A Novel Latent-Representation-Based Algorithm with Dynamic Obstacle Avoidance (LADy) for Autonomous Mobile Robots (AMRs)

Harishma Prakash[1], Prasina A[2], Samuthira Pandi V[3], Naregalkar AkshayKumar Rangnath[4], Rajalingam A[5], Sundar R[6]
Department of Mechatronics Engineering, Chennai Institute of Technology, Chennai, India[1]
Department of Electronics and Communication Engineering, Chennai Institute of Technology, Chennai, India[2]
Centre for Advanced Wireless Integrated Technology, Chennai Institute of Technology, Chennai, India[3]
Department of Electronics and Instrumentation Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India[4]
Department of Engineering and Technology, University of Technology and Applied Sciences, Shinas, Sultanate of Oman[5]
Department of Marine Engineering, AMET Deemed to be University, Chennai, India[6]

***Abstract*—Conventional path-planning algorithms are often tailored to industrial and warehouse settings, creating the necessity to integrate two or more planners, which requires more memory and computation limitations. To overcome this limitation, this study aims to develop a novel algorithm, semantic cost encoding-based A* with Dynamic obstacle avoidance, specifically designed for Autonomous Mobile Robots (AMRs) in warehouses. In this study, the proposed algorithm is benchmarked in Matplotlib against A* and RRT* showing 60.33% higher memory efficiency and 60.36% more efficiency in aspect of number of computed nodes than RRT*, while all equivalent to A* in a static environment, and benchmarked against D* lite for dynamic environment, as well as against a hybrid algorithm, a simplified interpretation of commercial AMR path planning approaches, to show 45.30% more memory-efficiency than the hybrid algorithm, which is much more preferred in a real time implementation than D* lite, for AMRs.**

Keywords*—Path planning; semantic encoding; Autonomous Mobile Robots (AMRs); warehouse navigation; dynamic environment

I. Introduction

The automation era has pushed many processes toward automatic functioning, such that warehouses are employing Autonomous Mobile Robots (AMRs) to perform their load-carrying functions. An important feature in the automation of AMRs is motion planning. Recent trends have evolved motion planning with the evolution of technology. Infusing AI technologies into motion planning approaches. For instance, reinforcement learning is implemented in motion planning for task-specific designs for efficient autonomous driving [28], and for enhanced optimality, it can reduce training complexity and improve sample utilization [29]. Further, deep reinforcement learning generates efficient autonomous motion planning, but implementation in real-world applications poses challenges [30]. For increased efficiency in production aspects, human-robot collaboration effectively improves flexibility and maintainability [31]. Though many modular designs are improved for motion planning approaches, for a warehouse's efficient operation with AMR-human collaboration that increases flexibility and maintainability requires resource allocation ratios, environment awareness for maximum warehouse efficiency [32]. Similar to humans' awareness of the complex working environment for their optimal performance, in the proposed algorithm, the environment awareness is induced through semantic encoding representations of the known environment map. The planner has consistent integration of semantic cost modeling across both global planning and dynamic replanning stages within a unified framework. Unlike classical A*, which is a static global planner, the proposed approach encodes a semantically weighted A*-based formulation within a dynamic framework to enable continuous adaptation in dynamic environments. In contrast to conventional hybrid approaches, where A* generates waypoints and D* Lite performs local tracking, the proposed method employs a unified architecture. The semantic cost encoding is consistently maintained within the replanning process, ensuring that dynamic obstacle avoidance is aligned with global contextual objectives. The algorithm, LADy, originally derived from a latent representation-based formulation, is now represented as a semantic cost-based dynamic planning framework.

This research contributes:

- Proposes a novel hybrid planning approach that integrates semantic cost encoded A* with dynamic planning functionalities, forming a unified framework.
- Developed an environment-aware path planning algorithm capable of operating in both static and dynamic environments.
- Introduced a semantic encoding mechanism to enhance the planner's understanding of context of environment.
- Provided comprehensive benchmarking of the proposed method against existing algorithms, A*, RRT*, D* Lite, and hybrid planning frameworks.
- Presented equation-level modifications to A*, contributing formal improvements tuned to proposed planner.

- A unified architecture employing context-aware semantic modelling in both global and dynamic planning phase.

The rest of the study is organized as follows: In Section II, related works are presented. In Section III, the planner methodology is proposed. Section IV covers the pseudocode of the proposed novel algorithm. Results of the study is presented in Section V. Section VI shows the metrics comparison according to the benchmarking data. Section VII discusses the results. Section VIII details complexity analysis of the proposed algorithm. Finally, Section IX concludes and proposes the future scope of the proposed algorithm.

II. Related Works

For the automation of mobility, an important function is the path planning, which enables a vehicle to navigate independently towards a goal. Motion planning plays an essential role in this process [1]. Motion planning algorithms are developed based on scenarios, static, dynamic, known, and unknown environments.

Not all algorithms satisfy the criteria and requirements; the algorithms require development on many approaches for a dynamic environment [2]. Similarly, the development requirements and standards increased with the need for advanced navigation, like in unknown environments [3]. For the effective operation of AMRs in dynamic environments for industrial applications [4]. For diverse robotic applications, path planning algorithms need to adapt to dynamic environments [5].

Autonomous driving solutions offer advantages over modular technologies and systems, but face challenges [6]. They require clarification on concerns and queries when a system is automated, and they need assurance on safety [7]. For an automated navigation system, path planning is crucial for safety, which is not a simple algorithm of conventional methods, but a hybrid to make the system optimal [8]. The evolution of path planning algorithms began with grid-based and advanced to sampling-based algorithms like Rapidly-exploring Random Tree * (RRT*) algorithm. Grid-based path planning needed improvements to support larger application spaces efficiently [9]. Across various robotic applications, sampling-based path planning methods offer more efficient solutions [10]. Grid-based planning solutions consume more computation and memory while optimized RRT* are more efficient in that case [11].

Though they are more efficient all developments have its own challenges while facing demands, such that RRT* faced important challenges that needed more research and development [12]. Path planning for robots requires algorithms that generate feasible and smooth paths, avoiding obstacles efficiently [13]. To develop path planning algorithms, we need to understand both conventional and modern approaches for current needs [14]. Although sampling-based approaches are efficient, they require a more systematic comparison for clarity in algorithm selection for specific applications [15]. Robot navigation is a sophisticated process that requires ensuring safety and optimality. For navigation, efficiency can be improved through many approaches. One such method is the fusion of global and local planning [16]. Though there are many approaches and methods to improve each algorithm from its fundamental to optimal and hybrid, the selection of an algorithm is specific to the functions of the robot. Various options in path planning algorithms make it challenging to confine to one to implement in a system according to requirements [17]. There are many robotics applications and various needs accordingly. For instance, multi-UAV delivery systems need collaborative planning frameworks to optimize for urban logistics [18].

In the case of AMR systems,

- Warehouse AMR dispatching needs multi-objective optimization to minimize costs and balance workloads efficiently [19].
- For smart warehousing, there is a need for an integrated system of robotic systems to IoT [20].
- For multi-AMR systems, we need traffic management to reduce conflicts and improve performance [21].

For multi-objective systems, conventional path planning methods are not sufficient. Such needs of path planning can be accomplished with a combination of grid-based algorithm like A* and greedy algorithms [22]. Similarly, many alterations and hybridizations of algorithms are needed to meet the criterion.

For modern trends, in a warehouse automation system, there are many new applications and requirements that need improved models with mathematical optimization for operation [23]. In automated vehicles, obstacle avoidance capabilities are supposed to be advanced to achieve true autonomous operations [24]. According to the environments and requirements, the autonomous navigation enabling algorithm should work in environments irrespective of the state of the environment. The developed algorithms should be modular so that it is capable of planning in open and complex environments. Especially, in a warehouse automation, AMRs should be capable of safe and optimal navigation in narrow aisles [25] and complex indoor environments [26]. It is a significant need that the development of algorithms for automated vehicles should balance all aspects like safety, energy optimization, and planning optimality in complex environments [27].

III. Proposed Methodology

A. Overview

The Semantic Cost-Based A* algorithm with Dynamic obstacle avoidance is a hybrid algorithm implemented by fusing two path-planning algorithms, both with different applications conventionally. The developed algorithm works efficiently for operating AMRs in warehouses. The algorithm takes in known and defined data, modifies A* functioning with the known data and applies it as base path to a D* lite inspired framework.

B. Inputs and Known Data

The algorithm proposed needs certain pre-defined and known data before it works as inputs. The general and default defined data are the start and goal points that can be

manipulated as per need. Then this algorithm works only in a completely known environment, but it can handle both static and dynamic obstacles. For a completely known environment along with the environment map, the algorithm should be defined with semantic encoding representations of the map based on user-defined environment preferences and previous AMR path histories, and current knowledge of the environment. Based on these semantic encoding representations a cost is assigned that modifies conventional A* in this algorithm.

C. Algorithmic Logic

The D* lite algorithm is inspired for replanning phase that works efficiently in a dynamic environment avoiding moving obstacles on its way. Semantic encoded costs in A* is integrated in a dynamic framework to enable context-aware navigation. The algorithm is split into three main phases that contribute to the efficiency of the algorithm:

- Semantic encoding defined map
- Semantic encoding cost modified A*
- Modified A* applied dynamic planner

These three phases together form the developed hybrid algorithm specifically designed for efficiency of the algorithm in commercial applications, unlike other papers published path planning algorithms.

D. Semantic Encoding Defined Map

The known map is provided as input for the developed path planning algorithm along with the start and goal points for the objective. The algorithm has the dynamic framework as base of the functioning which has A* layout to form the initial path with considerations of only the static obstacles as per the map. In the algorithm, the cost assignment of A* base layout that develops the initial path is modified.

In A* algorithm, the developed path is grid-based, that is the algorithm deploys nodes in the map throughout at first. Similarly, the algorithm deploys nodes throughout the map using a grid-based technique.

Now, at this point in the A* algorithm case, it assigns the actual cost $g(n)$ to the nodes on its path, unlike in this algorithm's case, before that actual cost assignment semantic costs to all the nodes on the map are assigned. Based on the semantic encoding representations defined by the user from data of previous AMR's path histories and awareness of the environment.

Each node in the map of the environment is assigned costs based on semantic encodings, primarily divided into:

- Traffic Area Ratings (t), given in Table I – denotes the crowd density.
- Task Busy Ratings (T), given in Table II– denotes the task activity level in that region that delays mobility.

For instance, the ratings that influence the path finding in the simulation performed for benchmarking in this study are given below,

TABLE. I. TRAFFIC RATING OF THE MAP GIVEN IN FIG. 2

Area type	*Traffic Rating(t)*	*Remarks*
Receiving aisle	7	Highest congestion
Packaging aisle	6	
Shipping aisle	6	
Maintenance area aisle	6	
Main storage aisle	5	Moderate congestion
Charging station aisle	5	
Equipment area aisle	4	
Loading docks	4	
High density storage	4	
Cold storage aisle	3	
Hazard area	3	
Quality Control area	2	
Office aisle	1	Lowest congestion
Default open areas	0.5	Free movement zone

TABLE. II. TASK BUSY RATING OF THE MAP GIVEN IN FIG. 2

Area type	*Task busy Rating(t)*	*Remarks*
Maintenance area aisle	8	Highest activity
Equipment area aisle	7	
High density storage	7	
Receiving aisle	7	
Shipping aisle	6	
Main storage aisle	5	Moderate activity
Packaging aisle	5	
Cold storage aisle	5	
Charging station area	5	
Quality Control area	5	
Loading docks	3	
Hazard area	2	
Office aisle	0	Lowest activity
Default open areas	0.5	No major task zone

E. Semantic Encoding Cost Modified A*

For a typical A* algorithm the cost function assigned to each node is influenced by two factors, actual cost $g(n)$ and estimated heuristic cost, $h(n)$ [see Eq. (1)]:

$$f(n) = g(n) + h(n) \quad (1)$$

For the algorithm, the above cost function is modified with the influence of the semantic cost, as below [see Eq. (2) and Eq. (3)]:

$$f(n) = d(n) + h(n) \quad (2)$$

where, $d(n)$ is the modified cost,

$$d(n) = g(n) + l(n) \quad (3)$$

Here, the modified cost is the summation of the actual cost $g(n)$ like in A* and the semantic cost, $l(n)$.

$$l(n) = \alpha \cdot t(n) + \beta \cdot T(n) \quad (4)$$

where,

α : Traffic weight parameter,

β : Task complexity weight parameter,

$t(n)$: normalized (0 – 1) traffic rating for node n,

$T(n)$: normalized (0 – 1) task complexity rating for node n,

implying the weight parameters for their significance in the environment with respect to the distance that is considered for cost evaluation, and the ratings are normalized to restrict dominance of influence in path finding. Eq. (2) is the cost function applied to the nodes of the path that is planned by the proposed algorithm, while the semantic costs, Eq. (4), are calculated for all nodes that cover free spaces in the map.

F. Modified A in D* lite Inspired Framework*

The modified A* cost function is used to assign costs to each node and find a path. The curated initial path is fed to the D* lite-like framework.

The dynamic replanning is implemented using localized A*-based replans, conceptually inspired by D* Lite. The local replanning plans the new path from the current position to the goal position. The complete flow of the algorithm's working is given in Fig. 1. Algorithm 1 details the proposed LADy algorithm.

G. Flowchart

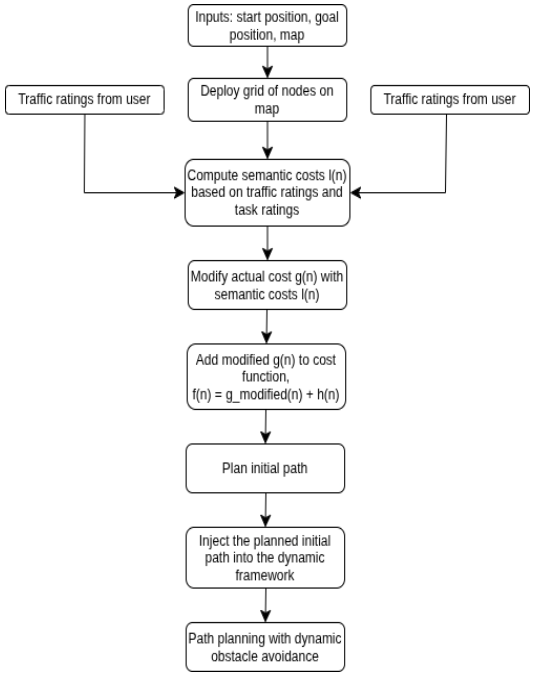

Fig. 1. Flowchart of the LADy algorithm working.

IV. PSEUDOCODE

- Input: warehouse_map, start_position, goal_position, alpha, beta
- Output: optimal_path, execution_complete

Key Functions:

- CalculateTrafficRating(node):
- Returns traffic congestion level for warehouse area
- CalculateTaskRating(node):
- Returns task complexity level for warehouse area
- CalculateSemanticCost(node):
- Combines traffic and task ratings with weight parameters
- ReplanFromCurrentPosition():
- Finds new path when obstacles block current route
- RevertToOriginalPath():
- Returns to initial optimal path when obstacles clear

Algorithm 1: Proposed LADy Algorithm

```
// PHASE 1: SEMANTIC ENCODING
for each free node in warehouse_map do
    traffic_rating ← CalculateTrafficRating(node) / 10.0 × 0.3
    task_rating ← CalculateTaskRating(node) / 10.0 × 0.3
    store traffic_rating and task_rating for node
end for

// PHASE 2: MODIFIED A* PLANNING
initialize all nodes with infinite cost and no parent
set start_position cost to zero
add semantic cost to start_position
create priority queue with start_position
while priority queue is not empty do
    current ← extract minimum cost node from queue
    if current equals goal_position then
        global_path ← reconstruct path using parent pointers
        break from loop
    end if
    for each valid neighbour of current do
        calculate base movement cost to neighbour
        add semantic cost for neighbour
        if total cost is better than existing cost then
            update neighbour cost and parent
            add neighbour to priority queue
        end if
    end for
end while

// PHASE 3: PATH INJECTION INTO DYNAMIC FRAMEWORK
initialize data structures for all nodes
path_length ← length of global_path
for each node in global_path do
    distance_to_goal ← calculate position from goal
    inject distance values into dynamic framework
end for
```

```
// PHASE 4: DYNAMIC EXECUTION
current_position ← start_position
current_path ← copy of global_path
while current_position not equals goal_position do
    |       detect new obstacles in environment
    |       if current path is blocked by obstacles then
    |       |       current_path ← replan from current position
    |       end if
    |       if original path is now better than current path then
    |       |       current_path ← revert to original optimal path
    |       end if
    |       next_position ← get next move from current path
    |       move robot to next_position
    |       current_position ← next_position
end while
return global_path, execution_complete
```

V. Results

For simulation and benchmarking against other renowned path planning algorithms, we consider factors like:

- Path length
- Time
- Memory
- Nodes
- Replans

Based on these factors, the proposed algorithm is benchmarked against three categories for an environment,

- Path planning algorithms for static environments: A*, RRT*
- Path planning algorithms for dynamic environments: D* lite
- Path planning algorithms used in current commercial AMRs: a simplified interpretation of commercial AMR path planning approaches

For the warehouse environment setup, a complex warehouse visualization was generated in the Matplotlib visualizer. Its description is given in Table III.

TABLE. III. Description of the Given Map in Fig. 2

Zone	*Code*	*Color*	*Description*
Perimeter walls	#2C3E50		Dark blue-gray
Storage racks	#8B4513		Saddle brown
Loading docks	#4682B4		Steel blue
Shipping area	#20B2AA		Light sea green
Receiving area	#32CD32		Lime green
Office complex	#FFD700		Gold
Break room	#FF69B4		Hot pink
Equipment storage	#FF8C00		Dark orange
Maintenance area	#DC143C		Crimson
Quality control	#9370DB		Medium purple
Packaging area	#00CED1		Dark turquoise
Fire exits	#FF0000		Red
Conveyor systems	#A0A0A0		Gray
Charging stations	#00FF00		Lime
Free space	#FFFFFF		White

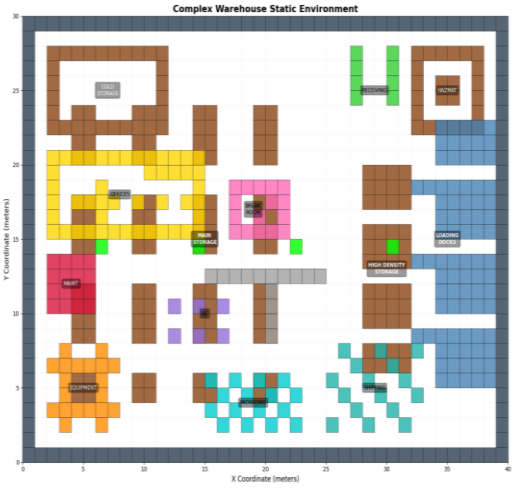

Fig. 2. Complex warehouse static setup for simulation.

A. Simulation in a Static Environment

For simulation in a static phase of the given warehouse environment, in Fig. 2, it is intended to benchmark against renowned algorithms for static environments, A* and RRT*. The study performed rigorous testing by simulating in Matplotlib, assigning a combination of 25 sets of start and goal points, and iterating the testing of each set 3 times to curate the average for a particular start and goal point.

*1) Benchmarking against A**: For the A* algorithm, whose path-finding process is grid-based, it deploys a grid of nodes on the map throughout and computes nodes that are explored as the path [33]. With simulation and testing of many input start and goal points set the A* algorithm's performance in the given environment in static state is analysed. For a glance at all testing results for A*, see the Data Availability section.

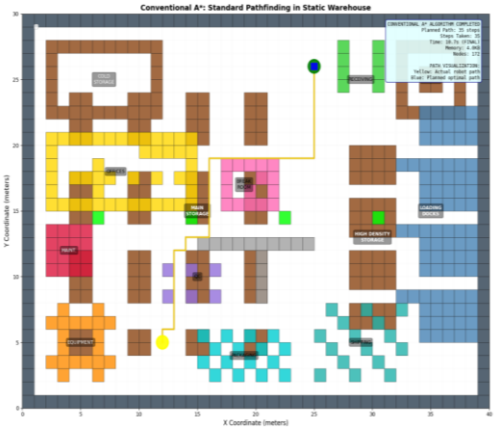

Fig. 3. A* simulation in the static state of the given map.

*2) Benchmarking against RRT**: For RRT* algorithm whose node distribution for path finding is random based on probability, deploys nodes randomly that is conditioned to reach the goal with less number of node distribution [34]. Similar to A*, the RRT* is also tested for its performance with input start and goal points sets and analyzed (see Fig. 4).

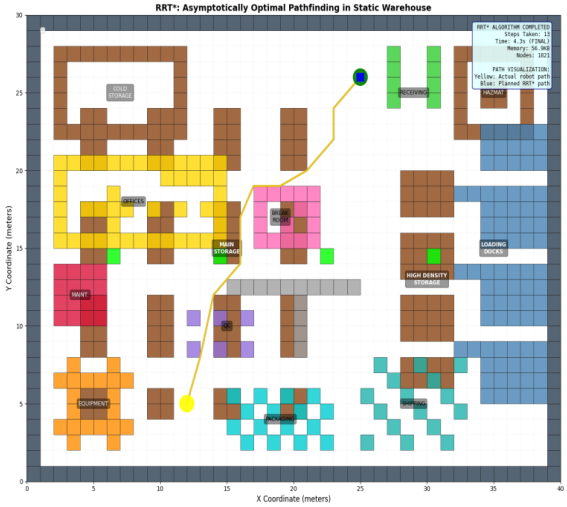

Fig. 4. RRT* simulation in the static state of the given map.

For a glance at all testing results for RRT*, see the Data Availability section.

In the case of performance metrics of A*, in Fig. 3, there are 4 details provided for the final results of the performance. The 'planned path' describes the number of steps initially planned to take by the robot to reach the goal from the start point. The 'steps taken' denotes the actual steps the robots take to reach the goal. Both the 'planned path' and 'steps taken' should be equal since this is a static environment case. And the 'time' denotes the time taken in total. The 'memory' denotes the memory it takes for computation, while the 'nodes' denote the computed nodes, that is, the explored nodes for the path. Similarly, in the same context, the performance metrics for RRT* are analyzed. Fig. 5 presents the LADy simulation in the static path of the given map.

3) LADy in a static environment

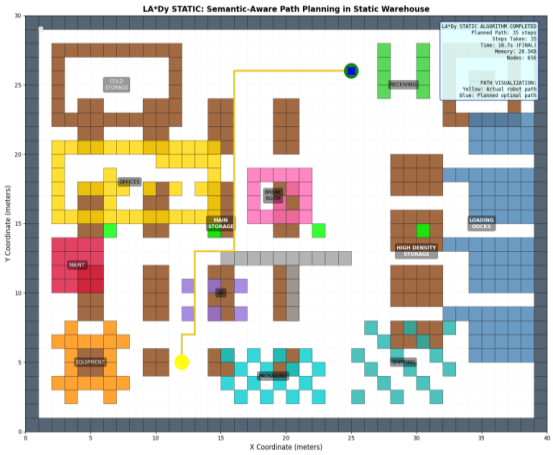

Fig. 5. LADy simulation in the static state of the given map.

B. Simulation in Dynamic Environment

For simulation in a dynamic phase of the given warehouse environment, in Fig. 2, it is intended to benchmark against renowned algorithms for dynamic environments, D* lite and a hybrid algorithm, a simplified interpretation of commercial AMR path planning approaches (see Fig. 6).

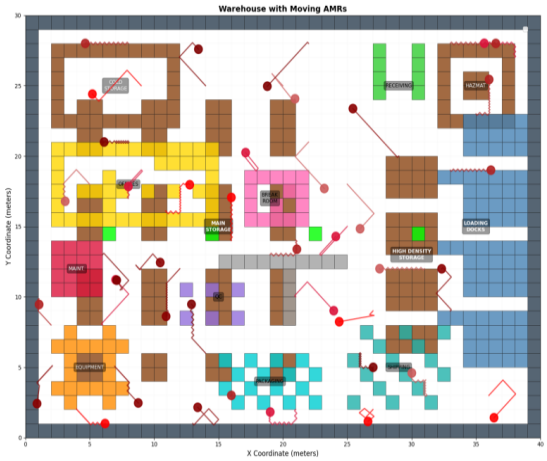

Fig. 6. Complex warehouse dynamic setup for simulation.

The study shows, performed rigorous testing by simulating in Matplotlib, like in a static simulation environment, assigning a combination of 25 sets of start and goal points, iterating the testing of each set 3 times to curate the average for a particular start and goal points. In the dynamic environment, along with the warehouse setup, 40 moving obstacles are deployed in the warehouse simulation setup that roam randomly in the free spaces.

The red circles are the moving obstacles in the free spaces, and the tail that follows is place for reference to its trail of movement in the warehouse simulation space. While random obstacle motion provides robust evaluation, deterministic and structured scenarios such as corridor blockages and congestion patterns will be explored in future work for controlled analysis.

1) Benchmarking against D lite*: For performance benchmarking, conventional D* lite algorithm from its publication was implemented and simulated in the developed simulation setup in Matplotlib [35]. In the simulation results, as in Fig. 7, comprise attributes like,

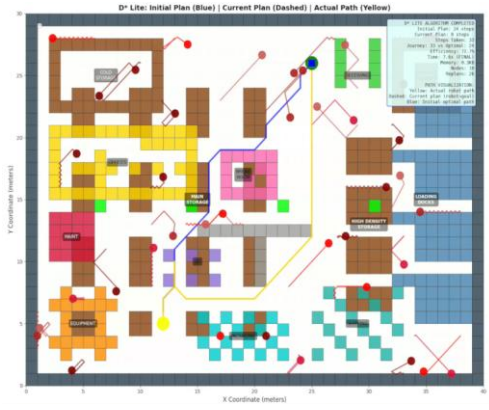

Fig. 7. D* lite simulation in the dynamic state of the given map.

- Initial plan : the initial path drafted by the algorithm
- Current plan : the remaining path to be covered by the robot from the current position to the goal position
- Steps taken : Steps covered by the robot
- Efficiency : efficiency of the final path the robot took with respect to the initial planned path

$$Efficiency = \frac{Initialplan}{Stepstaken} \times 100$$

- Time : time taken in total travel
- Memory : memory consumed on computation
- Nodes : number of computed nodes on the path
- Replans : number of replans due to obstacles

2) Benchmarking against hybrid algorithm: Similar to benchmarking against D* lite, the performance metrics, as in Fig. 8, of the hybrid algorithm in the given dynamic state of map in Fig. 6, is analyzed and compared with the proposed algorithm's performance. The simulation results' performance metrics of this algorithm shares the same context described for the D* lite algorithm above. The commercial AMR baseline is a simplified approach for conceptual comparison, not an exact replication of any proprietary systems.

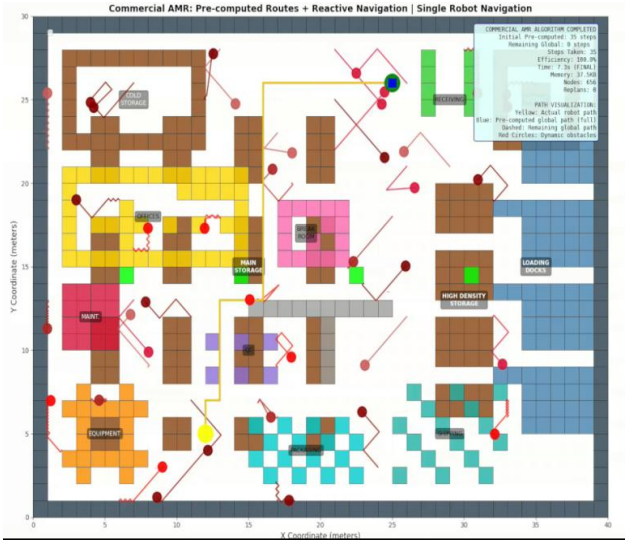

Fig. 8. Hybrid algorithm simulation in the dynamic state of the given map.

3) LADy in dynamic environment: As already described the methodology of LADy algorithm of its working, it works as per its methodology described irrespective of the state of the given map, either static or dynamic doesn't influence the working of the proposed algorithm, which is a key characteristic of the proposed algorithm. For simulation of LADy algorithm, as in Fig. 9, in the dynamic simulation setup it has implemented complementary functionalities in simulation code to simulate best possible real-time environment. The research implemented and simulated sensor range, that is, visibility of moving obstacles within a particular range, like in LiDAR used in real-time applications. The implementation of the mentioned sensor range can be noticed in the simulation recordings provided below.

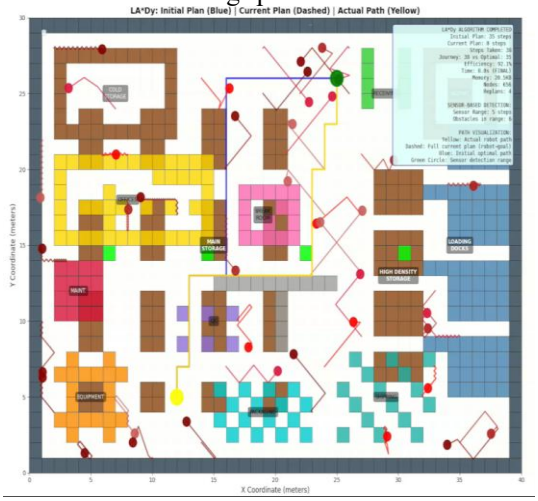

Fig. 9. LADy simulation in the dynamic state of the given map.

For a glance at all simulation results of LADy algorithm, see the Data Availability section.

VI. Metrics

With the simulation results curated with many combinations of start and goal points, set inputs performance metrics are cumulated for benchmarking the proposed algorithm LADy against the renowned algorithms. The metrics are factored based on the state of the environment, whether the environment is static or dynamic.

A. For a Static Environment

For a static environment, compare LADy algorithm against the A* and RRT*, as mentioned before, for the factors defined below:

- Path length (in steps)
- Time (in seconds)
- Memory (in Kilobytes)
- Nodes

1) LADy vs. RRT vs. A* (Path length)*

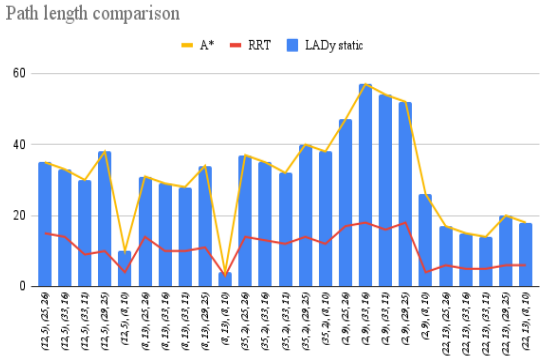

Fig. 10. Comparison chart of path length between LADy, RRT* and A*.

2) LADy vs. RRT* vs. A* (Time)

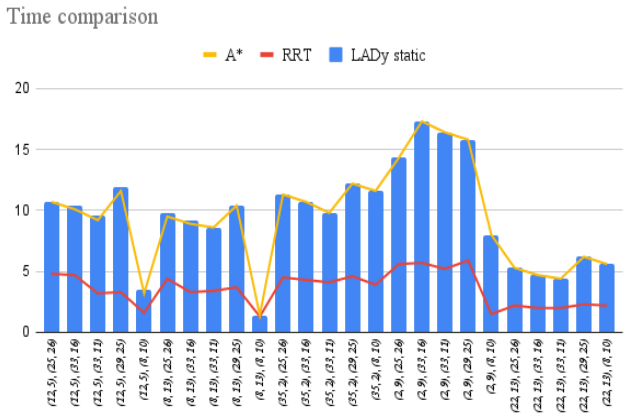

Fig. 11. Comparison chart of time between LADy, RRT* and A*.

3) LADy vs. RRT* vs. A* (Memory)

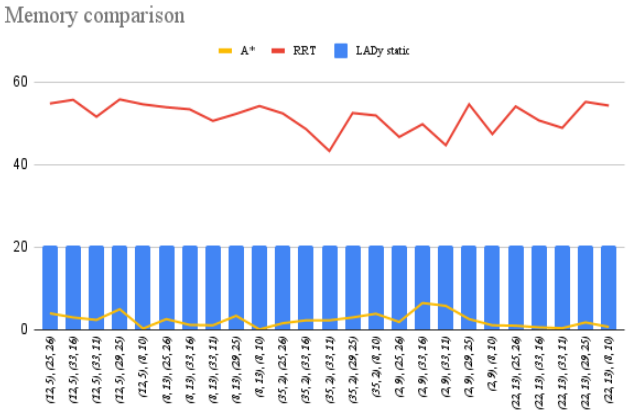

Fig. 12. Comparison chart of memory between LADy, RRT* and A*.

4) LADy vs. RRT* vs. A* (Nodes)

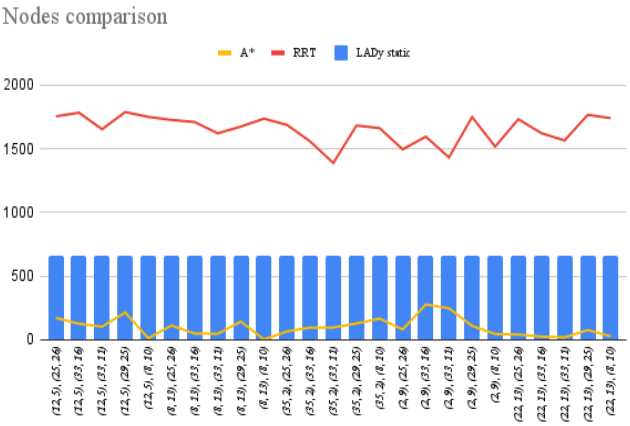

Fig. 13. Comparison chart of nodes between LADy, RRT* and A*.

As shown in the charts, both LADy and A* maintain the same length (Fig. 10) and nearly same time (Fig. 11) obviously in a static environment, since LADy itself has A* as its base layout of algorithm. While RRT* has less path length and time taken compared to A* and LADy algorithms.

Though, in memory consumption and number of computed nodes, A* and LADy are more efficient than RRT*. Even in paths computed by algorithms are at times less efficient, being a probability-based random sampling algorithm.

The chart shown in Fig. 10, shows the comparative analysis between LADy, RRT* and A* in the context of path length, while analysis in context of time is given in Fig. 11, memory is given in Fig. 12, and nodes are given in Fig. 13.

For instance, in the Fig. 14, the robot can be found to overlap obstacles in static setup planning path as per RRT* algorithm.

And for A*, the LADy algorithm is equivalent to A* performance in terms of path length and time, while A* is more efficient than LADy in terms of memory and computed nodes count since LADy pre-computes semantic cost data in each node that increase the computation memory and count of nodes but makes the proposed algorithm environment-aware and consistent and more suitable in real-time in this simulations' aspect while A* can't be implemented without modifications for such usage.

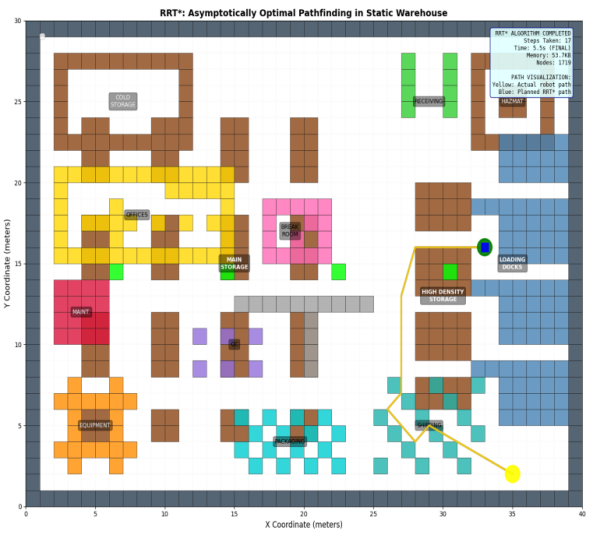

Fig. 14. Robot overlapping obstacle in RRT* simulation.

As mentioned above, the robot path can be seen in the above image, Fig. 14, overlapping the obstacles, and it can't be implemented without an additional approach to avoid collision with obstacles and get a smooth and realistic applicable path planned for real-time usage. For all performance metrics, refer the Data Availability section.

It covers all iterations and combinations of start and goal point sets.

B. For a Dynamic Environment

For dynamic environment, compare LADy algorithm against the D* lite and hybrid algorithm, a simplified interpretation of commercial AMR path planning approaches, as mentioned before for the factors defined below:

- Path length (in steps)
- Time (in seconds)
- Memory (in Kilobytes)
- Nodes
- Replans

1) LADy vs. D lite vs. Hybrid (Path length)*

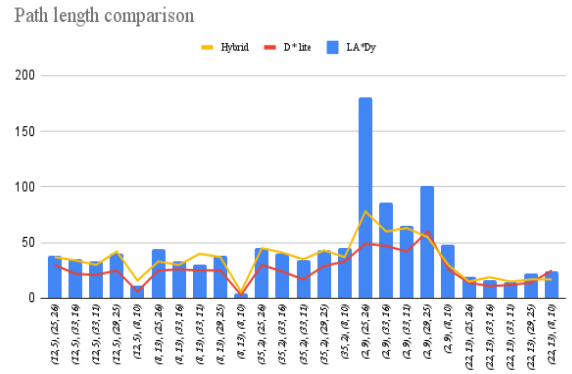

Fig. 15. Comparison chart of path length between LADy, D* LITE and hybrid.

2) LADy vs. D lite vs. Hybrid (Time)*

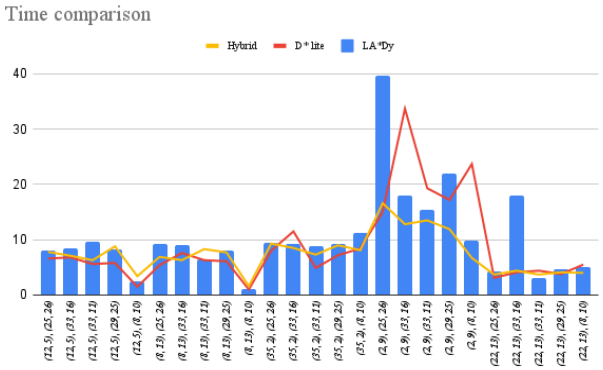

Fig. 16. Comparison chart of time between LADy, D* LITE and hybrid.

3) LADy vs. D lite vs. Hybrid (Memory)*

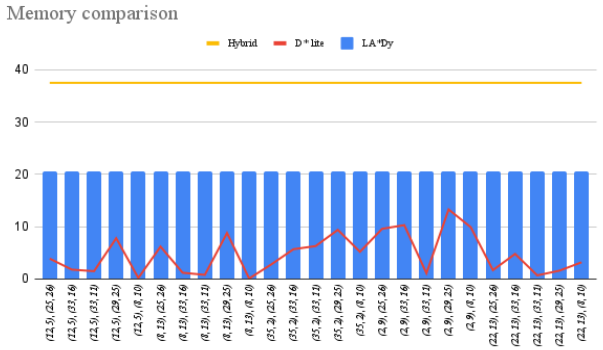

Fig. 17. Comparison chart of memory between LADy, D* LITE and hybrid.

4) LADy vs. D lite vs. Hybrid (Nodes)*

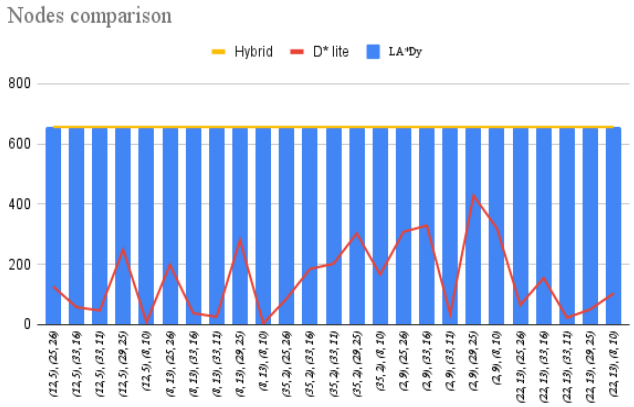

Fig. 18. Comparison chart of nodes between LADy, D* LITE and hybrid.

5) LADy vs. D lite vs. Hybrid (Replans)*

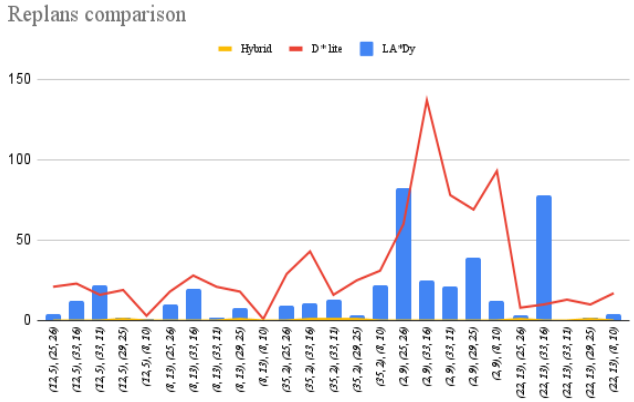

Fig. 19. Comparison chart of replans count between LADy, D* LITE and hybrid.

The iterations of simulation results are cumulated to average, yet still the moving obstacles deployed in the simulation are a random factor in the environment, so this should be considered in benchmark analysis. Experiments were conducted across multiple start–goal configurations with repeated tests to ensure consistency, although detailed statistical analysis is considered for future work.

VII. Discussion

From the above comparison charts, the results suggest that the proposed algorithm performs competitively under the tested conditions with respect to both D* lite algorithm, an optimal algorithm in terms of theoretical aspects for dynamic environments, and hybrid algorithm which is a simplified interpretation of commercial AMR path planning approaches, a specifically designed planning algorithm for warehouses with approaches of sensor fusion, reactive navigation dynamic obstacle avoidance.

In aspects of path length, as in Fig. 15, and time, as in Fig. 16, it approximately tallies the performance of D* lite which is theoretical optimal, and hybrid algorithm which is

application wise in real-time optimal. In aspects of memory, as in Fig. 17, and nodes, as in Fig. 18, both LADy and hybrid algorithm are consistent irrespective of the inputs for path planning. In aspects of replan counts, as in Fig. 19, the results show a wide variation and inconsistency due to the randomized moving obstacles deployment. While the hybrid algorithm has a lower count of replans since commercial AMR path planning approaches have their paths pre-defined for sets of start positions and goal positions, and they dodge obstacles at the moment reactively and not detour in most cases, which is the reason for the lower count of replans in the simulation results. The proposed method does not aim to establish computational improvements over D* Lite, but focuses on implementation with semantic cost integration.

VIII. Complexity Analysis

The computational complexity of the LADy algorithm is analyzed across its three distinct phases: semantic encoding, modified A* planning, and dynamic execution.

A. Phase 1: Semantic Encoding

The semantic encoding phase computes traffic and task ratings for all free navigable nodes in the warehouse environment.

Given a warehouse grid of dimensions $W \times H$ with $|O|$ static obstacles, the number of free nodes is, $|F| = W \times H - |O|$. For each free node, the algorithm performs constant-time area classification and rating calculation operations.

Time Complexity: $O(|F|) = O(W \times H)$

Space Complexity: $O(|F|)$ for storing traffic_ratings & task_ratings dictionaries

B. Phase 2: Modified A* Planning

The modified A* phase follows standard A* complexity with additional semantic cost calculations. In the worst case, the algorithm explores all free nodes, with each node having at most 4 neighbors in a grid-based environment. The semantic cost calculation adds constant overhead per node evaluation.

Time Complexity:

$$O\big(|F|log(|F|)\big) = O\big(W \times Hlog(W \times H)\big)$$

Space Complexity:

$O(|F|)$ For open set, closed set, and cost tracking structures

The logarithmic factor arises from priority queue operations, while the semantic cost computation $\big(\alpha \times t(n) + \beta \times T(n)\big)$ adds only constant overhead per node.

C. Phase 3: Dynamic Execution

The dynamic execution phase complexity depends on the number of replanning operations triggered by obstacle changes. Let R denote the number of replanning events during path execution. Each replanning operation has the same complexity as the initial modified A* planning.

Time Complexity per Replan: $O\big(|F|log(|F|)\big)$

Total Dynamic Phase Complexity: $O\big(R \times |F|log(|F|)\big)$

Space Complexity: $O(|F|)$ For maintaining dynamic structures and current path

D. Phase 4: Overall Algorithm Complexity

The total computational complexity of LADy combines all three phases:

Time Complexity:

$$O\big(W \times H + (R+1) \times W \times Hlog(W \times H)\big)$$

Space Complexity:

$$O(W \times H)$$

For typical warehouse scenarios, where, $W = 40$, $H = 30$, and $R \ll W \times H$. The algorithm demonstrates practical efficiency. The semantic encoding overhead is linear and performed only once during initialization, while the modified A* planning provides optimal initial paths with semantic awareness.

IX. Conclusion and Future Scope

The proposed algorithm has shown expected optimal results when benchmarked with A* and RRT* under static circumstances, and against D* lite and the hybrid algorithm used in current commercial AMRs under dynamic circumstances. In the field of Autonomous Mobile Robots operating in warehouses for respective functionalities and operations, typical paper-based path planning algorithms with original implementation don't produce optimal solutions in real-time when compared to simulations. To operate an Autonomous Mobile Robot in a controlled, known environment, yet dynamic, the renowned algorithms are modified or optimized according to their own functionalities and needs, differing between different operating warehouses. Combining two or more algorithms with a reactive nature, with the help of sensor fusion, an AMR functions in a warehouse. Yet, the presented algorithm, LADy, performs equivalent to an algorithm that is renowned in its specified state of environment. With curated benchmarking results, it is evident that LADy performs well optimally in a static environment, comparatively equivalent to A*, a renowned algorithm for static environments, and performs optimally in a dynamic environment comparatively equivalent to the hybrid algorithm that is implemented as a simplified interpretation of the path planning approaches used in commercial AMRs of warehouses.

As a future scope, the proposed, especially tailored motion planning algorithm for warehouse environments can be further improved with optimization and modularity for autonomous vehicles. The increasing trend of autonomous vehicles. As a driver, the vehicle not only should plan the path to the destination but also know the optimized path, which can't be achieved unless it is aware of the environment.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study.

Data Availability

The complete simulation results and benchmarking results of the LADy algorithm are publicly available in the supplementary repository:

https://github.com/Harishma356/LADy_simulation_results

References

[1] S. Teng et al., "Motion Planning for Autonomous Driving: The State of the Art and Future Perspectives," IEEE Trans. Intell. Veh., vol. 8, no. 6, pp. 3692-3711, Jun. 2023, doi: 10.1109/TIV.2023.3274536.

[2] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," Eng. Sci. Technol. Int. J., vol. 40, p. 101343, 2023.

[3] J. Jin et al., "Conflict-based search with D* lite algorithm for robot path planning in unknown dynamic environments," Comput. Electr. Eng., vol. 105, p. 108473, 2023.

[4] M. Aizat, A. Azmin, and W. Rahiman, "A survey on navigation approaches for automated guided vehicle robots in dynamic surrounding," IEEE Access, vol. 11, pp. 33934-33955, 2023.

[5] K. P. Jayalakshmi, V. G. Nair, and D. Sathish, "A comprehensive survey on coverage path planning for mobile robots in dynamic environments," IEEE Access, 2025.

[6] L. Chen et al., "End-to-end autonomous driving: Challenges and frontiers," IEEE Trans. Pattern Anal. Mach. Intell., 2024.

[7] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," IEEE Trans. Intell. Veh., vol. 9, no. 1, pp. 103-118, 2023.

[8] M. Reda et al., "Path planning algorithms in the autonomous driving system: A comprehensive review," Robot. Auton. Syst., vol. 174, p. 104630, 2024.

[9] M. Y. Yildirim and R. Akay, "An efficient grid-based path planning approach using improved artificial bee colony algorithm," Knowl.-Based Syst., p. 113528, 2025.

[10] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," IEEE Access, vol. 2, pp. 56-77, 2014.

[11] Yuan, L., Zhao, J., Li, W. et al. Improved Informed-RRT* Based Path Planning and Trajectory Optimization for Mobile Robots. Int. J. Precis. Eng. Manuf. 24, 435–446 (2023). https://doi.org/10.1007/s12541-022-00756-6.

[12] Zhang, Liding, et al. "Motion planning for robotics: A review for sampling-based planners." Biomimetic Intelligence and Robotics (2025): 100207.

[13] X. Yin et al., "Route planning of mobile robot based on improved RRT star and TEB algorithm," Sci. Rep., vol. 14, no. 1, p. 8942, 2024.

[14] A. Alexander et al., "A comprehensive survey of path planning algorithms for autonomous systems and mobile robots: Traditional and modern approaches," IEEE Access, 2025.

[15] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," Annu. Rev. Control Robot. Auton. Syst., vol. 7, 2023.

[16] Q. He et al., "Research on autonomous navigation of mobile robots based on IA-DWA algorithm," Sci. Rep., vol. 15, no. 1, p. 2099, 2025.

[17] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path planning for autonomous mobile robots: A review," Sensors, vol. 21, no. 23, p. 7898, 2021.

[18] J. Wen, F. Wang, and Y. Su, "A bi-layer collaborative planning framework for multi-UAV delivery tasks in multi-depot urban logistics," Drones, vol. 9, no. 7, p. 512, 2025.

[19] T. Xue, L. Xu, and Z. Zhou, "Multi-objective optimization method for AMR dispatching in warehouse environments," Procedia CIRP, vol. 134, pp. 897-902, 2025.

[20] V. R. Arvind, R. M. Shrinidhi, T. Deepa and M. Maheedhar, "Intelligent Warehousing: A Machine Learning and IoT Framework for Precision Inventory Optimization," in IEEE Access, vol. 13, pp. 169381-169414, 2025, doi: 10.1109/ACCESS.2025.3614679.

[21] R. Vrabič et al., "Bio-inspired traffic pattern generation for multi-AMR systems," Appl. Sci., vol. 15, no. 5, p. 2849, 2025.

[22] D. Xiang et al., "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot," Sci. Rep., vol. 12, no. 1, p. 13273, 2022.

[23] G. Pugliese et al., "AMR-assisted order picking: Models for picker-to-parts systems in a two-blocks warehouse," Algorithms, vol. 15, no. 11, p. 413, 2022.

[24] M. Aizat, N. Qistina, and W. Rahiman, "A comprehensive review of recent advances in automated guided vehicle technologies: Dynamic obstacle avoidance in complex environment toward autonomous capability," IEEE Trans. Instrum. Meas., vol. 73, pp. 1-25, 2023.

[25] D. Zotou et al., "Warehouse automation solution through a goods-to-person AMR," in Proc. 28th Int. Conf. Methods Models Autom. Robot. (MMAR), 2024.

[26] P. Y. Leong and N. S. Ahmad, "Exploring autonomous load-carrying mobile robots in indoor settings: A comprehensive review," IEEE Access, 2024.

[27] D. González et al., "A review of motion planning techniques for automated vehicles," IEEE Trans. Intell. Transp. Syst., vol. 17, no. 4, pp. 1135-1145, 2015.

[28] Z. Li et al., "A Survey of Reinforcement Learning-Based Motion Planning for Autonomous Driving: Lessons Learned from a Driving Task Perspective," arXiv:2503.23650, 2025.

[29] X. Ruan et al., "Improved DDPG-based path planning for mobile robots," Concurrency Comput. Pract. Exp., vol. 37, no. 25-26, p. e70317, 2025.

[30] R. Zhao et al., "A survey on recent advancements in autonomous driving using deep reinforcement learning: Applications, challenges, and solutions," IEEE Trans. Intell. Transp. Syst., 2024.

[31] A. Papadimitriou, D. Folinas, and I. Kostavelis, "Human-robot collaborative picking system for agile warehouses," Int. J. Adv. Manuf. Technol., pp. 1-18, 2025.

[32] V. Garg, J. D. Maywald, and M. Naman, "Optimising human-robot collaboration for efficiency in retail warehousing," Int. J. Retail Distrib. Manage., vol. 53, no. 10-11, pp. 1123-1138, 2025.

[33] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Trans. Syst. Sci. Cybern., vol. 4, no. 2, pp. 100-107, 1968.

[34] J. Nasir et al., "RRT*-SMART: A rapid convergence implementation of RRT," Int. J. Adv. Robot. Syst., vol. 10, no. 7, p. 299, 2013.

[35] S. Koenig and M. Likhachev, "D* lite," in Proc. 18th Nat. Conf. Artif. Intell., 2002.