

## Review

# A Comprehensive Review of Improved A\* Path Planning Algorithms and Their Hybrid Integrations

Doan Thanh Xuan <sup>\*</sup>, Nguyen Thanh Hung  and Vu Toan Thang 

School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam;  
hung.nguyenthanh@hust.edu.vn (N.T.H.); thang.vutoan@hust.edu.vn (V.T.T.)

<sup>\*</sup> Correspondence: xuan.doanthanh@hust.edu.vn

## Abstract

The A\* algorithm is a cornerstone in mobile robot navigation. However, the traditional A\* suffers from key limitations such as poor path smoothness, lack of adaptability to dynamic environments, and high computational costs in large-scale maps. This review presents a comprehensive analysis of 20 recent studies (2020–2025) on improved A\* variants and their hybrid integrations with complementary algorithms. The improvements are categorized into two core strategies: (i) geometric and structural optimization, heuristic weighting and adaptive search schemes in A\* algorithm, and (ii) hybrid models combining A\* with local planners such as Dynamic Window Approach (DWA), Artificial Potential Field (APF), and Particle Swarm Optimization (PSO). For each group, the mathematical formulations of evaluation functions, smoothing techniques, and constraint handling mechanisms are detailed. Notably, hybrid frameworks demonstrate improved robustness in dynamic or partially known environments by leveraging A\* for global optimality and local planners for real-time adaptability. Case studies with simulated grid maps and benchmark scenarios show that even marginal improvements in path length can coincide with substantial gains in safety and directional stability. This review not only synthesizes the state of the art in A\*-based planning but also outlines design principles for building intelligent, adaptive, and computationally efficient navigation systems.



Academic Editor:  
Farrokh Janabi-Sharifi

Received: 28 July 2025

Revised: 29 September 2025

Accepted: 4 October 2025

Published: 7 October 2025

**Citation:** Xuan, D.T.; Hung, N.T.; Thang, V.T. A Comprehensive Review of Improved A\* Path Planning Algorithms and Their Hybrid Integrations. *Automation* **2025**, *6*, 52. <https://doi.org/10.3390/automation6040052>

**Copyright:** © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Keywords:** A\* algorithm; path planning; mobile robotics; heuristic optimization; hybrid navigation systems; obstacle avoidance

## 1. Introduction

Autonomous path planning has been a cornerstone of mobile robotics research for over three decades, reflecting the growing complexity of dynamic environments and the need for intelligent, real-time decision-making. The foundational work by Kosaka & Kak (1992) [1] marked an early attempt to integrate vision and probabilistic models via the FINALE (Fast Indoor Navigation Allowing for Locational Errors) system, leveraging Kalman filters for indoor navigation under uncertainty. This seminal research laid the groundwork for later developments in map-based navigation and uncertainty handling. During the following decades, advances in localization, mapping, and real-time perception significantly shaped the path planning landscape. Patle et al. (2019) [2] provided a comprehensive survey of path planning strategies, broadly classifying them into global and local methods, as well as static and dynamic planning. Their work emphasizes the rise in intelligent and bio-inspired approaches—such as genetic algorithm (GA), particle swarm optimization (PSO), colony optimization (ACO), and artificial bee colony (ABC)—for effective obstacle

avoidance and trajectory generation in both static and uncertain environments. As robots became more autonomous, there was a pressing need to process environmental information more effectively and at higher speeds.

The evolution of robotic platforms also contributed to this progress. Taheri and Zhao. (2020) [3] reviewed omnidirectional mobile robot systems that are capable of holonomic motion, allowing them to navigate in narrow and complex spaces with enhanced maneuverability and flexibility. In parallel, several intelligent decision-making frameworks have emerged, combining fuzzy logic, artificial neural networks, and optimization techniques to improve adaptiveness in robotics and control systems, as shown by Cebollada et al. (2021) [4].

Real-world deployment scenarios such as intralogistics demanded new path planning paradigms that could operate under spatial and temporal constraints. Fracapane et al. (2021) [5] discussed the role of agent-based simulations and decentralized control in warehouse intralogistics, where autonomous mobile robots interact and negotiate with each other to optimize task allocation, reflecting broader concerns for scalability and robustness in dynamic multi-agent environments. Simultaneously, robust localization remained a prerequisite for accurate path execution. Panigrahi & Bisoy (2022) [6] categorized localization algorithms based on initial position awareness, and reviewed key methods including probabilistic approaches, simultaneous localization and mapping (SLAM), and AI-based techniques such as fuzzy logic, PSO, and neural networks. They emphasized the role of sensor fusion and feature-based mapping in reducing pose uncertainty and enhancing localization accuracy.

As robotic autonomy advanced, research began focusing on adaptable navigation systems capable of functioning in complex or partially known environments. Liu et al. (2023) [7] discussed a hybrid topological–metric mapping framework based on Mixed Representation (MR), which integrates the global efficiency of topological planning with the local precision of metric modeling. Their layered navigation strategy—employing global topological maps alongside subregion-based metric detail—enables scalable and flexible path planning without requiring exhaustive prior environment modeling. Simultaneously, Singh et al. (2023) [8] reviewed the transformative role of deep reinforcement learning (DRL) in mobile robot navigation. DRL enables robots to learn navigation strategies through interaction and feedback, often outperforming traditional planners in dynamic, unstructured environments. Further consolidating this shift, Loganathan & Ahmad (2023) [9] reviewed the integration of advanced sensing, SLAM, and high-definition mapping technologies to construct autonomous navigation pipelines. They emphasized the need to integrate global and local path planning modules, enabling robots to achieve both optimal route planning and real-time obstacle avoidance—an essential feature in dynamic or unpredictable environments.

The frontier of robotic navigation now includes emerging domains such as construction robotics, where planners must dynamically adapt to evolving physical environments. Zhao et al. (2025) [10] proposed a structured, three-phase planning framework—comprising construction task sequencing, element transit planning, and joint trajectory refinement—to coordinate high-level symbolic reasoning with low-level motion execution in autonomous in situ construction. Taleb et al. (2025) [11] emphasized the integration of mapping, localization, and path planning as a unified triad. Their review explores SLAM, semantic-level perception, and learning-enhanced planning algorithms as core enablers of intelligent robot navigation, particularly in complex and dynamic environments such as hospitals or emergency zones.

The A\* algorithm occupies a central position in path planning research, combining the accumulated cost function of Dijkstra's algorithm with the heuristic-driven exploration of

Best-First search. This hybrid formulation allows A\* to achieve deterministic optimality under admissible heuristics while maintaining superior computational efficiency compared with Dijkstra, which, though optimal, suffers from exhaustive node expansion in large or dense graphs. In contrast, Best-First search offers faster convergence by following heuristic estimates alone, yet frequently sacrifices path quality and reliability in cluttered environments. A\* therefore provides a balanced compromise, guiding the search efficiently toward the goal while preserving optimality. When contrasted with the Probabilistic Roadmap Method (PRM), A\* represents a fundamentally different paradigm. PRM is designed for high-dimensional or continuous spaces, where it constructs a roadmap through random sampling and connection of collision-free nodes. While PRM offers probabilistic completeness and improved scalability for multi-degree-of-freedom robotic systems, it lacks the deterministic guarantees of A\* and incurs significant preprocessing costs in dynamic environments. Overall, A\* is best suited for structured, low-dimensional domains, whereas Dijkstra, Best-First, and PRM retain situational relevance depending on the requirements of speed, and scalability.

The A\* search algorithm is a widely used and efficient pathfinding technique [12]. Within the framework of the A\* pathfinding algorithm, the evaluation function is explicitly defined in mathematical form as  $f(n) = g(n) + h(n)$  [13], where  $f(n)$  represents the total estimated cost of traveling from the start node to the goal node through the intermediate node  $n$ . The term  $g(n)$  denotes the actual accumulated cost from the initial node  $n_0$  to the current node  $n$ , calculated as

$$g(n) = \sum_{k=1}^K c(n_{k-1}, n_k) \quad (1)$$

where  $c(n_{k-1}, n_k)$  is the movement cost between two consecutive nodes along the known path. The term  $h(n)$  serves as the heuristic function, providing an estimate of the remaining cost from node  $n$  to the goal node  $n_{\text{goal}}$ , typically modeled as the Euclidean distance,

$$h(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} \quad (2)$$

or the Manhattan distance  $|x_n - x_g| + |y_n - y_g|$  among other metrics [14], depending on the characteristics of the search space. This two-component structure allows A\* to balance the accuracy of the known cost with the ability to predict the remaining cost, thereby improving search efficiency and preserving the optimality of the path compared with traditional exhaustive methods such as Dijkstra's algorithm.

Choosing an appropriate heuristic is critical [15]. A heuristic is admissible if it never overestimates the true cost and consistent if it obeys the triangle inequality. Common heuristics include Manhattan distance for grid maps or Euclidean distance for continuous spaces. A\* guarantees optimality only when the heuristic is admissible. Implementation details, such as tie-breaking, grid connectivity (4-way, 8-way), and data structures (priority queues), significantly affect performance. Amit Patel suggests breaking ties in favor of nodes with higher  $g(n)$  to encourage smoother, less "zigzag" paths [16].

In heuristic-based path planning, the choice of evaluation metric critically influences the efficiency and quality of the A\* algorithm. The Manhattan distance, defined as the sum of absolute coordinate differences, is particularly suitable for four-connected grid environments where movement is constrained to orthogonal directions. Its computational simplicity is advantageous, though it often underestimates true path cost in scenarios involving diagonal movement. The Euclidean distance, by contrast, calculates the straight-line distance between two points, providing an admissible heuristic that better reflects continuous motion. However, in grid-based applications, Euclidean estimation can overemphasize diagonal proximity, leading to expanded nodes that do not align well with the

allowed movement set. The octile distance addresses this limitation by integrating both orthogonal and diagonal movements into its formulation, assigning a weight of  $\sqrt{2}$  for diagonals and 1 for straight moves. This produces a more accurate and admissible heuristic for eight-connected grids, striking a balance between accuracy and computational cost. Finally, the Chebyshev distance measures the maximum absolute difference across coordinates, effectively assuming unrestricted diagonal and orthogonal movement at equal cost. While computationally efficient, it may over-simplify environments with anisotropic motion constraints. Collectively, these metrics highlight the necessity of aligning heuristic design with the robot's mobility model and grid topology to ensure both admissibility and search efficiency.

Path planning remains a core problem in mobile robotics, particularly in dynamic and partially observable environments. Traditional algorithms such as A\* are widely adopted for global path planning in grid-based representations [17]. Despite its widespread use, the traditional A\* algorithm exhibits several limitations that hinder its application in real-world robotic systems. Firstly, A\* does not account for the kinematic or dynamic constraints of vehicles—it only computes geometrically valid paths without considering turning radii, speed limits, or feasible angular transitions. As a result, the generated paths may include sharp, impractical turns that are infeasible for wheeled robots [18]. Secondly, A\* often produces jagged or sawtooth-like trajectories due to grid discretization, which reduces motion smoothness and increases friction, energy consumption, and mechanical wear [19]. Third, A\* focuses solely on minimizing distance to the goal and lacks obstacle avoidance buffers, causing paths to pass dangerously close to obstacles—an issue under positional uncertainty [20]. Moreover, in large or complex environments, A\* suffers from poor scalability, exploring many unnecessary nodes and consuming excessive memory and computation time. Its reliance on uniform grid structures also limits resolution and can result in “staircase” effects when using 4- or 8-connected grids [21]. Additionally, A\* lacks learning capability—it re-computes paths from scratch even for repetitive tasks in the same environment [22]. Finally, A\* is inherently an offline planner; it does not adapt to environmental changes, making it unsuitable for dynamic or partially observable settings unless combined with reactive or incremental methods [23].

In a 2024 review, Xu et al. [24] conducted a performance-oriented assessment of improved A\* variants for robotic path planning, notably introducing the SBREA (Slide-Rail corner adjustment and Bi-directional Rectangular Expansion A\*) algorithm. This method effectively enhanced path smoothness and reduced search redundancy by incorporating bidirectional rectangular expansion and an adaptive corner adjustment mechanism. Despite its contributions, the review lacked a systematic cross-comparison among studies, did not address hybrid algorithmic integrations, and overlooked the applicability of these methods in dynamic or large-scale environments.

To address these limitations, the present study provides a comprehensive and structured synthesis of 22 peer-reviewed articles, selected from an initial pool of 50 publications retrieved from the databases Scopus and Web of Science using targeted keywords such as “A\* algorithm,” “improved A\*,” “enhanced A\*,” “hybrid A\*,” “mixed A\*,” and “combined A\*.” The selected papers, published between 2020 and 2025, were classified into two principal categories. The first batch of 10 papers comprises A\* enhancements focusing on structural and geometric optimizations, heuristic weighting schemes, and adaptive search strategies. The second batch of 12 papers involves hybrid A\* algorithms that integrate A\* with complementary techniques such as Dynamic Window Approach (DWA), Artificial Potential Field (APF), and Particle Swarm Optimization (PSO), etc.

Our review further introduces comparative tables presenting quantitative performance indicators—such as percentage reductions in path length, number of expanded nodes, and

angular deviation—to support a rigorous performance analysis. Mathematical formulations of advanced heuristic evaluation functions and kinetic feasibility constraints are also analyzed to underscore practical applications in autonomous mobile robot navigation under dynamic and complex scenarios. Therefore, this work significantly extends existing literature by offering an application-driven, analytically grounded, and industry-relevant synthesis of recent advancements in A\*-based path planning.

## 2. Improved A\* Algorithm

Currently, classical path-planning approaches such as the traditional A\* algorithm [25], Rapidly exploring Random Tree (RRT) [26], and the artificial potential field method [27], etc., can be integrated to develop an improved A\* algorithm that exploits the strengths of each technique while mitigating their individual limitations.

To begin with, we reviewed the first ten studies focusing on improved A\* algorithms. Table 1 presents a chronological list of these studies, highlighting the percentage improvements achieved by each modified algorithm compared to the traditional A\* algorithm. The comparison is based on four key performance indicators, which correspond to the major limitations of the original A\* algorithm: the number of expanded nodes, total path length, processing time, and path angularity. These metrics provide a comprehensive assessment of enhancements in terms of computational efficiency, trajectory optimality, and smoothness.

**Table 1.** Comparative performance of 10 improved A\* algorithms versus the traditional A\*.

| Paper ID | Ref. | The Number of Expanded Nodes | Path Length  | Processing Time                                           | Path Angularity                                        | Note                                                                                                                                    |
|----------|------|------------------------------|--------------|-----------------------------------------------------------|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 1.1      | [28] | ☑ −95.76%                    | ☑ −0.15%     | ☑ −2.0%                                                   | ✗ unspecified                                          | Results of the field implementation experiment on the outdoor autonomous robot.                                                         |
| 1.2      | [29] | ✗ unspecified                | ☑ −1% to −2% | ✗ Processing time doubled due to computational complexity | ☑ −71% to −72%                                         | Simulation results on $20 \times 20$ and $40 \times 40$ grid maps.                                                                      |
| 1.3      | [30] | ☑ −65%                       | ☑ −11.41%    | ☑ −74.38%                                                 | ✗ unspecified                                          | Simulation results on the $40 \times 40$ grid map.                                                                                      |
| 1.4      | [31] | ☑ −62.52%                    | ☑ −13.13%    | ☑ −82.21%                                                 | ☑ −50%                                                 | Average simulation results on randomized $200 \times 200$ grid maps.                                                                    |
| 1.5      | [32] | ☑ −83.45%                    | ☑ −1.58%     | ☑ −99.8%<br>After the first execution of the algorithm    | ✗ unspecified                                          | Average results of four simulation experiments with varying start/end coordinates and obstacle configurations.                          |
| 1.6      | [33] | ☑ −71.17%                    | ☑ −3.13%     | ☑ −49.9%                                                  | ☑ −35.12%                                              | Average results of three experiments with different start and goal coordinates on the same obstacle layout in $80 \times 80$ grid maps. |
| 1.7      | [34] | ☑ −86.57%                    | ✗ unchanged  | ☑ −82.07%                                                 | ☑ The application of the Bézier curve; not quantified. | Simulation results on the $120 \times 120$ grid map.                                                                                    |

Table 1. Cont.

| Paper ID | Ref. | The Number of Expanded Nodes | Path Length | Processing Time | Path Angularity                                           | Note                                                                                                                                                                                                                                                                                                    |
|----------|------|------------------------------|-------------|-----------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.8      | [35] | ☑ −52%                       | ☑ −3.7%     | ☑ −49.4%        | ☑<br>The use of Cubic B-spline technique; not quantified. | Average value computed from two experiments with different start and goal coordinates but identical obstacle configuration, conducted on $30 \times 30$ grid maps.                                                                                                                                      |
| 1.9      | [36] | ☑ −13.4%                     | ✗ +21.9%    | ☑ −88.2%        | ✗ unspecified                                             | In this test, the size of the simulated terrain area is $2000 \times 1600$ . The results were compared with the traditional A algorithm (16 directions). The path length obtained using the improved A algorithm was longer than that of the traditional A* due to its preference for smoother terrain. |
| 1.10     | [37] | ☑ −48.31%                    | ☑ −6.6%     | ☑ −20.00%       | ☑ −1.86%                                                  | Simulation experiment on a $30 \times 30$ grid map.                                                                                                                                                                                                                                                     |

The evaluation function plays a pivotal role in the A\* algorithm, and its formulation often reflects the core improvements proposed in a given study. Analyzing the evaluation function thus provides valuable insights into the optimization strategies employed. Table 2 summarizes and compares the evaluation functions adopted in the selected ten studies, along with brief explanations of how each modification deviates from the traditional A\* formulation and contributes to performance enhancement.

The ten selected studies were categorized into four methodological groups. **Group 1** (IDs 1.4 and 1.5) retains the traditional A\* evaluation function; enhancements focus on other techniques rather than modifying the function itself, as detailed in later sections. **Group 2** (IDs 1.1, 1.3, 1.6, and 1.8) introduces a “guideline” mechanism to assist directional planning, which effectively reduces both the number of traversed nodes and processing time. **Group 3** (IDs 1.7, 1.9, and 1.10) involves modifications to the weighting parameters in the heuristic component  $h(n)$  to enhance performance in complex environments. **Group 4** comprises a single study (ID 1.2) that directly integrates the Artificial Potential Field (APF) method into the A\* evaluation function. In this method, APF serves as an enhancement within the A\* framework, distinguishing it from the hybrid approach discussed in Section 2, where A\* handles global path planning and APF performs local optimization. This layered combination aims to balance global optimality with real-time obstacle avoidance.

### 2.1. Group 1

Both studies (IDs 1.4 and 1.5) published in 2022 aim to overcome key limitations of the traditional A\* algorithm, including inefficiency in large-scale environments and the generation of non-smooth, suboptimal paths. Although they share a common objective—improving path planning for mobile robots—they differ in methodology, target application, and evaluation criteria.

**Table 2.** Evaluation functions of 10 improved A\* algorithms.

| Paper ID | Evaluation Function                             | Formula Detail                                                                                                                                                                                                                                                                                                       | Explanation                                                                                                                                                             |
|----------|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.1      | $f(n) = g(n) + h_1(n)\alpha_1 + h_2(n)\alpha_2$ | $f(n)$ : cost accumulated to reach point $n$ .<br>$h_1(n)$ : distance from the current point to the guideline.<br>$h_2(n)$ : distance from the nearest point on the guideline to the goal.<br>$\alpha_1, \alpha_2$ : weighting coefficients.                                                                         | Guideline, which was generated by a global planner or by humans. The guideline is employed to develop the heuristic function to express the driver's intention.         |
| 1.2      | $f(n) = g(n) + h'(n) + v(n)$                    | $h'(n)$ : distance heuristic, using diagonal distance.<br>$v(n)$ : potential field.                                                                                                                                                                                                                                  | Integrates artificial potential field method to include obstacle proximity and goal attraction.                                                                         |
| 1.3      | $f(n) = g(n) + h(n) + C(n)$                     | $C(n)$ : vertical distance from node to guideline.                                                                                                                                                                                                                                                                   | The guideline serves to orient the search along the axis connecting the start and goal points, thereby reducing unnecessary search areas.                               |
| 1.4      | $f(n) = g(n) + h(n)$                            |                                                                                                                                                                                                                                                                                                                      | No change compared to the traditional A algorithm.                                                                                                                      |
| 1.5      | $f(n) = g(n) + h(n)$                            |                                                                                                                                                                                                                                                                                                                      | No change compared to the traditional A algorithm.                                                                                                                      |
| 1.6      | $f(n) = g(n) + h(n) + \lambda \times d(n)$      | $d(n)$ : vertical distance from node to line(start, goal).<br>$\lambda$ : tuning weight.                                                                                                                                                                                                                             | Keeps nodes close to main path axis, enhancing search efficiency and directionality.                                                                                    |
| 1.7      | $f(n) = g(n) + w \times h(n)$                   | $h(n)$ : weighted dynamically.                                                                                                                                                                                                                                                                                       | When the node is far from the goal, the weight $w$ is increased to prioritize search speed. As the node approaches the goal, $w$ is decreased to enhance path accuracy. |
| 1.8      | $f(n) = g(n) + h(n) + \mu \times c(n)$          | $c(n) = 3.14 - \text{angle}(v_{\text{start}}, v_{\text{goal}})$ , where<br>$v_{\text{start}}$ is the vector from the parent node to the current node, and $v_{\text{goal}}$ is the vector from the current node to the goal.<br>$\mu$ : the weighting coefficient for angle deviation.                               | Angle-based penalty function keeps node trajectory aligned to ideal direction.                                                                                          |
| 1.9      | $f(n) = g(n) + \alpha h(n)W$                    | $\alpha$ : the weight coefficient (adjusted based on experimental requirements)<br>$W$ (weight for road factors): a penalty factor that reflects environmental conditions such as soil type, slope, or terrain resistance, which affects the traversability of unpaved paths (e.g., muddy, rocky, or sandy surfaces) | Accounts for soil/slope/terrain resistance in $h(n)$ to favor accessible paths.                                                                                         |
| 1.10     | $f(n) = g(n) + [(R + r)/R] \times h(n)$         | $R$ : distance(start, goal).<br>$r$ : distance(current, goal)                                                                                                                                                                                                                                                        | Dynamically adjusts $h(n)$ weight based on proximity to goal—larger weight when far.                                                                                    |

Wang et al. [31] proposed the EBS-A\* algorithm, integrating three main strategies: expansion distance, bidirectional search, and path smoothing. The expansion distance defines a safety buffer by marking adjacent nodes near obstacles as non-traversable, thus reducing collision risk and search space (Equation (3)). The buffer radius depends on robot velocity ( $V_i$ ), threshold velocity ( $V_r$ ), and safety radius ( $r$ ).

$$E(V_i) = \begin{cases} r & \text{if } V_i \leq V_r \\ \frac{V_i}{V_r} \cdot r & \text{if } V_i < V_r \leq V_{\max} \end{cases} \quad (3)$$

where  $V_i$  is the current velocity,  $V_r$  is the threshold velocity, and  $r$  is the robot's safety radius.

Bidirectional search simultaneously grows the tree from start and goal nodes, reducing search depth and time complexity. Path smoothing is applied using Bezier curves and acute-angle decomposition, yielding continuous curvature and a maximum turning angle of  $45^\circ$ , thereby enhancing motion stability.

In contrast, Martins et al. [32] introduced the IMO A\* algorithm, which improves traditional A\* by optimizing four criteria: processing time, path length, smoothness, and number of random points. A major innovation is the pre-serialization of the obstacle map using a pickle file, allowing instant retrieval in subsequent runs and reducing average processing time by 99.98%. A path-problem-aware executor then removes redundant intermediate points, improving smoothness by minimizing angle deviation. IMO A demonstrated several average improvements, including a 1.58% reduction in path length, 83.45% fewer sampling points, and a smoother trajectory characterized by a turning angle of  $155^\circ$ , compared to  $135^\circ$  for the traditional A algorithm. In contrast, EBS-A\* does not rely on pre-stored maps but integrates both pre-processing and post-processing stages, thereby enhancing safety, reducing computation time, and producing a smoother path.

To enhance path quality, the authors introduced a path-problem-aware executor that evaluates the raw path and simplifies it by deleting unnecessary intermediate points. Given two path nodes  $P_i$  and  $P_{i+1}$ , the total path length is computed as:

$$\text{Path Length} = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (4)$$

Path smoothness is measured by minimizing deviations from  $180^\circ$ , using:

$$\text{Smoothness Penalty} = \sum_{i=1}^{n_g} (180^\circ - \alpha_i) \quad (5)$$

The algorithmic flowchart presented in paper ID 1.5 is illustrated in Figure 1.

The EBS-A\* algorithm enhances performance through three strategies: expansion distance, bidirectional search, and path smoothing. Expansion distance ensures a safety margin from obstacles, bidirectional search accelerates convergence, and smoothing eliminates sharp turns to provide more continuous and practical paths. These improvements make EBS-A\* particularly suitable for industrial and service robotics in structured environments such as warehouses, airports, and healthcare facilities, where safety and reliable navigation are prioritized. However, these very improvements introduce important constraints. First, the algorithm assumes a static environment with a raster-type map, making it difficult to maintain performance when the operating space changes continuously or when map data are provided only partially—situations commonly encountered in outdoor applications or complex industrial settings. Second, the selection of the expansion-distance parameter

depends directly on the robot's geometric dimensions and kinematic characteristics (such as diameter and cruising speed) as well as on sensor accuracy. If this parameter is not properly tuned, it can either waste maneuvering space or, conversely, fail to provide a sufficient safety "buffer zone," thereby reducing applicability in real operational contexts. Third, employing Bézier curves requires continuous computation and can increase the processing burden when applied to large maps or systems with limited computational resources. Consequently, EBS-A\* is particularly well suited to scenarios in known indoor environments where the map is relatively stable and where optimizing trajectory smoothness and speed yields direct benefits for operational productivity.

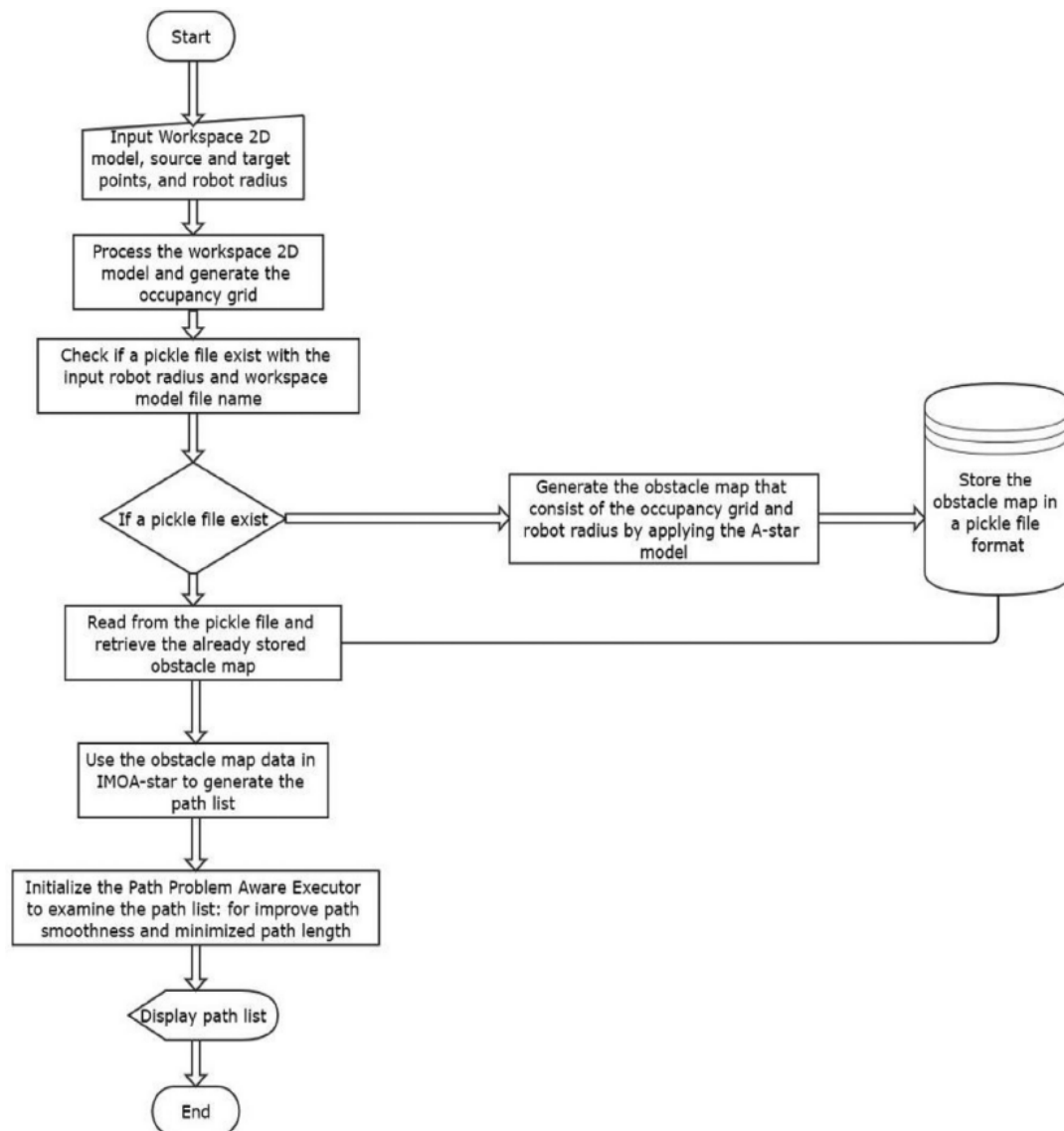


**Figure 1.** Flowchart for the IMOA-star proposed by Martins et al. [32].

In contrast, the IMOA\* algorithm introduces a multi-objective optimization approach aimed at reducing computational overhead in large workspaces. By storing obstacle maps and incorporating a path-problem-aware executor, it significantly improves planning speed, smoothness, and energy efficiency for non-holonomic mobile robots. This design is highly applicable to large-scale logistics and industrial environments where computational speed and scalability are essential. Nevertheless, this advantage comes with several noteworthy limitations. First, the initial setup phase—the creation of the pickle file—can take several hours for a large workspace, which is incompatible with the need for rapid

response in continuously changing environments. Moreover, IMO<sup>A</sup>\* also assumes static obstacles and a fixed spatial configuration; any map variation requires a complete regeneration of the database, undermining the method's time advantage. In addition, although the algorithm was demonstrated on powerful hardware (Intel Core i7), deploying it on resource-constrained computing devices (for example, low-cost robot embedded systems) may face challenges in terms of storage capacity and the bandwidth required to retrieve the pickle file.

In summary, EBS-A\* is tailored for dynamic local navigation with real-time adaptation, while IMO<sup>A</sup>\* targets large-scale environments requiring repeated queries and energy-efficient planning.

## 2.2. Group 2

All four studies in this group (IDs 1.1, 1.3, 1.6, and 1.8) incorporate a guideline—typically the axis between start and goal—to direct search trajectories and minimize unnecessary exploration. Each modifies the traditional A\* evaluation function by embedding application-specific terms into the heuristic or as a separate penalty.

Erkel et al. [28] proposed enhancements tailored for autonomous land vehicles, addressing A\*'s inefficiencies in handling curves, obstacle anticipation, and fixed step size. Their revised function is:

$$F(i) = G(i) + \alpha_1 H1(i) + \alpha_2 H2(i) \quad (6)$$

where  $G(i)$  is the cumulative cost,  $H1(i)$  is the distance from node  $i$  to a guideline (expressing driver intent), and  $H2(i)$  is the distance from the nearest point  $g(i)$  on the guideline to the goal. The weights  $\alpha_1$  and  $\alpha_2$  balance spatial alignment and goal-directed efficiency. In the original article, the author used the subscript  $i$ ; however, in Table 2, this has been replaced with  $n$  to ensure consistency with other published studies and to facilitate easier comparison. The meanings of these two indices are entirely equivalent. Figure 2 describes the physical meaning for  $H1$  and  $H2$ , as illustrated by Erke et al. [28].

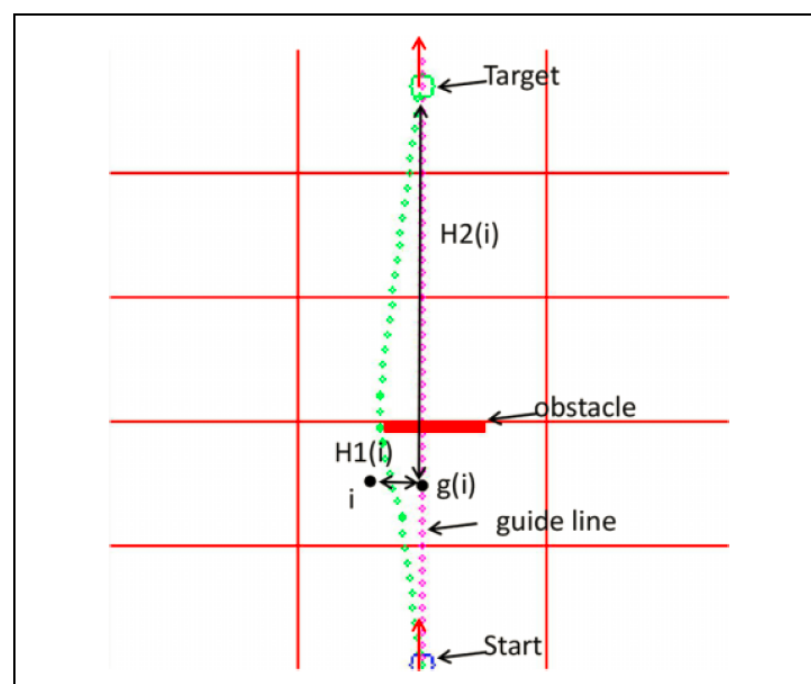


**Figure 2.** Physical meaning for  $H1$  and  $H2$  illustrated by Erke et al. [28].

To enhance obstacle avoidance, the algorithm identifies key points near blocked guidelines and regenerates sub-guidelines that circumvent the obstruction. The best candidate is chosen based on virtual-speed-based cost-time. Figure 3 shows the process of the key point-based A\* algorithm, as proposed by Erke et al. [28].

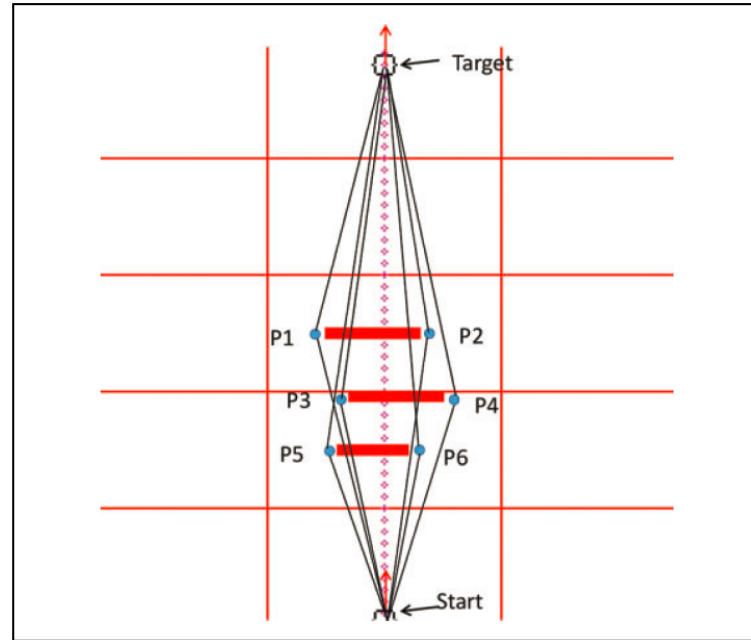


**Figure 3.** The process of the key point-based A\* algorithm proposed by Erke et al. [28].

Furthermore, a variable-step strategy adjusts search resolution: longer steps in open areas improve efficiency, while shorter steps near obstacles enhance safety. Combined, these techniques reduce node expansion and planning time, enabling robust real-time path planning in complex urban environments.

In contrast, XiangRong et al. [30] explicitly modify the evaluation function by appending a spatial constraint term  $C(n)$  to guide the search trajectory. Their cost function is  $f(n) = g(n) + h(n) + C(n)$ . The function  $C(n)$  quantifies the perpendicular distance between the current node  $n$  and a straight line—termed the “guide line”—connecting the start and goal positions. Mathematically, for a node  $n = (x_n, y_n)$ , start  $s = (x_s, y_s)$ , and goal  $g = (x_g, y_g)$ , the constraint is calculated as:

$$C(n) = \frac{|(x_g - x_s)(y_s - y_n) - (x_s - x_n)(y_g - y_s)|}{\sqrt{(x_g - x_s)^2 + (y_g - y_s)^2}} \quad (7)$$

Figure 4 illustrates  $C(n)$ , as defined by XiangRong et al. [30], representing a node’s deviation from the ideal trajectory. By penalizing nodes that deviate from the guide line, the algorithm minimizes unnecessary exploration, thereby reducing the search space and computational load. This constraint functions as a soft geometric prior, guiding the search toward globally aligned paths, even in complex obstacle environments.

Zhang et al. [33] propose a similar yet subtly different improvement for unmanned surface vehicles (USVs) operating in marine environments. Their evaluation function extends A\* by incorporating a linear deviation term  $d(n)$ , modulated by a tunable weight  $\lambda$ :  $f(n) = g(n) + h(n) + \lambda d(n)$ .

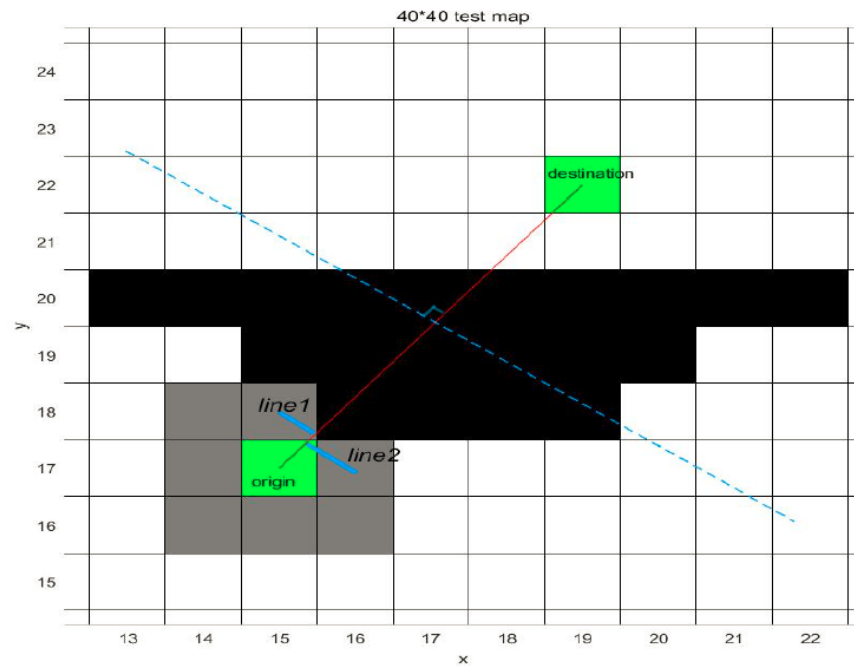


Figure 4.  $C(n)$  illustrated by XiangRong et al. [30].

Figure 5 shows the node vertical distance diagram, as proposed by Zhang et al. [33]. Unlike  $C(n)$ , which uses Euclidean distance,  $d(n)$  quantifies orthogonal deviation from a predefined navigation axis, measured relative to the grid. The weight  $\lambda$  adaptively controls the influence of this penalty, with higher values used in constrained environments such as narrow channels or dynamic obstacle zones. A directional smoothing filter is applied post-planning, highlighting that the evaluation function is part of a comprehensive trajectory optimization process.

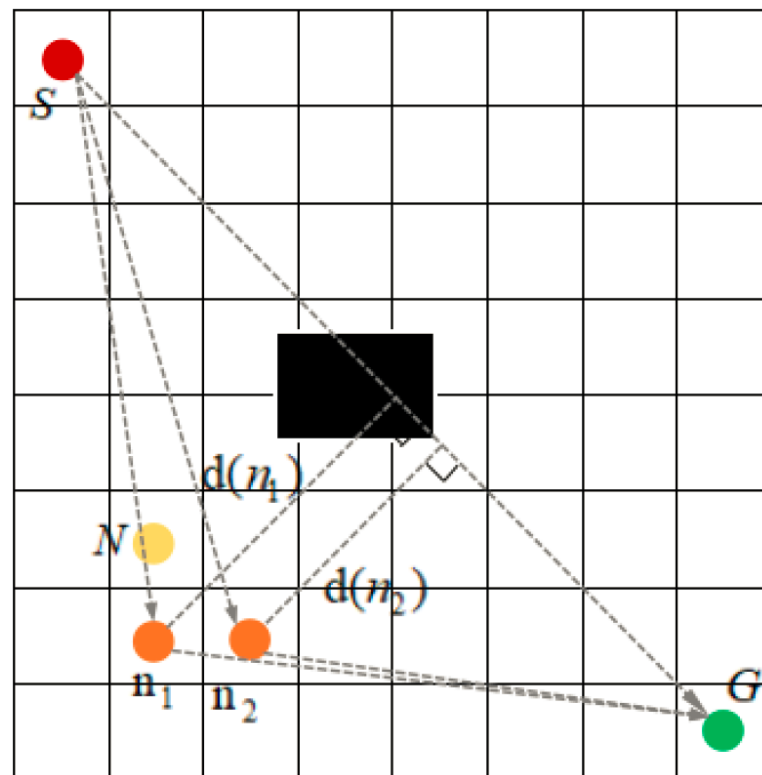


Figure 5. Diagram of node vertical distance suggested by Zhang et al. [33].

Yin et al. [35] proposed a mathematically rigorous enhancement by incorporating a directional angle cost  $c(n)$  into the evaluation function:

$$f(n) = g(n) + h(n) + \mu c(n) \quad (8)$$

where  $c(n) = \pi - \theta$ , and  $\theta$  is the angle between two vectors:  $\vec{v}_1$ , from node  $n$  to the start, and  $\vec{v}_2$ , from  $n$  to the goal. This formulation penalizes angular deviation, encouraging node selection aligned with the global path.

$$\theta = \arccos \left( \frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \|\vec{v}_2\|} \right), \quad c(n) = \pi - \theta \quad (9)$$

Figure 5 illustrates the node angle, as proposed by Yin et al. [30]. This formulation favors nodes aligned with the global path (i.e., when  $\theta \approx \pi$ ). The weight  $\mu$  adjusts the emphasis on angular consistency. This cost term proved effective in medical robotics, where smoother paths reduce mechanical strain and enhance the stability of sample transport. A cubic uniform B-spline smoothing stage is also applied to ensure physical feasibility for mobile platforms.

Study (ID 1.1) is applied in practice to autonomous land vehicles, enhancements such as guideline-based heuristics, key-point obstacle avoidance, and variable step sizes improve robustness and reduce computational cost, making the algorithm more adaptable to structured road environments. This approach helps stabilize the path and increase reliability, but it relies on the assumption of an available and trustworthy guideline—a condition difficult to achieve in dynamic environments or when map information is incomplete. In addition, selecting the optimal search-step parameter still requires experimentation and empirical tuning, thereby reducing the level of automation when scaling up. In static indoor settings, study (ID 1.3) reports improvements including bidirectional search, constraint functions, and guideline strategies effectively reduce memory consumption and limit unnecessary node expansion, enhancing efficiency for robots operating in constrained spaces. However, this method assumes that the straight line between the start and goal points is always geometrically valid and unobstructed by major obstacles. When the map has a complex layout or contains dynamic obstacles, the guide line may lose its validity, reducing the reliability of the solution and requiring the addition of a local replanning algorithm. In maritime applications, study (ID 1.6) applied the improved A\* for unmanned surface vehicles integrates bidirectional search and spline-based smoothing, reducing inflection points and producing more navigable paths under complex sea conditions, which are critical for environmental monitoring tasks. Nevertheless, this method still relies on the assumption of a static environment and a rasterized grid. For dynamic marine areas with waves, moving obstacles, or incomplete map information, the solution lacks a real-time adaptation mechanism; moreover, the grid pre-processing steps and B-spline smoothing further increase the computational burden on resource-constrained embedded systems. Similarly, study (ID 1.8), conducted in medical testing laboratories, employed an enhanced evaluation function, bidirectional strategy, and B-spline smoothing to enable mobile robots to navigate efficiently in highly structured yet cluttered environments, thereby ensuring stability and reducing redundant nodes. However, this method strongly depends on an accurate grid map and a stable laboratory spatial configuration; any changes in the setup or equipment layout require remapping and a complete recomputation, thereby reducing flexibility in real operational environments that change continuously.

Although significant progress has been made, several common limitations persist. Many methods depend on accurate prior maps and demonstrate reduced adaptability in

dynamic or rapidly changing environments. Moreover, path smoothing and optimization often increase computational complexity, limiting real-time responsiveness in large-scale scenarios. Overall, improved A\* variants have shown strong potential in logistics, health-care automation, and marine monitoring, and future work should integrate dynamic re-planning capabilities with multi-objective optimization to balance efficiency, safety, and adaptability in real-world applications.

These differences are reflected in the computational performance summarized in Table 1. Erkel et al.'s algorithm achieved a 95.76% reduction in computational load (1.426 vs. 33.654) [28]. XiangRong et al. [30] reported average reductions of 65% in search area and 74.38% in processing time across two experiments. Zhang et al.'s hybrid method reduced the number of nodes traversed by 71.17% on  $80 \times 80$  grid maps [33], while Yin et al. [35] showed a 52% reduction on  $30 \times 30$  grid maps. All Group 2 studies achieved over 50% improvement, confirming the strong efficiency gains in expanded nodes—a key factor in optimizing time and memory consumption.

### 2.3. Group 3

Group 3 also modifies the evaluation function of traditional A\*, but with a distinct focus on optimizing the weighting of the heuristic component  $h(n)$ . A comparative analysis of three recent algorithms [34,36,37] reveals shared strategies—such as guided search, dynamic weighting, and heuristic refinement—alongside key differences in how the heuristic term is adaptively tuned to improve path quality and computational efficiency.

All three aim to reduce redundant node expansion, enhance heuristic flexibility, and apply post-processing smoothing. They start from the traditional A\* formulation  $f(n) = g(n) + h(n)$ , using Euclidean distance as  $h(n)$  for greater accuracy in continuous-space environments, compared to Manhattan or diagonal metrics.

The first aspect analyzed is heuristic weight adaptation, which balances search speed and accuracy. This strategy dynamically adjusts the weight of the heuristic function  $h(n)$  during the search, allowing the algorithm to emphasize global exploration or local refinement depending on goal proximity and environmental complexity.

These algorithms incorporate dynamic heuristic weighting to adjust the influence of  $h(n)$  throughout the search. For instance, Han et al. [34] introduce a dynamic coefficient  $W$  in the evaluation function:  $f(n) = g(n) + W \times h(n)$ .

In Zhang & Zhang [37], a more structured weighting scheme is introduced:

$$f(n) = g(n) + \left( \frac{R+r}{r} \right) \cdot h(n) \quad (10)$$

where  $R$  is the total distance between the start and goal, and  $r$  is the distance from the current node to the goal. This formulation prioritizes speed in the early search phase and accuracy in later phases.

Han et al. [34] employ static weights for  $W$ , manually tuned based on goal distance (see their Table 2). In contrast, Zhang & Zhang [37] implement real-time heuristic modulation via the ratio  $(R+r)/R$ , enabling automatic, environment-sensitive adaptation.

Zhao et al. [36] introduced vector-based angular filtering, which restricts node expansion to directions aligned with the optimal search vector. Rather than evaluating 8, 16, or 24 neighbors, the algorithm limits the search to a six-directional neighborhood, defined as:  $\theta \in \{[0^\circ, 22.5^\circ] \cup (337.5^\circ, 360^\circ]; (22.5^\circ, 90^\circ]; (90^\circ, 157.5^\circ]; (157.5^\circ, 202.5^\circ]; (202.5^\circ, 270^\circ]; (270^\circ, 337.5^\circ]\}$

This directional constraint greatly reduces unnecessary lateral or regressive node expansions, cutting down on redundant processing without sacrificing path feasibility.

Secondly, we analyze differences in bidirectional search strategies. Unlike the traditional A\*, which performs a unidirectional forward search, improved variants launch simultaneous searches from both the start and goal nodes. These converge mid-way, significantly reducing node expansion and computation time. The effectiveness of such strategies depends on how the meeting point is defined and how partial paths are merged, with implementations varying across studies.

Among the three, only Zhang & Zhang [37] explicitly implement bidirectional A\*, growing search trees from both start and goal nodes.

Thirdly, we examine differences in path smoothing. While all three algorithms apply post-processing to improve trajectory feasibility and motion stability, they adopt distinct mathematical approaches. These include strategies for reducing sharp turns, removing redundant waypoints, and ensuring curvature continuity. The next subsection outlines each method—such as Bezier curve fitting, angle-threshold filtering, and minimum snap optimization—highlighting their trade-offs in smoothness, computational cost, and real-time applicability.

Zhao et al. [36] use second-order Bezier curves to connect segments of the raw path. The curve is defined by:

$$B(t) = (1 - t)^2 P_0 + 2t(1 - t)P_1 + t^2 P_2, \quad t \in [0, 1] \quad (11)$$

where  $P_0, P_1, P_2$  are three sequential control points. This interpolation smooths sharp corners and provides continuous first- and second-order derivatives (velocity and acceleration), which are critical for real-time motion execution.

Instead of curve fitting, Zhang & Zhang [37] employ an Inflection Point Elimination method using parent–grandparent filtering. This approach analyzes angular deviation among a node, its parent, and grandparent to identify and remove unnecessary inflection points. If a direct, obstacle-free line exists from  $P_i$  to  $P_{i-2}$ , the intermediate point  $P_{i-1}$  is eliminated. Iterative application of this rule yields a smoother, more direct path while preserving obstacle avoidance and geometric structure.

Han et al. [34] introduced a Grouping-Based Node Consolidation method, dividing the minimal node set into seven segments and applying Bezier smoothing to each. This simple post-processing technique offers a balance between smoothness and structural fidelity, though it is less adaptive than Zhang & Zhang's [37] inflection point pruning.

In study (ID 1.7), the introduction of dynamic weighting and flexible neighborhood selection significantly improves computational efficiency while maintaining accuracy in large-scale environments. However, this method depends on properly tuning the weight coefficient  $W$  during the search phases; if  $W$  is not optimally chosen, the resulting path may not be truly shortest or may become unstable when the environment changes. In addition, the five search directions are constrained by the initial vector from the start to the goal point, so when the obstacle configuration is complex or there are many junctions, the narrowed search range may overlook feasible passages, reducing reliability in dynamic environments or irregular spaces. Similarly, algorithms in study (ID 1.10) that incorporate bidirectional search and refined evaluation functions achieve faster convergence and lower path costs, while the integration of Bezier or B-spline curves ensures smoother trajectories, which are critical for stable robot motion. Nevertheless, this approach still relies on the assumption of a static environment and complete map information; if obstacles move or the map is continuously updated, the bidirectional strategy will struggle to synchronize the searches from both sides. Moreover, the algorithm does not account for the vehicle's kinematic constraints, so when applied to robots with limited turning radius or more complex dynamics, the smoothed path may still be mechanically infeasible. In off-road emergency rescue scenarios, study (ID 1.9) analyzed the inclusion of road and terrain factors allows the algorithm

to generate feasible and time-efficient paths across unstructured environments, demonstrating adaptability to harsh and dynamic terrains. However, the fixed grid-based map model assumes a stable laboratory layout; any changes in the arrangement or equipment require remapping and a complete recomputation, limiting flexibility in environments that change frequently.

Despite these advances, several limitations remain. Many improved versions rely heavily on accurate environmental models, which may reduce effectiveness in dynamic or uncertain contexts. Furthermore, the additional computational overhead introduced by smoothing and evaluation refinements can limit scalability for real-time large-scale systems. In practice, these algorithms have shown promising applications in autonomous vehicles, logistics robots, marine and field exploration, and healthcare automation, where both safety and efficiency are paramount. Future research should aim to combine dynamic re-planning with multi-objective optimization to balance robustness, adaptability, and real-time performance in diverse operating environments.

As shown in Table 1, algorithms in Group 3 demonstrate varying performance. Studies (IDs 1.7 and 1.9) achieved substantial reductions in processing time—82.07% and 88.2%, respectively. In contrast, study (ID 1.10) reported only a 20% reduction, indicating lower optimization effectiveness.

#### 2.4. Group 4

The final study in this section (ID 1.2) distinguishes itself by prioritizing path smoothness over processing time or node expansion.

To enhance trajectory smoothness and control efficiency for autonomous land vehicles (ALVs), Zhang et al. [29] proposed an improved A\* algorithm that integrates an artificial potential field (APF) into the heuristic. Traditional A\*, relying solely on distance-based heuristics, often produces paths with excessive turning points. To address this, the revised evaluation function is:  $f(n) = g(n) + h'(n) + v(n)$ ; where  $g(n)$  is the actual cost,  $h'(n)$  is a diagonal-distance-based heuristic, and  $v(n)$  represents APF-based repulsion from obstacles and attraction to the goal.

Specifically, the improved heuristic  $h'(n)$  is defined as:

$$\begin{aligned} h'(n) &= dx + dy + \left(\sqrt{2} - 2\right) \cdot \min(dx, dy) \\ dx &= |x_n - x_e|; dy = |y_n - y_e| \end{aligned} \quad (12)$$

Here,  $x_n$  and  $y_n$  represent the horizontal and vertical coordinates, respectively, of the current position  $n$ , while  $x_e$  and  $y_e$  correspond to the horizontal and vertical coordinates of the target point. The potential field term  $v(n)$  is computed via the projection of the synthetic force vector  $\vec{F}(k)$  (all repulsive forces and attraction) onto the direction vector  $\vec{e}_n$  toward the neighbor:

$$\begin{aligned} v(n) &= \left| \vec{F}(k) \right| \cdot \cos(\theta_n) \\ \theta_n &= \left\langle \vec{e}_n, \vec{F}(k) \right\rangle \end{aligned} \quad (13)$$

The improved A\* algorithm for autonomous land vehicles introduces a heuristic function combining distance and obstacle potential fields, enabling smoother and more practical paths. By reducing redundant turning points and integrating B-spline smoothing, the method enhances maneuverability and decreases posture adjustments, which is crucial for efficiency in logistics, rescue, and urban navigation. Its main advantage is improved path feasibility and stability, while the drawback lies in higher computational cost and longer planning times in dense environments. Nevertheless, this method still has clear limitations: (i) it relies heavily on the quality and accuracy of the static raster map; (ii) com-

puting the potential field and performing B-spline path smoothing significantly increases planning time (doubling it in large-scale tests), making real-time applications challenging; and (iii) the well-known local minima problem of the potential field approach can cause the algorithm to become trapped.

When combined with quartic B-spline smoothing, it preserves path length similar to traditional A\*, while reducing total turning angle by approximately 71–72%. Although computational time increases due to added complexity, overall ALV operation time decreases, as fewer directional adjustments are needed during execution.

### 3. A\* Algorithm Combined with Other Algorithms

This section analyzes ten studies that integrate the A\* algorithm with other path planning techniques. As summarized in Table 3, the studies are grouped into three categories: (i) A\* combined with the Dynamic Window Approach (DWA)—IDs 2.4, 2.8, and 2.10; (ii) A\* combined with Artificial Potential Field (APF)—IDs 2.2, 2.4, and 2.8; and (iii) A\* integrated with other algorithms. The table also provides a comparative evaluation of each improved algorithm against its respective baseline: ID 2.7 is compared with traditional DWA, ID 2.8 with standalone APF, and the remaining studies with traditional A\*. Reported values are averaged from experimental results in the respective papers. It should be noted that these comparisons are approximate, given variations in hybridization strategies and experimental setups independently.

#### 3.1. Dynamic Window Approach

In mobile robot navigation, integrating A\* with the Dynamic Window Approach (DWA) creates a hybrid path planning framework that leverages both global and local strategies. A\* effectively computes optimal global paths on static maps but struggles with dynamic changes, such as moving obstacles. To overcome this, DWA serves as a real-time local planner, accounting for kinematic constraints, velocity limits, and obstacle clearance. The combination enables efficient global route planning and responsive local obstacle avoidance, improving robustness and adaptability in dynamic or partially unknown environments.

A key advantage of combining DWA with A\* is its ability to ensure safe, smooth navigation under high environmental uncertainty. A\* supplies accurate global waypoints, while DWA prevents local minima entrapment by using these references for reactive avoidance. The resulting trajectories are dynamically feasible, avoiding abrupt turns or stops common in purely A\*-based planning. Operating within a constrained velocity space and relying on local computation, DWA reduces the need for continuous global replanning. These strengths have made the A\* + DWA hybrid widely adopted in autonomous systems navigating dynamic environments.

##### 3.1.1. Working Principle of DWA

The A\* + DWA combinations proposed by Zhenyang and Wei [41], Liu et al. [45], Liao et al. [47], adopt a shared formulation for the DWA evaluation function, which forms the basis of local trajectory optimization. The general objective function is expressed as:

$$G(v, \omega) = \alpha \cdot \text{heading}(v, \omega) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{velocity}(v, \omega) \quad (14)$$

where

- ✓  $v$  denotes the translational velocity, and  $\omega$  is the angular velocity;
- ✓  $\text{heading}(v, \omega)$  evaluates the alignment between the robot's orientation and the goal direction, typically measured using the cosine of the angle between them;

- ✓  $dist(v, \omega)$  quantifies the minimum distance from the predicted trajectory to any nearby obstacle, favoring trajectories with greater clearance;
- ✓  $velocity(v, \omega)$  encourages higher speeds when it does not compromise safety.

**Table 3.** Comparative analysis of a\* hybrid path planning algorithms based on key performance metrics.

| Paper ID | Ref. | The Number of Expanded Nodes                                                                                                                                                                            | Path Length                        | Processing Time                                                       | Path Angularity                                                                                                                                                                     | Hybrid Algorithm                   |
|----------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| 2.1      | [38] | ✓ The reduction may be attributed to the nature of the RRT algorithm, which does not explore the entire space exhaustively but rather expands the tree through randomly sampled points; not quantified. | ✓ −51.7%                           | ✓ −70.9%                                                              | ✓ In RRT, new nodes are generated along vectors directed toward the goal, which facilitates the formation of curved paths or paths with reduced angular deviations; not quantified. | RRT                                |
| 2.2      | [39] | ✗ unspecified                                                                                                                                                                                           | ✓ −22%                             | ✗ unspecified                                                         | ✓ −53%                                                                                                                                                                              | Minimum Snap Trajectory Generation |
| 2.3      | [40] | ✓ −88.85%                                                                                                                                                                                               | ✓ −5.58%                           | ✓ −77.05%                                                             | ✗ unspecified                                                                                                                                                                       | APF                                |
| 2.4      | [41] | ✓ −13.9%                                                                                                                                                                                                | ✗ insignificant change             | ✓ −36.6%                                                              | ✓ Path smoothing is achieved using the Dynamic Tangent Method; not quantified.                                                                                                      | DWA                                |
| 2.5      | [42] | ✓ −32%                                                                                                                                                                                                  | ✗ unspecified                      | ✓ −70%                                                                | ✓ Path smoothing is achieved using the Least Squares method; not quantified.                                                                                                        | APF                                |
| 2.6      | [43] | ✓ −17.2%                                                                                                                                                                                                | ✓ −9.68%                           | ✓ −24.54%                                                             | ✓ −72.59%                                                                                                                                                                           | Greedy                             |
| 2.7      | [44] | ✗ unchanged                                                                                                                                                                                             | ✓ −4.8%                            | ✓ −16%                                                                | ✓ −76%                                                                                                                                                                              | PSO                                |
| 2.8      | [45] | ✓ −58%                                                                                                                                                                                                  | ✓ −6.24%                           | ✓ −12%                                                                | ✓ −75%                                                                                                                                                                              | DWA                                |
| 2.9      | [46] | ✓ The reduction may be attributed to the decrease in the number of neighboring directions considered—from eight to five; not quantified.                                                                | ✗ No path shortening was observed. | ✓ The processing time is reduced; not quantified.                     | ✓ −51.96%                                                                                                                                                                           | APF                                |
| 2.10     | [47] | ✓ −51.8%                                                                                                                                                                                                | ✓ −17.9%                           | ✓ −43.4%                                                              | ✓ −53.5%                                                                                                                                                                            | DWA                                |
| 2.11     | [48] | ✗ unspecified                                                                                                                                                                                           | ✓ −5.5%                            | ✗ 50.7 times higher in terms of random sampling and optimization time | ✓ The path curvature is significantly increased owing to the continuous circular arc technique applied at the junction of two straight segments.                                    | RRT                                |
| 2.12     | [49] | ✓ It decreases because random node selection is probabilistic, thereby avoiding excessive exploration around local optima.                                                                              | ✗ unchanged                        | ✓ −12%                                                                | ✓ −24%                                                                                                                                                                              | RRT                                |

The weight coefficients  $\alpha$ ,  $\beta$ , and  $\gamma$  are adjustable parameters that balance goal orientation, safety, and efficiency. This local cost function allows real-time selection of the optimal motion command pair  $(v, \omega)$  at each control cycle, ensuring smooth, safe, and goal-directed movement. The DWA algorithm operates under the following constraints:

(i) *Physical velocity limits*

$$v \in [v_{\min}, v_{\max}], \omega \in [\omega_{\min}, \omega_{\max}] \quad (15)$$

(ii) Acceleration limits (feasible acceleration/deceleration over time  $\Delta t$ )

$$\begin{aligned} v &\in [v_t - a_v^{\max} \cdot \Delta t, v_t + a_v^{\max} \cdot \Delta t] \\ \omega &\in [\omega_t - a_\omega^{\max} \cdot \Delta t, \omega_t + a_\omega^{\max} \cdot \Delta t] \end{aligned} \quad (16)$$

(iii) Obstacle avoidance constraints

$$v \leq \sqrt{2 \cdot a_v \cdot \text{dist}(v, \omega)}, \quad \omega \leq \sqrt{2 \cdot a_\omega \cdot \text{dist}(v, \omega)} \quad (17)$$

The valid velocity window is the intersection of the three sets:

$$V_r = V_s \cap V_d \cap V_a \quad (18)$$

where

- ✓  $V_s$ : Velocity limits based on robot's maximum/minimum speed
- ✓  $V_d$ : Limits based on maximum acceleration
- ✓  $V_a$ : Limits based on safe stopping distance (to avoid collision)

### 3.1.2. Improved A\* Algorithm Combined with DWA

In recent years, integrating global A\* planning with local reactive DWA has gained traction for mobile robot navigation in complex, partially known environments. Three key studies (IDs 2.4, 2.8, 2.10) proposed distinct frameworks combining enhanced A\* with adaptive DWA, each introducing novel modifications to evaluation functions and coordination strategies to improve safety, efficiency, and path optimality.

To trace the evolution of evaluation strategies in hybrid A\*-DWA algorithms, Table 4 presents representative scoring functions from these key studies. These formulations progressively expand beyond distance and heading, incorporating kinematics, safety, dynamic feasibility, and motion smoothness. Notably, adaptive and direction-aware heuristics are used, with cost functions adjusted based on velocity or goal alignment. This reflects a shift from static, manually weighted evaluations to context-sensitive, integrated decision-making frameworks.

**Table 4.** Summary of evaluation functions in A\* + DWA algorithms.

| Paper ID | Representative Formula                                                                                                     | Key Differences in Evaluation Function                                             |
|----------|----------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| 2.4      | $G(v, \omega) = \alpha \text{Heading}(v, \omega) + \beta \text{Dist}_s(v, \omega) + \gamma \text{Vel}(v, \omega)$          | Emphasizes real-time control using DWA by tuning weights dynamically during motion |
| 2.8      | $G(v, \omega) = \alpha_1 \cdot \text{heading} + \alpha_2 \cdot \text{safety} + \alpha_3 \cdot \text{dynamic\_feasibility}$ | Evaluation function depends on robot kinematics                                    |
| 2.10     | $f(n) = g(n) + h(n) + \frac{(x_{\text{goal}} - x_n)^2 + (y_{\text{goal}} - y_n)^2}{2\sigma^2}$                             | Adjusts evaluation function conditionally based on direction of travel             |
|          | or : $f(n) = \begin{cases} g(n) + h(n), & v_x = 1 \\ g(n) + 2h(n), & v_x = -1 \end{cases}$                                 |                                                                                    |
|          | or : $f(n) = g(n) + h(n) + \frac{[(x_{\text{goal}} - x_n)^2 + (y_{\text{goal}} - y_n)^2]}{2c^2}$                           |                                                                                    |

All three studies adopt a two-tier planning architecture: an improved A\* algorithm generates the global path, while DWA handles local navigation via velocity sampling and evaluation. The A\* output—either a full path or key nodes—is fed to the DWA module as intermediate goals, aligning long-term planning with short-term control.

Despite this shared structure, each study proposes distinct enhancements to the evaluation functions at both global and local levels.

Zhenyang and Wei [41] studied to enhance both planning layers. The A\* algorithm integrates safe-distance constraints and dynamic tangential smoothing to eliminate redundant nodes and prevent diagonal cuts near obstacle corners. In the DWA module, the evaluation function is modified as:

$$G(v, \omega) = \alpha \cdot \text{Heading} + \beta \cdot \text{dist}_s + P \cdot \gamma \cdot \text{Vel} \quad (19)$$

Here, the adaptive weight  $P$  adjusts based on the robot's proximity  $D$  to the nearest obstacle, using:

$$P = \begin{cases} a_0 + b_0 \cdot \frac{D}{L}, & D < L \\ 1, & D \geq L \end{cases} \quad (20)$$

This ensures reduced speed near obstacles for safety, while allowing faster motion in open areas. Simulation results confirm that the adaptive DWA reduces iteration counts and runtime by 36.6%, despite a minor increase in path length.

To flexibly adjust the expansion bias toward either the goal or the start point, Liu et al. [45] introduced a dynamic weighting factor, denoted as  $P$ , which balances the priority of the heuristic function  $h(n)$ . This factor is calculated as follows:

$$P = \frac{|x_n - x_g| + |y_n - y_g|}{|x_s - x_g| + |y_s - y_g|}, \quad P \in [0, 1] \quad (21)$$

Based on this formulation, the new evaluation function is rewritten as follows:

$$f(n) = g(n) + e^P \cdot h(n) \quad (22)$$

This approach allows dynamic adjustment of the search strategy based on the position of node  $n$ , reducing node expansion and computation time. In this study, DWA is guided by key nodes from the improved A\* algorithm, operating in a “segmentally guided” manner. This prevents deviation from the global path and avoids local minima, ensuring close adherence to the global trajectory. Additionally, the DWA module optimizes the weighting coefficients  $\alpha$ ,  $\beta$ , and  $\gamma$  through experimental tuning, improving obstacle avoidance, reducing directional oscillations, and enhancing rotational smoothness.

Liao et al. [47] focused on dynamic weighting in A\* and kinematic feasibility in DWA. A dynamic factor  $P$  adjusts the heuristic term in:

$$f(n) = g(n) + P \cdot h(n) \quad (23)$$

enabling the algorithm to balance exploration and convergence based on node location. The DWA cost function is refined to consider minimum turning radius, obstacle clearance, and smooth heading transitions:

$$G(v, \omega) = \alpha_1 \cdot \text{heading} + \alpha_2 \cdot \text{safety} + \alpha_3 \cdot \text{dynamic\_feasibility} \quad (24)$$

DWA is guided by A\*-generated key points, improving global–local coordination and avoiding local minima. Experiments show enhanced path smoothness and directional stability.

Collectively, these three studies reflect the evolution of A\*-DWA fusion frameworks toward more adaptive, safe, and realistic path planning. Zhenyang and Wei's model [41] emphasizes velocity adaptation, Liu et al. [45] integrates kinematic constraints and dynamic heuristics, while Liao et al.'s approach [47] reformulates A\* for more intelligent global planning. Future systems may unify these advancements into a cohesive model combining nonlinear global reasoning, local kinodynamic feasibility, and adaptive obstacle avoidance.

### 3.2. Artificial Potential Field

Path planning in complex environments often requires balancing global optimality with real-time reactivity. The A\* algorithm and Artificial Potential Field (APF) method offer complementary strengths—A\* for global path generation and APF for local obstacle avoidance. Their fusion effectively combines long-range planning with responsive local navigation, enhancing overall path planning performance.

#### 3.2.1. Working Principle of Artificial Potential Field

The APF method models the robot's environment as a synthetic potential field composed of attractive forces from the goal and repulsive forces from obstacles.

- **Attractive potential function:**

$$U_{att}(p) = \frac{1}{2}k_{att}\|p - p_g\|^2 \quad (25)$$

where  $p = (x, y)$  represents the current position of the robot;  $k_{att}$  is the attraction coefficient that adjusts the intensity of the attractive force;  $\|\cdot\|$  denotes the Euclidean distance between the robot and the target.

- **Attractive force:**

$$F_{att}(p) = -\nabla U_{att}(p) = -k_{att}(p - p_g) \quad (26)$$

- **Repulsive potential function:**

$$U_{rep}(p) = \begin{cases} \frac{1}{2}k_{rep}\left(\frac{1}{\|p - p_0\|} - \frac{1}{q_0}\right)^2 & \text{if } \|p - p_0\| \leq q_0 \\ 0 & \text{otherwise} \end{cases} \quad (27)$$

where  $k_{rep}$  is the repulsion coefficient that adjusts the intensity of the repulsive force;  $p_0$  is the obstacle position;  $q_0$  denotes the influence radius of the obstacle.

Repulsive force is defined as follows.

$$F_{rep}(p) = -\nabla U_{rep}(p) \quad (28)$$

Resultant force is calculated as follows.

$$F_{total}(p) = F_{att}(p) + \sum_{i=1}^n F_{rep,i}(p) \quad (29)$$

Here,  $n$  represents the total number of obstacles. This total force vector dictates the instantaneous direction of the robot's motion.

### 3.2.2. Improved A\* Algorithm Combined with APF

In mobile robot navigation, integrating global A\* planning with the local reactivity of the APF method has become a key hybrid strategy. Three studies (IDs 2.3, 2.5, 2.9) present distinct yet convergent approaches to fusing improved A\* and APF, aiming to address the limitations of each. While all adopt a global–local coordination architecture, they differ in heuristic design, neighborhood selection, and force-field modeling. Their shared goal is to improve path planning in complex environments by reducing A\*'s node expansion and trajectory roughness, while overcoming APF's local minima and unreachable goal issues. In all cases, A\* generates a waypoint-based global path, which guides APF-driven local navigation.

Integrating A\* with the Artificial Potential Field (APF) method combines global path optimality with real-time local reactivity. A\* ensures an optimal path in static environments, while APF enables dynamic obstacle avoidance via attraction–repulsion forces. This fusion allows the robot to follow an optimal path while adapting to unforeseen obstacles through real-time sensor feedback—something standalone A\* cannot achieve without full replanning.

Chen et al. [40] introduced a directional three-neighbor A\* strategy, limiting expansion to nodes aligned with the movement trend to reduce redundancy. Its APF component adopts classical potential formulations for attraction and repulsion.

In contrast, Liu et al. [42] enhanced A\* by introducing a bias angle factor into the heuristic function,

$$f(n) = g(n) + h(n) + \lambda \cdot |\alpha - \beta| \quad (30)$$

where  $\alpha$  is the direction from the current node to the goal, and  $\beta$  is from the start to the goal. This dynamic angular penalty reduces sharp turns and improves path smoothness. While the APF component remains classical, its effectiveness is amplified through global–local fusion.

Sun et al. [46] presented the most mathematically extensive reformulation. It modifies both the heuristic and force-field components. The improved A\* uses a weighted angular heuristic:

$$h_{\text{new}}(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} + e^{-\lambda|x_n - x_g|} \cdot |\alpha - \beta| \quad (31)$$

This dynamically weighted term accelerates early search and decelerates near the goal. Furthermore, the APF algorithm is significantly enhanced with repulsive potential modified by a Euclidean power term  $n \in (0,1)$ , and the direction of resultant force adjusted via vector composition of gravitational and velocity components. The resulting repulsive force function is defined as:

$$U_{\text{rep}}(p) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left( \frac{1}{q(p)} - \frac{1}{q_0} \right)^n & \text{if } q(p) \leq q_0 \\ 0 & \text{otherwise} \end{cases} \quad (32)$$

where  $q(p)$  is the distance to the obstacle, and  $q_0$  is the maximum influence range. The resultant force  $\vec{F}_{\text{total}} = \vec{F}_{\text{att}} + \sum \vec{F}_{\text{rep},i}$ ,  $i$  ensures better obstacle avoidance and guarantees target reachability even in local optima scenarios.

In summary, all three algorithms seek to bridge global and local planning via A\*–APF integration. Chen et al.'s model [40] focuses on computational efficiency through neighbor

pruning, whereas the Liu et al.'s model [42] enhances path directness via angular heuristic control, and Sun et al.'s model [46] delivers the most robust solution by optimizing both search behavior and force-field dynamics.

Table 5 presents a comparative overview of these studies, highlighting key algorithmic modifications, hybridization mechanisms, obstacle-handling strategies, and their respective advantages and limitations. This synthesis clarifies the evolving trends and trade-offs among A\*–APF fusion approaches.

**Table 5.** Comparison of Representative A\*–APF Hybrid Algorithms in Terms of Modifications, Enhancements, and Performance Characteristics.

| Criteria          | Chen et al. [40]                   | Liu et al. [42]               | Sun et al. [46]                             |
|-------------------|------------------------------------|-------------------------------|---------------------------------------------|
| A* Modification   | Three-neighbor directional search  | Dynamic parent selection      | 5-directional + angle-based heuristic       |
| APF Enhancement   | Attractive-only, repulsion removed | Intermittent goal + smoothing | n-power repulsion + angle vector correction |
| Obstacle Handling | Rectangular grouping of obstacles  | APF for local deviation       | Real-time switching with sub-goals          |
| Switching Logic   | Region-triggered mode switch       | Turn-point adaptive control   | Goal queue with reactive jumping            |
| Main Benefit      | Reduced nodes and trap avoidance   | Smooth and continuous paths   | Highly adaptive and safe                    |
| Limitation        | No dynamic obstacle avoidance      | High computational load       | More complex and parameter-sensitive        |

### 3.3. Other Algorithms

#### 3.3.1. Improved A\* Algorithm Combined with Rapidly Exploring Random Tree

To overcome the slow convergence and suboptimal paths of classical sampling-based planners, Al-Ansarry and Al-Darraj [38] proposed a hybrid method combining the exploratory power of Rapidly exploring Random Tree (RRT) with the optimality of A\*. In this Hybrid RRT–A\* framework, RRT efficiently explores the configuration space  $C$  toward the goal region, after which A\* computes the locally optimal terminal segment.

During RRT expansion, new candidate nodes  $q_{new}$  are generated based on:

$$q_{new} \leftarrow q_{near} + \varepsilon \cdot \frac{q_{rand} - q_{near}}{\|q_{rand} - q_{near}\|} \quad (33)$$

where  $\varepsilon$  is the fixed step size and  $q_{rand} \in C_{free}$  is a random sample; and added to the tree only if they lie in the free space  $C_{free}$  and satisfy the collision-free condition:

$$(x - x_{obs})^2 + (y - y_{obs})^2 > r_{obs}^2 \quad (34)$$

where  $(x_{obs}, y_{obs})$  is the center and  $r_{obs}$  is the radius of a circular obstacle.

To optimize computational efficiency, the hybrid method introduces a dual-condition mechanism for A\* activation: A\* is triggered only when the newly added node  $q_{new}$  satisfies:

$$\|q_{new} - q_{goal}\| < T, \delta_{since \text{ last A*}} \geq \Delta \quad (35)$$

where  $T = 13$  is the goal-region threshold and  $\Delta = 7$  is the minimum step interval between consecutive A\* calls.

Once activated, A\* computes a locally optimal path segment using its standard evaluation function.

Figure 6 displays the Hybrid RRT-A\* flowchart proposed by Al-Ansarry and Al-Darraj [38].

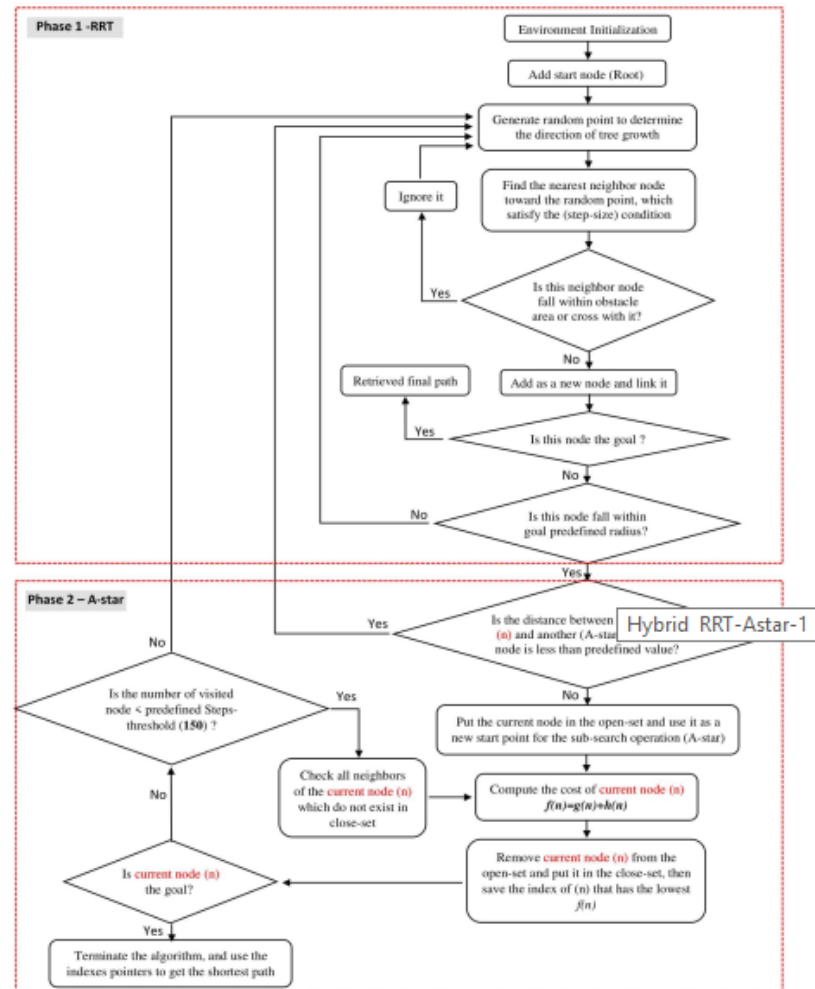


Figure 6. Hybrid RRT-A\* flowchart [38].

In the first study (ID 2.10), simulations on three  $75 \times 75$  grid maps showed that Hybrid RRT-A\* reduced path length by up to 51.7% and processing time by 70.9% compared to the traditional A\*. The integration notably enhanced performance in environments with 18–30 obstacles, confirming its effectiveness for real-time autonomous navigation.

In the second and third studies (IDs 2.11 and 2.12), the methodological emphases and application contexts differ substantially. The second study focuses on mobile robots operating in irregular and complex maps, combining vector-guided sector sampling, repulsive and attractive vector fields, beta-distribution point optimization, and adaptive corner smoothing. This integrated approach enhances both trajectory optimality and convergence efficiency. By contrast, the third study, designed for AGVs in digitalized workshop environments, replaces the Euclidean heuristic with the Chebyshev distance to introduce weak directionality and incorporates an inflection-point penalty, thereby reducing turning points and improving computational efficiency under industrial constraints. The results of this study demonstrate a 12% reduction in processing time and a 24% decrease in path angularity.

### 3.3.2. Improved A\* Algorithm Combined with Minimum Snap Trajectory Generation and Timed Elastic Band (TEB)

To address geometric discontinuities and dynamic infeasibility in traditional A\*-based planning, Ma et al. [39] proposed a hybrid framework combining an improved A\* planner with Minimum Snap Trajectory Generation and Timed Elastic Band (TEB) optimization. This approach ensures both global optimality and real-time local feasibility.

Initially, A\* generates a global path by minimizing the standard evaluation function:

$$f(n) = g(n) + h(n), \quad h(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} \quad (36)$$

To eliminate abrupt turns that violate non-holonomic constraints, the path is smoothed using Minimum Snap polynomials:

$$p(t) = \sum_{i=0} p_i t^i \quad (37)$$

$$\min \int_0^T \left( \frac{d^4 p(t)}{dt^4} \right)^2 dt = \min \sum_{i=1}^k p_i^T Q_i p_i \quad (38)$$

In this framework, snap—the fourth derivative of position—is minimized via a quadratic programming (QP) formulation with continuity and smoothness constraints across trajectory segments.

The Timed Elastic Band method further refines the trajectory in real time by modeling the path as an elastic structure subject to repulsive and attractive forces. Penalty functions enforce minimum distances from obstacles:  $f_{ob} = \varepsilon(-d_{min,j}, -r_{omin}, \dots)$ ,  $f_{path} = \varepsilon(d_{min,j}, r_{pmax}, \dots)$ , and minimize travel time via:  $f_k = \left( \sum_{i=1}^n \Delta T_i \right)^2$ .

Simulation and hardware tests on a 10 m × 5 m environment showed a 22% reduction in path length and a 53% improvement in smoothness compared to the traditional A\*. The hybrid approach significantly enhances motion safety, energy efficiency, and responsiveness.

### 3.3.3. Improved A\* Algorithm Combined with Greedy Algorithm

To overcome the inefficiency of traditional A\* in multi-goal path planning, Xiang et al. [43] proposed a hybrid approach combining an improved A\* algorithm with a greedy insertion mechanism. This method generates near-optimal, collision-free paths for autonomous mobile robots (AMRs) while minimizing total path length and angular discontinuities. The enhanced A\* employs a modified evaluation function:

$$F(n) = G(n) + H(n) + o(n), \quad o(n) = -\alpha C(n) + \beta I(n) \quad (39)$$

where  $C(n)$  promotes goal-directed expansion and  $I(n)$  penalizes directions with dense obstacles. The weights  $\alpha$  and  $\beta$  allow adaptive tuning between efficiency and safety. For multi-objective planning, an initial sequence of  $(n-m)$  goal points is selected and optimized using A\*. Then, a greedy insertion strategy is used to insert the remaining  $m$  points. Given an existing sequence:  $Path_k = \{S, X_1, X_2, \dots, X_k, G\}$ ; a new point  $X_{k+1}$  is inserted at position  $i^*$  that minimizes the total path cost:  $L_i = L_{prefix} + d(X_i, X_{k+1}) + d(X_{k+1}, X_{i+1})$ ,  $i^* = \arg \min_i [L_i]$ . This insertion continues until all targets are visited.

Figure 7 compares results on three grid maps (20 × 20, 30 × 30, and 50 × 50) with varying obstacle shapes and densities. Red paths indicate traditional A\*, while green paths represent those generated by the improved A\* combined with the Greedy algorithm. The hybrid method produces slightly shorter, smoother paths with fewer turning points.



computational cost, whereas coarse grids reduce computation but produce unrealistic “staircase” trajectories that violate the kinematic feasibility of real robots. Finally, A\* lacks adaptability; any environmental change necessitates recomputation from scratch, which is particularly prohibitive in large, dynamic scenarios such as warehouses, agricultural fields, or search-and-rescue operations.

In essence, A\*'s poor scalability arises from its exhaustive search nature and uniform grid reliance. This is why researchers in robotics and automation have moved toward improved or hybrid A\* variants (e.g., A\* + DWA, A\* + APF, A\* + PSO, A\* + RRT...), which incorporate adaptive heuristics, hierarchical maps, or learning-based strategies to significantly reduce computational load while maintaining path quality.

The findings indicate that while the traditional A\* algorithm has its limitations—such as trajectory angularity, computational inefficiency, and lack of adaptability to dynamic or large-scale environments—have motivated diverse enhancements. Studies focusing on geometric and heuristic refinements have demonstrated significant reductions in node expansion, path angularity, and computation time, while hybrid frameworks that combine A\* with methods such as DWA, APF, PSO, RRT, and Greedy algorithm have proven more robust in dynamic, uncertain, or multi-goal scenarios. Importantly, the hybrid approaches not only bridge global optimality with local adaptability but also offer improved safety margins and smoother trajectories, thereby moving closer to real-world applicability in autonomous mobile robots.

Despite these advances, several open challenges remain. First, most of the improvements have been validated in controlled simulation environments, with limited experimental evidence on physical robotic platforms operating in large-scale, unstructured, or highly dynamic environments. Second, many algorithms prioritize one performance metric (e.g., smoothness or computational efficiency) at the expense of others, leading to trade-offs that are not sufficiently addressed in a unified optimization framework. Third, current hybrid algorithms still lack generalization capability; their effectiveness is often contingent upon parameter tuning and problem-specific assumptions. Furthermore, the integration of learning-based methods, such as deep reinforcement learning, has only been explored in a limited capacity, leaving significant potential for adaptive and self-optimizing navigation strategies.

Future research should address these gaps through several promising directions. A first priority is the development of holistic evaluation benchmarks that incorporate path optimality, smoothness, safety, computation time, and energy efficiency under standardized testing scenarios. Such benchmarks will enable objective cross-comparison and accelerate the translation of algorithmic innovation into practical deployment. A second direction lies in the tight integration of A with machine learning and data-driven methods, enabling adaptive heuristic functions, real-time parameter optimization, and lifelong learning in continuously evolving environments. Third, multi-agent navigation represents a critical frontier, where collaborative path planning and decentralized control must ensure scalability and robustness in swarm robotics, warehouse intralogistics, and emergency rescue operations. Fourth, future systems must incorporate kinodynamic constraints and uncertainty modeling, ensuring that planned trajectories are not only geometrically optimal but also dynamically feasible and real-time movements under sensor noise, localization errors, and environmental disturbances. Lastly, hardware-in-the-loop validation and large-scale real-world experiments should become a standard component of evaluation to confirm robustness, safety, and computational feasibility in practical robotic systems.

In conclusion, the trajectory of research indicates that improved and hybridized A\* algorithms will remain a cornerstone of autonomous navigation. Their continued evolution, coupled with advances in sensing, computation, and machine learning, promises to deliver

intelligent, adaptive, and reliable robotic systems capable of safe and efficient operation in increasingly complex environments.

**Author Contributions:** Conceptualization, D.T.X.; methodology, D.T.X.; formal analysis, D.T.X.; investigation, D.T.X.; resources, D.T.X.; data curation, D.T.X.; writing—original draft preparation, D.T.X.; writing—review and editing, N.T.H. and V.T.T.; supervision, V.T.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is funded by the Hanoi University of Science and Technology (HUST) under the project number T2023-PC-010.

**Data Availability Statement:** No new data were created or analyzed in this study. Data sharing is not applicable to this article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Kosaka, A.; Kak, A.C. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *CVGIP Image Underst.* **1992**, *56*, 271–329. [CrossRef]
2. Patle, B.K.; Babu, L.G.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
3. Taheri, H.; Zhao, C.X. Omnidirectional mobile robots, mechanisms and navigation approaches. *Mech. Mach. Theory* **2020**, *153*, 103958. [CrossRef]
4. Cebollada, S.; Payá, L.; Flores, M.; Peidró, A.; Reinoso, O. A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data. *Expert Syst. Appl.* **2021**, *167*, 114195. [CrossRef]
5. Fragapane, G.; de Koster, R.; Sgarbossa, F.; Strandhagen, J.O. Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *Eur. J. Oper. Res.* **2021**, *294*, 405–426. [CrossRef]
6. Panigrahi, P.K.; Bisoy, S.K. Localization strategies for autonomous mobile robots: A review. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 6019–6039. [CrossRef]
7. Liu, L.; Wang, X.; Yang, X.; Liu, H.; Li, J.; Wang, P. Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.* **2023**, *227*, 120254. [CrossRef]
8. Singh, R.; Ren, J.; Lin, X. A Review of Deep Reinforcement Learning Algorithms for Mobile Robot Path Planning. *Vehicles* **2023**, *5*, 1423–1451. [CrossRef]
9. Loganathan, A.; Ahmad, N.S. A systematic review on recent advances in autonomous mobile robot navigation. *Eng. Sci. Technol. Int. J.* **2023**, *40*, 101343. [CrossRef]
10. Zhao, C.; Chen, J.-Y.; Sun, T.; Fan, W.; Sun, X.-Y.; Shao, Y.; Guo, G.-Q.; Wang, H.-L. Robotic motion planning for autonomous in-situ construction of building structures. *Autom. Constr.* **2025**, *171*, 105993. [CrossRef]
11. Taleb, M.A.; Korsoveczki, G.; Husi, G. Automotive navigation for mobile robots: Comprehensive review. *Results Eng.* **2025**, *27*, 105837. [CrossRef]
12. Introduction to A\*. From Amit's Thoughts on Pathfinding. Available online: <https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html> (accessed on 9 July 2025).
13. Liu, Y.; Gao, X.; Wang, B.; Fan, J.; Li, Q.; Dai, W. A passage time–cost optimal A\* algorithm for cross-country path planning. *Int. J. Appl. Earth Obs.* **2024**, *130*, 103907. [CrossRef]
14. Wang, X.; Li, G.; Bian, Z. Research on the A\* Algorithm Based on Adaptive Weights and Heuristic Reward Values. *World Electr. Veh. J.* **2025**, *16*, 144. [CrossRef]
15. Heuristics. From Amit's Thoughts on Pathfinding. Available online: <https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html> (accessed on 9 July 2025).
16. Amit's A\* Pages. From Red Blob Games. Available online: <https://theory.stanford.edu/~amitp/GameProgramming/> (accessed on 9 July 2025).
17. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
18. Wang, Y.; Linnett, J.A.; Roberts, J. Kinematics, kinematic constraints and path planning for wheeled mobile robots. *Robotica* **1994**, *12*, 391–400. [CrossRef]
19. Romero, S.; Valero, J.; García, A.V.; Rodríguez, C.F.; Montes, A.M.; Marín, C.; Bolaños, R.; Álvarez-Martínez, D. Trajectory Planning for Robotic Manipulators in Automated Palletizing: A Comprehensive Review. *Robotics* **2025**, *14*, 55. [CrossRef]

20. Torres, L.G.; Kuntz, A.; Gilbert, H.B.; Swaney, P.J.; Hendrick, R.J.; Webster, R.J., 3rd; Alterovitz, R. A Motion Planning Approach to Automatic Obstacle Avoidance during Concentric Tube Robot Teleoperation. In Proceedings of the 2015 IEEE international conference on robotics and automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2361–2367. [\[CrossRef\]](#)
21. Daniel, K.; Nash, A.; Koenig, S.; Felner, A. Theta\*: Any-angle path planning on grids. *J. Artif. Intell. Res.* **2010**, *39*, 533–579. [\[CrossRef\]](#)
22. Ayan, N.F.; Kuter, U.; Yaman, F.; Goldman, R.P. Hotride: Hierarchical ordered task replanning in dynamic environments. In Proceedings of the Planning and Plan Execution for Real-World Systems—Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop, Providence, RI, USA, 22–26 September 2007; pp. 26–27.
23. Vemula, A. Safe and Efficient Navigation in Dynamic Environments. Master's Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, 2017. Available online: <https://www.ri.cmu.edu/app/uploads/2017/08/thesis1-Anirudh-Vemula.pdf> (accessed on 9 July 2025).
24. Xu, X.; Zeng, J.; Zhao, Y.; Lü, X. Research on global path planning algorithm for mobile robots based on improved A\*. *Expert Syst. Appl.* **2024**, *243*, 122922. [\[CrossRef\]](#)
25. Sedighi, S.; Nguyen, D.; Kuhnert, K. Guided hybrid A-star path planning algorithm for valet parking applications. In Proceedings of the 2019 5th International Conference on Control, Automation and Robotics (ICCAR), Beijing, China, 19–22 April 2019; pp. 570–575. [\[CrossRef\]](#)
26. Shi, Y.; Yang, J.; Bu, S.; Zhu, L. Intelligent vehicle path planning algorithm based on improved RRT. *Comput. Technol. Autom.* **2019**, *38*, 81–86.
27. Shankar, M.; Sushnigdha, G. A Hybrid Path planning approach combining Artificial Potential Field and Particle Swarm Optimization for Mobile Robot. *IFAC-PapersOnLine* **2022**, *55*, 242–247. [\[CrossRef\]](#)
28. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. An improved A-Star based path planning algorithm for autonomous land vehicles. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420962263. [\[CrossRef\]](#)
29. Zhang, J.; Wu, J.; Shen, X.; Li, Y. Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 17298814211042730. [\[CrossRef\]](#)
30. XiangRong, T.; Yukun, Z.; XinXin, J. Improved A-star algorithm for robot path planning in static environment. *J. Phys. Conf. Ser.* **2021**, *1792*, 012067. [\[CrossRef\]](#)
31. Wang, H.; Lou, S.; Jing, J.; Wang, Y.; Liu, W.; Liu, T. The EBS-A\* algorithm: An improved A\* algorithm for path planning. *PLoS ONE* **2022**, *17*, e0263841. [\[CrossRef\]](#)
32. Martins, O.O.; Adekunle, A.A.; Olaniyan, O.M.; Bolaji, B.O. An Improved multi-objective a-star algorithm for path planning in a large workspace: Design, Implementation, and Evaluation. *Sci. Afr.* **2022**, *15*, e01068. [\[CrossRef\]](#)
33. Zhang, H.; Tao, Y.; Zhu, W. Global Path Planning of Unmanned Surface Vehicle Based on Improved A-Star Algorithm. *Sensors* **2023**, *23*, 6647. [\[CrossRef\]](#)
34. Han, C.; Li, B. Mobile Robot Path Planning Based on Improved A\* Algorithm. In Proceedings of the 2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 8–10 December 2023; pp. 672–676.
35. Yin, C.; Tan, C.; Wang, C.; Shen, F. An Improved A-Star Path Planning Algorithm Based on Mobile Robots in Medical Testing Laboratories. *Sensors* **2024**, *24*, 1784. [\[CrossRef\]](#)
36. Zhao, D.; Ni, L.; Zhou, K.; Lv, Z.; Qu, G.; Gao, Y.; Yuan, W.; Wu, Q.; Zhang, F.; Zhang, Q. A Study of the Improved A\* Algorithm Incorporating Road Factors for Path Planning in Off-Road Emergency Rescue Scenarios. *Sensors* **2024**, *24*, 5643. [\[CrossRef\]](#)
37. Yuanyuan, Z.; Miao, Z. Path Planning for Mobile Robots Based on Improved A-star Algorithm. In Proceedings of the 2025 International Conference on Artificial Life and Robotics (ICAROB2025), J:COM HorutoHall, Oita, Japan, 13–16 February 2025; pp. 87–90.
38. Al-Ansarry, S.; Al-Darraj, S. Hybrid RRT-A\*: An Improved Path Planning Method for an Autonomous Mobile Robots. *Iraqi J. Electr. Electron. Eng.* **2021**, *17*, 107–115. [\[CrossRef\]](#)
39. Ma, Z.; Qiu, H.; Wang, H.; Yang, L.; Huang, L.; Qiu, R. A\* algorithm path planning and minimum snap trajectory generation for mobile robot. In Proceedings of the 2021 4th International Conference on Robotics, Control and Automation Engineering (RCAE), Wuhan, China, 4–6 November 2021; pp. 284–288.
40. Chen, J.; Tan, C.; Mo, R.; Zhang, H.; Cai, G.; Li, H. Research on path planning of three-neighbor search A\* algorithm combined with artificial potential field. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 17298814211026449. [\[CrossRef\]](#)
41. Zhenyang, X.; Wei, Y. Mobile robot path planning based on fusion of improved A\* algorithm and adaptive DWA algorithm. *Proc. J. Phys. Conf. Ser.* **2022**, *2330*, 012003. [\[CrossRef\]](#)
42. Liu, L.; Wang, B.; Xu, H. Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm and Artificial Potential Field Method. *Electronics* **2022**, *11*, 3660. [\[CrossRef\]](#)
43. Xiang, D.; Lin, H.; Ouyang, J.; Huang, D. Combined improved A\* and greedy algorithm for path planning of multi-objective mobile robot. *Sci. Rep.* **2022**, *12*, 13273. [\[CrossRef\]](#)

44. Huang, C.; Zhao, Y.; Zhang, M.; Yang, H. APSO: An A\*-PSO hybrid algorithm for mobile robot path planning. *IEEE Access* **2023**, *11*, 43238–43256. [\[CrossRef\]](#)
45. Liu, Y.; Wang, C.; Wu, H.; Wei, Y. Mobile Robot Path Planning Based on Kinematically Constrained A-Star Algorithm and DWA Fusion Algorithm. *Mathematics* **2023**, *11*, 4552. [\[CrossRef\]](#)
46. Sun, Z.; Zhang, T.; Zhu, H.; Ma, T.; Bao, X.; Zhang, X. Path Planning for Mobile Robot Based on the Fusion Algorithm of Improved A\* and APF. In Proceedings of the 2024 6th International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT), Changzhou, China, 20–22 September 2024; pp. 113–118.
47. Liao, T.; Chen, F.; Wu, Y.; Zeng, H.; Ouyang, S.; Guan, J. Research on Path Planning with the Integration of Adaptive A-Star Algorithm and Improved Dynamic Window Approach. *Electronics* **2024**, *13*, 455. [\[CrossRef\]](#)
48. Duan, M.; Wang, Z.; Shao, X.; Ren, G. VGA\*-RRT\*: A mobile robot path planning algorithm for irregular and complex maps. *IEEE Access* **2025**, *13*, 50714–50726. [\[CrossRef\]](#)
49. Liu, N.; Hu, Z.; Wei, M.; Guo, P.; Zhang, S.; Zhang, A. Improved A\* algorithm incorporating RRT\* thought: A path planning algorithm for AGV in digitalised workshops. *Comput. Oper. Res.* **2025**, *177*, 106993. [\[CrossRef\]](#)
50. Foad, D.; Ghifari, A.; Kusuma, M.B.; Hanafiah, N.; Gunawan, E. A systematic literature review of A\* pathfinding. *Procedia Comput. Sci.* **2021**, *179*, 507–514. [\[CrossRef\]](#)

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.