Iraqi Journal of Computers, Communications, Control and Systems Engineering

Volume 26 | Issue 1 Article 9

4-4-2026

Path Planning for Robotic Systems in Dynamic Environments: Using RRT*, RRT-D* Lite, and Bi-RRT*-D* Lite

Hameed Salman Hameed
College of Control and Systems Engineering, University of Technology-Iraq, Baghdad, Iraq,
cse.22.03@grad.uotechnology.edu.iq

Omar Farouq Lutfy
College of Control and Systems Engineering, University of Technology-Iraq, Baghdad, Iraq,
omar.f.lutfy@uotechnology.edu.iq

Firas A. Raheem
College of Control and Systems Engineering, University of Technology-Iraq, Baghdad, Iraq,
firas.a.raheem@uotechnology.edu.iq

IJCCCE 2026;26(1):117–133

RESEARCH ARTICLE

Path Planning for Robotic Systems in Dynamic Environments: Using RRT*, RRT-D* Lite, and Bi-RRT*-D* Lite

Hameed Salman Hameed *, Omar Farouq Lutfy, Firas A. Raheem

College of Control and Systems Engineering, University of Technology-Iraq, Baghdad, Iraq

ABSTRACT

Path planning is one of the essential challenges in the area of robotic systems. In recent years, Rapidly-exploring Random Tree star (RRT*) has been a preferred path planner for robots due to its probabilistic completeness. This paper explores using three path planning algorithms for a mass point robotic system navigating dynamic environments with RRT*, Bidirectional Rapidly-exploring Random Tree (Bi-RRT*), and a novel hybrid approach (Bi-RRT*-D Lite*). The study shows the performance of these algorithms in terms of the path length, the path generating time, the number of search attempts, and the percentage error. In addition, it compares the Bi-RRT* and the hybrid approach of Bi-RRT*-D*Lite relative to RRT*. We implement each algorithm in MATLAB and evaluate its effectiveness through simulations involving mazes with static and dynamic obstacles. Our findings demonstrate the advantages of the new hybrid approach in handling dynamic obstacles and showing superior efficiency in complex environments. The results showed that the reduction percentages of iteration, path generating time, and path length for Bi-RRT* relative to RRT are 91%, 86%, and 0.33%, respectively. On the other hand, the hybrid approach gives the percentages of improvements of 97%, 97%, and 6.8% compared to RRT.

Keywords: Path planning, Dynamic environment, Hybrid approach, Rapidly-exploring random tree star

Highlights

1. Studying the limitations of RRT* in path planning.
2. Improvement of RR*.
3. Suggesting a unified framework.

Received 17 July 2025; revised 4 August 2025; accepted 16 September 2025.
Available online 4 April 2026

* Corresponding author.
E-mail addresses: cse.22.03@grad.uotechnology.edu.iq (H. S. Hameed), omar.f.lutfy@uotechnology.edu.iq (O. F. Lutfy), Firas.A.Raheem@uotechnology.edu.iq (F. A. Raheem).

https://doi.org/xx.xxxxx/2617-3352.1521

1. Introduction

Path planning is a critical aspect in the design phase of robotic systems, ensuring the generation of the shortest and safest trajectories. By optimizing travel time and energy consumption, effective path planning enables precise execution of repetitive tasks with fewer errors and reduced downtime. It also facilitates multi-tasking and coordination among multiple robots and rapid adaptation to dynamic environmental changes, ultimately saving computational resources and minimizing operational delays. Furthermore, path planning guarantees the physical feasibility of trajectories, preventing unsafe or impractical motions.

The challenges of path planning in dynamic environments are multifaceted, involving accurate obstacle detection, maintaining optimal paths while avoiding collisions, and ensuring computational efficiency and adaptability [1]. Unlike static environments, where obstacles are stationary and predictable, dynamic environments introduce uncertainty, requiring robots to quickly adapt to unexpected changes. In this regard, optimal path planning must consider safety, efficiency, and computational cost, particularly in complex, changing environments [2–5].

In this context, recent advances in path planning algorithms are summarized as follows. Zhang et al. [6] introduced a predictive algorithm that integrates RRT with dynamic movement prediction to reduce path costs and improve collision avoidance. Wu et al. [7] developed Fast-RRT, enhancing RRT with a fast-optimal module for quicker and more stable path finding. Ding et al. [8] presented Expanding Path RRT (EP-RRT), leveraging heuristic sampling to improve path extension in challenging environments. Li et al. [9] proposed enhancements to D* Lite to address safety margins and eliminate redundant nodes, while Gao et al. [10] combined D* Lite with the Dynamic Window method for improved operational efficiency. Yu et al. [11] developed an enhanced D* Lite for better computational efficiency, and Yao et al. [12] improved the classic A* algorithm using a bidirectional search mechanism integrated with D* Lite for increased adaptability. Other studies have focused on hybrid approaches. For instance, Zhang et al. [13] improved D* Lite for automated guided vehicles, addressing local optima and slow convergence. Jin et al. [14] introduced a conflict-based search with D* Lite for mixed environments. Lin et al. [15] developed a deep learning hybrid algorithm to improve navigation in extensive map scenarios. Zhu et al. [16] enhanced D* Lite for multi-target path planning in uncrewed surface vehicles. Hybrid methods combining RRT* with D* Lite, such as Bi-RRT*-D* Lite, have shown promise in balancing optimality and adaptability. Fan et al. [17] proposed a UAV trajectory planner using bidirectional APF-RRT*, improving convergence and search efficiency. Hameed et al.[18] proposed APF-IRRT*-SB, a path-planning algorithm that integrates APF and an improved RRT* with low-randomness sampling. The approach aims to enhance efficiency and path quality in dynamic, complex environments. Its effectiveness is demonstrated through comparative experiments on mass point and two-link robot scenarios. Xu et al. [19] addressed the limitations of RRT and PRM algorithms in path planning by proposing a fusion method that combines PRM with probability-based bidirectional RRT (P-Bi-RRT). The approach divides the planning area, applies PRM for pre-planning, and connects regions using optimal matching points with P-Bi-RRT or direct links. The resulting path is further optimized by removing redundant nodes and shortening the path length. Shu et al. [20] identified the inefficiency of RRT* and its variants in clustered environments with narrow passages, primarily due to high memory use and slow initial solution finding. The Locally Guided Multiple Bi-RRT* (LGM-BRRT*) method was proposed, which integrates an improved bridge test and local guidance strategy. The approach accelerates pathfinding, improves memory efficiency,

and remains straightforward. Xin et al. [21] addressed randomness in tree expansion by optimizing the artificial potential field in Bi-RRT*. Ma et al. [22] introduced a probability-smoothing Bi-RRT for better initial solution quality and convergence speed. Fan et al. [23] proposed a bidirectional Informed-RRT* for robust and smooth trajectory design. Wang et al. [24] improved Bi-RRT with greedy expansion and sampling space limitations for efficient path planning in medical robotics. Yang et al. [25] analyzed RRT* and Bi-RRT*, highlighting Bi-RRT*'s faster convergence and improved collision avoidance. Zhang et al. [26] presented a Bi-RRT-based planner for smooth and rapid trajectories in serial manipulators.

Previous works in robotic path planning for dynamic environments exhibit several limitations, including poor adaptability to sudden environmental changes, high computational demands for frequent replanning, and a tendency to generate suboptimal or impractical paths. Many methods also assume perfect knowledge of obstacle motion, resulting in reduced robustness when facing uncertainty. These weaknesses highlight the need for more efficient, adaptive, and reliable approaches, which motivated the comprehensive framework proposed in this study.

To overcome these challenges, this study integrates RRT*, RRT-D* Lite, and Bi-RRT*-D* Lite into a unified framework that achieves optimal path generation, efficient real-time replanning, and greater robustness to dynamic changes. By leveraging the strengths of each algorithm, the proposed approach offers a practical solution for reliable robotic path planning in real-world dynamic environments [27].

In conclusion, path planning in dynamic environments with moving obstacles is inherently complex, demanding algorithms capable of real-time adaptation, efficient obstacle detection, and robust trajectory optimization. In this regard, advanced hybrid algorithms, such as the proposed Bi-RRT*-D* Lite, offer a promising solution for navigating the intricacies of modern robotic applications.

The contribution of this study is to mitigate these challenges by suggesting a new hybrid approach. This approach integrates the Bi-RRT* with D*Lite algorithms for robotic systems. The Bi-RRT*accelerates convergence to the shortest feasible path while maintaining the asymptotic optimality properties of RRT*. The D*Lite can update the path locally, ensuring that the robot can adapt to changes without replanting the entire path. Moreover, a smoothing approach (Gaussian filter) was applied to explore and optimize a feasible path.

The structure of this study is as follows: Section 2 presents the materials and methods, detailing the proposed RRT*, D* Lite, and Bi-RRT*-D* Lite algorithms, along with smoothness and collision detection techniques. Section 3 presents the simulation results comparing the hybrid approach with the traditional RRT*, along with a discussion emphasising the efficiency and safety benefits of the proposed hybrid method. Finally, Section 4 concludes the study.

2. Materials and methods

*2.1. RRT**

The RRT* is excellent for global path planning in high-dimensional spaces, where the initial path is found efficiently even in intricate environments with dynamic obstacles. More specifically, the RRT* continuously refines the path over time, aiming for asymptotic optimality. Given enough time, the RRT* will eventually find the most optimal (shortest) path in the environment [28]. The algorithm steps are as follows:

Algorithm 1: RRT*.

***Initialize the tree**:
a. $T = [q_{init}]$: Start the tree with the initial configuration.
b. $Cost(q_{init}) = 0$: The cost to reach the initial configuration is zero.
c. $Parent(q_{init}) = null$: The initial configuration has no parent.
***Generate K random samples**: Set K as the number of iterations for random sampling.
***For each iteration k from 1 to K**:
a. Sample a random configuration:
Generate a random configuration q_{rand} in the Cartesian space X_{free}.
b. Find the nearest node in the tree:
- Find the nearest node q_{near} in the tree T to the random configuration q_{rand} using the Euclidean distance:
- $q_{near} = \arg\min_{q \in v} \|q - q_{rand}\|$, where *q_near*: nearest node to q_{rand}

c. Generate a new configuration q_{new}:
Move from $q_{near} =$ towards q_{rand} with a predefined step size ϵ to generate a q_{new} $q_{new} = q_{near}\epsilon . \frac{q_{rand} - q_{near}}{\|q_{rand} - q_{near}\|}$ (1)
d. Collision Check:
- Check if the path from q_{near} to q_{new} is free of collisions using the collision_checking_function.
- If a collision is detected, discard q_{new} and return to step 3. Otherwise, proceed to the next step.

e. Add the new node to the tree:
- Add q_{new} to the tree T.
- Set the parent of q_{new} to q_{near}.
- Calculate the cost of q_{new} : $c(q_{new}) = c(q_{near}) + \|(q_{new}) - (q_{near})\|$

*B- **Rewiring** (Optimization Step):*
1. Identify the neighboring nodes $N(q_{new})$ within the rewiring radius r_{rewire}:
$N(q_{new}) = \{q \in vT \ \|q - q_{new}\| \leq r_{rewire}\}$
2. For each q_{neigh} in $N(q_{new})$, check if q_{new} provides a better parent with a lower cost: $c(q_{near}) + \|(q_{new}) - (q_{neigh})\| < q_{neigh}$
If the above condition is true, reassin q_{new} as the new parent of q_{neigh}, and update the cost: $c(q_{neigh}) = c(q_{near}) + \|(q_{new}) - (q_{neigh})\|$.

*C- **Repeat for K iterations**:*
Repeat Steps 3 and 4 for K random samples to expand the tree and optimize the path.

*D- **Return the optimal path:***
1. After K iterations, the tree T is generated with the optimal path from the initial configuration q_{init} to the goal configuration q_{goal}.
2. The final path can be extracted from the tree by tracing back from q_{goal} to q_{init} using the parent nodes.

The algorithm starts by initializing the tree with the initial configuration, setting the cost to zero, and leaving the parent undefined. Random configurations are generated within the free space for a specified number of iterations (K), and the nearest node in the tree is identified. A new node is generated by moving towards the random sample with a predefined step size [29–33]. The algorithm checks for collisions along the path from the nearest node to the new node, and if clear, the new node is added to the tree. Rewiring is performed to optimize the tree by adjusting parent nodes and minimizing the path cost, and after K iterations, the optimal path is obtained by tracing back from the goal to the start configuration using the parent nodes [34, 35].

2.2. D* Lite

D* Lite is an incremental search algorithm for efficient path re-planning in dynamic environments. It maintains a priority queue of nodes and updates the path when changes occur, such as moving obstacles [36, 37]. When an obstacle is detected or removed, the algorithm updates the affected nodes and their neighbours, ensuring consistency in the graph [37]. The real-time re-planning capability allows quick adjustments, making D* Lite suitable for dynamic, time-sensitive applications. Its incremental updates reduce computational load, enabling real-time operation in environments with frequent changes [10, 38]. The D* Lite updates node costs and queues when the environment changes. The D* Lite clears the path for all segments of the robotic arm, ensuring collision-free operation by incrementally updating the affected nodes and recalculating path costs around obstacles. The D* Lite algorithm steps are shown below.

Algorithm 2: D* Lite.

1. **Initialization**
   a. Set Up Environment: Initialize the occupancy grid with obstacles.
   b. Start and Goal Setup: Define start and goal positions.
   c. Initialize Costs: rhs(s) = ∞, g(s) = ∞, rhs(sgoal) = 0.
   d. Insert into Open List: Add the goal to the open list with the initial key.
   e. key(sgoal) = h(sstart, sgoal) + km
2. **Environment Update**
   a. Obstacle Movement: Update dynamic obstacle positions.
   b. Rebuild Occupancy Grid: Update the grid based on new obstacle positions.
   c. Change Detection: Compare the old grid with the new one and identify changed cells.changed = detect Changed Cells(oldOcc, newOcc)
3. **Path Re-planning**
   a. Update rhs Values: Update rhs for the affected cells. rhs(s) = $min_{s' \in neighbors(s)}(cost(s,s') + g(s'))$.
   b. Compute the key for Each Cell: key(s) = min(g(s), rhs(s)) + h(s) + km.
   c. Update Open List: Remove the affected cells from the open list and reinsert them with the new keys.
4. **Replan Trigger** (*Threshold*)
   a. Time Threshold: Replan if the elapsed time since the last update exceeds a threshold. if toc(startTime) − last_update_time > threshold.
   b. Replan the Path: Recompute the optimal path from the start to the goal.
   c. Recompute the Path: path= dstar_get_path (data).
5. **Goal Check**
   a. Check Goal Achievement: Compare the current position with the goal position. if ‖ scurrent − sgoal ‖≤ goal_tolerance.
   b. Terminate if the Goal is reached: If the robot reaches the goal within tolerance, stop.

2.3. Bi-RRT*-D*Lite

On the other hand, the Bidirectional Rapidly-exploring Random Tree (Bi-RRT) is efficient for complex, high-dimensional spaces in path planning, where two trees grow simultaneously, one from the start, and one from the goal [3]. In particular, Tree 1 grows from the start point towards random samples, while Tree 2 grows from the goal point towards

random samples. Then, the trees attempt to connect with each other to generate more efficient and faster paths in complex environments[39]. However, the Bi-RRT* may still be computationally intensive in very complex spaces, no efficient re-planning in dynamic environments, limited adaptability to unknown or partially known environments, path quality may degrade or become invalid due to dynamic obstacles, high computational cost if the environment changes frequently, and the quality of the path may require smoothing post-processing[40]. To overcome most of these limitations, we will use a hybrid approach, the Bi-RRT*-D* Lite with a Gaussian function for smoothing, to incrementally re-plan the path quickly when obstacles or the environment change, avoiding full re-planning from scratch. The graph/grid-based approach complements the Bi-RRT* by handling discrete environmental changes effectively, and the ability to re-plan on-the-fly enables the robot to adapt to new information as it explores. The hybrid Bi-RRT*-D*Lite pays an up-front cost to build the grid, but each repair is incremental and very fast. The steps of the hybrid approach algorithm are as follows:

Algorithm 3: Bi-RRT*±D Lite.

1. Initialization
   a. Define environment bounds, start and goal configurations.
   b. Initialize two trees for Bi-RRT*:
   • Tree_start rooted at the start node.
   • Tree_goal rooted at goal node.
   c. Initialize the D* Lite planner:
   • Construct an occupancy grid from static/dynamic obstacles.
   • Set D* Lite start and goal cells based on grid mapping.
   d. Set parameters:
   • Maximum iterations, step size (EPS), goal bias, replanning frequency, robot radius, tolerances, etc.
2. For a maximum number of iterations or until trees connect:
   a. Sample random configuration q_rand, biased toward the goal.
   b. Alternate tree expansions:
   • Odd iterations: extend Tree_start toward q_rand.
   • Even iterations: extend Tree_goal toward q_rand.
   c. Extend chosen tree:
   • Find the nearest node q_near to the chosen tree.
   • Steer toward q_rand within a step size EPS → new node q_new.
   • Collision check edge (q_near to q_new).
   • If free, add q_new to tree.
   d. Try to connect opposite tree **to** q_new**:**
   • Find the nearest node in the opposite tree.
   • Attempt connection if collision-free.
   • If connected, trees are joined, proceed to path reconstruction.
   e. Visualize trees and the current best path (optional).
3. Path Reconstruction and Smoothing
   a. After trees connect:
   • Reconstruct the path from the start to the connection node in Tree_start.
   • Reconstruct the path from the goal to the connection node in Tree_goal.
   • Concatenate and smooth the full path using Gaussian or spline smoothing.
   b. This path is used as the initial global reference path.
4. Dynamic Re-planning with D Lite*
   • Initialize D* Lite with occupancy grid, start, and goal cells.
   • At each control loop iteration or fixed replanning intervals:

Algorithm 3: Continued.

a. Update the occupancy grid:
• Reflect the positions of the dynamic obstacles.
b. Detect the changed cells in the grid compared to the previous occupancy.
c. *Update D Lite graph:**
• Update the vertices affected by changes.
• Recompute the shortest path using the D* Lite incremental updates.
d. *Extract a new local path from D Lite**:
• Convert the grid path to world coordinates.
• Update the robot's reference path accordingly.
5. Execution and Monitoring
• The robot follows the path updated by D* Lite.
• If the path becomes blocked or deviates:
• Trigger replanning by Bi-RRT* or rely on D* Lite for local adjustments.
• The loop continues until the robot reaches the goal within tolerance or max attempts.
6. Termination
• Stop when the robot reaches the goal or no feasible path is found.
• Output the final path and the metrics (length, time, and cost).

3. Results and discussions

The main results are demonstrated through the simulation under MATLAB® (R2023b) environment, which is operated on a PC with an Intel® Core(TM) i7-1355U (3rd Gen, 1.70 GHz) processor and 8 GB of RAM. The simulation focuses on path planning for a mass point in a dynamic environment, comparing the efficiency of the three approaches: the RRT*-only, the Bi-RRT*, and the novel hybrid Bi-RRT-D* Lite*. The results for the three adopted algorithms will be addressed in detail in the following subsections.

3.1. RRT* implementation

The first case study shows the RRT* alone with a complex (hybrid) environment, i.e., it has static and dynamic obstacles. Thus, the environment in this study is considered to contain several fixed obstacles, a fixed maze, and only two moving obstacles oscillating vertically. In addition, the workspace is defined by Cartesian coordinates, with the window as set $\chi = \{(x, y) : x \in [1500\ 0],\ y \in [0\ 1500]\}$ length-unit, where all units are in mm; the start and the goal point coordinates are $\chi_{start} = (1500,\ 50)\ mm$, $\chi_{end} = (1500,\ 50)\ mm$, respectively. The other properties are as follows: The mass point starts from the goal point toward the goal with a radius of 10 *mm*, a step size (*EPS*) of 100 *mm*, goal gias of 0.7, goal tolerance of 30 *mm*, and maximum nodes number of 2000 nodes. In Fig. 2, there are 8 subfigures, Fig. 2(a… f) for the RRT*, where the first figure is the environment with hybrid static and dynamic obstacles. The subsequent figures are the evolving generated path utilizing the RRT* algorithm. The start point is the unfilled blue circle, while the goal, which is the end point, is the green circle. Besides the stationary environments, two moving obstacles oscillate in the vertical y-axis. The main properties to be compared are: the path length, the path generating time, the number of search attempts, and the percentage error for RRT*; these results will be compared with those of the Bi-RRT* and the new hybrid approach, the Bi-RRT*-D*Lite (see Table 1).

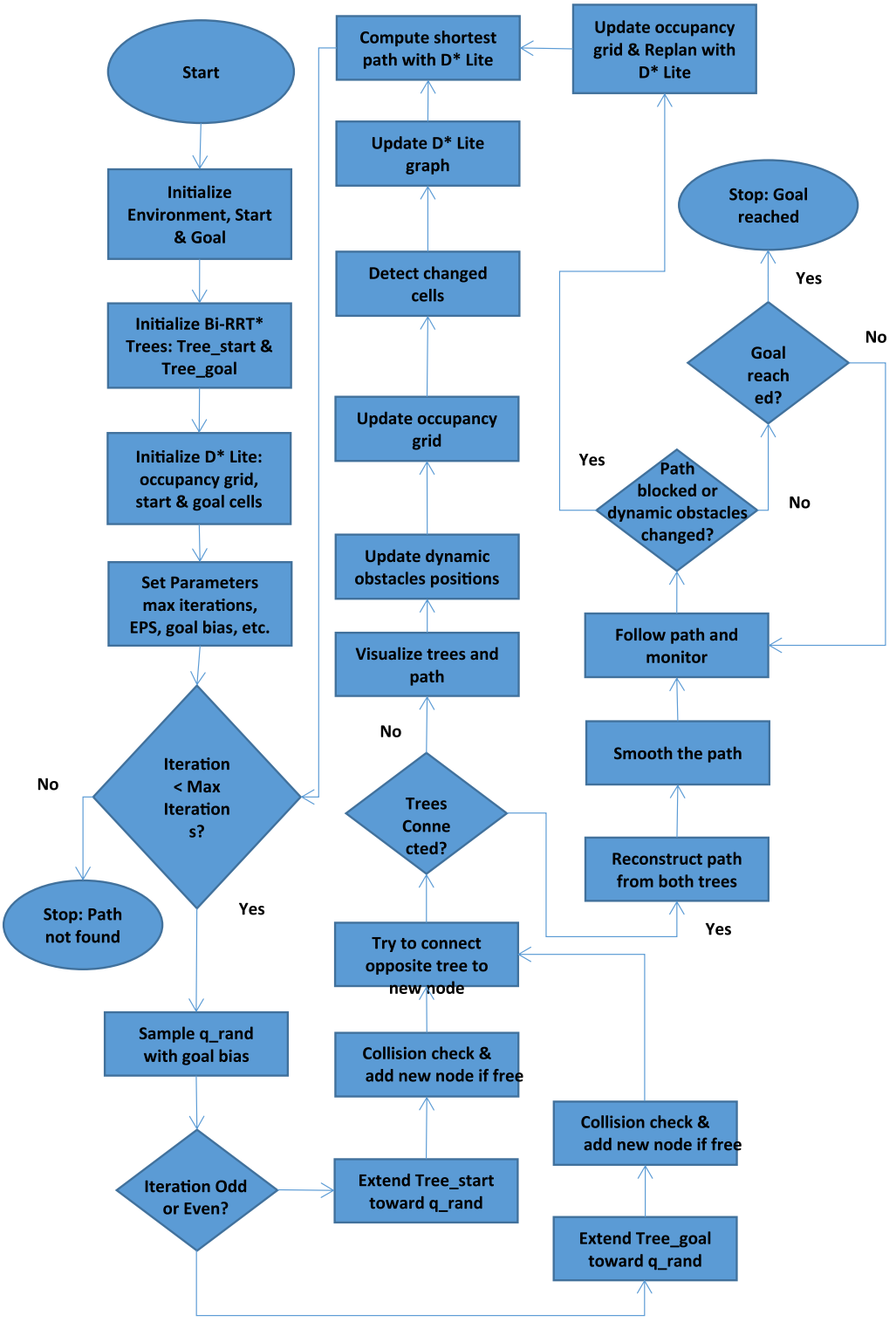

Fig. 1. Flowchart -BI-RRT*-D*lite approach [19].

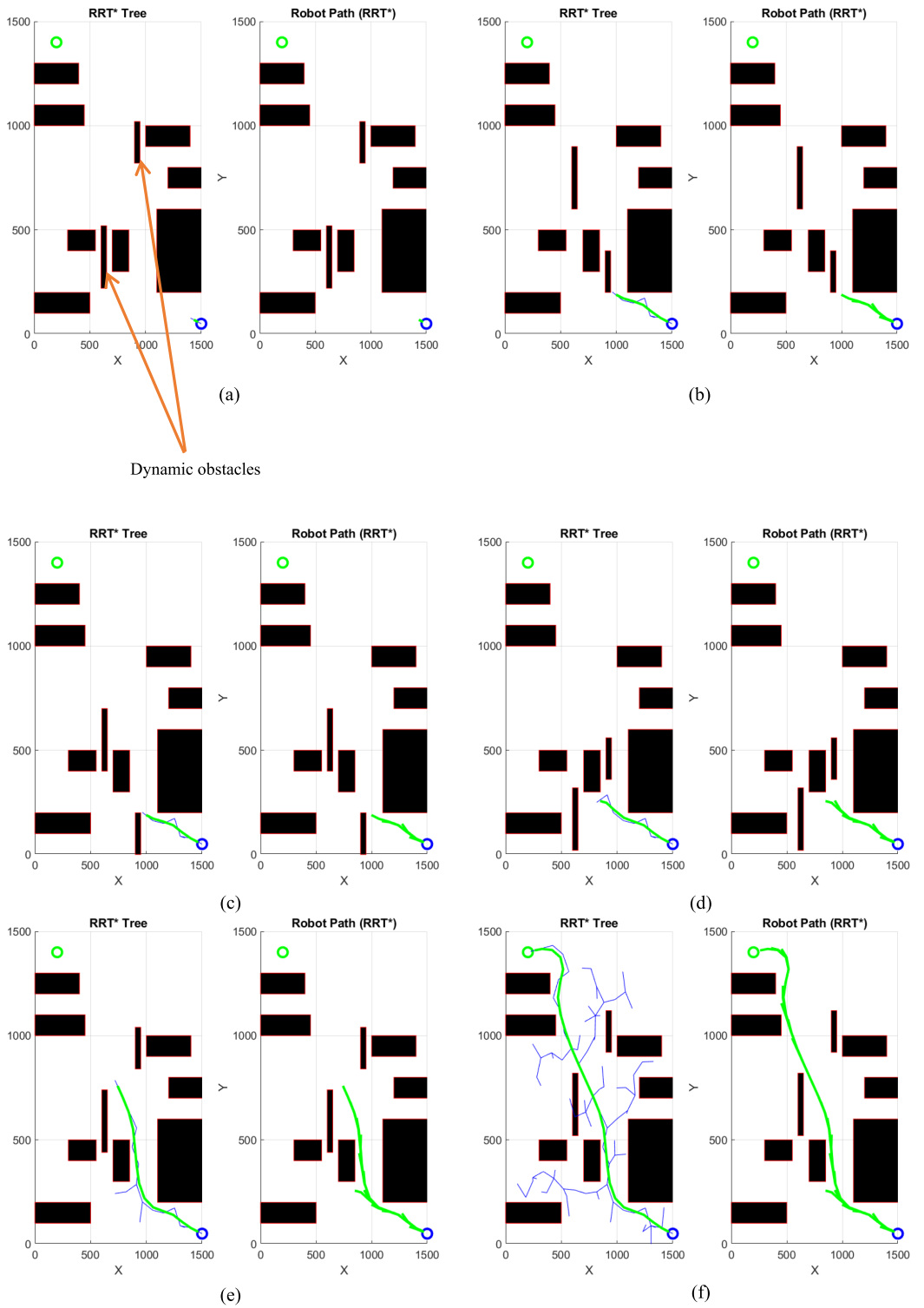

Fig. 2. Evaluation and performance analysis of the rrt* method in complex environments with static obstacles and a dynamic environment: (a) the start point is in a blue circle, the goal point is in a green circle, the rrt* tree is in black, and the final smooth path is in green. Fig. 2 (b-f) show the mass point motion while avoiding the obstacles.

Table 1. Results for RRT*, BI-RRT*-D*LITE.

Algorithm Type	Direct start-goal Distance (mm)	Total Path Length (mm)	Iteration No.	Rath Generating Time (s)	Percentage Error (%)
RRT*	1874.17	2118.7	1782	30.77	13.05
Bi-RRT*	1874.17	2111.76	158	4.24	12.68
Bi-RRT*-D* Lite	1874.17	1974.27	47	1.01	5.34

Table 2. Reduction percentages relative to RRT*.

Method	Path Length	Time	Search Attp.
Bi-RRT*	0.33	86	91
Bi0RRT*-D*Lite	6.8	97	97

3.2. Bi-RRT* implementation

The second algorithm utilized in this study is the Bi-RRT*, which is implemented under the same previous environment as that of the previous algorithm, the RRT*. In this scheme, the BI-RRT*is considered an extension of RRT*, by which the tree grows here with two extended trees. Thus, this extension happens as follows: one new branch grows from the start, while another tree grows from the goal. Nonetheless, attempts have been made to connect the generated trees during the exploration phase. The primary usage of each tree is based on the RRT* logic. This can be implemented by rewiring and optimal parent selection; they take turns growing toward the sampled points. To this end, whenever they get closer to each other, by a reasonable distance, they attempt to unite by their connection and form a complete path from the start to the goal point using the same previous environment. As in the previous implementation, Fig. 3 has its own 8 sub-figures for the same sequence of time evolving for the bi-RRT* algorithm. Table 1 shows the path planning results for the path length, the path generating time, the number of search attempts, and the percentage error. The Iteration reduction percentage [41] (see Eq. (1)), the path generating time reduction percentage, and the path reduction percentage compared with the RRT*-only are 91%, 86%, and 0.33 %, respectively, (see Table 2).

$$\text{Percentage Reduction} = \frac{Old\ Value\ \ (RRT^*) - New\ Vlaue}{Old\ Value\ \ (RRT^*)} * 100\% \quad (1)$$

Where:
Old Value**:** The value for the baseline algorithm (RRT***)**, since all reductions are relative to RRT***.**
New Value**:** The value for the compared algorithms (Bi-RRT* or Bi-RRT*-D* Lite**)**.

3.3. BI-RRT*-D*Lite

The third algorithm, which is adopted in this study, is the Bi-RRT*-D*Lite. This method is the upgraded version of the Bi-RRT*. To implement the utilized algorithm, the same previous environment is built. Thus, the main feature of the algorithm is that the main tree generates two local trees. Yet, this planning is a local re-planning; it is more efficient than global re-planning. However, this scenario does the tree growth: one from the start point, whilst the other growth one is generated from the goal point, and afterwards, the two grown trees attempt to connect with each other during the exploration phase. Similar

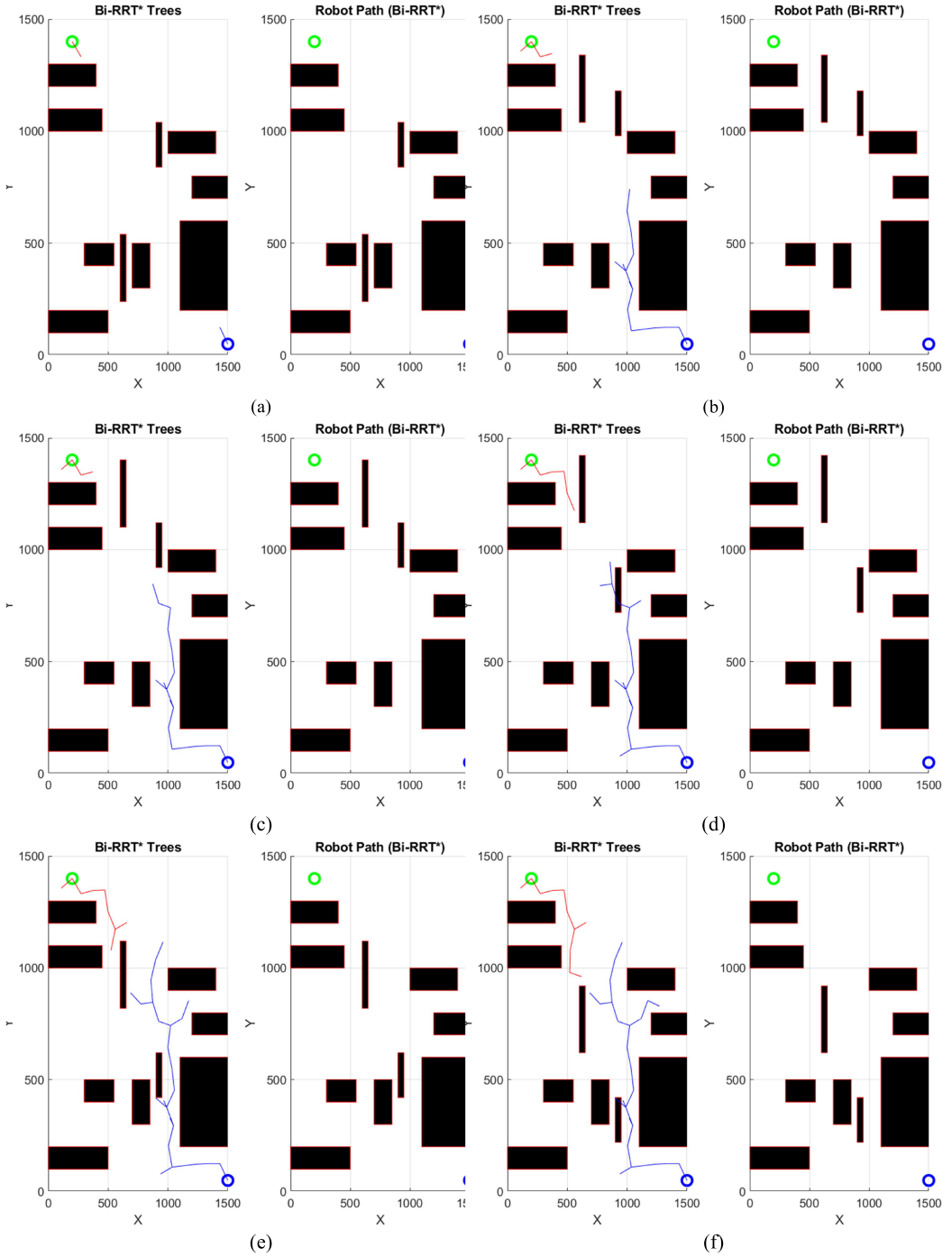

Fig. 3. Evaluation and performance analysis of the bi-rrt* method in complex environments with static obstacles and a dynamic environment: (a) the start point is in a blue circle, the goal point is in a green circle, odd iterations: extend **tree_start** toward q_rand in blue, even iterations extend **tree_goal** toward q_rand in red, (bf) the bidirectional tree growing. (g) the bidirectional tree connection (h) shows the smoothing of the final path, and the final smooth path is green.

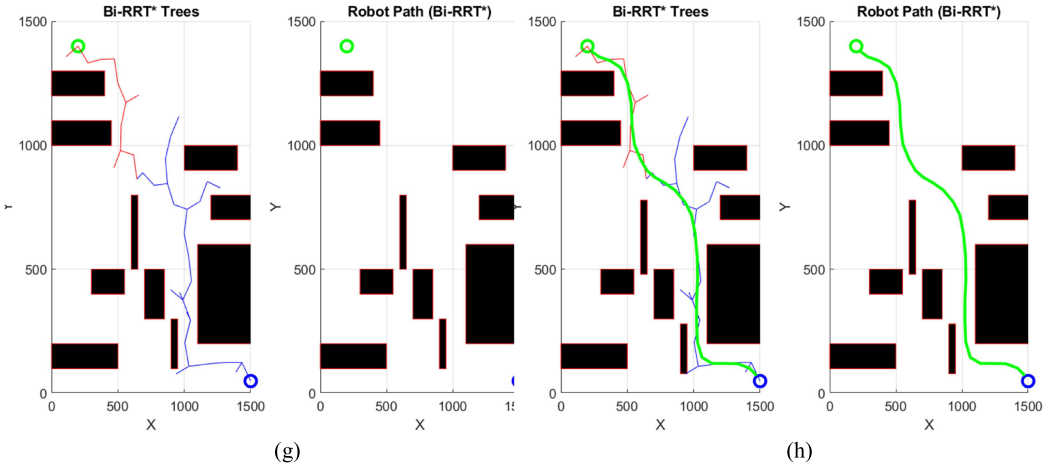

Fig. 3. Continued.

to Bi-RRT* growth, each tree uses RRT* logic. Therefore, with rewiring and optimal parent selection, they take turns growing toward sampled points. For a fair comparison, the environment properties are the same as those of the other two cases: RRT* and Bi-RRT*. When these trees are close enough to each other, they attempt a connection, forming a complete path from start to goal (see Fig. 4). This leads to minimizing the path generating time and the number of search attempts, as shown in Table 1. Therefore, we will get a higher reduction percentage of iteration, path generating time, and path reduction percentage compared with those of the RRT* at 97 %, 97%, and 6.8%, respectively.

The RRT* algorithm, operating with a single tree expanding from the start to the goal, successfully found feasible paths in a complex hybrid environment containing static and dynamic obstacles. However, it required a relatively high number of iterations and path generation time. In contrast, the Bi-RRT* algorithm demonstrated significant improvements in efficiency by growing two trees simultaneously, from the start and the goal, and attempting to connect them. This bidirectional search substantially reduced the number of iterations and path-generating time compared to RRT*. The path length obtained by the Bi-RRT* was marginally shorter than that of the RRT*, indicating improved path optimality alongside faster convergence due to better exploration of the environment. Integrating *D Lite** with Bi-RRT*, forming the hybrid Bi-RRT-D Lite** approach, further enhanced the performance by enabling local re-planning capabilities. This allowed dynamic adjustment to environmental changes, which is particularly useful with moving obstacles. The Bi-RRT*-D* Lite method achieved the highest reduction in iterations and the path-generating time, outperforming RRT* and Bi-RRT*. This confirms the hybrid approach's superior computational efficiency and adaptability.

Fig. 5 presents a grouped bar chart comparing the performance of RRT*, Bi-RRT*, and Bi-RRT*-D* Lite across four key metrics: path length, number of iterations, generating time, and percentage error. The results demonstrate that the Bi-RRT*-D* Lite consistently outperforms the other algorithms, achieving the shortest path length, the lowest number of iterations, the fastest generating time, and the smallest percentage error. These improvements highlight the efficiency and accuracy of the proposed Bi-RRT*-D* Lite approach for path planning in dynamic environments.

Regarding path quality, the Bi-RRT*-D* Lite produced the shortest and safest trajectories, reflecting the benefit of dynamic re-planning combined with the bidirectional tree growth,

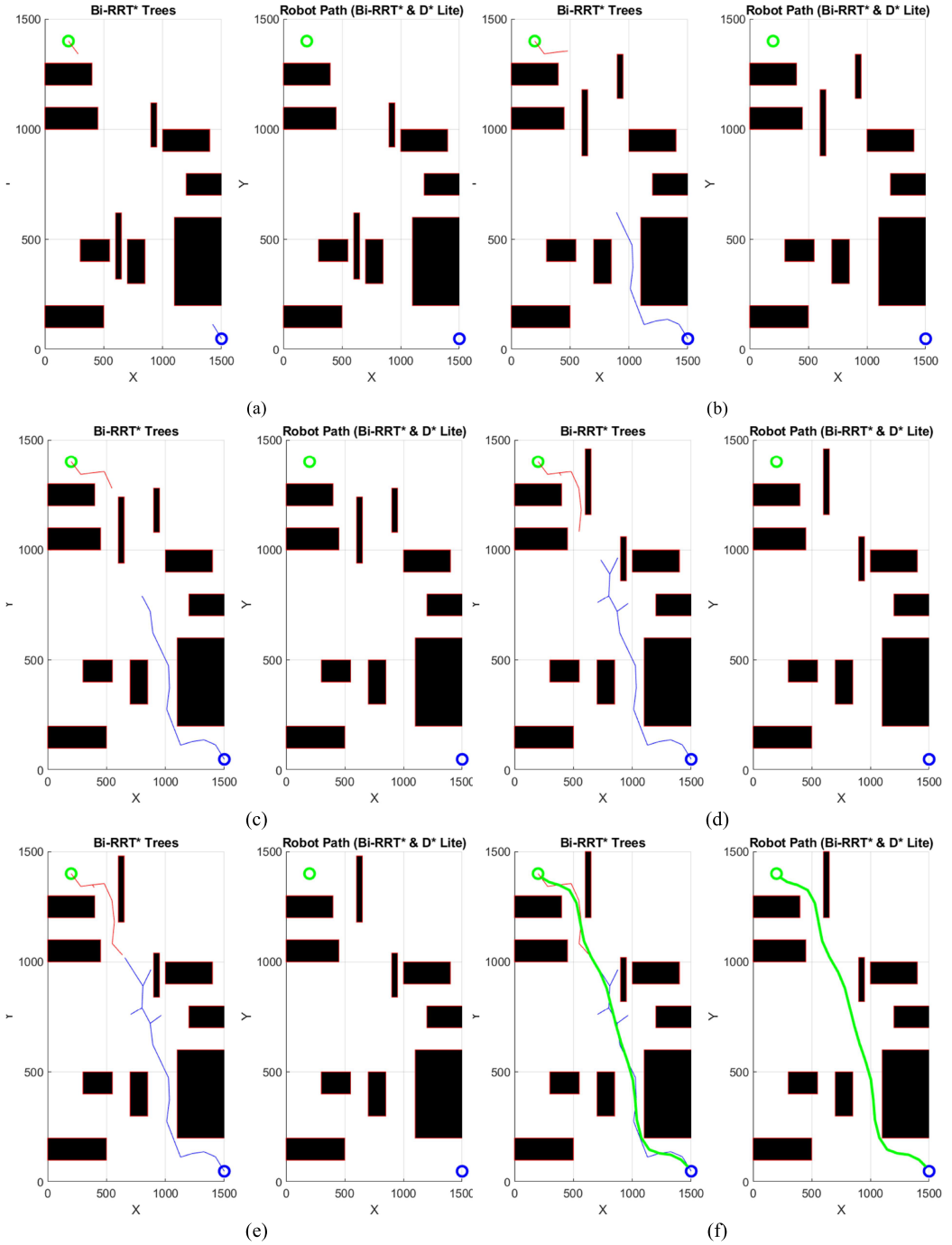

Fig. 4. Evaluation and performance analysis of the bi-rrt*-d*lite method in complex environments with static obstacles and a dynamic environment: (a) the start point is in a blue circle, the goal point is in a green circle, odd iterations: extend tree_start toward q_rand in blue, even iterations extend tree_goal toward q_rand in red, (b-d) the bidirectional tree growing. (e) the bidirectional tree connection (f) the smoothing of the final path, and the final smooth path is green.

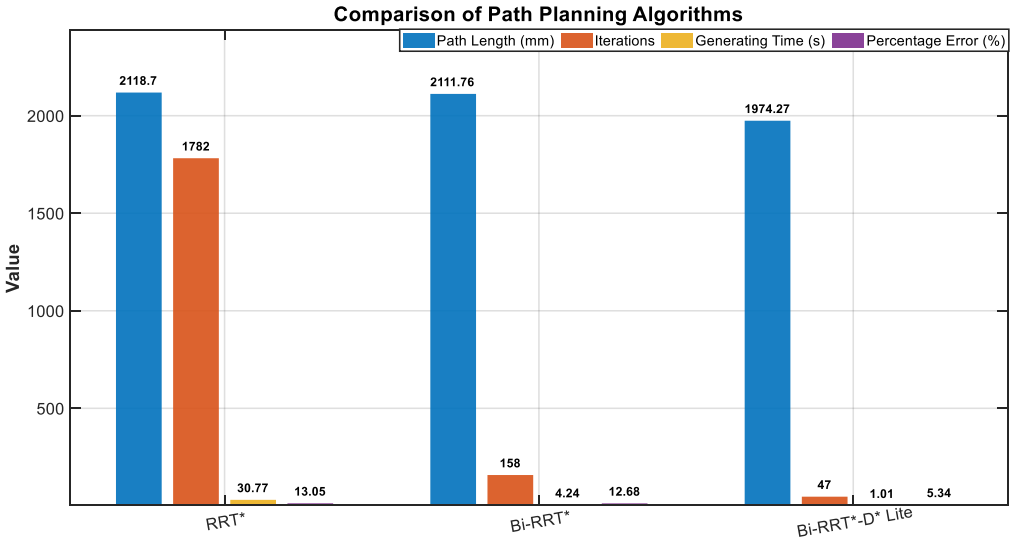

Fig. 5. Comparison results of path planning algorithms.

which reduces unnecessary exploration and redundancy. The results also showed a clear decrease in the percentage error (deviation from the direct start-goal distance) for the Bi-RRT*-D* Lite, illustrating improved accuracy in the planned paths. While the RRT* method is simpler to implement, its limitations become apparent in complex, dynamic environments due to slower convergence and less adaptability to changes. The Bi-RRT* offers a balanced trade-off between complexity and performance, providing faster path generation without the overhead of dynamic re-planning. Overall, the Bi-RRT*-D* Lite algorithm is the most robust and efficient method among the three methods, particularly suited for real-world robotic applications where the environments are dynamic and the safety-critical path planning is essential.

4. Conclusions

In this study, which demonstrates the effectiveness and comparison of combining the Bi-RRT* and the D* Lite algorithms for path planning in dynamic environments for a mass point, many concluding remarks can be listed here. In particular, such hybridization improves the efficiency of path generation and optimizes the path planning in terms of the time and the number of search attempts required to complete the task. Consequently, the results showed that the Bi-RRT*-D* Lite algorithm provides a more efficient path compared to that of the traditional RRT*, with significant reductions in all of the following: the iteration reduction percentage, the path generating time reduction percentage, and the path reduction percentage, where they are 97%, 97%, and 6.8%, respectively. The simulation results underlined the advantageous feature of combining the Bi-RRT* and the D* Lite, with the hybrid approach, resulting in faster, safer, and more efficient navigation of the robotic system in such complex dynamic environments. Additionally, by applying a Gaussian function as a filter to smooth the generated paths, one can observe a noticeable improvement in motion efficiency, reducing, or shrinking, unnecessary path deviations.

Some suggestions can be drawn as future work guidelines that can be addressed here. One suggestion is to adopt a robotic arm manipulator instead of just a single point in the

environment. However, the latter is challenging, such as an arm manipulator's physical body being considered rather than just one point in the workable environment.

On the other hand, another suggestion is to enhance the real-time path adjustments by allowing faster obstacles due to their detection and response. The final future guideline is to extend the study to contain multiple robots and optimize collaborative path planning. In this regard, reinforcement learning can improve decision-making and generate optimal path planning. Moreover, the suggestion can be extended to integrate energy constraints in minimizing power consumption while maintaining path optimality.

Acknowledgments

I want to express my appreciation, respect, and special thanks to my supervisor, *Prof. Dr. Omar Farouq Lutfy* and *Asst. Prof. Dr. Firas A. Raheem* for all the patience and time spent in providing the necessary guidance and scientific explanations that enabled me to complete this work. His expertise and insight have been invaluable, and his belief in my abilities has motivated me to push through the most challenging times.

Conflict of interest statement

The author, Hameed S. Hameed, declares no conflict of interest. The author has no relevant financial or non-financial interests to disclose.

Data availability

The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request. The authors confirm that the data supporting the findings of this study are available within the article and its supplementary materials.

References

[1.] N. AbuJabal, M. Baziyad, R. Fareh, B. Brahmi, T. Rabie, and M. Bettayeb, "A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots," *Sensors (Basel, Switzerland),* vol. 24, no. 24, p. 8089, 2024.

[2.] L. Bo *et al.*, "3D UAV path planning in unknown environment: A transfer reinforcement learning method based on low-rank adaption," *Advanced Engineering Informatics,* vol. 62, p. 102920, 2024.

[3.] L. Zhang *et al.*, "Motion planning for robotics: A review for sampling-based planners," *Biomimetic Intelligence and Robotics,* p. 100207, 2025.

[4.] D. K. Muhsen, F. A. Raheem, Y. Yusof, A. T. Sadiq, and F. Al Alawy, "Improved rapidly-exploring random tree using firefly algorithm for robot path planning," *Journal of Soft Computing and Computer Applications,* vol. 1, no. 2, p. 1, 2024.

[5.] S. Nahavandi *et al.*, "A comprehensive review on autonomous navigation," *ACM Computing Surveys,* vol. 57, no. 9, pp. 1–67, 2025.

[6.] Z. Zhang, B. Qiao, W. Zhao, and X. Chen, "A predictive path planning algorithm for mobile robot in dynamic environments based on rapidly exploring random tree," *Arabian Journal for Science and Engineering,* vol. 46, no. 9, pp. 8223–8232, 2021.

[7.] Z. Wu, Z. Meng, W. Zhao, and Z. Wu, "Fast-RRT: A RRT-based optimal path finding method," *Applied sciences,* vol. 11, no. 24, p. 11777, 2021.

[8.] J. Ding, Y. Zhou, X. Huang, K. Song, S. Lu, and L. Wang, "An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling," *Journal of Computational Science,* vol. 67, p. 101937, 2023.

[9.] X. Li, Y. Lu, X. Zhao, X. Deng, and Z. Xie, "Path planning for intelligent vehicles based on improved D* Lite," *The Journal of Supercomputing,* vol. 80, no. 1, pp. 1294–1330, 2024.

[10.] Y. Gao, Q. Han, S. Feng, Z. Wang, T. Meng, and J. Yang, "Improvement and fusion of D* lite algorithm and dynamic window approach for path planning in complex environments," *Machines,* vol. 12, no. 8, p. 525, 2024.
[11.] J. Yu *et al.*, "Path planning of unmanned surface vessel in an unknown environment based on improved D* Lite algorithm," *Ocean Engineering,* vol. 266, p. 112873, 2022.
[12.] Z. Yao, J. Sun, C. Han, and L. Jin, "A path-planning algorithm integrated improved A* and D* Lite with bat clustering," *Journal of Computational Methods in Sciences and Engineering,* vol. 25, no. 3, pp. 2159–2184, 2025.
[13.] Y. Zhang, J. Luo, X. Cai, Y. Chen, E. Peng, and X. Zou, "AGV path planning for logistics warehouse by using an improved D* Lite algorithm," in *International conference on the Efficiency and Performance Engineering Network*, 2022: Springer, pp. 1018–1027.
[14.] J. Jin, Y. Zhang, Z. Zhou, M. Jin, X. Yang, and F. Hu, "Conflict-based search with D* lite algorithm for robot path planning in unknown dynamic environments," *Computers and Electrical Engineering,* vol. 105, p. 108473, 2023.
[15.] Z. Lin, L. Lu, Y. Yuan, and H. Zhao, "A novel robotic path planning method in grid map context based on D* lite algorithm and deep learning," *Journal of Circuits, Systems and Computers,* vol. 33, no. 04, p. 2450057, 2024.
[16.] X. Zhu, B. Yan, and Y. Yue, "Path planning and collision avoidance in unknown environments for USVs based on an improved D* lite," *Applied Sciences,* vol. 11, no. 17, p. 7863, 2021.
[17.] J. Fan, X. Chen, and X. Liang, "UAV trajectory planning based on bi-directional APF-RRT* algorithm with goal-biased," *Expert systems with applications,* vol. 213, p. 119137, 2023.
[18.] M. T. Hameed, F. A. Raheem, and A. R. Nasser, "Hybrid Path Planning for Robotics: APF-RRT Enhancement with Sobol Sequence Integration," *International Journal of Intelligent Engineering & Systems,* vol. 18, no. 3, 2025.
[19.] J. Xu, Z. Tian, W. He, and Y. Huang, "A fast path planning algorithm fusing prm and p-bi-rrt," in *2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)*, 2020: IEEE, pp. 503–508.
[20.] X. Shu, F. Ni, Z. Zhou, Y. Liu, H. Liu, and T. Zou, "Locally Guided Multiple Bi-RRT∗ for Fast Path Planning in Narrow Passages," in *2019 IEEE international conference on robotics and biomimetics (ROBIO)*, 2019: IEEE, pp. 2085–2091.
[21.] P. Xin, X. Wang, X. Liu, Y. Wang, Z. Zhai, and X. Ma, "Improved bidirectional RRT* algorithm for robot path planning," *Sensors,* vol. 23, no. 2, p. 1041, 2023.
[22.] G. Ma, Y. Duan, M. Li, Z. Xie, and J. Zhu, "A probability smoothing Bi-RRT path planning algorithm for indoor robot," *Future Generation Computer Systems,* vol. 143, pp. 349–360, 2023.
[23.] H. Fan, J. Huang, X. Huang, H. Zhu, and H. Su, "BI-RRT*: An improved path planning algorithm for secure and trustworthy mobile robots systems," *Heliyon,* vol. 10, no. 5, 2024.
[24.] L. Wang, Y. Zhang, and C. Guo, "Path planning for a prostate intervention robot based on an improved Bi-RRT algorithm," *IEEE/ASME Transactions on Mechatronics,* vol. 30, no. 1, pp. 668–678, 2024.
[25.] R. Yang, P. Cai, and L. Wang, "Comparison of strategies for optimizing bi-rrt* on mobile robots," in *2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE)*, 2021: IEEE, pp. 328–334.
[26.] M. Zhang, Y. Ma, and N. Xi, "Bi-RRT* based trajectory optimization and obstacle avoidance for a serial manipulator," in *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2021: IEEE, pp. 163–168.
[27.] B. Tao and J.-H. Kim, "Deep reinforcement learning-based local path planning in dynamic environments for mobile robot," *Journal of King Saud University-Computer and Information Sciences,* vol. 36, no. 10, p. 102254, 2024.
[28.] M. Yao, H. Deng, X. Feng, P. Li, Y. Li, and H. Liu, "Global Path Planning for Differential Drive Mobile Robots Based on Improved BSGA* Algorithm," *Applied Sciences,* vol. 13, no. 20, p. 11290, 2023.
[29.] D. K. Muhsen, F. A. Raheem, and A. T. Sadiq, "A Systematic Review of Rapidly Exploring Random Tree RRT Algorithm for Single and Multiple Robots," *Cybernetics and Information Technologies,* vol. 24, no. 3, 2024.
[30.] Z. Shen, J. Wilson, R. Harvey, and S. Gupta, "Smarrt: Self-repairing motion-reactive anytime rrt for dynamic environments," *arXiv preprint arXiv:2109.05043,* 2021.
[31.] X. He, Y. Zhou, H. Liu, and W. Shang, "Improved RRT*-Connect Manipulator Path Planning in a Multi-Obstacle Narrow Environment," *Sensors,* vol. 25, no. 8, p. 2364, 2025.
[32.] Q. Chai and Y. Wang, "RJ-RRT: improved RRT for path planning in narrow passages," *Applied Sciences,* vol. 12, no. 23, p. 12033, 2022.
[33.] Z. Liu, R. M. Sampurno, R. R. D. Abeyrathna, V. M. Nakaguchi, and T. Ahamed, "Development of a collision-free path planning method for a 6-DoF orchard harvesting manipulator using RGB-D camera and Bi-RRT algorithm," *Sensors,* vol. 24, no. 24, p. 8113, 2024.

[34.] J. O. Adibeli and Y.-K. Liu, "Radiation and ergonomic-aware A-star algorithm for nuclear decommissioning path planning," *Nuclear Engineering and Design,* vol. 439, p. 114123, 2025.
[35.] D. K. Muhsen, A. T. Sadiq, and F. A. Raheem, "A survey on swarm robotics for area coverage problem," *Algorithms,* vol. 17, no. 1, p. 3, 2023.
[36.] V. Sangeetha, R. Krishankumar, K. Ravichandran, and S. Kar, "Energy-efficient green ant colony optimization for path planning in dynamic 3D environments," *Soft Computing,* vol. 25, no. 6, pp. 4749-4769, 2021.
[37.] S. Sulaiman and A. Sudheer, "Modeling of a wheeled humanoid robot and hybrid algorithm-based path planning of wheel base for the dynamic obstacles avoidance," *Industrial Robot: the international journal of robotics research and application,* vol. 49, no. 6, pp. 1058–1076, 2022.
[38.] Y. Li, F. Yang, X. Zhang, D. Yu, and X. Yang, "Improved D* lite algorithm for ship route planning," *Journal of Marine Science and Engineering,* vol. 12, no. 9, p. 1554, 2024.
[39.] K. C. Ugwoke, N. A. Nnanna, and S. E.-Y. Abdullahi, "Simulation-based review of classical, heuristic, and metaheuristic path planning algorithms," *Scientific Reports,* vol. 15, no. 1, p. 12643, 2025.
[40.] D.-S. Huang, K.-H. Jo, and X.-L. Zhang, *Intelligent Computing Theories and Application: 14th International Conference, ICIC 2018,* Wuhan, China*, August 15–18,* 2018*, Proceedings, Part II*. Springer, 2018.
[41.] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. John wiley & sons, 2019.