collaboratively. In [24], the authors leverage Time-to-Collision in a custom layer to quantify the severity of collision risks at various locations, thereby enhancing the safety and efficiency of trajectory planning. However, these methods rely solely on real-time sensing and do not fully exploit the historical obstacle patterns. The study in [25] introduces a custom layer to manage decentralized multi-robot traffic using a modified Dijkstra algorithm with directional filters, but it also lacks the ability to detect and adapt to changing obstacle patterns [26]. We introduce a Cooperative Costmap with Lifelong Learning that overcomes the limitations of previous approaches. Our framework features a lightweight learning mechanism that integrates seamlessly into the ROS 2 navigation stack and adapts continuously using both real-time sensor data and accumulated past observations, ensuring robustness to changing environments. The method is entirely self-supervised—each robot derives state labels from its own traversals, eliminating manual-labeling costs and reflecting the philosophy of self-supervised learning [27]. Furthermore, parameter updates rely on a simple multiplicative rule, keeping the computational load minimal and making the framework far more efficient than computation-heavy reinforcement-learning pipelines.

3. Approach

This section introduces the Cooperative Costmap with Lifelong Learning framework, which enables global path optimization in industrial environments with unpredictable obstacle patterns. The framework aggregates sensor data from individual AMRs into a cooperative costmap and leverages a lifelong learning mechanism to continuously adapt the costmap to dynamic obstacle patterns within a multi-AMR system.

3.1. Obstacle influence model

Several studies indicate that principles from social-network theory can illuminate robot communication and coordination. The study in [28] mentions that multi-robot teams are coordinated more efficiently when modeled as social networks instead of traditional robot-specific schemes. Under this paradigm, a multi-robot system is viewed as a social entity—a collection of loosely coupled, intelligent agents that continuously collaborate, structurally mirroring a human social network. Building on this perspective, in [29], the authors show — through theoretical analysis and physical multi-robot experiments — that state propagation in both social systems and in engineered robot networks follows the same underlying dynamics. Collectively, these findings strongly support the use of social-network models in the design of cooperative robotic systems.

Building on the preceding social-network perspective, we adopt the issue-propagation concept from social-network theory to model inter-robot information sharing in multi-robot systems. As described in [10], the issue-attention cycle characterizes how public attention in a social issue surges rapidly and then gradually diminishes. This pattern serves as a basis for modeling the temporal influence of an issue. Based on this concept, the study in [11] analyzes public responses to the 2019 measles outbreak in Austria using a lognormal distribution, demonstrating that the influence of the issue tends to rise following propagation and then decay gradually in a long-tailed, skewed fashion. The lognormal distribution is shown in Eq. (1), where the rate of decay — representing the spread of the distribution — is determined by the standard deviation in logarithmic space.

$$f(x;\,\mu,\sigma) = \frac{1}{x\,\sigma\sqrt{2\pi}}\,\exp\!\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right]. \tag{1}$$

We draw an analogy between the way an obstacle-occurrence issue spreads among AMRs and the propagation of issues in social networks. Instead of requiring each AMR to individually encounter obstacles and replan in real time, information detected by one AMR is shared with others—mirroring how a social issue propagates through a network. As demonstrated in [11], we also model the temporal influence of such information using a lognormal distribution, which effectively captures the initial increase followed by a gradual decay over time. This model is referred to as Obstacle Influence, and its output is normalized to the range $[0, 1]$ to represent the relative intensity of the influence. The value at a given time shows the extent to which obstacle detection influences the navigation decisions of other AMRs, with the shared information subsequently incorporated into their costmaps.

An ‘unexpected obstacle’ is any obstruction absent from the preloaded map that triggers real-time path replanning during AMR navigation [30]. In this study, we narrow the term ‘obstacle’ to refer to objects that completely halt an AMR’s movement along its global planner

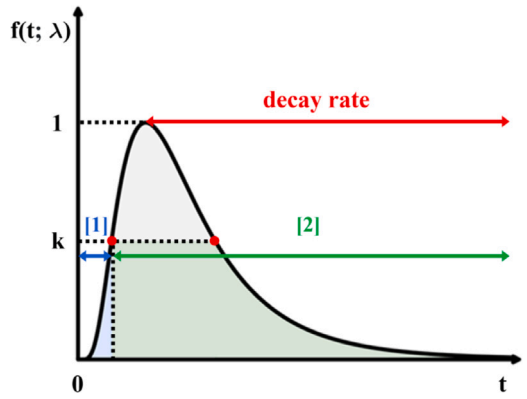

Fig. 1. Lognormal obstacle influence curve.

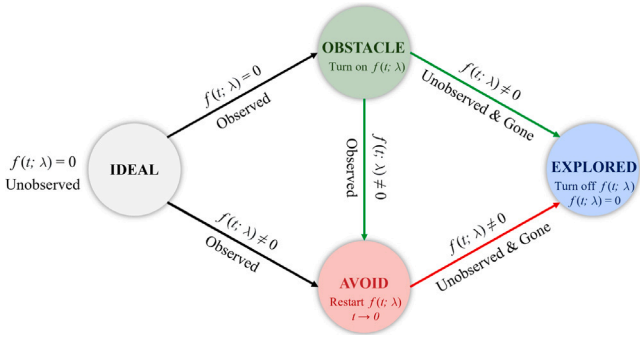

Fig. 2. Region state-transition diagram.

path, forcing the planner to regenerate an alternative detour route; such blockages can impose severe delays on the entire manufacturing workflow. Consequently, minor or only partially obstructing obstacles — ones that the local planner can readily handle while the AMR proceeds along its original global path — are excluded from the scope of this study. When an obstacle blocks the AMR's current global path and triggers replanning, the Obstacle Influence model is immediately activated for the affected region. Here, a 'region' is a spatial unit that may correspond to a single grid cell — the smallest grid element, a square whose side length equals the map resolution — or to any collection of adjacent cells, depending on how coarsely or finely the user discretizes the workspace. If the replanned detour path is markedly longer than the original global path because the AMR takes an alternate aisle, we infer the presence of a passage-blocking obstacle. Once an AMR encounters such an obstacle, its occurrence is propagated to the costmaps of all other AMRs through the Obstacle Influence model. In Fig. 1, Segment [1] represents the inherent system delays — including communication and processing latencies — from the moment an obstacle is detected to the point at which the information is incorporated into the costmaps of the other AMRs. After this delay, in Segment [2], the Obstacle Influence value rises to its peak at 1. To ensure that the peak occurs at a desired time t_0, the model parameters μ and σ are adjusted such that Eq. (2) holds.

$$\exp\left(\mu - \sigma^2\right) = t_0. \tag{2}$$

The resulting lognormal distribution, expressed using a single parameter — the decay rate λ — is presented in Eq. (3):

$$f(t;\lambda) = \frac{\sqrt{\lambda/\pi}}{t}\exp\left[-\lambda(\ln t)^2\right], \qquad t > 0,\ \lambda = \frac{1}{2\sigma^2} > 0. \tag{3}$$

The decay rate of the Obstacle Influence model is controlled by a parameter λ, which determines how quickly the influence diminishes over time. When the value of Obstacle Influence exceeds a predefined threshold, k, the corresponding obstacle-occurrence information is inserted into the costmap of each AMR, allowing the planner to classify the blocked aisle as impassable and preemptively select an alternative route. The threshold is empirically set to 0.5 by default, but it can be tuned depending on the industrial environment. Over time, as shown in Segment [2] of Fig. 1, the influence gradually decays. Once the influence value falls below the threshold, the obstacle-occurrence information is removed from the costmaps, allowing AMRs to re-enter the region. This mechanism is implemented through a custom costmap layer plugin that leverages Obstacle Influence values, as detailed in Section 3.3.

Within the Cooperative Costmap framework, the decay rate of the Obstacle Influence model dictates how soon an AMR can safely re-enter a region that was previously blocked. To predict unpredictable obstacle patterns effectively, we introduce a lifelong learning mechanism that adjusts the decay rate of each region in the Cooperative Costmap according to the historical obstacle observations accumulated by the AMRs. Specifically, in regions where obstacles tend to disappear quickly, the decay rate is increased, allowing the Obstacle Influence value to reach the threshold more rapidly. This leads to the timely removal of obstacle-occurrence information from the costmap, enabling faster re-entry and improved regional traversability. In contrast, in regions where obstacles tend to persist for longer periods, the decay rate is reduced, causing the influence value to decline more slowly. As a result, obstacle-occurrence information remains on the costmap longer, preventing premature re-entry and reducing the likelihood of repeated obstacle encounters.

3.2. Lifelong learning mechanism for obstacle influence

Fig. 2 outlines the workflow built around the four state types that an AMR assigns to each region it traverses and the corresponding actions. As these states are maintained on a per-region basis, the spatial distribution of obstacles is captured implicitly. While navigating, the AMR tags every region it encounters with one of the four labels; the chosen label can trigger an update to that region's Obstacle Influence value $f(t;\lambda)$, whose lognormal decay is designed to approach — but never actually reach — zero unless the reset is issued. After the mission, each region's decay rate is updated according to the state labels, thereby realizing the proposed lifelong-learning mechanism.

IDEAL indicates that $f(t;\lambda) = 0$ and no AMR has ever observed an unexpected obstacle in that region. **OBSTACLE** is assigned the first time an AMR detects an unexpected obstacle in a region where $f(t;\lambda) = 0$; at that moment $f(t;\lambda)$ becomes active for the region. The AMR immediately reroutes around the region. **AVOID** denotes that the obstacle is encountered again—either (i) when the same AMR again returns to a region already labeled OBSTACLE or AVOID, or (ii) when another AMR encounters a region whose residual influence satisfies $0 < f(t;\lambda) < k$. Here, the influence has decayed below the threshold k, so the region is traversable on the costmap, yet $f(t;\lambda) > 0$ means a past detection; the AMR immediately reroutes around the region and resets $f(t;\lambda)$ to start again from $t = 0$. **EXPLORED** is applied when that region currently labeled OBSTACLE or AVOID is traversed while $f(t;\lambda) > 0$ yet no obstacle is found; the state label switches to EXPLORED and $f(t;\lambda)$ is reset to zero to indicate that the obstacle has disappeared and that its influence has completely dissipated. These dynamic labels, with the state-transition diagram in Fig. 2, enable all AMRs to maintain a consistent, continually updated estimate of obstacle persistence throughout the workspace.

At the end of each navigation task, the system updates the decay rate λ for each region using the recent state observations of the AMR. The update is based on a comparison between the current travel time (T_{travel}) derived from the recorded state history and the predefined reference time (T_{ref}). The reference time is a fixed value based on the

environment, representing the travel time when taking an immediate detour instead of the shortest path; this value can be estimated offline from a pre-loaded map or empirically calibrated during actual operation. The ratio $\phi = T_{\text{ref}}/T_{\text{travel}}$ thereby incorporates the obstacle's temporal duration into the update, indicating how long the obstacle impeded the AMR relative to the detour. By comparing current travel times against this reference, the system infers obstacle persistence in each region and adjusts λ accordingly: λ remains constant for regions labeled IDEAL or OBSTACLE, indicating either no obstacle encounters or only a single detection.

State-based decay rate adjustment rules

$$\phi = \frac{T_{\text{ref}}}{T_{\text{travel}}}, \qquad \lambda \leftarrow \lambda\,\phi$$

- **AVOID**: $\phi < 1$ because T_{travel} is *always* greater than T_{ref} (slower decay)
- **EXPLORED**: $\begin{cases} \phi < 1 & \text{(slower decay)} \\ \phi > 1 & \text{(faster decay)} \end{cases}$

Decay-rate updating operates analogously to the paradigm of simultaneous state-and-parameter estimation in adaptive control [31]: after each mission, every region treats its current value of λ as a weight and rescales it multiplicatively according to the most recent observed state. Repeated application of this rule gradually aligns the decay rate of every region with its observed obstacle duration. These region-specific adjustments are then applied to the costmap of each AMR, allowing other AMRs to dynamically update or remove obstacle-related costs in their own costmaps.

3.3. Obstacle influence layer plugin for cooperative costmap

A costmap is a grid-based representation of obstacle and environmental cost information, which allows an AMR to perform path planning by minimizing the total accumulated cost during navigation. The costmap typically consists of two components: the global costmap, which covers the robot's entire workspace and derives its baseline costs from the pre-loaded static map, and the local costmap, which updates costs in real time using obstacle data detected by onboard sensors [32]. This study focuses on a 2D costmap, where each cell is assigned a cost value [33] ranging from 0 (free space) to 254 (lethal obstacle). A 2D costmap typically consists of three fundamental layers. The first is the static layer, which assigns costs based on preloaded static map information. The second is the obstacle layer, which assigns costs to obstacles detected by real-time sensor data. The third is the inflation layer, which inflates the area around lethal obstacles by assigning additional costs, thereby ensuring a safe buffer zone to prevent robot-obstacle collisions [34]. The open-source ROS 2 Nav2 stack, commonly used for the development of robot applications [35], does not provide built-in specialized functions to automatically handle dynamic obstacles [34]. To address dynamic obstacles within Nav2, the ROS 2 navigation stack allows for user-defined custom layers to be integrated as plugins, enabling flexible extensions of the costmap with additional semantic information tailored to specific field scenarios [36]. Consequently, although a custom costmap layer is added, the existing planner and controller in the ROS 2 navigation stack continue to operate seamlessly without any need for modification.

We implement the Obstacle Influence function, parameterized by the decay rate λ, via a custom costmap layer plugin called the Obstacle Influence Layer (as shown in Fig. 3). This layer captures obstacle-occurrence information and updates the cost values over time. As described in Section 3.2, when an AMR detects an unexpected obstacle during navigation, the corresponding region is immediately updated: the STATE label is changed to reflect obstacle-occurrence information, and the system evaluates whether to activate the Obstacle Influence model. After navigation, the appropriate decay rate λ is determined based on state observations of the AMR, enabling the system to adapt quickly to its environment by learning from experience [20]. The influence value computed using λ is then converted into a cost and applied to the Obstacle Influence Layer. This cost directly affects the AMR's global path planner, effectively discouraging it from traversing the obstructed region. The system estimates whether the AMR is likely to pass through a specific region via the aisle, based on its current position and goal position. This estimation is recorded in the `inAisle` flag and used to selectively update the Obstacle Influence Layer depending on whether the AMR is expected to traverse the region. For each region, if the Obstacle Influence value falls below a predefined threshold (as described in Section 3.1), the cost is set to 0, indicating that the region is traversable. If the value exceeds the threshold, the cost is set to 253, signaling that the region should be avoided. Through this plugin design, the Obstacle Influence Layer dynamically incorporates time-varying obstacle information into the costmap. Ultimately, this mechanism enables the construction of a Cooperative Costmap that continuously aggregates obstacle observations from all AMRs, thereby minimizing localized risks and enhancing overall path planning efficiency.

4. Experiments

We evaluate the effectiveness of the proposed Cooperative Costmap with Lifelong Learning framework in simulated environments that closely resemble real-world industrial settings. The constructed simulation environment spans a rectangular area of 25.9 m by 41.6 m, which serves as a simplified representation of warehouse delivery aisles (as shown in Fig. 4(a) and (b)). The simulation runs on an Ubuntu 22.04 environment with ROS 2 Humble. The experiments are conducted using multiple DeliveryRobot models, provided by GazeboSim [37]. To emulate a realistic logistics center, we utilize object models such as trashcans, buckets, and boxes provided by Amazon.com, Inc., along with lifts [38], shelves [39], pallets [40,41], and charging stations [42] from the GazeboSim library. Before running the navigation tests, a static map of the environment is generated using the Google Cartographer SLAM algorithm [43]. This standard preparatory step establishes a single, shared global coordinate frame, so that every subsequent pose estimate and costmap layer refers to the same physical locations [44]. In addition, RViz is employed to visualize the AMR trajectories and environment dynamics, enabling intuitive performance assessment of the proposed framework. The simulation environment parameters and obstacle characteristics influence the outcomes [45], and the specific settings used in Gazebo are summarized in Table 1.

The experiments deploy multiple AMRs into simulated industrial environments containing both long-lasting and short-lived obstacles — two representative persistence levels frequently encountered on warehouse floors — and compare the navigation performance of AMRs with the standard ROS 2 navigation stack. Lift models act as unexpected obstacles that halt AMR passage through the aisle and thereby delay operations; their arrivals follow a Poisson distribution with a minimum inter-arrival time of 1500 s, capturing realistic randomness and environmental variability. In the long-lasting obstacle scenario, lifts move at an average speed of 0.05 m/s, similar to a scissor-lift table that slowly traverses a rack during loading operations. In the short-lived obstacle scenario, they move at an average speed of 0.1667 m/s (Fig. 5(a)), mimicking a forklift that briefly crosses an aisle. DeliveryRobots, serving as AMRs, are equipped with 2D LiDAR sensors (1° angular resolution, 5 Hz scan rate) and navigate from their start points to goal points at an average speed of 0.5 m/s while performing transport tasks (Fig. 5(b) and (c)). The simulation models the specifications of real robot sensors as well as the constraints on acceleration and rotational speed to ensure physically consistent dynamics.

We focus on a smart-manufacturing architecture with integrated intralogistics, where unpredictable obstacles — such as forklifts or

Table 1
Gazebo simulation settings and obstacle characteristics.

Gravity (m/s^2)			Physics type	Dimensions of Gazebo environments (L × W m)	Average speed of long-lasting obstacles (m/s)	Average speed of short-lived obstacles (m/s)	Dimensions of obstacles (L × W × H m)
X	Y	Z					
0	0	−9.8	ode	25.9 × 41.6	0.05	0.1667	3.2 × 2.4 × 2.1

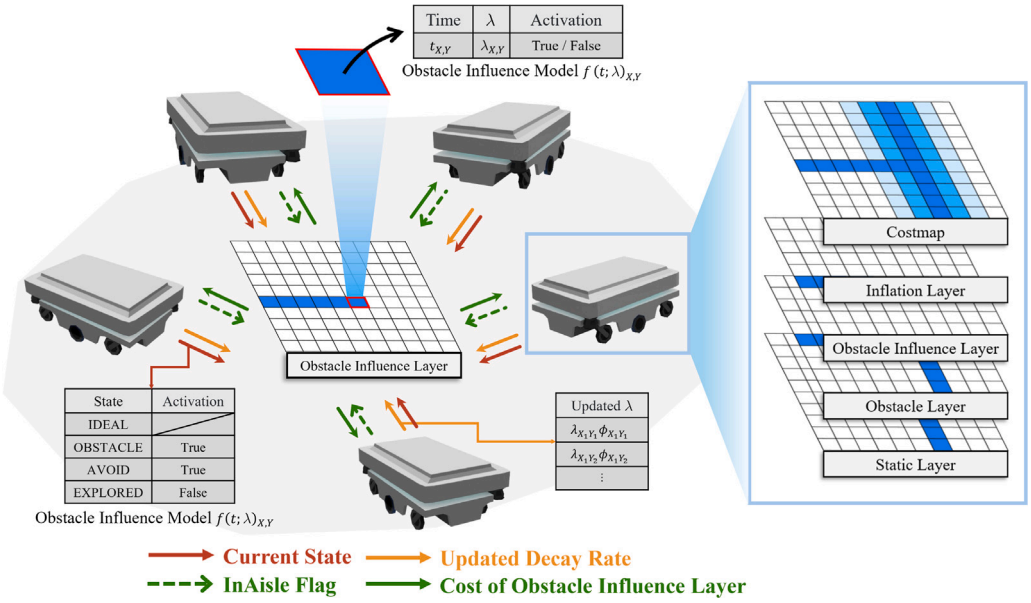

Fig. 3. Obstacle influence layer workflow.

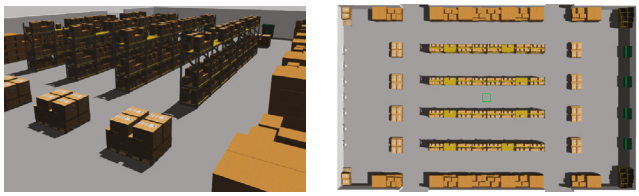

Fig. 4. Gazebo simulation environments (a) angled view; (b) top-down view.

AGVs — move through designated work zones for operational efficiency [46]. These movements create recurring obstacle patterns that the system can learn and adapt to, similar to a 'milk-run' distribution system [47]. Because these patterns tend to persist long enough for observations to accumulate, the framework can infer an environment-specific optimal decay rate by monitoring the obstacles encountered while AMRs perform their assigned tasks, even when human operators have no prior knowledge of the patterns.

4.1. Performance of lifelong learning mechanism

This section presents a 30,000 s simulation experiment conducted using a custom Python-based simulator developed to approximate Gazebo environment settings. This lightweight simulator lets us repeat trials rapidly and vary settings — down to very fine region granularity — in a controlled way, enabling efficient testing of the lifelong learning mechanism. The source code of this simulator is provided in Supplementary Materials A.

The purpose of this simulation is to assess the effects of the proposed lifelong learning mechanism under two distinct environmental conditions: one aisle with long-lasting obstacles and the other aisle with short-lived obstacles. Within this controlled setting, we evaluate three different decay rate configurations.

4.1.1. Three distinct decay rate configurations

Six AMRs are deployed to perform navigation tasks under three decay rate configurations: Transient, Permanent, and Adaptive. In the Transient configuration, the decay rate of the Obstacle Influence model is fixed at 1.0. Obstacle-occurrence information is added and then almost immediately removed from the costmap, so AMRs can traverse the aisle as soon as the obstacle is absent. In the Permanent configuration, the decay rate is set to a near-zero value (1×10^{-6}). Once an obstacle is recorded, its trace never fades, and the aisle remains blocked until the costmap is cleared manually. The Adaptive approach begins with a decay rate of 1.0 and gradually adjusts it over time through lifelong learning, dynamically determining how long obstacle-occurrence information remains on the costmap, based on each region's characteristics. Fig. 6 visualizes the learned decay rates of the Adaptive method in the simulator. Regions closer to red indicate decay rates near 0, while regions closer to blue indicate decay rates near 1.

Notably, the Transient and Permanent settings resemble common behaviors in the default ROS 2 Nav2 stack. In real-world deployments, when an AMR encounters an obstacle in environments such as narrow aisles with limited sensor visibility, the associated cost often remains in the costmap indefinitely—unless the region is explicitly revisited. This behavior is analogous to the Permanent case. To address this, users sometimes invoke the built-in `ClearEntireCostmap` service call, which instantly removes obstacle-related traces on the costmap, yielding behavior similar to the Transient case. In contrast, the proposed Adaptive approach offers a balanced, feedback-driven alternative. Rather than relying on static solutions, it learns to retain or remove obstacle-related traces over time based on accumulated environmental interactions.

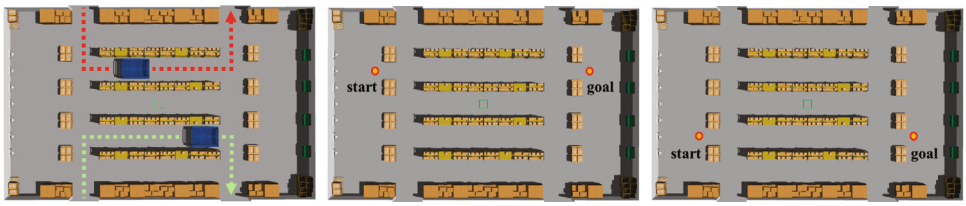

Fig. 5. Route details (a) lift-model paths through the long-lasting (red) and short-lived (green) obstacle aisles; (b) start/goal points for each DeliveryRobot in the long-lasting aisle; (c) start/goal points in the short-lived aisle.

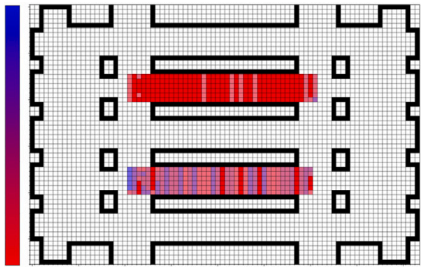

Fig. 6. Learned decay rate produced by the Python simulator.

4.1.2. Task throughput analysis

Fig. 7(a) illustrates how the total task throughput — the sum of tasks completed in the long-lasting obstacle aisle and the short-lived obstacle aisle, averaged over each 1800 s window — varies throughout the experiment, providing an intuitive view of how productivity changes. During the initial warm-up phase, the Adaptive method exhibits low task throughput due to insufficient data on obstacle patterns. However, the AMRs accumulate sufficient observations to allow the system to effectively learn the optimal decay rate of the Obstacle Influence model for each region. As a result, the Adaptive approach demonstrates consistently superior performance (green line in Fig. 7(a)). Notably, it also exhibits a narrower standard-deviation band — the shaded ribbon indicating ± 0.5 standard deviations around the sliding-window mean — than the Transient approach, indicating robust adaptability.

The Transient method, which quickly erases obstacle traces, frequently suffers performance fluctuations due to reactive detours and stalls caused by unexpected obstacle encounters—disruptions that become even more pronounced when AMRs confront long-lasting obstacles (red line in Fig. 7(b)). However, it can often select the shortest path when short-lived obstacles are present, yielding high average task throughput (red line in Fig. 7(c)). In contrast, the Permanent method retains obstacle traces almost indefinitely after a single observation. This results in overly conservative behavior, with AMRs persistently avoiding the corresponding regions — even after the obstacle has disappeared — leading to consistently low task throughput (blue lines in Fig. 7(b) and (c)). The Adaptive method continuously learns the obstacle patterns and dynamically adjusts the decay rate of the Obstacle Influence model. Consequently, the proposed Adaptive method minimizes direct encounters with obstacles and reduces unnecessary detours. This approach not only improves task throughput but also improves stability in dynamic operational scenarios.

4.1.3. Travel time analysis

Fig. 8(a) shows how the mean travel time — averaged over tasks completed in the long-lasting obstacle aisle and the short-lived obstacle aisle during each 1800 s window — varies over time. Because the same factors that increase productivity in Section 4.1.2 are at work here, the Adaptive method achieves the shortest travel time (green line in Fig. 8(a)). Furthermore, the preceding task throughput analysis offers an intuitive, high-level perspective, but the remainder of this section provides a more granular examination of how each decay rate configuration separately shapes travel-time performance under long-lasting and short-lived obstacle scenarios.

Fig. 8(b) shows the change in the average travel time in the long-lasting obstacle scenario. When an unexpected obstacle appears, the Transient approach — lacking obstacle-occurrence information on the costmap — causes the AMRs to reattempt the blocked path repeatedly until the obstacle disappears completely. This results in frequent replanning and delays — **replanning storms** — leading to wide standard-deviation bands and significant spikes in travel time during obstacle-active intervals (red line in Fig. 8(b)). In the Permanent configuration, previously detected obstacles are always treated as active because of their persistent presence in the costmap. Consequently, AMRs must consistently choose detour routes regardless of whether the obstacle has disappeared, resulting in unnecessarily long paths (blue line in Fig. 8(b)). As the obstacle-occurrence interval increases, this conservative behavior becomes increasingly inefficient.

Fig. 8(c) illustrates the change in the average travel time in the short-lived obstacle scenario. As obstacles appear only briefly and vanish quickly, replanning storms have little impact. However, this setting brings another issue to the forefront: **phantom detours**—unnecessary avoidance of regions where obstacles are no longer present. The Transient approach avoids phantom detours entirely, as it clears obstacle-related traces from the costmap almost immediately. As a result, it maintains favorable travel times on average, with only minor delays caused by a brief obstacle-occurrence (red line in Fig. 8(c)). In contrast, the Permanent approach persistently avoids any region where an obstacle has previously been detected, regardless of whether the obstacle still exists. While this strategy yields consistent travel times by avoiding stalls, it introduces phantom detours in environments leading to unnecessarily longer paths than those produced by the Transient approach (blue line in Fig. 8(c)).

After approximately 3000 s, the Adaptive approach begins to leverage accumulated memory, resulting in a sharp reduction in travel time (as shown in Fig. 8). Furthermore, during obstacle-free intervals, AMRs can consistently select the optimal path without phantom detours. When obstacles appear, they can avoid falling into replanning storms. The Transient method can be interpreted as having an extremely short memory, whereas the Permanent method retains an extremely long memory. Fixing the memory duration in this way may be beneficial for certain obstacle patterns but can lead to suboptimal navigation in others. In contrast, the proposed Adaptive method learns the appropriate memory duration based on actual AMR navigation experience, enabling it to make optimal decisions across different obstacle patterns.

4.1.4. Robustness to sudden obstacle pattern shifts

As a supplementary experiment presented in this section, we evaluate the robustness of the proposed framework by stress-testing its lifelong learning mechanism against abrupt obstacle pattern shifts. Even after the system has fully converged on the optimal decay rate for the initial obstacle pattern, we suddenly introduce a completely different pattern and observe whether the framework can rapidly re-adapt

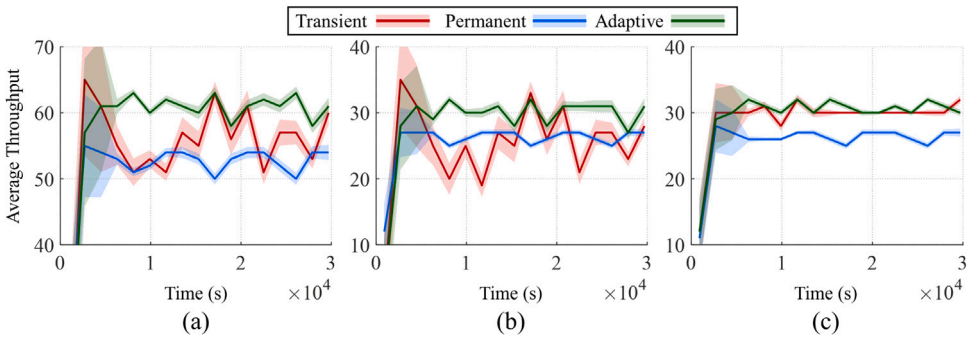

Fig. 7. Average task throughput for the three decay rate configurations (a) sum of averaged tasks completed in the long-lasting obstacle aisle and the short-lived obstacle aisle; (b) only long-lasting obstacle aisle; (c) only short-lived obstacle aisle.

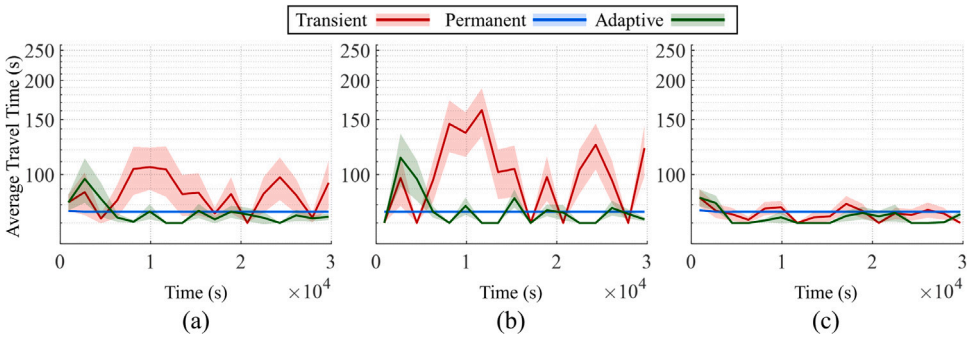

Fig. 8. Average travel time for the three decay rate configurations (a) mean travel time completed in the long-lasting obstacle aisle and the short-lived obstacle aisle; (b) only long-lasting obstacle aisle; (c) only short-lived obstacle aisle.

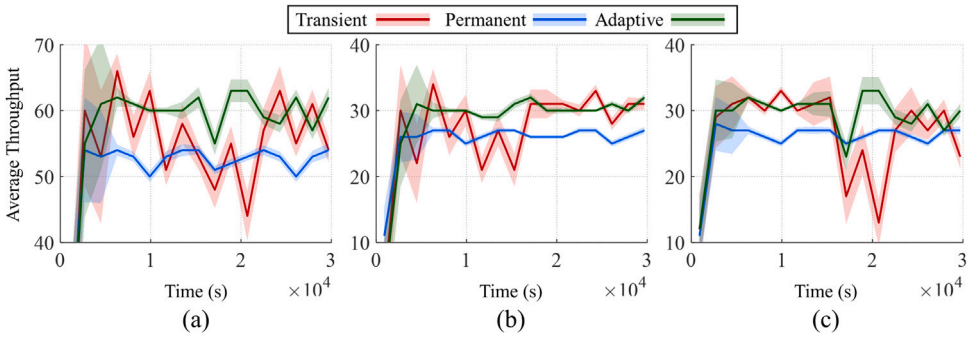

Fig. 9. Average task throughput under sudden obstacle pattern shifts for the three decay rate configurations (a) sum of averaged tasks completed in an environment that alternates between long-lasting and short-lived obstacles; (b) sudden transition from long-lasting to short-lived obstacles; (c) sudden transition from short-lived to long-lasting obstacles.

and maintain stable performance. Specifically, two types of sudden transitions are considered: a region that initially contains long-lasting obstacles and later transitions to short-lived obstacles, and the opposite case where short-lived obstacles transition to long-lasting obstacles. We simulate these two scenarios in which the obstacle patterns switch at a random point after 15,000 s, and examine how each method adapts to this transition. After the switch, we jointly analyze task throughput and travel time across the three decay rate configurations to provide a comprehensive assessment.

Fig. 9(a) illustrates how the total task throughput — the sum of tasks completed in both aisles, averaged over each 1800 s window — changes over time, providing insights into how productivity varies as the environment alternates between long-lasting and short-lived obstacles. Before the switch, all three methods perform as described in Section 4.1.2. After the pattern change, the Adaptive method experiences only a brief dip, quickly relearns an appropriate decay rate, and then returns to its earlier high performance (green line in Fig. 9(a)). The Transient approach improves task throughput when the obstacle pattern shifts from long-lasting to short-lived, but its throughput drops sharply when the shift is reversed—mirroring the behavior explained in Section 4.1.2 (red lines in Fig. 9(b) and (c)). In contrast, the Permanent approach remains consistently conservative, producing persistently low throughput in every scenario (blue lines in Fig. 9(b) and (c)). Because the Adaptive framework continuously updates the decay rate of the Obstacle Influence model, it ultimately sustains the highest productivity despite abrupt pattern changes. The next subsection analyzes travel time to explain in detail how each decay rate configuration performs under switches between long-lasting and short-lived obstacle regimes.

Fig. 10(a) shows how the mean travel time — averaged over tasks completed in an environment that alternates between long-lasting and short-lived obstacles during each 1800 s window — varies over time. Whenever the obstacle pattern switches between long-lasting and short-lived, the Adaptive approach responds in a manner consistent with the dynamics observed in task throughput: immediately after the pattern

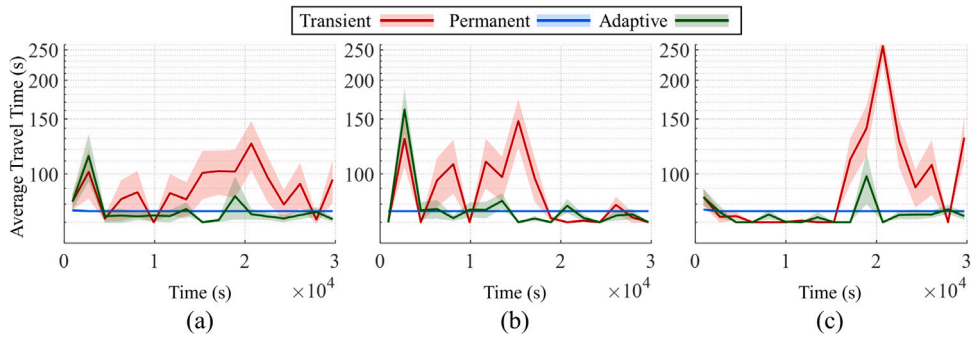

Fig. 10. Average travel time under sudden obstacle pattern shifts for the three decay rate configurations (a) mean travel time completed in an environment that alternates between long-lasting and short-lived obstacles; (b) sudden transition from long-lasting to short-lived obstacles; (c) sudden transition from short-lived to long-lasting obstacles.

shift, travel time spikes briefly, but as new data accumulate, the learned decay rate stabilizes and performance returns to a steady, favorable level (green line in Fig. 10(a)). Fig. 10(b) shows a case that starts with long-lasting obstacles and later switches to short-lived ones. During the initial phase, the Transient method suffers frequent stalls caused by prolonged blockages, producing wide standard-deviation bands and elevated travel time—exactly as observed in Section 4.1.3. After the pattern change, the short duration of the obstacles reduces stall frequency, and the Transient method's travel time settles below that of the Permanent method. By contrast, the Adaptive method experiences phantom detours due to residual avoidance behavior learned from the prior long-lasting obstacles. However, this does not trigger severe stalls or repeated replan cascades, and as a result, it avoids the extreme spikes seen in Fig. 10(c). This illustrates that the proposed lifelong learning mechanism can dynamically adapt — without significant time overhead — even when obstacle durations decrease abruptly.

Fig. 10(c) illustrates the other case that initially contains short-lived obstacles and later switches to long-lasting ones. During the early phase, the Permanent approach unnecessarily avoids areas where the obstacle has already disappeared, resulting in a higher average travel time than the other methods. After the pattern change, the Transient approach experiences a sharp rise in travel time and a significant widening of its standard-deviation band, due to stalls triggered by frequent replanning in response to long-lasting obstacles. In contrast, the Adaptive method initially also exhibits a spike in travel time due to replan cascades, but quickly stabilizes as it learns the changed obstacle patterns.

These results demonstrate that the Adaptive method is effective not only in responding to decreases in obstacle duration but also autonomously adjusting to increased obstacle persistence, dynamically refining its behavior over time based on learned obstacle patterns. Our approach is analogous to the mechanism in reinforcement learning, in which the policy is updated at the end of each episode based on the received reward. Rather than directly learning actions or motion policies, we retain the existing path-planning algorithm and instead learn to adjust a parameter of the Obstacle Influence model. This design eliminates the need to train or maintain large parameter sets typically required for complex behavioral policies. We incrementally refine a single parameter, the decay rate λ, which governs the temporal persistence of obstacles. This parameter is updated over time in response to task outcomes, effectively serving as a reward-driven adaptation mechanism. This approach allows the system to integrate the experiences of distributed and independent AMRs into a shared influence model, while avoiding the need for complex learning algorithms or extensive sensor data.

4.2. Gazebo-based simulation evaluation

In this section, we validate the performance of the proposed Cooperative Costmap with Lifelong Learning framework using the Gazebo simulator. The source code used to construct this simulator is provided in Supplementary Materials A. The objective is to evaluate the effectiveness of the learned region-specific decay rate, as established in Section 4.1. As illustrated in Fig. 11, obstacle-occurrence is reflected in the costmap. The Transient and Permanent methods approximate the behaviors observed in the standard ROS 2 navigation stack, as mentioned in Section 4.1.1.

As discussed in Section 4.1, we refer to the proposed framework as Adaptive. In the Gazebo simulation setup, a lift model — serving as an unexpected obstacle — moves along either a long-lasting obstacle aisle or a short-lived obstacle aisle. We evaluate the performance of Adaptive using the Gazebo simulator, comparing it against two fixed decay rate baselines: Transient and Permanent. For the long-lasting obstacle aisle, we compare Adaptive with Transient, since Adaptive has learned to apply a slow decay rate, whereas Transient uses a contrastingly fast decay rate. In this experiment, each AMR (DeliveryRobot) starts from a predefined location and navigates to its goal (Fig. 5(b)). As shown in the resulting navigation paths (Fig. 12(b) and (d)), Adaptive makes a decision to proactively detour around the obstacle when it is present for prolonged persistence. In contrast, Transient makes a decision that attempts to pass directly through the obstructed aisle and repeatedly retries the blocked path, as it lacks the ability to retain obstacle-occurrence information. In particular, Fig. 12(b) illustrates the momentary replanning behavior of the Transient method: whenever an obstacle is encountered, the AMRs replan and, shortly after, reattempt the blocked path multiple times until the obstacle completely disappears. This leads to a longer travel time and distance and ultimately to fewer completed tasks in the same amount of time. For the short-lived obstacle aisle, we compare Adaptive with Permanent, since Adaptive has learned to apply a fast decay rate, while Permanent uses a contrastingly slow decay rate. The DeliveryRobot starts from a start point and attempts to reach the goal point (Fig. 5(c)). The resulting paths (Fig. 12(f) and (h)) show that Adaptive makes a decision to intelligently enter aisles where obstacles were previously present but have since disappeared, based on learned patterns, thus reducing overall navigation time. In contrast, Permanent makes the AMR avoid the aisle unnecessarily, even after the obstacle has disappeared. This results in an increase in travel time and distance and, again, fewer tasks completed at the same time. Detailed AMR behaviors for each method are available in Supplementary Materials B.

4.3. Performance analysis

Table 2 presents a quantitative performance analysis of the three approaches — Transient, Permanent, and Adaptive — after the learning process stabilizes (that is, $t \geq 5000$ s). Route 0 is the start-to-goal path through the long-lasting obstacle aisle, while Route 1 is the start-to-goal path through the short-lived obstacle aisle. We define four categories based on the travel time for task completion: G1, which

Table 2
Performance of Transient, Permanent, and Adaptive strategies on Routes 0 and 1.

Model	Route	G1 (≤75 s)		G2 (=76 s)		G3 (77–140 s)		G4 (>140 s)		Journey time (s)		Tasks
		P (%)	T (s)	P (%)	T (s)	P (%)	T (s)	P (%)	T (s)	Mean	Std.	
Transient	0	86.48	70.00	0.00	–	0.00	–	13.52	332.73	105.52	91.76	355
	1	91.43	70.01	0.00	–	6.43	112.96	2.14	148.33	74.45	15.68	420
Permanent	0	0.00	–	100.00	76.00	0.00	–	0.00	–	76.00	0.00	366
	1	0.00	–	100.00	76.00	0.00	–	0.00	–	76.00	0.00	365
Adaptive	0	89.02	70.01	2.39	76.00	7.40	102.19	1.19	174.40	73.78	14.88	419
	1	95.33	70.00	0.23	76.00	3.50	99.13	0.93	162.50	71.90	11.03	428

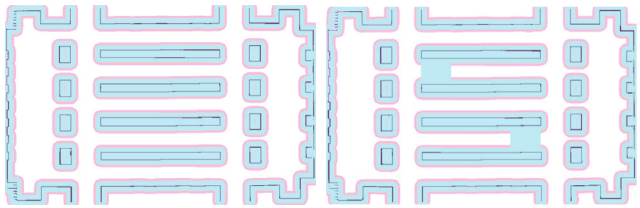

Fig. 11. (a) the default configuration of a costmap; (b) the proposed costmap version with the Obstacle Influence Layer added.

covers tasks completed within 75 s through a direct traversal via the shortest path; G2, for tasks completed in exactly 76 s by following a pre-planned detour; G3, encompassing tasks finished in 77–140 s due to partial abrupt detours and real-time replanning; and G4, which includes tasks requiring more than 140 s, reflecting significant delays caused by repeated avoidance or stalling. We denote the proportion of tasks completed within each time range as P and the corresponding average travel time as T. Across both routes, the Adaptive model demonstrates the highest task throughput and the lowest average travel time compared to other approaches. Specifically, it increases total tasks completed by up to 18.0% and reduces travel time by up to 30.1%. These gains stem from (i) the highest utilization of the shortest path (G1, averaging about 92%) and (ii) the suppression of stalling (G4) to approximately 1%. The detailed analysis is as follows.

In the long-lasting obstacle aisle, where obstacles persist and block the shortest path for extended durations:

- **Transient** fails to anticipate prolonged blockages, resulting in a G4 ratio of approximately 13.5%, an increased average travel time of approximately 105.5 s, and the completion of 355 tasks over 30,000 s.
- **Permanent**, by always choosing a detour (G2 = 100%), avoids stalls but cannot exploit available shortest-path opportunities, leading to an average travel time of 76.0 s and the completion of 366 tasks.
- **Adaptive** effectively learns obstacle patterns, achieving the highest G1 ratio (approximately 89.0%), selectively detouring (G2 + G3) only when necessary. It suppresses G4 to 1.2%, records the lowest average travel time of approximately 73.8 s, and achieves the highest task completion count of 419.

In the short-lived obstacle aisle, where obstacles disappear quickly and AMRs are relatively less prone to stalls:

- **Transient** shows reduced stalling (G4 ≈ 2.1%), significantly improves its average travel time to approximately 74.5 s, and completes 420 tasks.
- **Permanent** continues to always detour, maintaining a fixed average travel time of 76.0 s and completing 365 tasks.
- **Adaptive** increases G1 utilization to approximately 95.3%, suppresses G4 to approximately 0.9%, achieves the best average travel time of 71.9 s, and completes the highest number of 428 tasks.

By continuously updating the decay rate of the Obstacle Influence model in a lifelong learning manner, the Adaptive approach effectively balances risk and opportunity. Through a flexible combination of shortest-path traversal and selective detouring informed by learned obstacle patterns, it consistently maximizes task throughput and minimizes travel time in response to obstacle dynamics. Statistical testing confirms that these improvements are far from incidental. A Welch ANOVA across the three methods shows a highly significant effect on journey time for both routes (Route 0: Welch F = 26.87, $p < 0.001$; Route 1: Welch F = 9.34, $p < 0.001$).

These statistically robust gains translate into tangible productivity. Assume that operators always pick the best fixed strategy for each obstacle pattern—Permanent for long-lasting environments and Transient for short-lived environments. Even against this idealized mix and within the same 8-h shift using three AMRs, the proposed Adaptive strategy completed 61 additional missions, an increase of about 7.8%. That corresponds to roughly 20 extra tasks per AMR. This gain arises because the Adaptive planner flexibly combines shortest-path traversal with learned, selective detours, effectively suppressing congestion-induced delays. In real deployments, where obstacle patterns can change and the ideal strategy cannot be predetermined, the advantage of the Adaptive method becomes even more pronounced.

5. Conclusion

This paper proposes the Cooperative Costmap with Lifelong Learning framework to enhance multi-AMR navigation in dynamic industrial environments characterized by unpredictable obstacle patterns. Inspired by social-network theory, we model the temporal influence of obstacle-occurrence through the Obstacle Influence function, enabling navigation decisions without prior knowledge of obstacle patterns. The simulation results demonstrate that, relative to the behavior of the Transient and Permanent strategies — which replicate the default ROS 2 navigation stack — the proposed framework achieves up to 18.0% improvement in task throughput and a 30.1% reduction in average travel time, validating its practical benefits for multi-AMR systems.

The primary goal of this study is to evaluate how effectively the proposed framework keeps AMRs moving when an unexpected obstacle blocks their global path and would otherwise force a detour or stall. Collaborative information sharing is a powerful paradigm for boosting system performance [22,23,48], and we exploit this principle in mobile-robot navigation. Implemented as a lightweight plug-in on top of the standard ROS 2 Nav2 stack, the cooperative framework can be deployed immediately on any robot already running Nav2. All low-level collision-avoidance responsibilities remain with Nav2's Dynamic-Window-Based (DWB) local planner, so developers can add the Cooperative Costmap without altering proven safety components or worrying about inter-robot collisions. If an AMR itself becomes an 'unexpected obstacle,'

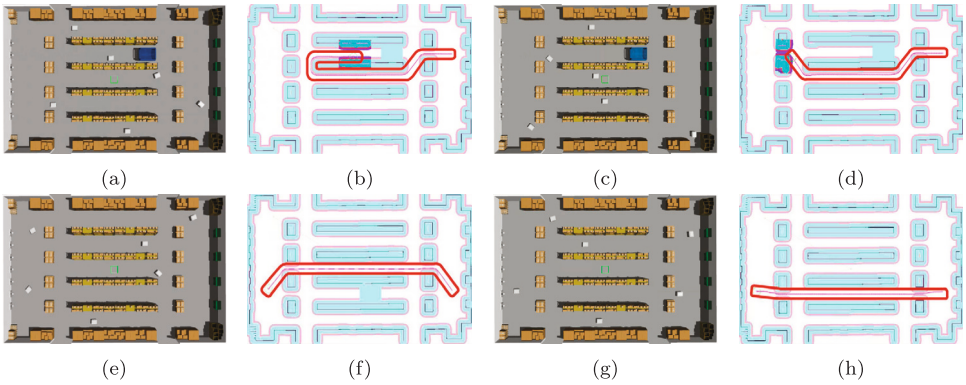

Fig. 12. AMR (DeliveryRobot) trajectory comparisons in Gazebo and RViz (a–b) Transient; (c–d) Adaptive; (e–f) Permanent; (g–h) Adaptive.

the framework simply treats it like any other obstacle, requiring no additional logic for robot–robot interactions.

As the framework shares obstacle information among AMRs, its advantages scale with fleet size—larger teams gain wider coverage and plan more efficient routes. Future work will therefore strengthen the Cooperative Costmap for high-density deployments by modeling and mitigating three practical limits: (i) physical congestion in narrow aisles, which can overload the local planner, (ii) network-level stress from frequent obstacle broadcasts, which may cause broadcast overhead and increase both latency and bandwidth usage [49], and (iii) instability of the learning process, which may arise in environments where obstacle patterns change frequently. Evaluating and optimizing these aspects in real warehouse testbeds will be the next step toward large-scale, production-ready adoption.

CRediT authorship contribution statement

Jiyeong Chae: Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization. **Sanghoon Lee:** Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Hyunkyo Seo:** Visualization, Validation, Software, Methodology. **Kyung-Joon Park:** Writing – review & editing, Resources, Project administration, Funding acquisition.

Funding

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP), South Korea grant funded by the Korea government (MSIT) (No. RS-2025-02219277, AI Star Fellowship Support)

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Kyung-Joon Park reports financial support was provided by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2025-02219277, Al Star Fellowship Support). Jiyeong Chae reports financial support was provided by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2025-02219277, Al Star Fellowship Support). Sanghoon Lee reports financial support was provided by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2025-02219277, Al Star Fellowship Support). Hyunkyo Seo reports financial support was provided by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2025-02219277, Al Star Fellowship Support). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary material

A.1. Project repositories

You can access the GitHub repositories related to this paper below:

- Gazebo Simulator for Cooperative Costmap
- Lifelong Learning of Obstacle Influence

A.2. Supporting video

You can access the video related to this paper below:

- Cooperative Costmap with Lifelong Learning Framework – Demonstration Video

Data availability

Data will be made available on request.

References

[1] H. Ali, R. Safdar, Y. Zhou, Y. Yao, L. Yao, Z. Zhang, W. Ding, F. Gao, A novel dynamic machine learning-based explainable fusion monitoring: application to industrial and chemical processes, Mach. Learn.: Sci. Technol. 6 (1) (2025) 1–38.
[2] H. Ali, R. Safdar, M.H. Rasool, H. Anjum, Y. Zhou, Y. Yao, L. Yao, F. Gao, Advance industrial monitoring of physio-chemical processes using novel integrated machine learning approach, J. Ind. Inf. Integr. 42 (2024) 1–22.
[3] P. Wei, X. Yu, Z. Di, X. Dai, B. Wang, Y. Zeng, Design of robot automatic navigation under computer intelligent algorithm and machine vision, J. Ind. Inf. Integr. 28 (2022) 1–11.
[4] R. Chand, B. Sharma, S.A. Kumar, Systematic review of mobile robots applications in smart cities with future directions, J. Ind. Inf. Integr. 45 (2025) 1–23.
[5] Z. Shen, Y. Duan, Y. Cong, W. Li, H. Du, W. Zhu, Deep reinforcement learning-based multi-AMR path planning algorithm, in: Chinese Control Conference, CCC, IEEE, 2023, pp. 1984–1987.
[6] H. Ali, R. Safdar, W. Ding, Y. Zhou, Y. Yao, L. Yao, F. Gao, Intelligent machine learning-based multi-model fusion monitoring: application to industrial physio-chemical systems, Control Eng. Pract. 162 (2025) 1–25.
[7] H. Ali, R. Safdar, Y. Zhou, Y. Yao, L. Yao, Z. Zhang, M.H. Rasool, F. Gao, Robust statistical industrial fault monitoring: A machine learning-based distributed CCA and low frequency control charts, Chem. Eng. Sci. 299 (2024) 1–24.

[8] A. Pandey, S. Pandey, D. Parhi, Mobile robot navigation and obstacle avoidance techniques: A review, Int. Robot. Autom. J. 2 (3) (2017) 1–12.
[9] W. Gao, D. Hsu, W.S. Lee, S. Shen, K. Subramanian, Intention-Net: Integrating planning and deep learning for goal-directed autonomous navigation, in: Conference on Robot Learning, CoRL, PMLR, 2017, pp. 185–194.
[10] A. Downs, Up and down with ecology-the "issue-attention cycle", Public Interes. 28 (1972) 38–51.
[11] F. Arendt, S. Scherr, Investigating an issue–attention–action cycle: A case study on the chronology of media attention, public attention, and actual vaccination behavior during the 2019 Measles outbreak in Austria, J. Heal. Commun. 24 (7–8) (2019) 654–662.
[12] L. Manuguerra, F. Cappelletti, M. Rossi, M. Germani, Eco-design tool to support the design of industrial electric vehicles. The case studies of an electric shuttle and an autonomous mobile robot, J. Ind. Inf. Integr. 39 (2024) 1–15.
[13] A.O. Agunloye, S.D. Ramchurn, M.D. Soorati, Learning to imitate spatial organization in multi-robot systems, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2024, pp. 3463–3469.
[14] Y. Chang, N. Hughes, A. Ray, L. Carlone, Hydra-Multi: Collaborative online construction of 3D scene graphs with multi-robot teams, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2023, pp. 10995–11002.
[15] Y. Dai, S. Yang, K. Lee, Sensing and navigation for multiple mobile robots based on Deep Q-Network, Remote. Sens. 15 (19) (2023) 4757.
[16] S.H. Arul, A.S. Bedi, D. Manocha, When, what, and with whom to communicate: Enhancing RL-based multi-robot navigation through selective communication, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2024, 7695–7695.
[17] L. Chen, Y. Wang, Z. Miao, M. Feng, Z. Zhou, H. Wang, D. Wang, Toward safe distributed multi-robot navigation coupled with variational Bayesian model, IEEE Trans. Autom. Sci. Eng. 21 (4) (2023) 7583–7598.
[18] Q. Wang, J. Li, L. Yang, Z. Yang, P. Li, G. Xia, Distributed multi-mobile robot path planning and obstacle avoidance based on ACO–DWA in unknown complex terrain, Electron. 11 (14) (2022) 2144.
[19] H.J.H. Saleh, J.R.T. Arrang, L.M.O. de Barros Cardoso, Applications of geospatial AI in human geography and spatial networks: A literature review, EDRAAK 2025 (2025) 76–84.
[20] H.A. MD, Meta-learning approaches for causal discovery in dynamic healthcare and robotics environments, Mesopotamian J. Artif. Intell. Heal. 2025 (2025) 136–153.
[21] E. Sani, A. Sgorbissa, S. Carpin, Improving the ROS 2 navigation stack with real-time local costmap updates for agricultural applications, in: IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2024, pp. 17701–17707.
[22] S. Jeong, T. Ga, I. Jeong, J. Oh, J. Choi, Layered-cost-map-based traffic management for multiple AMRs via a DDS, Appl. Sci. 12 (16) (2022) 8084.
[23] M. Graba, A. Kraiem, S. Kelouwani, A. Amamou, Multi-layered costmap-based navigation of heterogenous mobile robots for material handling application, in: Dans CIGI Qualita MOSIM, 2023, pp. 1–7.
[24] M. Graba, S. Kelouwani, L. Zeghmi, B. Allani, A. Amamou, K. Agbossou, M. Mohammadpour, A proposed adaptive navigation architecture to improve safety of self-guided vehicles in dynamic manufacturing environments, J. Intell. Manuf. (2025) 1–22.
[25] S.M. Langbak, C. Schou, K.D. Hansen, Multi-autonomous mobile robot traffic management based on layered costmaps and a modified Dijkstra's algorithm, Procedia Comput. Sci. 232 (2024) 53–63.
[26] K. Balasubramani, U.M. Natarajan, Improving bus passenger flow prediction using Bi-LSTM fusion model and SMO algorithm, Babylon. J. Artif. Intell. 2024 (2024) 73–82.
[27] M. Sallam, M.A. Shnan, Enhancing semantic image retrieval using self-supervised learning: A label-efficient approach, Babylon. J. Mach. Learn. 2025 (2025) 42–60.
[28] D. Rodic, A.P. Engelbrecht, Social networks as a coordination technique for multi-robot systems, in: Intelligent Systems Design and Applications, Springer, 2003, pp. 503–513.
[29] N. Horsevad, D. Mateo, R.E. Kooij, A. Barrat, R. Bouffanais, Transition from simple to complex contagion in collective decision-making, Nat. Commun. 13 (1) (2022) 1442.
[30] K. Honda, R. Yonetani, M. Nishimura, T. Kozuno, When to replan? an adaptive replanning strategy for autonomous navigation using deep reinforcement learning, in: International Conference on Robotics and Automation, ICRA, IEEE, 2024, pp. 6650–6656.
[31] M.G. Younis, Optimal control of dynamical systems using calculus of variations, Babylon. J. Math. 2023 (2023) 1–6.
[32] A. West, T. Wright, I. Tsitsimpelis, K. Groves, M.J. Joyce, B. Lennox, Real-time avoidance of ionising radiation using layered costmaps for mobile robots, Front. Robot. AI 9 (2022) 1–13.
[33] P.D.C. Cheng, M. Indri, F. Sibona, M. De Rose, G. Prato, Dynamic path planning of a mobile robot adopting a costmap layer approach in ROS2, in: International Conference on Emerging Technologies and Factory Automation, ETFA, IEEE, 2022, pp. 1–8.
[34] X. Han, Y. Leng, H. Luo, W. Zhou, A novel navigation scheme in dynamic environment using layered costmap, in: Chinese Control and Decision Conference, CCDC, IEEE, 2017, pp. 7123–7128.
[35] S. Macenski, F. Martín, R. White, J.G. Clavero, The marathon 2: A navigation system, in: International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 2718–2725.
[36] D.V. Lu, D. Hershberger, W.D. Smart, Layered costmaps for context-sensitive navigation, in: International Conference on Intelligent Robots and Systems, IROS, IEEE, 2014, pp. 709–715.
[37] Open-RMF, DeliveryRobot, 2024, URL https://fuel.gazebosim.org/1.0/Open-RMF/models/DeliveryRobot.
[38] OpenRobotics, Mecanum lift, 2023, URL https://fuel.gazebosim.org/1.0/OpenRobotics/models/Mecanumlift.
[39] MovAi, Shelf, 2022, URL https://fuel.gazebosim.org/1.0/MovAi/models/shelf.
[40] MovAi, Pallet, 2022, URL https://fuel.gazebosim.org/1.0/MovAi/models/pallet.
[41] MovAi, Pallet_box_mobile, 2022, URL https://fuel.gazebosim.org/1.0/MovAi/models/pallet_box_mobile.
[42] MovAi, Tugbot-charging-station, 2022, URL https://fuel.gazebosim.org/1.0/MovAi/models/Tugbot-charging-station.
[43] W. Hess, D. Kohler, H. Rapp, D. Andor, Real-time loop closure in 2D LIDAR SLAM, in: International Conference on Robotics and Automation, ICRA, IEEE, 2016, pp. 1271–1278.
[44] J. Chae, H. Seo, S. Lee, Y. Park, H.-S. Park, K.-J. Park, PINMAP: A cost-efficient algorithm for glass detection and mapping using low-cost 2-D LiDAR, IEEE Trans. Instrum. Meas. 74 (2025) 1–14.
[45] V. Govindan, J. Jayaprakash, C. Park, J.R. Lee, I.N. Cangul, Optimization-based design and control of dynamic systems, Babylon. J. Math. 2023 (2023) 30–35.
[46] S. Moon, S. Lee, K.-J. Park, Learning-enabled flexible job-shop scheduling for scalable smart manufacturing, J. Manuf. Syst. 77 (2024) 356–367.
[47] H.S. Kilic, M.B. Durmusoglu, M. Baskak, Classification and modeling for in-plant milk-run distribution systems, Int. J. Adv. Manuf. Technol. 62 (2012) 1135–1146.
[48] S.D. Shamsi, A.S.H.M. Ali, W.D. Shamsi, Hybrid Cooperative Spectrum Structured (HCSS) approach for adaptive routing in cognitive radio Ad Hoc networks, Mesopotamian J. Cybersecur. 5 (1) (2025) 23–38.
[49] F.A. Al-Ibraheemi, M.T. Tally, S. Nadweh, I.A. Al Sayed, J.F. Tawfeq, H.M. Gheni, P. Shah, R. Sekhar, A cognitive energy-driven routing strategy for ultra-efficient data transfer in wireless sensor networks, Appl. Data Sci. Anal. 2025 (2025) 131–143.