IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 April 2025, accepted 26 May 2025, date of publication 2 June 2025, date of current version 12 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3575168

RESEARCH ARTICLE

Efficient Pathfinding on Grid Maps: Comparative Analysis of Classical Algorithms and Incremental Line Search

AMR ELSHAHED, MAJID KHAN BIN MAJAHAR ALI, AHMAD SUFRIL AZLAN MOHAMED, FARAH AINI BINTI ABDULLAH, AND TS. LEE JIAN AUN

School of Mathematical Sciences, Universiti Sains Malaysia (USM), George Town 11800, Malaysia

Corresponding author: Majid Khan Bin Majahar Ali (majidkhanmajaharali@usm.my)

This work was supported by the Universiti Sains Malaysia, Research University Team (RUTeam) Grant Scheme under Grant R502-KR-RUT001-0000000406-K134.

ABSTRACT Pathfinding on grid maps is a cornerstone problem in robotics, autonomous navigation, and game development. Classical algorithms such as A*, Dijkstra's, and Breadth-First Search (BFS) are known for their ability to guarantee optimal solutions, while others like Depth-First Search (DFS) and Greedy Best-First Search prioritize speed over optimality. However, these algorithms can be computationally inefficient in large-scale or real-time environments. This paper introduces Incremental Line Search (ILS), a novel optimization strategy that incorporates Bresenham's line algorithm to define a dynamically adjustable corridor between the start and goal points. By restricting the search to this focused region and expanding only when necessary, ILS significantly reduces computational overhead while preserving optimality in cases where the underlying algorithm guarantees it, and delivering near-optimal paths in others. Extensive experiments were conducted to evaluate the performance of ILS across various metrics, including execution time, visited nodes, and path length, over a total of 6000 grid maps generated with obstacle densities of 10%, 20%, and 30%. On average, ILS achieved a **87.31%** reduction in execution time and a **71.44%** reduction in node expansions compared to their standard counterparts. Despite these improvements, path quality was preserved: Best-First Search showed a **63.24%** improvement in path length, while DFS improved by **86.70%**, confirming that ILS not only accelerates computation but also leads to smoother paths in most scenarios. These findings affirm ILS as a highly effective and resilient optimization technique for demanding pathfinding tasks. Its dynamic adaptability renders ILS especially suitable for applications in robotics, autonomous systems, and resource-constrained environments, setting the stage for future advancements in multi-agent navigation and real-time optimization.

INDEX TERMS Path planning, shortest path problem, heuristic algorithms.

I. INTRODUCTION

Path planning is a critical component in a wide range of applications, including autonomous vehicles, robotic systems, and video game AI. In the context of self-driving cars, an effective path planner is essential to ensure that the vehicle can navigate safely and efficiently without human intervention [23]. These systems rely on robust algorithms to compute collision-free routes in real time, directly impacting both passenger safety and ride comfort. Similarly, autonomous aerial vehicles (AAVs) require efficient path planning to successfully perform tasks such as surveillance and package delivery, often within complex and cluttered environments [16], [22]. A safe and reliable trajectory is vital for the successful completion of such missions. Given the critical nature of these scenarios, substantial research efforts have been devoted to advancing path planning techniques. This paper focuses on enhancing the efficiency of grid-based pathfinding algorithms.

Pathfinding on grid maps remains a fundamental challenge in robotics, video games, and autonomous navigation systems. At its core, the problem involves computing

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

the shortest path between two points on a grid, while avoiding obstacles and minimizing computational overhead. Traditionally, algorithms such as Dijkstra's [12] and A* [16] have been widely adopted for their ability to provide optimal solutions. However, these algorithms often face scalability issues in large or real-time applications, especially in dynamic and obstacle-dense environments.

Recent advancements in pathfinding research aim to address these limitations by integrating heuristic functions, geometric constraints, and adaptive mechanisms. Techniques like Weighted A* [13], Theta* [17], and Jump Point Search (JPS) [15] have demonstrated significant improvements in performance by prioritizing promising paths and reducing redundant node expansions. Nevertheless, a critical trade-off still persists between computational efficiency, path optimality, and the ability to generalize across diverse environments.

In response, this paper introduces a novel method called Incremental Line Search (ILS), which combines classical pathfinding with geometric optimization. ILS utilizes Bresenham's line algorithm [1] to define a focused corridor around the straight line connecting the start and goal points. By restricting exploration to this corridor, ILS substantially reduces the number of visited nodes while preserving path optimality.

The key contributions of this study are as follows:

1) Integration of classical algorithms (A*, Dijkstra's, BFS, DFS, and Best-First Search) within the ILS framework for enhanced efficiency.
2) Comprehensive evaluation of standard versus ILS-enhanced algorithms across key metrics such as runtime, node expansions, and path quality.
3) Demonstration of ILS's effectiveness through extensive experiments, including real-world satellite data, highlighting its suitability for real-time and resource-constrained applications.

The remainder of this paper is organized as follows: Section II presents a review of related pathfinding approaches. Section III details the ILS methodology and algorithmic adaptations. Section IV discusses experimental results and analysis. Section V addresses limitations and future research directions. Section VI concludes the study.

II. RELATED LITERATURE

Pathfinding on grid-based maps has been one of the fundamental problems in artificial intelligence, robotics, and navigation for decades. This section reviews significant advances in pathfinding algorithms, from classical approaches and heuristic methods to recent innovations incorporating geometric and heuristic-based optimizations.

A. CLASSICAL PATHFINDING ALGORITHMS

Classical pathfinding algorithms, such as Dijkstra's [12] and Breadth-First Search (BFS) [25], provide foundational solutions to shortest-path problems. Dijkstra's algorithm ensures optimal paths in weighted graphs, but its computational overhead scales poorly with graph size. BFS, while effective for unweighted graphs, lacks scalability in large environments due to its exhaustive exploration of equidistant nodes. Depth-First Search (DFS) [26], conversely, trades completeness for reduced memory usage but oftentimes results in suboptimal paths.

A significant limitation of classical algorithms is their inability to prioritize promising paths efficiently. This challenge led to the development of heuristic-driven methods.

B. HEURISTIC-BASED METHODS

The A* algorithm [16] significantly transformed pathfinding by incorporating heuristics to prioritize search efforts, which led to a substantial decrease in node expansions. A* ensures optimal solutions when the heuristic is admissible and consistent. Variants like Weighted A* (wA*) [13] allow bounded suboptimality to achieve faster runtimes by scaling the heuristic, making them suitable for time-critical applications.

Jump Point Search (JPS) [15] further optimizes A* by exploiting grid regularity to eliminate unnecessary expansions in uniform-cost environments. Studies by [18] showed that JPS achieves near-linear performance in large grids, making it highly efficient for robotic navigation.

C. GEOMETRIC AND HYBRID APPROACHES

Pathfinding algorithms are now more capable due to the addition of geometric constraints that cut down on unnecessary calculations. Initially designed for drawing straight lines in computer graphics, Bresenham's line algorithm has found a new purpose in pathfinding, helping to focus searches within defined corridors. This evolution reflects a growing trend of blending classic algorithms with geometric ideas to improve efficiency.

Theta* [17], for instance, allows any-angle pathfinding by performing line-of-sight checks between nodes, resulting in more natural paths compared to A*. Hybrid algorithms such as Safe A* [20] and Adaptive Window Search [21] integrate safety and adaptive heuristics, enabling efficient navigation in dynamic and uncertain environments.

D. RECENT ADVANCES AND COMPARATIVE OVERVIEW

Pathfinding on grid maps has been studied extensively, leading to a variety of classical algorithms and more recent learning-based approaches. Graph-based search algorithms form the foundation of classical path planning. Breadth-First Search (BFS) and Dijkstra's algorithm are guaranteed to find an optimal path on an unobstructed grid (for unweighted and weighted graphs, respectively), but they are often slow on large maps due to exhaustive exploration. The A* algorithm [16] improves efficiency by using a heuristic to guide search toward the goal, drastically reducing explored nodes while preserving optimality (given an admissible heuristic) [21]. However, even A* can produce paths that

are sub-optimal in Euclidean distance and appear ''jagged'' because movement is constrained to grid connections [21]. Researchers have addressed this by allowing *any-angle* movement: Theta* is a variant of A* that permits line-of-sight moves between non-adjacent grid nodes, yielding shorter, more natural paths [17]. Theta* maintains near-optimal path length while significantly smoothing the trajectory by not strictly adhering to the grid geometry [17]. Another enhancement to A* on uniform-cost grids is Jump Point Search (JPS), which prunes symmetric expansions on the grid to speed up search without losing optimality [15]. JPS skips over straight-line sequences of nodes and only considers ''jump points,'' thereby accelerating search by eliminating redundant steps [15].

For dynamic environments, variants of A* such as D* and D* Lite were introduced to update paths efficiently when the grid's obstacle configuration changes. D* Lite, for example, can replan an optimal path significantly faster than running A* from scratch upon each change [19]. This is crucial for robotic navigation in unknown or changing environments, but these algorithms still operate on the grid connectivity and thus inherit the same grid-constrained path geometry (i.e., no direct any-angle shortcuts).

Beyond classical searches, learning-based methods have gained traction recently for path planning. Reinforcement learning approaches, particularly Deep Q-Networks (DQNs), learn a navigation policy by interacting with the environment. These methods can handle complex, high-dimensional state spaces and can adapt to uncertainties by learning from experience [22]. For instance, deep reinforcement learning has been applied to train robots to navigate without a predetermined map. However, a known drawback is that purely learned planners do not guarantee optimal paths and require extensive training data. Improved variants of Q-learning have been proposed to overcome slow convergence and suboptimal exploration. Ma et al. introduce a continuous local search Q-learning (CLSQL) that divides the global task into local regions, using Euclidean distance as a guiding heuristic to accelerate learning [29]. Xing et al. propose an enhanced Q-learning algorithm with a dynamic exploration factor and heuristic reward shaping to improve path learning efficiency in grid maps [23]. These approaches demonstrate faster convergence and better adaptation in unknown environments. Nonetheless, learning-based planners still struggle to match classical algorithms' guaranteed optimality in static, known grid scenarios, and their performance depends heavily on training quality. A recent survey by Singh et al. provides a comprehensive review of deep reinforcement learning algorithms for mobile robot path planning, highlighting that while such methods are promising in complex scenarios, challenges related to generalization and reliability still persist [7].

Table 1 summarizes key characteristics of representative path planning methods, highlighting gaps that motivate our proposed approach. We compare classical graph search algorithms, any-angle techniques, and learning-based methods across criteria such as optimality, ability to handle dynamic changes, and path realism. As shown in the table, no single existing method simultaneously offers *optimal any-angle paths* and *efficient incremental re-planning* without heavy computation. This gap motivates the development of our Incremental Line Search (ILS) method, which aims to combine the strengths of classical and any-angle planning.

TABLE 1. Comparison of path planning methods and their key characteristics.

Method	Optimal Path	Any-Angle	Dynamic Update
BFS [25]	Yes	No	No
Dijkstra [15]	Yes	No	No
A* [16]	Yes	No	No
D* Lite [19]	Yes	No	Yes
Theta* [17]	Near-Yes	Yes	No
JPS [11]	Yes	No	No
Deep Q-Learning [22]	No	Potential	N/A
Improved Q-Learning [29]	No	Potential	Partial
ILS (Proposed)	Yes (expected)	Yes	Limited

Note: ''Optimal Path'' refers to the algorithm's ability to guarantee the shortest possible path on a static grid. ''Any-Angle'' denotes the capability to generate smooth trajectories that are not constrained to grid-aligned movements. ''Dynamic Update'' indicates whether the method supports efficient re-planning in environments with changing conditions. Classical algorithms such as A*, Dijkstra, and D* Lite provide path optimality but often produce jagged, grid-constrained trajectories and lack adaptability to dynamic contexts. In contrast, any-angle planners like Theta* generate smoother paths but typically offer limited flexibility. The proposed Incremental Line Search (ILS) method seeks to balance these trade-offs by combining computational efficiency, smooth path generation, and partial adaptability. It yields an optimal path within the bounds of the current search corridor; however, global optimality is contingent on whether the shortest path resides within that region.

III. METHODOLOGY

This section presents the Incremental Line Search (ILS) methodology and its integration with classical pathfinding algorithms. It outlines the algorithmic modifications, experimental setup, evaluation metrics, and analysis protocols, offering a comprehensive overview of the proposed approach and its benefits.

A. ALGORITHMIC ADAPTATIONS

The proposed Incremental Line Search (ILS) methodology utilizes Bresenham's Line Algorithm [1] to define a search corridor between the start and goal points. This algorithm provides an efficient way to approximate a straight-line path in a discrete grid space using integer-only calculations, reducing computational overhead.

Theorem 1 (Line Approximation for Corridor Definition): Given two integer endpoints (x_0, y_0) *and* (x_1, y_1) *on a 2D grid,*

the line approximation method computes a discrete set of grid points that define a search corridor for pathfinding.

Let the horizontal and vertical distances between the endpoints be defined as:

$$\Delta x = |x_1 - x_0|, \quad \Delta y = |y_1 - y_0| \tag{1}$$

The corridor definition process consists of the following steps:

1) ***Initialization:*** *Compute an incremental decision variable to determine the pixel selection along the path*:

$$D = 2\Delta y - \Delta x \tag{2}$$

2) ***Grid Point Selection:*** *Determine the next grid point by evaluating the decision criterion to minimize the deviation from the ideal line*:
   - *If* $D < 0$*, increment the x-coordinate:* $(x + 1, y)$.
   - *If* $D \geq 0$*, increment both coordinates:* $(x+1, y+1)$.
3) ***Decision Variable Update:*** *Adjust* D *iteratively to maintain minimal error in the corridor definition*:

$$D_{new} = \begin{cases} D + 2\Delta y, & \text{if } D < 0 \\ D + 2(\Delta y - \Delta x), & \text{otherwise} \end{cases} \tag{3}$$

The final corridor is constructed by expanding an initial width proportionally to the smaller dimension of the grid, ensuring an efficient search region within the Incremental Line Search (ILS) methodology.

In the ILS methodology, the search corridor is initially defined along the Bresenham line connecting the start and goal points, with a fixed initial width. If no valid path is found within this corridor, its width is incrementally expanded by a predefined step size until a feasible path is discovered or the maximum allowable corridor width is reached.

By leveraging Bresenham's algorithm, ILS significantly reduces the number of explored nodes by constraining the search to a focused region. While the method guarantees an optimal path within the currently defined corridor, it does not ensure global optimality if the shortest path lies outside this region. Nevertheless, when the optimal path falls within the corridor or when the corridor expands sufficiently, ILS preserves the optimality guarantees of the underlying base algorithm.

1) **Region Definition:**
   - Compute a straight-line approximation between the start and goal points using Bresenham's line algorithm.
   - Define a corridor around this line, with an initial width proportional to a fixed percentage of the smaller grid dimension (denoted as *initial corridor width*).

   Bresenham's line algorithm [1] is employed to efficiently approximate a straight-line path between the start and goal points using only integer arithmetic. By avoiding computationally expensive operations such as multiplication and division, it is particularly well-suited for grid-based path planning. The algorithm incrementally identifies the grid cells intersected by the ideal straight line, and these cells form the initial corridor that guides the search process.
   This corridor constrains the search space to a narrow region surrounding the Bresenham line, significantly reducing the number of nodes explored. Only grid cells within this corridor are considered during path expansion. If no valid path is found, the corridor is incrementally expanded outward, forming the basis for dynamic corridor adjustment. This adaptive mechanism enhances computational efficiency while preserving the completeness of the search process.
2) **Dynamic Corridor Expansion:**
   - The width is incrementally expanded if no path is found within the initial corridor.
   - Expansion is conducted in fixed-width steps until a valid path is found or the maximum corridor width is reached.
   - The *maximum width* is defined as the smaller of the two grid dimensions, ensuring exhaustive exploration only when necessary.

   To understand this process in detail, Figure 1 illustrates how the search corridor is incrementally widened during the dynamic corridor expansion phase.

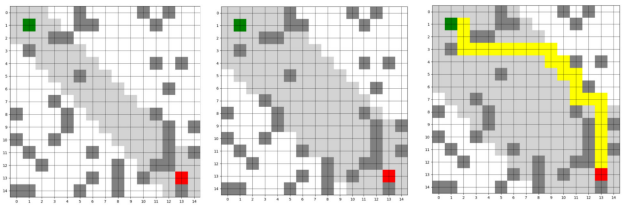

FIGURE 1. Illustration of the incremental line search (ILS) methodology demonstrating corridor expansion steps for pathfinding in a 200 × 200 grid.

3) **Algorithm Constraints:**
   - Modify pathfinding algorithms (A*, BFS, DFS, Dijkstra's, Best-First Search) to restrict their exploration to the defined corridor.
   - Nodes outside the corridor are excluded from the neighbor exploration process, significantly reducing computational overhead.
4) **Comparative Evaluation:**
   - Test each algorithm in its standard configuration and the ILS-constrained version.
   - Compare metrics such as execution time, visited nodes, and path lengths between the configurations.

To provide an overview of the ILS process, including corridor expansion and iterative pathfinding, Figure 2 presents a flowchart of the algorithm's operational steps.

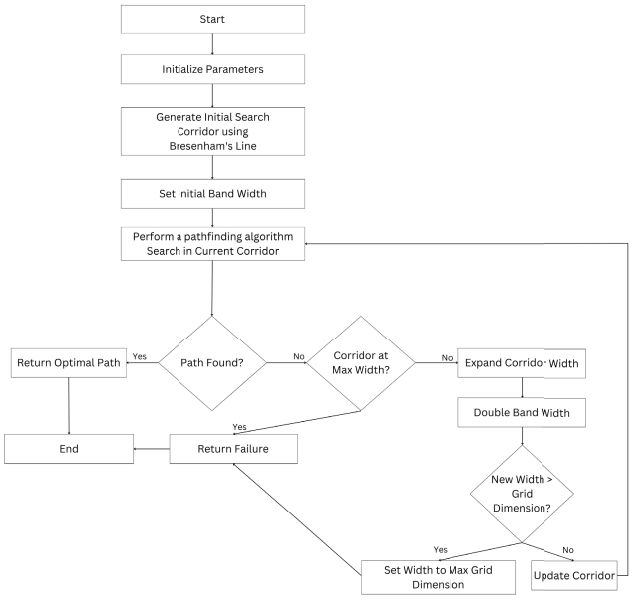

FIGURE 2. Flowchart of the incremental line search (ILS) algorithm.

1) ALGORITHM FOR INCREMENTAL LINE SEARCH

Algorithm 1 presents the pseudocode for implementing the ILS methodology. It highlights the way that the search region is constrained and dynamically expanded for classical pathfinding algorithms.

Below is a step-by-step explanation of the key operations:

Line 1-2: The algorithm takes as input a graph $\mathcal{G} = (V, E)$, the start node s, the goal node g, the chosen pathfinding algorithm $\mathcal{A}$ (e.g., A*, Dijkstra, BFS), the initial corridor width w_0, the maximum allowable corridor width $w_{\max}$, and a fixed step size Δw for expansion. The output is an optimal path P or failure if no path is found.

Line 4: A straight-line approximation from s to g is computed using Bresenham's line algorithm. This line acts as the center axis of the initial search corridor.

Line 5: An initial corridor of width w_0 is created around the Bresenham line, limiting the area within which the algorithm will search for a path.

Line 6-7: The pathfinding algorithm $\mathcal{A}$ is executed, but its search is constrained to only the nodes inside the currently defined corridor.

Lines 8-11: If a valid path P is found within the current corridor, it is immediately returned and the algorithm terminates successfully.

Lines 12-13: If no path is found, the corridor width is increased by Δw, and the corridor is recalculated accordingly. This allows the search space to grow progressively outward, only as needed.

Lines 14-16: If the corridor width exceeds the predefined maximum $w_{\max}$, the algorithm terminates with failure, indicating that no valid path could be found within reasonable limits.

This incremental corridor strategy enables significant reductions in the number of explored nodes and runtime compared to full-grid searches. It also maintains completeness, as the corridor can expand until it covers the entire grid if necessary.

Algorithm 1 Incremental Line Search (ILS)

1: **Input:** Graph $\mathcal{G} = (V, E)$, Start node s, Goal node g, Pathfinding algorithm $\mathcal{A}$, Initial corridor width w_0, Maximum corridor width $w_{\max}$, Step size Δw
2: **Output:** Optimal path P or *Failure*
3: **Step 1: Initialize Corridor**
4: Compute a straight-line approximation from s to g using Bresenham's line algorithm
5: Define an initial corridor of width $w \leftarrow w_0$ around the line
6: **while** True **do**
7: **Step 2: Execute Pathfinding in Corridor**
8: Run algorithm $\mathcal{A}$, restricting exploration to nodes within the defined corridor
9: **Step 3: Check for Valid Path**
10: **if** A valid path P is found **then**
11:
12: **return** P {Terminate with the optimal path}
13: **else**
14: Incrementally expand the corridor width: $w \leftarrow w + \Delta w$
15: Update the corridor region accordingly
16: **end if**
17: **Step 4: Terminate if Maximum Width Exceeded**
18: **if** $w > w_{\max}$ **then**
19:
20: **return** *Failure* {Terminate if no valid path exists within constraints}
21: **end if**
22: **end while**

B. ALGORITHM ADAPTATIONS TO ILS

To evaluate the effectiveness of the Incremental Line Search (ILS) framework, we integrated it into several well-known classical pathfinding algorithms, each with minor adaptations to conform to the corridor-based search space restriction. Below is a summary of how each algorithm was adapted:

A* + ILS: The standard A* algorithm was modified to include line-of-sight checks during neighbor expansion. If a node's parent has a clear line of sight to a neighbor, the algorithm connects them directly, skipping the current node. This adaptation follows the principle of Theta* [17] and reduces path length and node expansions.

Dijkstra + ILS: Since Dijkstra's algorithm is a variant of A* with no heuristic, the same line-of-sight logic was applied. This allows the algorithm to produce smoother paths without relying on heuristic guidance.

BFS + ILS: For unweighted graphs, BFS was paired with a post-processing step. After the standard BFS path was found,

a line-of-sight simplification was performed on the path to remove unnecessary intermediate waypoints, improving geometric quality.

DFS + ILS: DFS was constrained to explore only within the defined ILS corridor. While DFS does not guarantee optimality, restricting its depth-first traversal within a narrow corridor led to shorter paths and faster convergence in structured environments.

Best-First Search + ILS: Similar to A*, the heuristic-guided expansion in Best-First Search was limited to nodes inside the corridor. Since this algorithm aggressively follows the heuristic, ILS helped eliminate redundant detours by narrowing the area of exploration.

In all cases, nodes outside the corridor were excluded from consideration during neighbor expansion. This design ensured a fair and consistent application of ILS across algorithms while allowing each to leverage its inherent strengths within a reduced search space.

C. EXPERIMENTAL SETUP

1) HARDWARE AND SOFTWARE

The experiments were conducted on a MacBook Air (M1, 2020) with 8 GB RAM, utilizing Python 3.8 as the implementation language. The system ran on macOS Sonoma 14.5.

2) DATASET AND GRID CONFIGURATION

A 200 × 200 grid was chosen as the standard testbed for evaluating pathfinding algorithms. To analyze performance under varying environmental complexity, three distinct datasets were generated with obstacle densities of 10%, 20%, and 30%, representing sparse, moderate, and dense scenarios, respectively. Obstacles were distributed randomly using a uniform Bernoulli distribution, and all grids were post-processed to ensure the presence of at least one valid path between the start and goal points. The start and goal positions were fixed in the top-left and bottom-right corners, and the top row and rightmost column were kept obstacle-free to prevent accidental disconnection. All datasets and source code are publicly accessible at https://github.com/Amr-path/Incremental.Line.Search.git.

D. EVALUATION METRICS

The following metrics were employed to assess algorithm performance:

- **Execution Time (s):** The total time required to compute the path.
- **Visited Nodes:** The total number of grid cells explored during the pathfinding process.
- **Path Length:** The total length of the computed path in terms of grid cells traversed.
- **Percentage Improvements:** Improvements achieved by ILS over the standard search configuration, calculated as:

$$\text{Improvement } (\%) = \frac{\text{Metric}_{\text{Standard}} - \text{Metric}_{\text{ILS}}}{\text{Metric}_{\text{Standard}}} \times 100 \tag{4}$$

E. ANALYSIS PROTOCOL

The comparative analysis was conducted as follows:

1) **Algorithm Execution:** Each algorithm was executed on identical grids in both standard and ILS configurations.
2) **Comparisons:** Metrics such as execution time, visited nodes, and path lengths were compared between the two configurations.
3) **Statistical Validation:** A paired t-test was performed to validate the significance of observed differences.
4) **Visualization:** Results were visualized through bar plots and charts, highlighting the percentage improvements achieved by ILS.

IV. ANALYSIS

This section provides an in-depth evaluation of the performance of standard and ILS-version pathfinding algorithms. The analysis highlights improvements in execution time, visited nodes, and path length while providing visual insights into the results.

A. COMPARISON OF EXPLORED NODES: STANDARD VERSUS ILS FOR A*

Figure 3 provides a visual comparison of the nodes explored by the standard A* algorithm and the ILS-optimized version. The standard A* explores a larger area due to its exhaustive heuristic search, while the ILS adaptation effectively limits exploration to a narrow corridor, significantly reducing computational overhead.

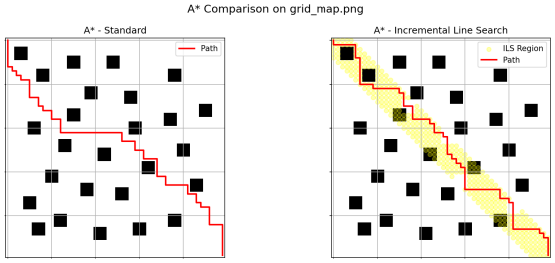

FIGURE 3. Comparison of explored nodes: standard A* vs. ILS-optimized A*.

B. RESULTS OVERVIEW

We present the results of evaluating the Incremental Line Search (ILS) method in comparison with standard pathfinding algorithms. The goal is to analyze ILS's effectiveness in improving computational efficiency and path quality.

To ensure reproducibility and robustness across different map structures, we generated a total of 2000 random grid maps of size 200 × 200 for each obstacle density scenario

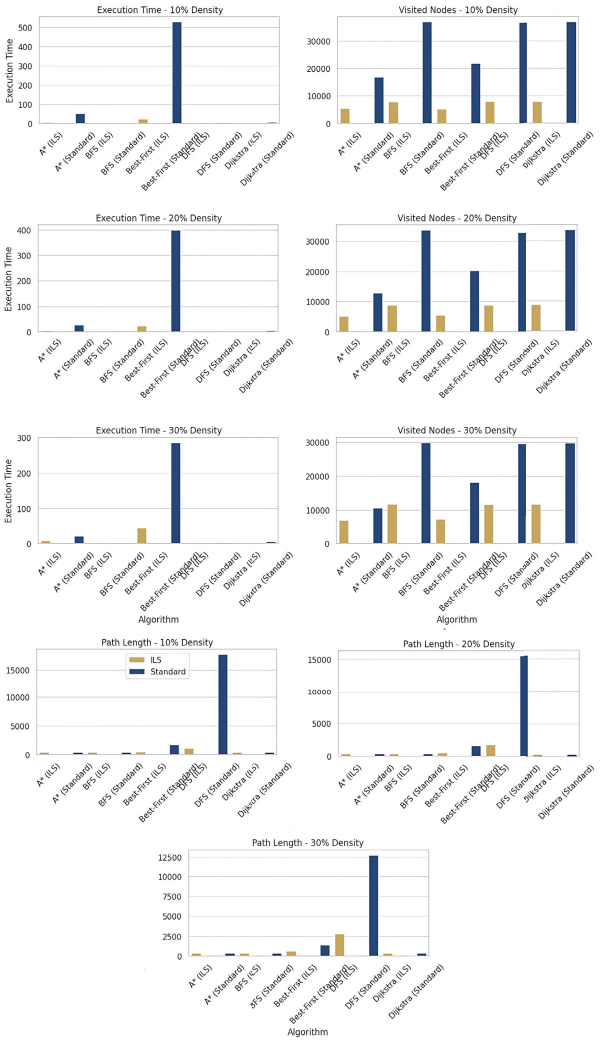

FIGURE 4. Comparison of visited nodes across 10%, 20%, and 30% obstacle densities for all algorithms and their ILS versions.

(10%, 20%, and 30%), representing sparse, moderate, and dense environments, respectively. Each classical pathfinding algorithm (A*, Dijkstra, BFS, DFS, and Best-First Search) was evaluated in two forms: (1) standard implementation, and (2) with ILS integration. All algorithms used 8-neighbor connectivity (horizontal, vertical, and diagonal movement), and all results were averaged over multiple randomly generated maps.

Exploring different scenarios with varying obstacle densities allows for a more comprehensive evaluation of how the ILS algorithm behaves in diverse environments. This approach helps identify the conditions under which ILS performs most effectively and reveals potential limitations. By analyzing its behavior across sparse, moderate, and dense grids, we gain insights into the algorithm's adaptability, efficiency, and robustness in handling varying levels of environmental complexity.

Figure 4 presents a comparison between all standard algorithms and their ILS-enhanced versions in the dense

TABLE 2. Average execution time with improvement by ILS.

Algorithm	10%	20%	30%
A* (Standard)	118.13	87.12	63.00
A* (ILS)	6.13	5.72	10.86
A* Improvement	**94.81%**	**93.43%**	**82.77%**
BFS (Standard)	0.84	0.71	0.67
BFS (ILS)	0.17	0.20	0.26
BFS Improvement	**79.15%**	**71.71%**	**61.12%**
Best-First (Standard)	527.02	399.09	286.34
Best-First (ILS)	23.61	22.77	44.24
Best-First Improvement	**95.52%**	**94.29%**	**84.55%**
DFS (Standard)	3.01	2.19	1.99
DFS (ILS)	0.23	0.21	0.32
DFS Improvement	**92.33%**	**90.58%**	**84.13%**
Dijkstra (Standard)	7.67	6.19	5.08
Dijkstra (ILS)	0.58	0.61	1.03
Dijkstra Improvement	**92.49%**	**90.09%**	**79.72%**

TABLE 3. Average visited nodes with improvement by ILS.

Algorithm	10%	20%	30%
A* (Standard)	29881.48	26616.26	23680.53
A* (ILS)	6898.19	7456.46	9856.90
A* Improvement	**76.91%**	**71.99%**	**58.38%**
BFS (Standard)	36920.00	33600.00	29933.00
BFS (ILS)	7918.50	8871.50	11820.83
BFS Improvement	**78.55%**	**73.60%**	**60.51%**
Best-First (Standard)	21904.80	20189.70	18275.83
Best-First (ILS)	5285.80	5516.30	7077.50
Best-First Improvement	**75.87%**	**72.68%**	**61.27%**
DFS (Standard)	36820.00	32762.00	29600.00
DFS (ILS)	7941.50	8847.00	11654.67
DFS Improvement	**78.43%**	**73.00%**	**60.63%**
Dijkstra (Standard)	36919.00	33599.00	29931.33
Dijkstra (ILS)	7915.80	8870.50	11819.83
Dijkstra Improvement	**78.56%**	**73.60%**	**60.51%**

scenario. The results highlight that even under tighter constraints, the corridor-based pruning mechanism employed by ILS consistently reduces search effort and execution time across all evaluated methods, while preserving or improving path quality.

To assess the impact of obstacle density on algorithmic performance, we present a series of comparative tables that summarize average results across three distinct environmental settings: sparse (10%), moderate (20%), and dense (30%) obstacle distributions. Each table reports key performance metrics: path length, execution time, and visited nodes for a range of classical search algorithms and their ILS-enhanced counterparts. The results are organized to enable clear, side-by-side comparisons, revealing trends in efficiency and solution quality as environmental complexity increases. Additionally, each table highlights the percentage improvement achieved by ILS over the standard approach, shown in bold directly beneath each algorithm pair. These findings provide valuable insights into the strengths, limitations, and trade-offs of each method under varying levels of navigational difficulty.

C. EXECUTION TIME

ILS demonstrated significant reductions in execution time across all algorithms and obstacle densities:

TABLE 4. Average path length with improvement by ILS.

Algorithm	10%	20%	30%
A* (Standard)	4123.04	3666.36	3068.73
A* (ILS)	562.16	705.80	934.67
A* Improvement	**86.37%**	**80.75%**	**69.54%**
BFS (Standard)	399.00	399.00	399.00
BFS (ILS)	399.00	399.00	399.00
BFS Improvement	**0.00%**	**0.00%**	**0.00%**
Best-First (Standard)	1741.90	1597.40	1372.00
Best-First (ILS)	506.10	534.40	655.67
Best-First Improvement	**70.95%**	**66.55%**	**52.21%**
DFS (Standard)	17674.50	15536.40	12773.33
DFS (ILS)	1107.20	1797.20	2819.33
DFS Improvement	**93.74%**	**88.43%**	**77.93%**
Dijkstra (Standard)	399.00	399.00	399.00
Dijkstra (ILS)	399.00	399.00	399.00
Dijkstra Improvement	**0.00%**	**0.00%**	**0.00%**

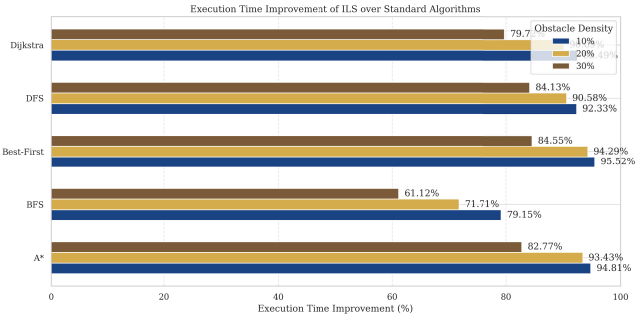

FIGURE 5. Execution time comparison for standard and ILS configurations.

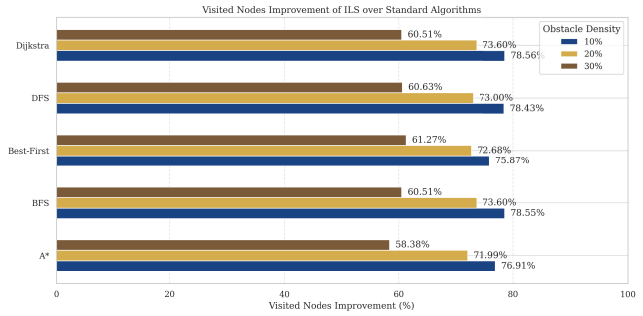

FIGURE 6. Visited nodes comparison for standard and ILS configurations.

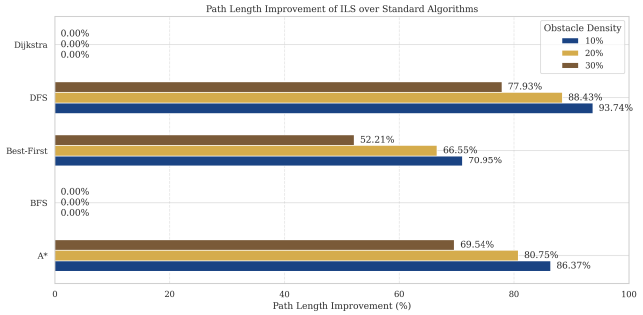

FIGURE 7. Path length comparison for standard and ILS configurations.

- **A*** achieved a reduction of **94.81%**, **93.43%**, and **82.77%** for 10%, 20%, and 30% obstacle densities, respectively.
- **Best-First Search** exhibited the highest reduction at 10% (**95.52%**), and consistently high improvements at 20% and 30% (**94.29%** and **84.55%**).
- **DFS** also showed impressive reductions, with improvements of **92.33%**, **90.58%**, and **84.13%**.
- **BFS** and **Dijkstra** maintained reductions above **60%** across all densities.

D. VISITED NODES

ILS significantly reduced the number of nodes visited during search:

- **DFS** and **BFS** demonstrated improvements up to **78%** in the 10% obstacle case, and **60%** or more at higher densities.
- **A***, **Dijkstra**, and **Best-First Search** also consistently reduced explored nodes by **58–78%**.

E. PATH LENGTH AND OPTIMALITY

The results showed meaningful path length improvements for non-optimal algorithms:

- **DFS** saw the most substantial improvement, with path lengths reduced by up to **93.74%**.
- **Best-First Search** also achieved strong gains, reducing path length by **70.95%** at 10% density.
- **A*** showed moderate improvement in Euclidean path length (**69.54–86.37%**) due to smoother paths.
- **BFS** and **Dijkstra** retained the same discrete grid path lengths, as expected, since they always return optimal paths in uniform-cost grids.

F. INSIGHTS AND IMPLICATIONS

- ILS was particularly beneficial for algorithms like **DFS** and **Best-First**, which often suffer from inefficient exploration.
- The corridor constraint helped even optimal algorithms like **A*** and **Dijkstra** reduce computation time and visited nodes.
- The results across all densities highlight the adaptability of ILS in both sparse and dense environments.
- ILS is suitable for applications requiring fast computation and constrained resource usage, such as robotics, embedded navigation, and real-time systems.

V. REAL-WORLD APPLICATION OF ILS

This section demonstrates the practicality of the Incremental Line Search (ILS) methodology through experiments conducted on real-world data derived from a satellite image. The selected location corresponds to coordinates `5.31361, 100.26908`, featuring a complex environment with varied terrain. The satellite image was transformed into a grid-based map to simulate real-world navigation challenges.

FIGURE 8. Original satellite image of the environment.

FIGURE 9. Grid map derived from the satellite image.

A. SETUP AND METHODOLOGY

The satellite image was processed into a binary grid map, marking obstacles and traversable areas. Figure 8 displays the original satellite image, while Figure 9 shows the resulting grid map after conversion. Classical pathfinding algorithms, along with their ILS adaptations, were executed on this grid map. Key performance metrics, such as execution time, visited nodes, and path length, were recorded and analyzed to evaluate the impact of the ILS methodology.

B. ALGORITHM IN ACTION

To illustrate the difference in behavior, Figure 10 compares the DFS algorithm with its ILS-enhanced counterpart. The ILS methodology limits the search space, resulting in fewer explored nodes and faster execution times while maintaining acceptable path quality.

C. RESULTS AND ANALYSIS

Table 5 summarizes the performance of standard and ILS versions of various pathfinding algorithms. The results demonstrate significant improvements in execution time and visited nodes with minimal impact on path length.

D. DISCUSSION

The experimental results provide compelling evidence for the effectiveness of the Incremental Line Search (ILS) methodology when integrated with classical pathfinding algorithms. By constraining the search space to a dynamically adjustable

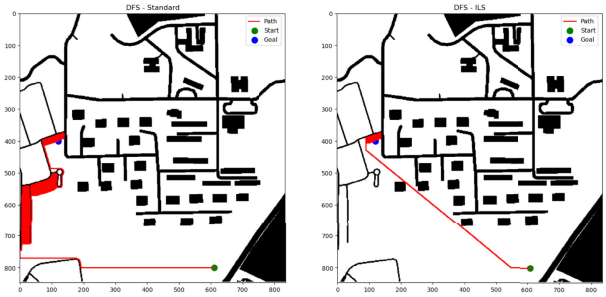

FIGURE 10. Comparison of DFS (left) and DFS - Incremental line search (right).

TABLE 5. Performance comparison of pathfinding algorithms on satellite data.

Algorithm	Time (s)	Visited Nodes	Path Length
A* (Standard)	98.923362	42,160	893
A* (ILS)	13.114038	17,670	891
Dijkstra (Standard)	18.736390	288,979	891
Dijkstra (ILS)	0.463445	42,352	891
BFS (Standard)	1.295967	289,031	891
BFS (ILS)	0.102061	42,386	891
DFS (Standard)	0.191055	12,591	6,211
DFS (ILS)	0.040886	2,464	1,447

corridor defined using Bresenham's line algorithm, ILS achieves significant reductions in execution time and node expansions while preserving, and in several cases improving, path optimality.

Across all tested scenarios, ILS consistently outperformed standard algorithms. On average, ILS achieved a **87.31%** reduction in execution time and a **71.44%** reduction in the number of visited nodes. Notably, Best-First Search recorded a **91.45%** reduction in execution time and a **69.94%** drop in node expansions, while A* achieved time savings of up to **94.81%** in sparse environments. These reductions translate into substantial computational efficiency, particularly critical for real-time and embedded systems.

ILS also enhances path quality in several cases. For example, Best-First Search showed a **63.24%** average reduction in path length across all densities, and DFS improved by **86.70%**, transforming from a highly inefficient strategy into a viable alternative when combined with ILS. A*, Dijkstra, and BFS maintained their optimal path lengths, as expected, demonstrating that the corridor-based restriction does not compromise completeness or optimality.

These improvements were validated over 6000 randomized grid maps with varying obstacle densities (10%, 20%, and 30%). As obstacle density increased, ILS continued to provide measurable benefits, although the gap between standard and ILS variants narrowed. This is due to limited availability of line-of-sight shortcuts in cluttered environments; however, ILS still provided consistent improvements across all metrics, even in dense scenarios.

Compared to other optimization techniques like Theta* [17] and Jump Point Search (JPS) [15], ILS distinguishes itself through its algorithm-agnostic nature and ease of integration.

While Theta* modifies the internal neighbor-expansion logic of A*, and JPS relies on pruning rules tied to uniform-cost grids, ILS can be added as a preprocessing or constraint layer to a wide range of algorithms, including uninformed searches like DFS and BFS. This makes ILS particularly valuable in heterogeneous or legacy systems where re-architecting the planner is impractical.

Overall, ILS not only achieves execution time reductions averaging over **87%** and node visitation improvements over **71%**, but also maintains or improves path quality. Its simplicity, generalizability, and effectiveness across a spectrum of search algorithms and environments underscore its potential as a robust optimization layer for practical pathfinding applications. Future directions may include adaptive corridor widths based on local obstacle distributions, or combining ILS with online re-planning in dynamic maps.

To contextualize our findings, we compare the observed performance of ILS-enhanced algorithms with patterns reported in prior studies.

The reduction in path length achieved by ILS (and Theta*-like algorithms) aligns with established results in any-angle path planning. Nash et al. [17] demonstrated that Theta*, by permitting moves not restricted to grid-aligned edges, finds paths closer to the true Euclidean shortest path. Similarly, our implementation of A*-ILS, which integrates Bresenham-based line-of-sight checks, consistently produced equal or shorter paths than grid-constrained A*, confirming that geometric shortcutting improves path optimality.

The improved efficiency of A* over uninformed methods such as BFS and Dijkstra is consistent with the well-documented benefits of heuristic guidance [16]. In our sparse grid experiments, A*-ILS further reduced computation time compared to A*, as it could skip over many intermediate nodes. This behavior resembles the optimization technique used in Jump Point Search (JPS), which prunes symmetric paths in uniform grids to speed up the search [15].

In denser environments, where obstacles often block direct sightlines, ILS naturally reverts to standard A* behavior. Our results showed that in such scenarios, A*-ILS and A* had similar runtimes, with ILS incurring only a minor overhead due to additional visibility checks. This behavior is expected, as line-of-sight pruning is less beneficial when long direct paths are unavailable.

The conceptual similarity between ILS and Theta* suggests comparable performance, and indeed our experiments showed that the path lengths obtained by A*-ILS closely matched those from Theta*. However, ILS offers greater flexibility: it can be used as a post-processing path smoother or embedded within various algorithms. Nash et al. [17] noted that Theta* achieves near-optimal paths with only a modest increase in computational effort an observation echoed in our own measurements.

While our experiments focus on static environments, insights from dynamic planning are also relevant. D* Lite, for instance, is designed to reuse prior search efforts and update paths efficiently as the environment changes [19]. Although ILS in its current form does not retain state between runs, its line-of-sight optimization could be advantageous if embedded within a dynamic re-planning framework. The combination of ILS's geometric efficiency with the incremental replanning of D* Lite could potentially yield both responsiveness and path realism an opportunity for future research.

In comparison to learning-based approaches, ILS remains entirely analytical and requires no training. Deep reinforcement learning (DRL) methods have shown strong performance in complex or partially observable environments. However, DRL approaches often lack guarantees of path optimality and may suffer from sample inefficiency. In contrast, ILS leverages full map knowledge to produce optimal or near-optimal paths in static settings with significantly lower computational overhead.

Overall, our findings reinforce known characteristics of classical and any-angle planners while demonstrating that ILS delivers comparable gains with a flexible and algorithm-agnostic design.

E. LIMITATIONS AND APPLICABILITY

While the proposed ILS methodology offers clear benefits on 2D grid maps, it is important to recognize its limitations and the scenarios in which it may not be suitable.

1) HIGH-DIMENSIONAL SPACES

The current ILS framework is designed for 2D grids and relies heavily on the geometric concept of line-of-sight in a planar space. Extending this approach to high-dimensional configuration spaces (e.g., a robotic manipulator operating in a 6D joint space or an autonomous drone navigating in continuous 3D space) presents significant challenges. In such scenarios, the number of potential line-of-sight checks increases combinatorially, and the definition of a ''grid'' may no longer be practical. Sampling-based algorithms like RRT* and PRM are typically preferred in these contexts due to their ability to handle continuous and high-dimensional planning spaces efficiently [27]. Applying ILS in such environments would require redefining the notion of visibility and introducing advanced collision-checking mechanisms, which could negate the performance advantages observed in 2D grids.

2) DYNAMIC AND UNCERTAIN ENVIRONMENTS

The ILS algorithm, in its current form, assumes a static environment during the search process. In dynamic scenarios where obstacles may move or change over time, ILS would need to be re-executed from scratch or integrated into an incremental re-planning framework. In contrast, algorithms like D* Lite are specifically designed for dynamic environments and can update existing paths efficiently when local changes occur [19]. A naive implementation of ILS may incur unnecessary recomputation if an obstacle appears along a previously computed line-of-sight path.

To improve applicability in such settings, future extensions could integrate ILS with dynamic planners to allow partial re-evaluation of the corridor or maintain tree-based memory for incremental updates. Additionally, reactive layers or learned controllers may be necessary to ensure safety and adaptability in unpredictable conditions, albeit at the cost of strict path optimality [28].

In summary, while ILS demonstrates strong performance in static 2D grid environments, its limitations in high-dimensional, dynamic, or size-constrained scenarios must be considered.

By addressing these limitations, we ensure that ILS is applied effectively within its domain of strength and extended thoughtfully for broader use cases.

VI. CONCLUSION AND FUTURE WORK

This section summarizes the key findings of the study and outlines directions for future research. It begins by revisiting the primary outcomes and contributions of the proposed Incremental Line Search (ILS) methodology. It then discusses potential enhancements and extensions to the ILS framework, particularly its application to more complex, high-dimensional, and dynamic environments.

A. CONCLUSION

This paper presented a comparative analysis of classical pathfinding algorithms, including A*, BFS, DFS, Best-First Search, and Dijkstra, alongside their adaptations using the Incremental Line Search (ILS) methodology. The results confirm that ILS significantly reduces computational overhead both in execution time and visited nodes while maintaining or improving path quality in many cases.

Across obstacle densities of 10%, 20%, and 30%, ILS achieved substantial reductions in execution time, with improvements of up to **95.52%** for Best-First Search and **94.81%** for A*. Visited nodes were reduced by an average of **73.7%** across all algorithms and scenarios, demonstrating the efficiency of the corridor-based pruning mechanism. Path length improvements were most pronounced in DFS and Best-First Search, with reductions of up to **93.74%** and **70.95%**, respectively, particularly in sparse environments where long, direct line-of-sight jumps are feasible.

These findings validate the robustness and generalizability of the ILS framework. Its lightweight integration, consistent performance gains, and minimal impact on path optimality make it highly suitable for real-time and resource-constrained applications in robotics, autonomous vehicles, and game development.

B. FUTURE WORK

The Incremental Line Search (ILS) methodology provides a foundation for further research and development in several key directions. One potential enhancement involves implementing dynamic corridor widths that adjust based on obstacle density and path complexity, potentially improving computational efficiency and robustness. Another promising area is extending the ILS approach to three-dimensional environments, such as 3D grids, enabling many applications for aerial robotics and autonomous drones.

Additionally, integrating machine learning techniques, such as reinforcement learning or neural networks, could allow for dynamic predictions of the most efficient corridor shape and size, further optimizing ILS's performance. Multi-agent pathfinding represents another exciting avenue where ILS could be adapted to handle scenarios involving multiple agents, requiring coordination and collision avoidance strategies.

Finally, real-world validation of ILS in scenarios such as urban navigation and warehouse robotics would be instrumental in assessing its practical applicability. These advancements could significantly enhance ILS's scalability, adaptability, and effectiveness, paving the way for more efficient and intelligent pathfinding systems.

REFERENCES

[1] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.

[2] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.

[3] A. Nash, K. Daniel, S. Koenig, and A. Feiner, "Theta: Any-angle path planning on grids," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2007, pp. 1177–1183.

[4] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 2, pp. 144–148, Jun. 2012.

[5] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robot. Auto. Syst.*, vol. 174, Apr. 2024, Art. no. 104630.

[6] S. Ghambari, M. Golabi, L. Jourdan, J. Lepagnot, and L. Idoumghar, "UAV path planning techniques: A survey," *RAIRO-Operations Res.*, vol. 58, no. 4, pp. 2951–2989, Jul. 2024.

[7] R. Singh, J. Ren, and X. Lin, "A review of deep reinforcement learning algorithms for mobile robot path planning," *Vehicles*, vol. 5, no. 4, pp. 1423–1451, Oct. 2023.

[8] H. Han, J. Wang, L. Kuang, X. Han, and H. Xue, "Improved robot path planning method based on deep reinforcement learning," *Sensors*, vol. 23, no. 12, p. 5622, Jun. 2023.

[9] Y. Zhong and Y. Wang, "Cross-regional path planning based on improved Q-learning with dynamic exploration factor and heuristic reward value," *Expert Syst. Appl.*, vol. 260, Jan. 2025, Art. no. 125388.

[10] T. Ma, J. Lyu, J. Yang, R. Xi, Y. Li, J. An, and C. Li, "CLSQL: Improved Q-Learning algorithm based on continuous local search policy for mobile robot path planning," *Sensors*, vol. 22, no. 15, p. 5910, Aug. 2022.

[11] Z. Shi, K. Wang, and J. Zhang, "Improved reinforcement learning path planning algorithm integrating prior knowledge," *PLoS ONE*, vol. 18, no. 5, May 2023, Art. no. e0284942.

[12] E. W. Dijkstra, "A note two problems connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 1959, doi: 10.1007/bf01386390.

[13] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artif. Intell.*, vol. 1, nos. 3–4, pp. 193–204, Jan. 1970, doi: 10.1016/0004-3702(70)90007-X.

[14] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta: any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, pp. 533–579, Feb. doi: 10.1613/jair.2994.

[15] D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 1–15. [Online]. Available: https://www.aaai.org

[16] 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. SSC-4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[17] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta: Any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, pp. 533–579, Oct. 2010.
[18] D. Harabor and A. Botea, "Hierarchical path planning for multi-size agents in heterogeneous environments," in *Proc. IEEE Symp. Comput. Intell. Games*, Dec. 2008, pp. 258–265, doi: 10.1109/CIG.2008.5035648.
[19] S. Koenig and M. Likhachev, "D Lite," in *Proc. AAAI Conf. Artif. Intell.*, 2002, pp. 476–483.
[20] X. Zhong, J. Tian, H. Hu, and X. Peng, "Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment," *J. Intell. Robotic Syst.*, vol. 99, no. 1, pp. 65–77, Jul. 2020, doi: 10.1007/s10846-019-01112-z.
[21] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations," *Ocean Eng.*, vol. 223, Mar. 2021, Art. no. 108709, doi: 10.1016/j.oceaneng.2021.108709.
[22] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based UAV path planning," *Knowl.-Based Syst.*, vol. 158, pp. 54–64, Oct. 2018, doi: 10.1016/j.knosys.2018.05.033.
[23] B. Xing, M. Yu, Z. Liu, Y. Tan, Y. Sun, and B. Li., "A review of path planning for unmanned surface vehicles," *J. Mar. Sci. Eng.*, vol. 11, no. 8, p. 1556, 2023. [Online]. Available: https://www.mdpi.com/2077-1312/11/8/1556
[24] K. Zheng, "Autonomous obstacle avoidance and trajectory planning for mobile robot based on dual-loop trajectory tracking control and improved artificial potential field method," *Actuators*, vol. 13, no. 1, p. 37, Jan. 2024. [Online]. Available: https://www.mdpi.com/2076-0825/13/1/37
[25] R. C. Holte, T. Mkadmi, R. M. Zimmer, and A. J. MacDonald, "Speeding up problem solving by abstraction: A graph oriented approach," *Artif. Intell.*, vol. 85, nos. 1–2, pp. 321–361, Aug. 1996, doi: 10.1016/0004-3702(95)00111-5.
[26] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artif. Intell.*, vol. 27, no. 1, pp. 97–109, Sep. 1985, doi: 10.1016/0004-3702(85)90084-0.
[27] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/
[28] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Proc. Robot., Sci. Syst.*, Jul. 2017, pp. 1–15.
[29] W. Ma, D. Wu, Y. Jiang, and H. Guo, "CLSQL: A Q-learning-based continuous local search algorithm for path planning," *Appl. Sci.*, vol. 12, no. 7, p. 3594, 2022, doi: 10.3390/app12073594.

AMR ELSHAHED received the B.Sc. degree in mathematics from the Faculty of Science and Arts, Qassim University, Saudi Arabia, and the M.Sc. degree in mathematics from the School of Mathematical Sciences, Universiti Sains Malaysia, where he is currently pursuing the Ph.D. degree. His research focuses on pathfinding algorithms, with an emphasis on developing more efficient methods for determining the shortest path on grid maps derived from satellite images.

MAJID KHAN BIN MAJAHAR ALI received the B.Sc. degree in mathematics with economics and the Ph.D. degree in mathematics from Universiti Malaysia Sabah. In May, he joined the School of Mathematical Sciences, Universiti Sains Malaysia (USM), as a Lecturer specializing in operational research. He currently serves as a Senior Lecturer and an Appointed Fellow with USM, focusing on seaweed cultivation, solar drying systems, processing, modeling, and simulation. His research applies the IoT, big data, and simulation methods to improve model predictions of moisture losses during drying in controlled and uncontrolled environments. His work extends to modeling challenges in engineering and biological systems, such as tissue culture and aquamarine studies. To address these problems, he employs machine learning using eight selection criteria approaches and robustness regression techniques to overcome multicollinearity and outlier issues. His research bridges theoretical statistical methods with practical applications, transforming mathematical models into impactful industrial solutions.

AHMAD SUFRIL AZLAN MOHAMED received the B.I.T. degree (Hons.) from Multimedia University, Malaysia, the M.Sc. degree from The University of Manchester, U.K., and the Ph.D. degree from the University of Salford, U.K. He is currently with the School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang, Malaysia. His research interests include image processing, video tracking, facial recognition, and medical imaging.

FARAH AINI BINTI ABDULLAH received B.Sc. and M.Sc degrees in mathematics and information technology from Universiti Sains Malaysia (USM), the Ph.D. degree in computational mathematics from The University of Queensland. Since 2009, she has been with the School of Mathematics, USM, focusing on computational mathematics, computational biology, and fractional differential modeling. Her research interests include addresses fractional dynamical systems in biological and environmental contexts, emphasizing environmental modeling, and statistical analysis.

TS. LEE JIAN AUN Co-founded LeadAlways Technology (M) Sdn Bhd as a Technology Arm of LeadAlways Group of companies. LATECH specialized in software and enterprise solutions servicing more than 100 SME, MNC, FI, Bhd and Public Sector. Transitioning into an AI and IOT oriented segment, LATECH is ready to bring advance and innovative solution to tackle industrial challenges with deep tech. LATECH have also recently partnered with Springtech Ventures Sdn Bhd an AI company to further advance the Research and Development of AI and deep tech technologies with joining as part of the boards of directors of Springtech Ventures.