IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 August 2024, accepted 27 August 2024, date of publication 30 August 2024, date of current version 10 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3452277

RESEARCH ARTICLE

Accelerated Path Planning for Large-Scale Grid Maps

DUOHANG SUN, ZHE SUN, AND PEINAN SHAO
The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 200000, China
Corresponding author: Duohang Sun (duohangsun99@outlook.com)

ABSTRACT Path planning is a critical task in automated navigation and seeks to identify optimal collision-free routes for autonomous systems such as unmanned vehicles, aircrafts, and surface ships within a specified environment. The rapid computation of optimal paths in large-scale grid maps remains a major challenge. This paper presents an enhanced HPA* pathfinding algorithm that utilizes an abstract representation of grid maps to facilitate fast navigation. The enhancements to the HPA* algorithm include a detailed examination of neighborhood branch extensions, the incorporation of a high-quality heuristic function, and the implementation of repulsive force fields. The effectiveness of this improved HPA* algorithm in large-scale grid environments is demonstrated by extensive experiments in real-world settings. In terms of computational efficiency, for path planning in real scenarios, the computation time of the HPA* Improved algorithm can be reduced by more than 95% relative to the A* algorithm, and the computation time can still be reduced by more than 80% relative to the HPATheta* algorithm, even though the computation time can be reduced by more than 80%. Even compared to HPATheta* algorithm, the computation time can still be reduced by more than 80%. Specifically, the improved HPA* algorithm significantly reduces the time required to generate path results while also enhancing path safety. This study provides valuable insights into advanced pathfinding techniques that could advance automated navigation systems.

INDEX TERMS Abstract map, improved HPA* algorithm, large-scale maps, path planning.

I. INTRODUCTION

Path planning, which involves the computation of a sequence of waypoints that describe the desired trajectory for an agent's movement, is pivotal in the field of automated navigation. This process is critical for autonomous navigation systems, such as unmanned vehicles [1], unmanned aircraft [2], and unmanned surface ships [3], which must find an optimal, collision-free path from an initial position to a target location within a specified environment. The optimal path is typically defined by multiple criteria (i.e., minimization of travel distance, travel time, and energy consumption, etc.). Therefore, path planning is considered an optimization problem [4] where the primary challenge is to solve for an objective function under a set of relevant parameters. This objective function encapsulates several key questions: (1) *What is the current location of the agent?* (2) *What is the destination?* (3) *What is the most efficient route from the current location to the destination that satisfies the given objective function?*

Strategies to address these challenges associated with the interdependencies inherent to the path planning process are categorized based on the operational context and underlying principles. These classifications include static versus dynamic path planning, global versus local path planning, and offline versus online path planning [5], [6], [7]. Global path planning usually assumes that the environment is static and involves the identification of an optimal path from the start point to the end point within the whole environment. Meanwhile, local path planning presupposes a dynamic environment and zeroes in the most feasible trajectory of moving from the current position to the target position. Path planning for large-scale scenarios typically involves global path planning. In large-scale maps, path planning can further be divided into two categories:'' preprocessing — path planning'' and ''direct path panning.'' The former is advantageous for large-scale environments where the initial map

The associate editor coordinating the review of this manuscript and approving it for publication was P. K. Gupta.

processing step can significantly streamline the subsequent path finding process. In this method, techniques such as the artificial potential field method [8], [9] and abstract hierarchical method [10] are employed to simplify environmental complexity, which facilitates more efficient planning. This abstract hierarchical method is appropriate for scenarios that require immediate responses and adaptability. The core algorithms used in path planning include traditional search-based methods such as Depth-first search, Breadth-first search, Best-first search, Dijkstra's algorithm, and the A* algorithm, along with its variants (Hybrid A*, D*, Theta*, and Lifelong Planning A*). Additionally, the Rapidly-explore random tree (RRT) algorithm and its derivatives, such as RRT* and Informed RRT*, are extremely useful for navigating complex and dynamic environments. Notably, path planning also benefits from swarm intelligence algorithms, including ant colony and particle swarm optimization, as well as evolutionary algorithms [11], [12], which simulate natural processes to evolve and adapt solutions.

Recent studies have explored various methods to improve computational efficiency and performance. For example, Gu et al. developed a fast linearized virtual element method (VEM) for nonlinear time-fractional diffusion equations [13]. Liu et al. investigated large-scale video processing using double-buffer optimization, ant colony algorithms, and distributed online concurrent editing [14]. Wang et al. introduced a dimensionality reduction technique combined with fuzzy clustering to reduce computational overhead and enhance model generalization [15]. Liu et al. proposed a low-cost link aggregation method for concurrent video streaming transmission, significantly improving performance metrics [16].

The transition from these general methods to more specific path-planning algorithms that operate on grid maps also requires effective search strategies. These algorithms are essentially search-based, and the path is continuously constructed by moving from the current position to a subsequent position, with incremental queries being generated until the target position is reached. While simple search algorithms such as Depth-first search, Breadth-first search, and Best-first search are not specifically tailored for path planning and may incur higher computational costs, they lay the groundwork for more optimized solutions. For instance, Dijkstra's algorithm — which aims to find the shortest path from the start position to the target position [17] — uses a data structure to store the shortest distance from the start node to the current node and traverses all the nodes within a loop to obtain the shortest path.

The A* algorithm employs heuristics derived from Dijkstra's algorithm [18]. The evaluation function of the A* algorithm and its variants typically consist of two parts:

$$f(n) = g(n) + h(n) \tag{1}$$

Here, $g(n)$ represents the actual distance from the current node n to the start position, and $h(n)$ represents the heuristic value, which is typically the estimated distance from node n to the target. Unlike Dijkstra's algorithm, the A* algorithm and its variants use heuristics to reduce the number of node extensions, thereby accelerating the search process. However, although its heuristic values reduce the number of expanding nodes to some extent, the A* algorithm remains inadequate in large-scale environments because it does not scale well when grid sizes expand. Therefore, Ammar et al. introduced two new temporally linearly relaxed versions of Dijkstra's algorithm (RD) and the A* algorithm for global path planning in large-scale grid environments (RA*), accurately approximating the optimal path without revesting any of the cells [19]. Harabor et al. proposed the Jump Point Search (JPS) algorithm as an optimization of the A* algorithm. Notably, the JPS algorithm could accelerate pathfinding on grid maps by reducing access to open and closed lists and limiting symmetric path exploration through inflection points. However, JPS was found to underperform in unstructured environments and those with multiple pathways and high randomness [20]. To address this problem, Chen et al. developed a hybrid optimization algorithm that combines an improved ant colony algorithm with JPS to enhance path accuracy and minimize turns [21]. Furthermore, Sang et al. proposed a hybrid strategy that combines the improved A* algorithm with the artificial potential field method, transforming the global path into a series of subgoal points to improve pathfinding effectiveness and reduce the risk of local minima entrapments [22]. Alajlan et al. experimentally demonstrated that a genetic algorithm could find optimal paths in large-scale grid environments as effectively as the A* algorithm [23]. Zhong et al. developed a hybrid path planning method that combines the A* algorithm with an adaptive windowing approach for global path planning, real-time tracking, and obstacle avoidance [24]. Finally, Holte et al. explored the application of graph abstraction for accelerating the pathfinding process between two nodes, leading to more rapid solutions [25]. Subsequently, hierarchical A* algorithms were adapted to accelerate path planning under various scenarios [26]. Zheng et al. proposed to use the PRM algorithm for two-stage path planning for long-distance off-road scenarios, which is similar in nature to the hierarchical idea, combined with the improvement of the PRM algorithm, which improves the computational efficiency of the algorithm, and carries out the enhancement of the applicability of long-distance path planning in large-scale scenarios [27]. For long-range, large-scale, high-resolution path planning for grid maps, computational efficiency remains the primary challenge, along with the need to ensure that the suboptimality of paths is within reasonable limits.

This paper explores three fundamental and effective approaches—abstract graph construction, neighborhood expansion, and heuristic functions—to decrease the number of visited nodes, enhance path security and ensure paths closely approximate the true shortest paths. Extensive experiments and analysis confirm the effectiveness and superiority of our proposed method in large-scale environments.

II. RELATED LITERATURE

The idea of larger-scale hierarchical maps has a long history. Holte et al. first proposed the hierarchical A* algorithm in 1996 [26]. This algorithm transforms the original graph into an abstract graph and hierarchizes it, thus reducing node accesses and speeding up path planning. Subsequently, Harabor et al. proposed a hierarchical path planning algorithm (HPA*) for grid maps that could be applied to a wide range of agents of different sizes [28]. The HPA* algorithm achieves hierarchy by dividing the grid map into n square clusters and determining the paths of different agents by studying the connectivity between the square clusters and labeling them on a hierarchical abstraction map. Similarly, the CH algorithm for grid maps — first showcased in a Grid-Based Path Planning Competition — constructs a directed acyclic graph (DAG) and reduces map complexity by hierarchizing it to accelerate the grid map path planning [29]. Pelechano and Fuente proposed the development of a hierarchical representation based on a multilevel k-way partitioning algorithm (MLkP) that has labeled sub-paths, which can then be accessed online by the Hierarchical NavMesh Path-finding algorithm (HNA*) that is employed for navigation meshes [30]. Ting et al. proposed a path planning method for a grid map that combines the hierarchical approach with an ant colony algorithm, where the grid map is a simulation of the unstructured lunar surface [31]. Chagas et al. proposed a hierarchical path planning algorithm for HPATheta* because path planning was too time-consuming when dealing with large-scale real virtual terrains, and this algorithm could compute smooth paths that took terrain features into account [32]. Overall, for large-scale grid maps, the hierarchical abstraction approach has emerged as an effective way to reduce computing time, allowing path-planning activities to be completed within a reasonable timeframe. Moreover, since a grid map can also be considered an image, researchers like Moghadam et al. have used the Sobel operator to perform a convolution operation and explore the effect of weight differences between neighboring nodes on the current node as a measure of weighted grid map complexity [33]. Meanwhile, Kim and Sull have proposed the application of convolution to reduce the search space of the grid map and decrease the time required for path planning [34].

Although all of the above scholars have made many contributions to the efficiency of path planning, further algorithmic improvements are still needed for large-scale high-resolution scenarios that include both on-road and off-road environments.

III. ENVIRONMENTAL MODELING

Path planning involves a variety of map types, including occupancy grid maps, Voronoi graphs, visibility graphs, and probabilistic roadmaps [12]. Each type of map possesses unique characteristics that influence the strategies and outcomes of path planning. Occupancy grid maps, for instance, are particularly effective for modeling large-scale environments because they discretize the space into smaller grid cells. These maps depict the occupancy status of spatial locations and can store diverse information. However, the accuracy of this method in the context of path planning is limited by the resolution it offers. Furthermore, while higher resolutions yield more complex data, they also require increased memory and computational resources.

In real-world scenarios, large-scale grid maps must be considered from two perspectives: the large number of grids (exceeding ten million) and the broad geographic area. For instance, a 5000 × 5000 grid map encompasses 25,000,000 cells, and it covers an area of 10 × 10 km if each cell represents a 2 × 2 m region. These grid maps are typically constructed based on remote sensing images or LiDAR data, and elevation details are derived from geological terrain analyses to produce corresponding accessibility maps. This dual consideration of both scale and detail is crucial for effectively deploying large-scale grid maps in path-planning applications [35].

Assuming that the grid map has a total height of h and a width of w, the grid map can be mathematically expressed as:

$$\text{Grid} = \left\{ cell_{(i,j)} \,|\, 0 \le \text{i} < \text{h}, 0 \le \text{j} < \text{w}, \text{i}, \text{j} \in \text{z} \right\} \quad (2)$$

The value of each $cell_{(i,j)}$ term indicates the state of the grid (i, j):

$$cell_{(i,j)} = \begin{cases} 0, & \text{if grid (i, j) is a non} - \text{obstacle grid} \\ 1, & \text{if grid (i, j) is free} \end{cases} \quad (3)$$

Under these conditions, a path can be represented as a collection of non-obstacle grids in partial order:

$$\text{Path} = \left\{ \begin{array}{l} cell_{(i1,j1)}, cell_{(i2,j2)}, \ldots, cell_{(in,jn)}, cell_{(ik,jk)} \in \\ \text{Grid} \cap cell_{(ik,jk)} = 0 \end{array} \right\} \quad (4)$$

Here, $cell_{(i1,j1)}$ denotes the starting position of the agent, and $cell_{(in,jn)}$ denotes the target position.

The path cost is denoted as:

$$Cost\ (Path) = \sum_{k=1}^{n-1} d\ (k), \quad (5)$$

Here, $d(k)$ is the Euclidean distance between the k$^{\text{th}}$ grid $cell_{(ik,jk)}$ and the k+1 grid $cell_{(ik+1,jk+1)}$ in the constituent path, as follows:

$$d\ (k) = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \quad (6)$$

We present an abstract simulation of real environments using three map patterns, as illustrated in Figure 1. The maps are named according to the format "MapType_Size_X," where "MapType" includes corridor, room, and random arrangement, and "Size" denotes the dimensions in the format Size×Size. The variable "X" holds different meanings depending on the map type.

Map patterns represent the obstacle distribution within a particular region. In structured environments, there are

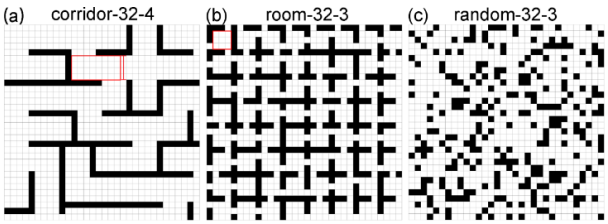

FIGURE 1. Schematic diagrams of the three different map types. (a) Map type "Corridor-32-4" corresponds to a corridor-style map with a width of 4 units and dimensions of 32 × 32. Here, "X" represents the corridor width. (b) Map type "Room-32-3" describes a room-style map featuring a single room that is 3 × 3 units in size, within a total area of 32 × 32. Here, "X" indicates the room size. Map type (c) "Random-32-3" is a random map, which is also sized 32 × 32 units, where obstacles occupy 30% of the grid area. Here, "X" specifies the obstacle coverage percentage.

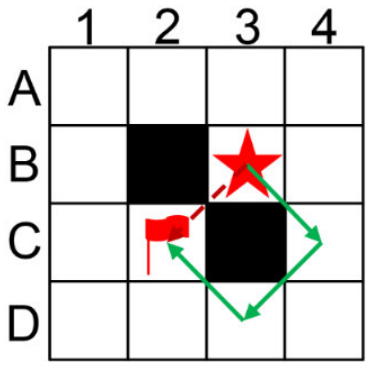

FIGURE 2. Prohibition of diagonal obstacle crossing. The red dashed path is not traversable due to diagonal barriers, while the solid green arrows show the feasible route.

well-defined connections between obstacles. However, random environments feature randomized and unstructured obstacles without clear connections. The transition from structured to unstructured environments in the different maps mirrors real-world scenarios wherein regular streets and buildings co-exist with randomly placed obstacles. When using higher resolution for high-precision modeling of larger geographic areas, the number of grids in the grid maps obtained from the modeling grows dramatically due to the increased resolution of the maps. Path planning in such high-resolution environments is significantly more demanding in terms of memory, computation, and time. This paper aims to develop a method that accelerates path planning in large-scale grid maps while optimizing path costs.

Assuming that the actual geographic area represented by each cell is relatively small, if a grid cell is found to contain an obstacle, the accessible area around this cell is considered extremely limited. The geographic range represented by each cell in the original map is quite small, corresponding to a geographic region sized 2 × 2 m. In cases where a cell acts as an obstacle, the available space for access in that area is severely limited. Consequently, this study assumes that traversal between diagonal obstacles is not feasible, as illustrated in Figure 2. Here, with the start position corresponding to (B, 3), the target position corresponding to (C, 2), and the obstacles located at (B, 2) and (C, 3), the red dashed path is prohibited. Meanwhile, the green route represents the shortest permissible path from the start to the target location.

IV. METHODOLOGY AND NUMERICAL ANALYSIS

In this section, we describe our proposed algorithm, which explores how large-scale grid map path planning can be accelerated through three perspectives: abstract graph construction, neighborhood extension, and the use of heuristic functions.

A. ABSTRACT GRAPH CONSTRUCTION

We employ an abstract map construction scheme, as depicted in Figure 3. Specifically, this approach involves the processing of an original grid map with blocks (top-left, Figure 3), the calculation of the external and internal connectivity of each block (upper-middle and lower-middle, Figure 3), and the subsequent transformation of the grid map into an abstract map (bottom-right, Figure 3). The calculation process for each step is as follows.

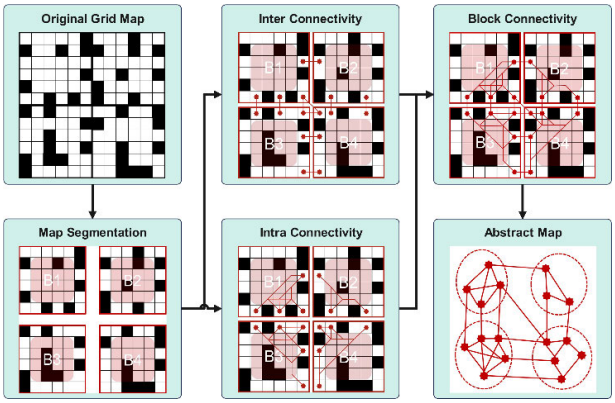

FIGURE 3. Schematic diagram of the proposed grid map abstraction process. (a) Top-left, original grid map; (b) Bottom-left, map segmentation; (c) Upper-middle, inter-connectivity; (d) Lower-middle, intra-connectivity; (e) Top-right, block connectivity; (f) Bottom-right, abstract map.

Grid Initialization: In our setting, the original grid map size is defined as X×X, and the block size is Y×Y. Following abstraction, the grid map is reduced to a size of (X/Y) in each dimension, resulting in a total of $(X/Y)^2$ blocks. This forms the basis for the hierarchical grid map.

Inter-Connectivity Calculation: After segmenting the original map into blocks, it is crucial to establish the connectivity between the blocks to assess their external connectivity. This step ensures the presence of valid connections between the blocks. For instance, as shown in Figure 4, blocks B1, B4, and B2 lack interconnections; hence, if the start or end point of a path lies in B2, path planning would fail due to the lack of connections.

Intra-Connectivity Calculation: While inter-connectivity determines how well-connected a block is externally, it is not sufficient to determine whether a block is suitable for path formation. Intra-connectivity, on the other hand, assesses the traversability within a particular block. Consider the scenario

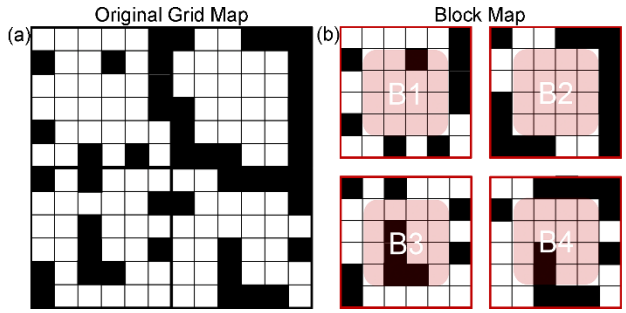

FIGURE 4. Schematic diagrams of the original grid map and block map for the process of Inter-connectivity calculation. (a) Original grid map. (b) Block map. After dividing (a) into 6 × 6 units, B2 becomes completely isolated from the other blocks (B1, B3, and B4). Note that B2 is not entirely enclosed.

depicted in Figure 5. Although there exists a nominal connection between B1 and B2, and between B2 and B3, B2 does not provide actual access to the other areas within B1 or B3 through these external links. This highlights the importance of intra-connectivity in functional path planning within a grid map.

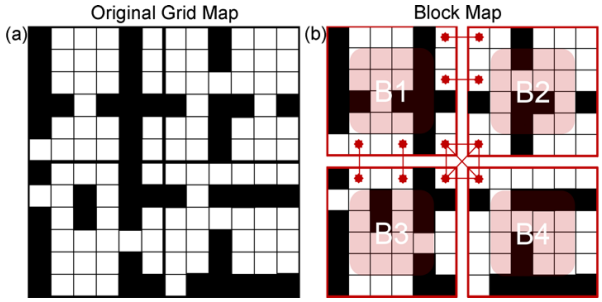

FIGURE 5. Schematic diagrams of the original grid map and block map for the process of intra-connectivity calculation. (a) Original grid map. (b) Block map. In the block map (b), certain blocks are non-traversable internally, which impedes movement between connected blocks.

After calculating the internal and external connectivity, the original large-scale grid can be transformed into an abstract map. The abstract path from the start location to the target location can be calculated using this abstract map, as shown in Figure 6:

$$AbsPath = \left\{ \begin{array}{l} abscell_{(i1,j1)}, abscell_{(i2,j2)}, \ldots, abscell_{(in,jn)}, \\ abscell_{(in,jn)} \in Grid \end{array} \right\} \tag{7}$$

where $abscell_{(i1,j1)}$ is the starting position of the agent, and $abscell_{(in,jn)}$ denotes the target position. Meanwhile, $abscell_{(ik,jk)} 1 < k < n$ denotes the abstract points on the abstract map after hierarchical abstraction that correspond to the boundary grids after dividing the original grid map into blocks. The final path is obtained through replanning based on the abstract path.

$$Path = \bigcup_{k=1}^{n-1} plan\,(k) \tag{8}$$

Here, $plan(k)$ represents the set of the partial orders of $Path_{(k,k+1),}$ and $Path_{(k,k+1)}$ is the path with $abscell_{(ik+1,jk+1)}$ as the start position and $abscell_{(ik+1,jk+1)}$ as the target position.

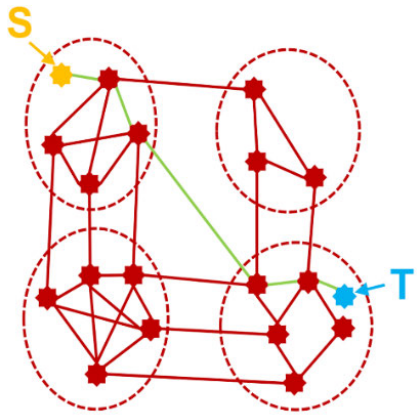

FIGURE 6. Schematic diagram of abstract graph path planning, with S (top-left, yellow) representing the start point and T (bottom-right, blue) denoting the target point.

The abstract graph path planning algorithm (Algorithm 1) initializes the open and closed lists for single-path planning. It identifies the block containing the target position (TargetPoint) and adds the starting position (StartPoint) to the open list, along with the associated data. The algorithm continues as long as the open list is not empty, selecting the node with the lowest f(n) value from the open list and exploring its neighbors in the AbstractGraph. When the current node and TargetPoint are encountered in the same block and a path between them exists, the algorithm delineates this as AbstractPath. This path is then generated using Algorithm 2 (ConstructPath) and outputted. To reduce any unnecessary detours caused by block entrances, the waypoints in AbstractPath are minimized using the dis_she technique, thereby streamlining the path and enhancing the planning speed. Finally, Algorithm 3 (RefinedPath) transforms AbstractPath into a detailed path that aligns with the original grid map.

B. NEIGHBORHOOD EXPANSION

Previous studies have often failed to focus sufficiently on the impact of neighborhood expansion on path planning. For example, Daniel et al. limited their testing to a random 500 × 500 grid map with an obstacle grid scale of 0.2 [36]. Rivera et al. explored path planning within 2^k neighborhoods and introduced a heuristic function suitable for these settings [37], [38]. Their finding demonstrated that planning for 2^k neighborhoods is as competitive as that using arbitrary angles. Meanwhile, Kramm et al. developed a suboptimality bound function for 2^k neighborhoods [39].

In this section, we describe path planning across four-neighborhood, eight-neighborhood, sixteen-neighborhood, and thirty-two-neighborhood branching extension scenarios (Figure 7), mainly focusing on (1) time cost, (2) path cost,

Algorithm 1 Implementation of AbstractPathPlanning

```
Input: AbstractGraph, GridMap, StartPoint, TargetPoint
Output: An abstract path from the start point to the target point
Initialize openlist, closelist;
targetClusterID = Clobalpos2ClusterID(TargetPoint, AbstractGraph);
h_value = weighted_Octi_heuristic(StartPoint, TargetPoint);
openlist.push_back(StartPoint, 0.0, h_value, StartPoint, 0);
while openlist not empty do
  Current_node = openlist.top;
  closelist.push_back(current_node);
  currentClusterID = Globalpos2ClusterID(currentPos, AbstractGraph);
  if currentClusterID == targetClusterID then
    cost_target = ComputeCostBetween(GridMap, current_node, Target-
Point, AbstractGraph);
    if cost_target > 0.0 then
      dis_she = 75.0
      return ConstructPath(closelist, dis_she);
  ExploreNeighbors(current_node, clusterID, TargetPoint, AbstractGraph,
GridMap);
return failure;
```

Algorithm 2 Implementation of ConstructPath

```
Input: closelist, dis_she;
Output: An Abstract Path
item = closelist.back;
current_node = item.cor;
parent_index = 0;
path.push_back(current_node);
While current_node != item.parent_cor do
  node_last = path.index;
  Parent_index = item.parent_index;
  if distance(node_last, current_node) > dis_she then
  path.push_back(item.cor);
  Item = closelist[parent_index];
  path.push_back(closelist[0].cor);
return path;
```

Algorithm 3 Implementation of RefinedPath

```
Input: GridMap, AbstractPath
Output: A Refined Path
path = null;
for i ← 1 to (n-1) do
  start = AbstractPath[i-1];
  target = AbstractPath[i];
  path.append(Astar(GridMap, start, target));
return path;
```

and (3) path smoothing. First, we briefly introduce the generation process of the neighborhood direction vector.

The 2^k neighborhood direction vector is generated via a $2^{(k-1)}$ neighborhood computation, where the minimum number of neighborhood branches is 4. Moreover, the four-neighborhood scenario has the minimum number of directions. The four-neighborhood direction vector can be denoted by D_4, as follows:

$$D_4 = \{(1,0), (0,1), (-1,0), (0,-1)\} \tag{9}$$

Meanwhile, the eight-neighborhood direction vector D_8 can be deduced from D_4 by assuming that $D_4(1)$ denotes the vector (1,0), $D_4(2)$ denotes the vector (0,1), and so on, as follows:

$$D_8 = \left\{ \begin{matrix} D_4(1), D_4(1) + D_4(2), D_4(2), D_4(2) + D_4(3) \\ D_4(3), D_4(3) + D_4(4), D_4(4), D_4(4) + D_4(1) \end{matrix} \right\} \tag{10}$$

Similarly, the sixteen-neighborhood D_{16} can be computed from D_8 and the thirty-two-neighborhood D_{32} can be computed from D_{16}.

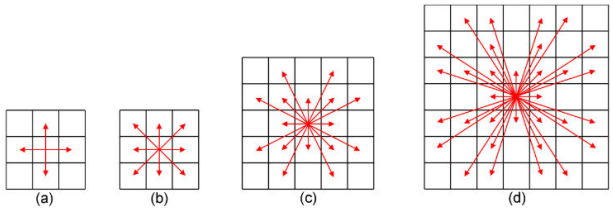

FIGURE 7. Number of neighborhood branches. (a) Four-neighborhood, (b) Eight-neighborhood, (c) Sixteen-neighborhood, (d) Thirty-two-neighborhood.

We further discuss the time cost in these modes. The interactions between map patterns and neighborhood expansion strategies significantly influence the time efficiency of path planning. As indicated in Table 1, the number of neighborhood branches varies based on the design of the grid maps. Performance improvement is observed with multi-neighborhood branching as the grid map size increases because the strategy allows longer steps in a single search iteration. This approach decreases the total number of search nodes required, thereby accelerating the path-planning process. For instance, the sixteen-neighborhood expansion strategy integrates an inner eight-neighborhood search with a step size of 1 and an outer eight-neighborhood search with a step size of 2. This configuration prioritizes exploration from points in the outer eight-neighborhood area that display the lowest expected heuristic cost, effectively minimizing the number of external iterations. However, it must be noted that the benefits of increased neighborhood branching depend on both the map pattern and the target location. Specifically, since the outer branches in configurations of sixteen neighborhoods or higher are not immediately adjacent to the current centroid, additional computations become necessary to verify the feasibility of connections. Thus, the anticipated time benefits are only realized if the valid connections at these outer extension points account for a significant portion of the overall path.

In terms of distance cost, the implementation of a multi-neighborhood branching structure enhances the diversity of path segments and facilitates more direct connections. In Figure 8, the starting point is located at (A, 1), and the target point is at (C, 5). The cost of the path, denoted as $Cost_{D_x}$, demonstrates a relationship wherein $Cost_{D_4} \geq Cost_{D_8} \geq Cost_{D_{16}} \geq Cost_{D_{32}}$. This trend is consistent across various environments, as illustrated in Table 2. Path planning can be conceptualized as fitting the shortest possible route

TABLE 1. Comparison of the conventional A* Algorithm and improved A* Algorithm.

Map type	Time cost (ms)			
	Four branches	Eight branches	Sixteen branches	Thirty-two branches
Corridor-32-4	0.461	0.567	1.000	1.842
Room-32-3	0.376	0.251	0.332	0.467
Random-32-2	0.543	0.278	0.298	0.482
Random-32-3	0.418	0.222	0.242	0.370
Random-64-2	6.039	1.140	1.329	2.224
Random-64-3	4.084	1.482	1.486	1.918
Random-100-2	23.561	4.015	4.127	5.310
Random-100-3	21.245	4.856	4.913	6.107
Random-500-2	N/A	1090.728	485.816	671.022
Random-1000-2	N/A	12459.047	3694.525	5326.611

between the start and target locations, and it typically involves connecting a few pivotal points along straight lines. These connections leverage the full angular range allowed by the neighborhood branches, thus approximating the actual shortest path. However, it is worth noting that while increasing the number of neighborhood branches may decrease distance costs, it concurrently raises the time costs. Therefore, the balance between these benefits must be carefully managed.

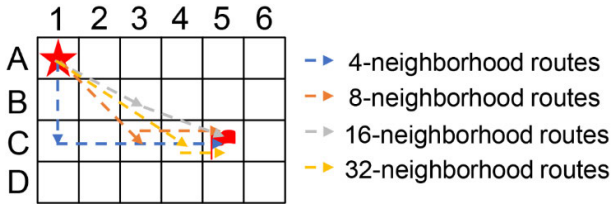

FIGURE 8. Schematic diagram of neighborhood branching effects on potential paths.

In path planning for a four-neighborhood branching system, the angle between two adjacent directions is 90°. This is also the minimum turn angle in this system. For an eight-neighborhood system, this angle reduces to 45°. Meanwhile, in a sixteen-neighborhood configuration, the angles between adjacent directions decrease to 26.565° and 18.435°. Furthermore, in a thirty-two-neighborhood system, the angles between adjacent directions can be as small as 18.435°, 8.13°, 7.125°, and 11.31°. Thus, as the number of neighborhood branches increases, the variety of steering angles available also increases, resulting in progressively smoother routes.

Neighborhood branching also impacts the time efficiency of the A* algorithm due to the repetitive expansion of nodes, which necessitates frequent access to both the open and closed lists. As the map scale enlarges, the number of nodes in these lists increases, thereby prolonging access times. As demonstrated by the JPS algorithm, under practical application conditions, many grid cells do not require expansion.

TABLE 2. Comparison of the PATH LENGTHof exploring different neighborhoods on different maps.

Map type	Path Length			
	Four branches	Eight branches	Sixteen branches	Thirty-two branches
Corridor-32-4	67.018	57.166	55.627	55.286
Room-32-3	49.160	40.050	38.957	38.634
Random-32-2	44.061	35.802	34.859	34.497
Random-32-3	45.781	37.269	36.113	34.955
Random-64-2	93.594	75.617	73.368	72.782
Random-64-3	96.912	78.520	76.207	73.257
Random-100-2	148.620	120.160	116.846	115.455
Random-100-2	152.059	122.978	119.567	117.043
Random-500-2	N/A	597.239	583.443	578.090
Random-1000-2	N/A	1171.474	1143.677	1133.772

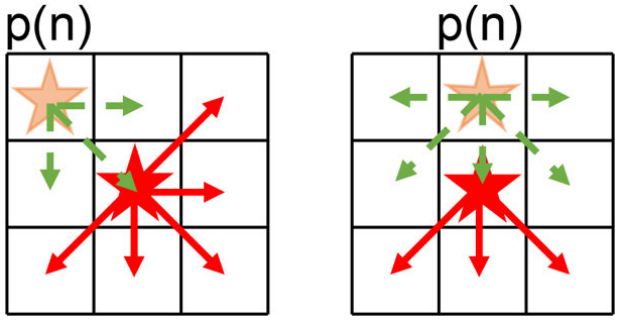

FIGURE 9. Schematic diagram of neighborhood extension, where red stars denote the current node *n* and orange stars denote the parent node of n, i.e., p(n).

For example, when node n's parent node is p(n), and the expansion direction of p(n) and node n is similar to that depicted in Figure 9(a), some common neighbors between p(n) and node n do not require a path through node n due to the availability of shorter alternate routes. Consequently, node n only requires expansion in five specific directions. Similarly, under the conditions shown in Figure 9(b), node n may only need expansion in three directions. This approach significantly prevents redundant expansions. Theoretical analysis suggests that the parent of node n has a 50% probability of diagonal extension and a 50% probability of linear extension, necessitating expansions in five and three directions, respectively. This also greatly reduces redundant computations.

C. HEURISTIC FUNCTIONS

The $h(n)$ heuristic function in the A* algorithm (refer to Equation significantly influences the algorithm's efficiency, with its informativeness being directly correlated with the speed of path planning. If $h(n) \leq H(n)$, where $H(n)$ is the actual cost from node n to the goal, the algorithm will

generally succeed at finding the optimal path, assuming that one exists. Furthermore, if $h_1(n) \leq h_2(n) \leq H(n)$, where $h_1(n)$ and $h_2(n)$ are different heuristic functions, the path planning algorithm that uses $h_1(n)$ and $h_2(n)$ can surely find the optimal path. In terms of time efficiency, however, $t_2 < t_1$, illustrating the principle of heuristic consistency in the A* algorithm. When $h(n) = 0$, this algorithm transforms into Dijkstra's algorithm. When $h(n) = H(n)$, the algorithm will ideally be both cost-optimal and time-efficient. However, if h(n) > H(n), the algorithm may not find a cost-optimal path, even if one exists. Common path planning heuristics — such as the Manhattan, Euclidean, Chebyshev, and Octile distances — are depicted in Figure 10 for two grid map points S(x_S, y_S) and T(x_T, y_T).

The Manhattan function is of the form:

$$Manhattan\,(S,T) = |x_T - x_S| + |y_T - y_S| \tag{11}$$

The Euclidean function is of the form:

$$Euclid\,(S,T) = \sqrt{(x_T - x_S)^2 + (y_T - y_S)^2} \tag{12}$$

The Chebyshev function is of the form:

$$\text{Chebyshev (S,T)} = \max\left(|x_T - x_S|, |y_T - y_S|\right) \tag{13}$$

The Octile function is of the form:

$$\text{Octile (S,T)} = \sqrt{2} \times \min\left(|x_T - x_S|, |y_T - y_S|\right) + ||x_T - x_S| - |y_T - y_S|| \tag{14}$$

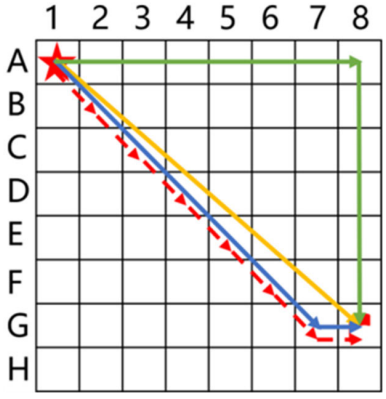

FIGURE 10. Schematic of the distance formulae. The green line segments indicate Manhattan distances; the yellow line segments represent Euclidean distances; the blue line segments represent Octile distances; and the red dashed lines represent Chebyshev distances. Although the schematic for the Chebyshev distance appears very similar to that of the Octile distance, they are not identical. The Chebyshev heuristic calculates the minimum number of moves required to transition from the current position to the target position in an eight-neighborhood branch on an obstacle-free map.

In path planning problems, the Manhattan distance often exceeds the actual distance (i.e., $Manhattan(n) \geq H(n)$); however, for maze-like terrains or serpentine corridors, it is possible that $Manhattan(n) \leq H(n)$, yielding impressive results due to the computational simplicity. Typically, the relationship among the heuristics is as follows: $Chebyshev(n) \leq Euclid(n) \leq Octile(n) \leq Mahattan(n)$. Both Figure 11 and Table 3 corroborate this hierarchy, demonstrating that path planning based on the Manhattan function does not yield the least costly path. Conversely, since the Octile function more closely approximates $H(n)$, it offers fewer visited points than the Euclidean and Chebyshev distances, as detailed in Table 4. Figure 12 further validates the superiority of the Octile function; for the map Random-100-3, the Octile function visits 32.15% fewer points than the Euclidean function.

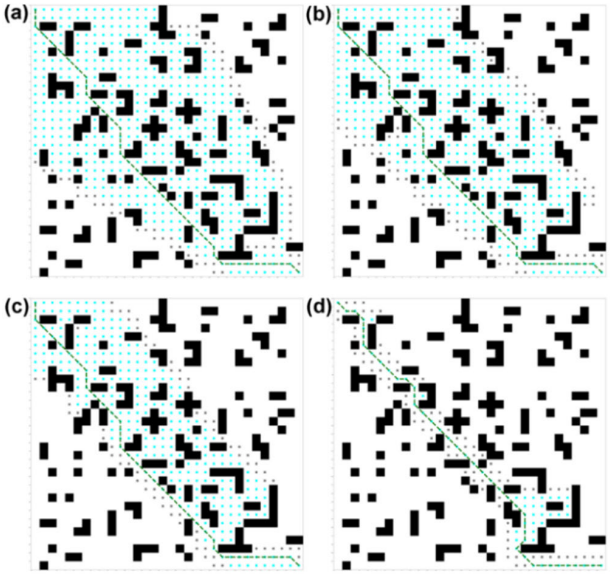

FIGURE 11. An eight-neighborhood search is performed for a random grid map sized 32 × 32, with blue points stored in the close list and gray points stored in the open list, all of which constitute the set of access points. (a) Search performed using the Chebyshev function heuristic term. (b) Search performed using the Euclidean heuristic term. (c) Search performed using the Octile function heuristic term. (d) Search performed using the Manhattan function heuristic term.

TABLE 3. Heuristic Functions and associated planning time and path cost relationships.

Random-32-2	Eight branches		Sixteen branches	
	Path length	Time cost (ms)	Path length	Time cost (ms)
Manhattan	50.527	0.203	51.725	0.287
Chebyshev	48.527	1.191	47.458	2.165
Euclid	48.527	0.821	47.458	1.119
Octile	48.527	0.447	47.458	0.535

TABLE 4. Heuristic functions and the associated number of access points.

Random-32-2	Eight branches		Sixteen branches	
	Open list	Close list	Open list	Close list
Manhattan	82	66	121	62
Chebyshev	72	516	109	423
Euclid	78	426	127	376
Octile	73	287	121	193

Unlike in corridor systems, where the obstacles are structured, the obstacles in high-resolution grid maps of real outdoor environments are typically unstructured. The exploration process of the A* algorithm can be likened to flooding.

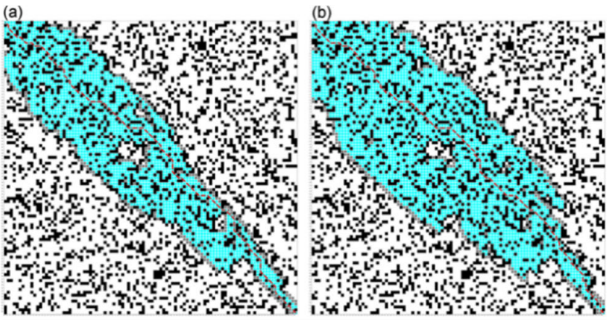

FIGURE 12. For the map Random-100-3, based on the eight-neighborhood expansion approach, (a) the Octile function is used as a heuristic, and (b) the Euclid function is used as a heuristic. The points stored in the close list are shown in bright blue, and the points stored in the open list are shown in gray. The planned route is indicated in red. The Octile-based function ensures both the shortest route and reduced access. (Cost of the route: 152.3086; number of points accessed in the open list in [a]: 247; number of points accessed in the close list: 1895; number of points accessed in the open list in [b]: 256; number of points accessed in the close list: 2901).

Assuming that the start position is S and the target is T, the vector from S to T is denoted as $\overrightarrow{ST}$, and $\overrightarrow{MO}$ is orthogonal to this vector. The expansion behavior of A* demonstrates that nodes closer to T are more likely to expand toward $\overrightarrow{MO}$. However, as shown in Figure 13(a), the optimal shortest path on the grid map is a straight line following $\overrightarrow{ST}$. In environments with highly randomized obstacles, the paths closer to $\overrightarrow{ST}$ are shorter. Thus, during exploration, suppressing expansion toward $\overrightarrow{MO}$ and encouraging movement along $\overrightarrow{ST}$ reduces the number of expansions and accelerates path planning.

In this paper, we propose a weighted heuristic function based on the Octile distance, which includes several components: d_{Si} represents the Euclidean distance from the starting position S to node i; d_{iT} denotes the Euclidean distance from node i to the target T; d_{ST} indicates the total Euclidean distance from S to T; and Octile (i, T) denotes the Octile distance from node i to the target T.

$$\text{Weighted Octile (i)} = \frac{d_{Si} + d_{iT}}{d_{ST}} \times Octile\,(i) \tag{15}$$

As demonstrated in Figure 13(b), given that $\frac{d_{Si}+d_{iT}}{d_{ST}} \leq \frac{d_{Sj}+d_{jT}}{D_{ST}}$ and $Octile(i) < Octile(j)$, $weightedOctile\,(i) < Octile\,(i) < Octile\,(j) < weighedOctile(j)$. This condition enhances the efficiency of the A* algorithm by reducing the expansion of nodes, thereby increasing the likelihood of identifying an optimal path more quickly.

We evaluate the weighted Octile function with an eight-way neighborhood branching approach. The experimental results (Tables 5 and 6) demonstrated that under identical conditions, compared with the conventional Octile function, the weighted Octile function not only reduces time consumption by approximately 20% in randomized environments but also decreases the number of access points by roughly 10%. This decrease in access points would lead to a consequent decrease in memory usage, without significantly increasing the path length.

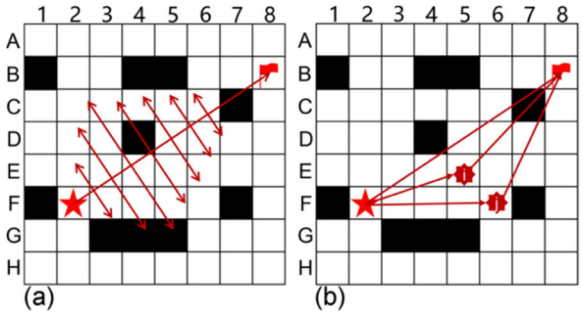

FIGURE 13. (a) Trend of A* algorithm extension. (b) Schematic of the weighted heuristics.

D. CONVOLUTION-BASED REPULSIVE FORCE FIELD CONSTRUCTION

In high-resolution, large-scale grid field maps wherein the obstacles are randomized instead of structured, it is essential to minimize routes through densely obstructed spaces for safety and to navigate more reliably in less crowded areas (Figure 14). This approach aids in managing the unpredictability of the environment and provides agents with greater maneuverability. An optimal balance is required among factors such as the shortest path, safety, and planning speed. The artificial potential field method is a crucial technique for path planning in this context, guiding agents from the start position to the target position by mimicking electromagnetic or gravitational fields. This method includes two main components: a repulsive field generated by obstacles, which intensifies as the proximity to the obstacle increases, and an attractive field produced by the target, which grows stronger as the agent nears the goal. These fields not only inform agents about nearby obstacles and targets but also significantly improve route planning and safety. Through the application of this concept, we focus on constructing a repulsive field that robustly captures the negative impact of obstacles, effectively broadening the agents' awareness of their surroundings and adding avoidance measures. Using a similar approach, Chen et al. combined the A* algorithm with the artificial potential field method to speed up and refine path planning [40].

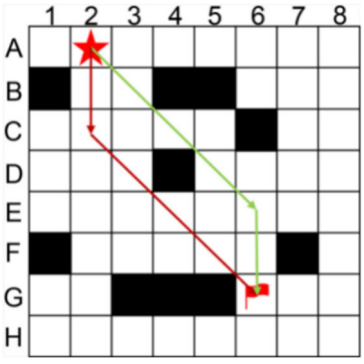

FIGURE 14. Schematic diagram of symmetric paths. Both the red and green paths represent the shortest routes from the start position to the goal position; however, fewer obstacles are encountered in the red path, and it is consequently safer.

TABLE 5. Comparison of weighted_Octile and Octile functions.

Map type	Time cost (ms)		Path length		Open list		Close list	
	Weighted	Octile	Weighted	Octile	Weighted	Octile	Weighted	Octile
Random-32-2	0.201	0.254	36.988	36.932	65.969	66.200	64.262	98.108
Random-32-3	0.202	0.397	38.703	38.683	56.531	56.938	76.875	101.156
Random-64-2	0.916	1.308	77.063	77.032	135.765	147.500	266.016	329.734
Random-64-3	1.240	1.576	80.024	79.935	113.691	117.912	441.618	511.427
Random-100-2	3.275	4.269	121.498	121.002	223.550	248.180	622.830	756.450
Random-100-3	3.530	5.155	124.548	124.393	192.485	213.719	794.564	967.901

TABLE 6. Comparison of weighted Octile and Octile functions (scaled version).

Map type	Time cost	Path length	Number of nodes
Random-32-2	-20.87%	+0.15%	-20.74%
Random-32-3	-49.12%	+0.05%	-15.62%
Random-64-2	-29.97%	+0.04%	-15.81%
Random-64-3	-21.32%	+0.11%	-11.76%
Random-100-2	-23.28%	+0.41%	-15.75%
Random-100-3	-31.53%	+0.12%	-16.47%

The conventional repulsive field approach for grid maps is quite inefficient owing to its time-consuming nature. In the traditional potential field construction method, each grid containing an obstacle is treated as a distinct obstacle, which significantly increases the number of perceived obstacles in comparison to methods that use polygons to represent obstacles in continuous spaces. This increases computational demands and prolongs the planning time. Conversely, convolution-based repulsive field construction, which does not focus on individual obstacles, not only ensures that each grid has an adequate field of view but also expedites the construction of potential fields. Accordingly, it accelerates the path-planning process. Therefore, in this paper, we propose a convolution-based, non-obstacle-oriented modeling approach, designed to collect information about surrounding obstacles at the current location and thereby enrich the heuristic data used in the analysis.

$$\begin{bmatrix} 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 1 & 1 & 1 & 0.6 \\ 0.6 & 1 & 0 & 1 & 0.6 \\ 0.6 & 1 & 1 & 1 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \end{bmatrix} \tag{16}$$

The convolution kernel is designed as illustrated in (16). The kernel's size determines the field of view available for the current grid. The values decrease radially from the center, reflecting the diminishing influence of distant obstacles. By employing different convolution kernels, more sophisticated functionalities can be achieved. For instance, a 32 × 32 grid map based on the convolution method yields a more balanced and stable repulsion field than one based on the traditional method, instead of being heavily polarized, as depicted in Figure 15. Furthermore, the potential field created with the convolution method more accurately represents the influence of surrounding obstacles, providing a contour trend that peaks at obstacles. However, this trend is less pronounced in the conventional potential field. Additionally, the potential field map based on the convolution method displays more uniformly distributed repulsive values, while the one based on the conventional method exhibits distinct bifurcations. These characteristics make the convolution-based repulsive field more effective at enhancing heuristic functions, providing a superior representation of the current grid and facilitating a more nuanced differentiation of repulsive values. It is noteworthy that the primary function of the kernel is to generate a repulsive force.

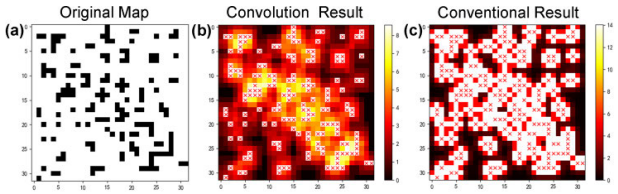

FIGURE 15. Results of the repulsive field construction under both convolution and conventional conditions, with the red X symbols marking the obstacle grid corresponding to the original map.

Table 7 presents a time-based comparison of the construction of the repulsive field based on the convolution method and the conventional method. The results indicate that the convolution method offers a significant time advantage over the conventional approach. Furthermore, as the size of the grid map and the number of obstacles increase, the conventional method becomes even more time-consuming. However, the convolution approach retains its efficiency even under these conditions. Notably, it is less sensitive to the number of obstacles because it does not attempt to assess the impact of each obstacle individually. In contrast, the conventional method, which considers the effects of each obstacle on the current grid, incurs redundant computations when the distance of the obstacle exceeds a certain threshold, leading to inefficiency.

Thus, this paper derives a heuristic function based on the Octile function mixed with a convolutional repulsive field function.

$$h(i) = weighted\ Octile(i) + conv(i) \tag{17}$$

Here, $Octile(i)$ denotes the node i to the target position T, and $conv(i)$ denotes the convolutional repulsive field value of node i.

The routes derived using this method were compared with those generated by the A* algorithm, which lacks

TABLE 7. Comparison of time costs during the construction of conventional and convolutional repulsive fields.

Map type	Conventional potential field method (ms)	Convolution potential field method (ms)
Random-32-2	0.312	0.039
Random-32-3	0.405	0.039
Random-64-2	4.999	0.144
Random-64-4	7.456	0.143
Random-100-2	29.374	0.339
Random-100-3	38.622	0.340

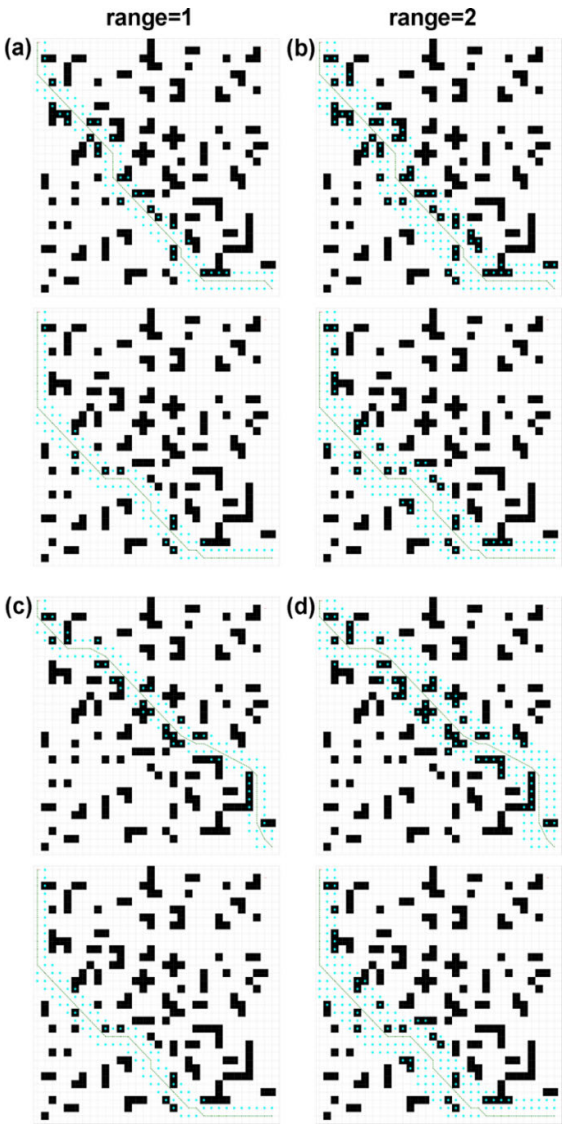

FIGURE 16. Schematic diagram of path comparison. In the Random_32 × 32_2 map, blue indicates the route surroundings based on the standard A* algorithm, while yellow depicts those based on the convolutional repulsive field A*. Routes (a) and (b) are derived based on eight-directional movement.

potential field function integration. This comparison particularly emphasized the number of obstacles within proximity ranges of 1 and 2 units (Figure 16). It was noted, however, that the paths derived from the sixteen-neighborhood approach using this method may occasionally overlap with those obtained through the eight-neighborhood branching computation. Table 8 compares the performance of the A* algorithm that uses the weighted Octile function and the A* algorithm that employs the convolutional kernel described in (17). The results show that although the inclusion of the convolutional repulsive field slightly increases path and time costs, it notably diminishes the number of obstacles surrounding the path.

V. EXPERIMENTAL VALIDATION

All the experiments reported in this paper were conducted on an Ubuntu 22.04 system with an AMD®Ryzen 9 5900hx CPU and 32 GB of RAM. The code was implemented in C++.

A. EXPERIMENTS BASED ON THE IMPROVED ALGORITHM

To ensure algorithmic robustness, the experimental setup mandated that the straight-line distance between the start and target positions on any given map contain at least one edge. Specifically, on a Size × Size grid map, the target was positioned at (0,0), and the starting position at [x, Size - 1]. To assess the comprehensive route planning capacity of the algorithm across the entire map, the set of initial positions included the coordinates [0, Size-1] and [Size-1, Size-1], along with an additional 10 to 30 randomly selected points within the range [1, 2, …, Size-2].

Figure 17 shows the environments in which the experiments were conducted, namely, Shanghai-1-1024, Shanghai-2-1024, Random-1000-2, and Simulation-200.

Figure 17(a) and Figure 17(b) were derived from the Benchmarks for Grid-Based Pathfinding standards (*http://movingai.com/benchmarks/*). Two maps, 1024 × 1024 in size and containing 65 start–target location pairs, were selected and tested. Meanwhile, Figure 17(c) was randomly generated based on an obstacle rate of 0.2 and a size of 1000 × 1000. Figure 17(d), which shows the results of the simulation (size, 2000 × 2000), was generated via manual illustration.

We compared the effectiveness of two types of algorithms: unimproved A* algorithms (A* 8d and A* 16d) and improved methods (Theta*, HPTheta*, HPA* 8d improved, HPA* 16d improved). Theta* is an any-angle path planning algorithm, while HPA* 8d improved and HPA* 16d improved represent the enhancements proposed in this paper.

In our experiments, the grid was segmented into 10 × 10 blocks of 50 × 50 for every map. The convolutional kernel from (16) and the heuristic function from (17) were utilized, and a method for preventing the duplicate extensions of neighboring nodes was also incorporated. Meanwhile, the standard A* algorithm employed the Octile heuristic and a similar method for avoiding repeated neighbor expansions. The improved HPA* algorithm segmented path calculations, using a counting method to determine the number of access points for the entire path. Notably, the maximum number of access points across one segment represented the count for that entire path planning session.

TABLE 8. Comparison of the conventional A* algorithm and convolutional A* algorithm.

Map type	Time cons. (ms)		Distance		Range=1		Range=2	
	A*	Conv A*	A*	Conv A*	A*	Conv A*	A*	Conv A*
Random-32-2	0.226	0.306	37.214	38.346	0.225	0.207	0.203	0.199
Random-32-3	0.209	0.262	38.970	40.176	0.296	0.260	0.282	0.260
Random-64-2	0.933	1.712	77.238	79.916	0.226	0.197	0.212	0.190
Random-64-3	1.167	1.464	80.054	82.047	0.286	0.264	0.271	0.271
Random-100-2	4.257	5.194	121.592	124.59	0.204	0.185	0.203	0.180
Random-100-3	5.010	6.263	124.393	127.052	0.309	0.281	0.293	0.281

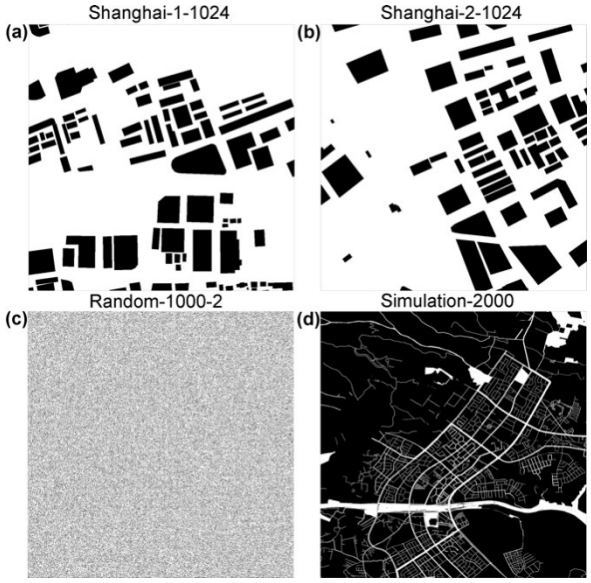

FIGURE 17. Schematic diagram of experimental environments. (a) Shanghai-1-1024; (b) Shanghai-2-1024; (c) Random-1000-2; (d) Simulation-2000.

TABLE 9. Comparison of the conventional A* Algorithm and improved A* Algorithm.

Random-500-2	Time cost (ms)	Path length	Range=1 ratio	Number of nodes
8d A*	1090.728	597.239	0.216	22781.64
HPA* 8d improved	59.633	599.086	0.206	849.214
16d A*	485.816	583.444	0.198	10724.33
HPA* 16d improved	60.200	590.060	0.193	690.450

TABLE 10. Percentage of improvement with the improved A* algorithm versus the conventional A* Algorithm.

Random-500-2	Time cost (ms)	Path length	Range=1 ratio	Number of nodes
HPA* 8d improved vs 8d A*	-94.53%	+0.31%	-4.63%	-96.27%
HPA* 16d improved vs 16d A*	-87.61%	+1.12%	-2.52%	-93.56%

Tables 9 and 10 illustrate that the improved HPA* algorithm significantly enhances path planning performance by reducing time costs and the number of access points while minimally increasing the path cost. Specifically, when applied using an eight-neighborhood exploration approach, the algorithm reduces the time spent by 94.53% and lowers the number of access points by 96.27%, thus substantially decreasing the memory footprint. Under these conditions, the path cost increases by a mere 0.31%, and the implementation of a convolutional repulsive field decreases the obstacles around the path by 4.63%. In contrast, the sixteen-neighborhood exploration approach incurs a path cost increase of 1.12% due to chunking behavior, which has a lower effect on sixteen-neighborhood exploration than on eight-neighborhood exploration, reducing the path cost by 1.2% in comparison to the conventional eight-neighborhood A* algorithm. Notably, the sixteen-neighborhood regular A* method demonstrates considerable improvement over its eight-neighborhood counterpart, demonstrating its suitability for path planning in large-scale randomized grid maps.

TABLE 11. Quantitative results and comparison of the impact of cluster size.

Random-500-2	Time cost (ms)	Path length
8d cluster_size=50	59.633	599.086
8d cluster_size=100	406.322	597.981
16d cluster_size=50	60.200	590.061
16d cluster_size=100	412.185	587.651

TABLE 12. Quantitative results and comparison of time cost and path length on a Random-1000-2 map following the application of conventional and improved A Algorithms.

Random-1000-2	Time cost (ms)	Path length	Range=1 ratio
8d_A*	12459.047	1171.474	0.217
HPA* 8d improved	262.362	1174.170	0.209
16d_A*	3694.525	1143.677	0.202
HPA* 16d improved	264.734	1161.538	0.196

Table 11 demonstrates that cluster size significantly impacts pathfinding efficiency. When the cluster size increases from 50 to 100, a much longer time is required for path planning. However, the increase in cluster size yields a path that is closer to the optimal one achievable by the A* algorithm, though a more nuanced balance is necessitated given the diminishing returns in path cost reduction at a cluster size of 100.

Table 12 shows the performance of the improved HPA* algorithm on maps configured with the Random-1000-2

TABLE 13. Comparison of the results of different algorithms based on the Shanghai-1-1024 MAP.

Shanghai-1-1024	Time (ms)	Length	Nodes	Ratio (range=1)	Time reduction	Length reduction	Node reduction	Ratio reduction
Benchmarks	N//A	1228.423	N//A	N/A	N/A	N/A	N/A	N/A
A* 8d	44070.401	1226.169692	123945.154	0.0439	1	1	1	1
A* 16d	40868.516	1181.669	77978.185	0.0286	7.27%	3.63%	37.09%	34.85%
Theta*	46808.119	1168.899	112884.985	0.0216	-6.21%	4.67%	8.92%	50.80%
HPATheta*	619.61	1178.621	2363.615	0.138	98.59%	3.88%	98.09%	-214.35%
HPA* improved	568.406	1194.893	2351.938	0.0021	98.71%	2.55%	98.10%	95.22%

TABLE 14. Comparison of the results of different algorithms based on the Shanghai-2-1024 MAP.

Shanghai-2-1024	Time (ms)	Length	Nodes	Ratio (range=1)	Time reduction	Length reduction	Node reduction	Ratio reduction
Benchmarks	N/A	1241.275	N/A	N/A	N/A	N/A	N/A	N/A
A* 8d	32331.899	1239.962	88656.98462	0.0330	1	1	1	1
A* 16d	32806.056	1199.613	69134.2	0.0310	-1.47%	3.25%	22.02%	6.17%
Theta*	38845.536	1189.962	62153.738	0.0278	-20.15%	4.03%	29.89%	15.72%
HPATheta*	675.705	1196.933	2066.923	0.0874	97.91%	3.47%	97.67%	-164.96%
HPA* improved	467.697	1207.053	1617.861	0.0021	98.55%	2.65%	98.18%	93.63%

TABLE 15. Comparison of the results of different algorithms based on the Random-1000-2 MAP.

Random-1000-2	Time (ms)	Length	Nodes	Ratio (range=1)	Time reduction	Length reduction	Node reduction	Ratio reduction
A* 8d	16833.683	1325.49	91187.169	0.2123	1	1	1	1
A* 16d	8626.65	1299.992	45552.661	0.205	48.75%	1.92%	50.04%	3.44%
Theta*	33739.17	1280.217	117090.063	0.1745	-100.43%	3.42%	-28.41%	17.80%
HPATheta*	421.274	1287.658	2256.714	0.2086	97.50%	2.85%	97.53%	1.74%
HPA* improved	354.489	1309.986	1296.286	0.1732	97.89%	1.17%	98.58%	18.42%

parameter and a cluster size of 50, using the convolution kernel outlined in (17) as a heuristic. This comparison confirms the effectiveness of the improved algorithm.

Tables 13 and 14 show the results obtained from tests based on Figure 17 (a) and (b). Sixty-five start–target tasks were randomly selected from the dataset for testing. The time, length, and nodes were calculated based on the average values across 65 tasks. The length parameter represented the average length of the shortest path of the task in a given dataset. In the experiments, A* 8d and A* 16d both used the Octile heuristic function, Theta* and HPTheta* [32] both used the Euclid heuristic function, and HPA* improved used 16d neighborhood exploration, the weighted Octile function, and the repulsive field constructed via the convolutional method for obstacle avoidance.

For the Shanghai-1-1024 map, the Improved HPA* algorithm achieves a 98.71% reduction in path planning time when compared to the standard A* 8d algorithm. Additionally, it maintains a consistent path length with a minimal decrease of 2.55%. Despite the considerable increase in path length relative to the standard A* 16d and Theta* algorithms due to hierarchical structuring, the Improved HPA* algorithm outperforms HPATheta* with respect to both time and path length reductions. Additionally, the Improved HPA* algorithm increases the obstacle rate along the path, improving the distance maintained from obstacles. In terms of memory usage, this algorithm reduces the average number of nodes accessed by 98.10%, significantly reducing memory consumption when compared to the A* 8d algorithm.

For the Shanghai-2-1024 map, the Improved A algorithm significantly reduces path planning time by 98.55% when compared to the standard A* 8d algorithm, while increasing the path length by only 2.65%. Additionally, it achieves a 30.78% time reduction relative to the HPATheta* algorithm. The obstacle rate of the improved HPA* algorithm around the path is as low as 0.0021, representing a 93.63% reduction in comparison to the standard A* 8d algorithm, thus ensuring that the paths avoid close proximity to obstacles. In terms of memory usage, the improved HPA* algorithm accesses an average of 98.18% fewer nodes than the standard A* 8d algorithm, thereby reducing memory consumption significantly.

In the Random-1000-2 map, the Improved A* algorithm achieves a significant time reduction of 97.89% versus the

TABLE 16. Comparison of the results of different algorithms based on the Simulation-2000 MAP.

Simulation-2000	Time (ms)	Length	Nodes	Ratio (range=1)	Time reduction	Length reduction	Node reduction	Ratio reduction
A* 8d	59892.975	2694.535833	165863.5	0.2123	1	1	1	1
A* 16d	82219.054	2614.572083	141080.625	0.205	-37.28%	2.97%	14.94%	3.44%
Theta*	86815.133	2588.887	176175.25	0.1745	-44.95%	3.92%	-6.22%	17.80%
HPATheta*	598.303	2584.714783	2518.696	0.2306	99.00%	4.08%	98.48%	-8.62%
HPA* improved	506.502	2611.160	2232.83	0.0610	99.15%	3.09%	98.65%	71.27%

standard A* 8d algorithm. Despite the challenges posed by the random obstacle generation and chaotic environments in such maps, the Improved A* algorithm also reduces the obstacle rate around the path by 18.42%. This algorithm prioritizes the identification of open, symmetric paths. Additionally, while the path length it yields is lower than that provided by the HPATheta* algorithm, it also minimizes traversal through obstacle clusters.

In the Simulation-2000 map, which contains a 2000 × 2000 region, the Improved A* algorithm demonstrates significant efficiency enhancements. First, it reduces the path planning time by 99.15% and the path length by 3.09% when compared to the standard A* 8d algorithm. Additionally, it decreases the number of accessed nodes by 98.65% and lowers the path obstacle rate by 71.27% in comparison to A* 8d. When compared to the HPATheta* algorithm, the Improved A* algorithm yields a planning time and obstacle rate reduction of 15.34% and 73.55%, respectively. Specifically, the obstacle rate reduces significantly from 95.22% to 18.42%. Thus, this algorithm offers a slight optimization over HPATheta*, which also employs hierarchical theory.

B. PRACTICAL SCENARIOS

In this paper, the area spanning from 25°6′ 34.72″ to 24°58′ 3.68″ N and 121°10′ 11.46″ to 121°24′ 2.06″ E was selected as the experimental area, including both the urban road network and off-road environments, as shown in FIGURE18 (a). In path planning for large-scale scenarios, a vehicle can be abstracted as a mass point. Hence, the data of the selected area should be matched to motorized vehicle information to obtain more practical path planning results. To align with the characteristics of motorized vehicles, we leveraged remote sensing satellite imagery with a 2.5-meter resolution in conjunction with Digital Elevation Model (DEM) data, thereby enhancing the practical applicability of our path planning outcomes. Ground accessibility information was extracted through semantic segmentation of remote sensing images in tandem with slope analysis derived from DEM data, as shown in FIGURE18 (b). This integrated approach yielded slope data, enabling the generation of an accessibility map for the designated area, comprising approximately 63 million grid cells.

Furthermore, we proceeded with tests on three distinct paths of varying lengths, originating from a common starting point and terminating at different target locations, as outlined in Table 17. We juxtaposed the performance of fundamental pathfinding algorithms, namely A* and Theta, against a hierarchical planning methodology, encompassing

HPATheta and the enhanced HPA* Improved algorithm introduced in this study. In the context of real-world, large-scale path planning, computational efficiency emerges as a pivotal challenge. The outcomes, presented in Table 18, underscore the remarkable capability of the HPA* Improved algorithm in reducing computation time by a staggering 99.51% in comparison to A, while maintaining paths that are merely 1.6% less optimal. In a specific instance, when benchmarked against HPATheta for the path traversing from [8144, 1157] to [100, 4000], HPA* Improved demonstrated an impressive 84.49% reduction in computation time. The efficacy of our path planning approach is vividly demonstrated in Figure 19, where the paths generated by HPA* Improved exhibit a close resemblance to those produced by A* and Theta*, as evident in Figures 19(b) and 19(c), respectively.

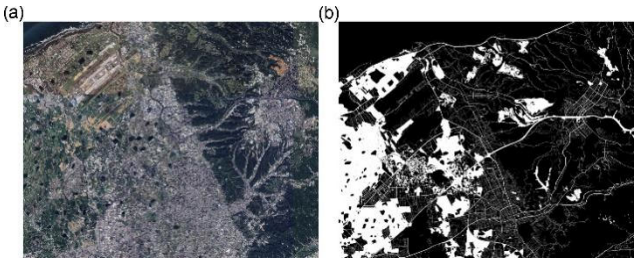

FIGURE 18. (a) Remote sensing imagery for selected areas (b) Finalized pass/fail grid maps.

TABLE 17. Detailed information on the real-life experimental scenarios.

Path No.	Start	Target	Actual straight-line distance (km)
1	[8144, 1157]	[100, 4000]	21.329
2	[8144, 1157]	[3850, 4583]	13.733
3	[8144, 1157]	[7438, 2816]	4.507

Table 18 shows the calculation of the three paths. The data clearly demonstrate that the algorithm designed in this paper showcases a substantial improvement in the efficiency of path

TABLE 18. The results of different algorithms based on the real scenarios.

Start	Target	AStar		Theta		HPATheta*		HPA* Improved	
		Length	Time(ms)	Length	Time(ms)	Length	Time (ms)	Length	Time (ms)
[8144, 1157]	[100, 4000]	10039.400	251927.000	9582.740	815596.000	9884.160	7956.712	10209.704	1233.810
[8144, 1157]	[3850, 4583]	6194.670	49304.560	5902.650	89059.800	6025.830	2504.230	6235.400	439.501
[8144, 1157]	[7438, 2816]	2058.900	10060.741	1953.820	18080.000	2061.810	1093.650	2059.491	184.539

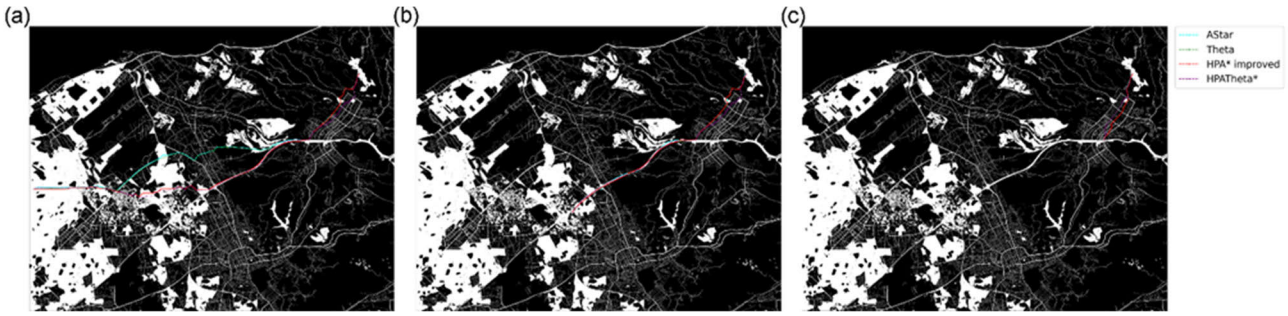

FIGURE 19. (a) Path planning results. Where AStar results are shown using cyan colored dashed lines, theta is shown using green dashed lines, HPA* Improved is shown using red dashed lines, and HPATheta* is shown using purple dashed lines. (a) Starting at [8144, 1157] and ending at [100, 4000], (b) Starting at [8144, 1157] and ending at [3850, 4583], (c) Starting at [8144,1157] and ending at [7438, 2816].

planning in complex environments. A maximum improvement of about 141-fold can be achieved, and the path cost can be controlled within a reasonable range.

VI. CONCLUSION AND FUTURE OUTLOOK

This paper presents an enhanced HPA* algorithm tailored to improving path planning in the large-scale grid maps of real-world environments. The enhancements focus on accelerating planning speed, ensuring safety, and minimizing traversal through densely obstructed areas. The method proposed in this paper encompasses several key steps: (1) Transformation of the Grid Map: The grid map path planning problem is reformulated as an abstract map search problem by converting the original grid map into an abstract map. (2) Analysis of Neighborhood Branches: We examined how the number of neighborhood branches affects path planning, especially in terms of time and distance costs. Our analysis indicates that the sixteen-neighborhood exploration mode is superior to the eight-neighborhood extension mode for more randomized maps. (3) Heuristic Function and Safety Enhancements: following a comparative analysis of various basic heuristic functions, we introduce the weighted Octile function, which effectively reduces the number of access points and accelerates path planning. Additionally, we propose a convolutional repulsive field-based method to enhance path security. Our experimental results demonstrate that the enhanced HPA* algorithm significantly reduces the number of access points, decreases path planning time, and enhances the reliability of movement in environments containing randomized obstacles.

In the future, we aim to develop a more rapid method for constructing abstraction graphs that adapt to dynamically changing environments. This effort seeks to enhance the responsiveness and efficiency of the A* algorithm in complex scenarios and environments.

REFERENCES

[1] C. Hua, R. Niu, B. Yu, X. Zheng, R. Bai, and S. Zhang, "A global path planning method for unmanned ground vehicles in off-road environments based on mobility prediction," *Machines*, vol. 10, no. 5, p. 375, May 2022. [Online]. Available: https://www.mdpi.com/2075-1702/10/5/375

[2] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based UAV path planning," *Knowl.-Based Syst.*, vol. 158, pp. 54–64, Oct. 2018, doi: 10.1016/j.knosys.2018.05.033.

[3] B. Xing, M. Yu, Z. Liu, Y. Tan, Y. Sun, and B. Li, "A review of path planning for unmanned surface vehicles," *J. Mar. Sci. Eng.*, vol. 11, no. 8, p. 1556, 2023. [Online]. Available: https://www.mdpi.com/2077-1312/11/8/1556

[4] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," *Robotics*, vol. 12, no. 4, p. 93, Jun. 2023. [Online]. Available: https://www.mdpi.com/2218-6581/12/4/93

[5] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, Aug. 2021. [Online]. Available: https://www.mdpi.com/2624-8921/3/3/27

[6] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Defence Technol.*, vol. 15, no. 4, pp. 582–606, Aug. 2019, doi: 10.1016/j.dt.2019.04.011.

[7] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *Int. J. Phys. Sci.*, vol. 7, pp. 1314–1320, Feb. 2012.

[8] J. Liu, Y. Yan, Y. Yang, and J. Li, "An improved artificial potential field UAV path planning algorithm guided by RRT under environment-aware modeling: Theory and simulation," *IEEE Access*, vol. 12, pp. 12080–12097, 2024, doi: 10.1109/ACCESS.2024.3355275.

[9] K. Zheng, "Autonomous obstacle avoidance and trajectory planning for mobile robot based on dual-loop trajectory tracking control and improved artificial potential field method," *Actuators*, vol. 13, no. 1, p. 37, Jan. 2024. [Online]. Available: https://www.mdpi.com/2076-0825/13/1/37

[10] J. D. Tenenberg, "Abstraction in planning," Ph.D. dissertation, Dept. Comput. Sci., Univ. Rochester, Rochester, NY, USA, 1988.

[11] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proc. Int. Conf. Ind. Eng. Syst. Manage. (IESM)*, Oct. 2013, pp. 1–8.

[12] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path planning for autonomous mobile robots: A review," *Sensors*, vol. 21, no. 23, p. 7898, Nov. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/7898

[13] Q. Gu, Y. Chen, J. Zhou, and J. Huang, "A fast linearized virtual element method on graded meshes for nonlinear time-fractional diffusion equations," *Numer. Algorithms*, pp. 1–37, 2024.

[14] H. Liu, Q. Chen, and P. Liu, "A novel memory concurrent editing model for large-scale video streams in edge computing," *Mathematics*, vol. 11, no. 14, p. 3175, Jul. 2023.

[15] Z. Wang, E.-H. Kim, S. KwunOh, W. Pedrycz, Z. Fu, and J. H. Yoon, "Reinforced fuzzy rule-based neural networks realized through streamlined feature selection strategy and fuzzy clustering with distance variation," *IEEE Trans. Fuzzy Syst.*, early access, Jul. 9, 2024, doi: 10.1109/TFUZZ.2024.3422414.

[16] H. Liu, Q. Chen, and P. Liu, "An optimization method of large-scale video stream concurrent transmission for edge computing," *Mathematics*, vol. 11, no. 12, p. 2622, Jun. 2023.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959, doi: 10.1007/bf01386390.

[18] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[19] A. Ammar, H. Bennaceur, I. Chăari, A. Koubăa, and M. Alajlan, "Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments," *Soft Comput.*, vol. 20, no. 10, pp. 4149–4171, Oct. 2016, doi: 10.1007/s00500-015-1750-1.

[20] Q. Lei, L. H. Ling, and L. Jianlong, "Principle explanation and performance analysis of jump point search algorithm," *J. Xinjiang Univ.*, vol. 33, no. 1, pp. 80–87, 2016, doi: 10.13568/j.cnki.651094.2016.01.013.

[21] T. Chen, S. Chen, K. Zhang, G. Qiu, Q. Li, and X. Chen, "A jump point search improved ant colony hybrid optimization algorithm for path planning of mobile robot," *Int. J. Adv. Robotic Syst.*, vol. 19, no. 5, Sep. 2022, Art. no. 172988062211279, doi: 10.1177/17298806221127953.

[22] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations," *Ocean Eng.*, vol. 223, Mar. 2021, Art. no. 108709, doi: 10.1016/j.oceaneng.2021.108709.

[23] M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, and A. Ammar, "Global path planning for mobile robots in large-scale grid environments using genetic algorithms," in *Proc. Int. Conf. Individual Collective Behaviors Robot. (ICBR)*, Dec. 2013, pp. 1–8.

[24] X. Zhong, J. Tian, H. Hu, and X. Peng, "Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment," *J. Intell. Robotic Syst.*, vol. 99, no. 1, pp. 65–77, Jul. 2020, doi: 10.1007/s10846-019-01112-z.

[25] R. C. Holte, T. Mkadmi, R. M. Zimmer, and A. J. MacDonald, "Speeding up problem solving by abstraction: A graph oriented approach," *Artif. Intell.*, vol. 85, nos. 1–2, pp. 321–361, Aug. 1996, doi: 10.1016/0004-3702(95)00111-5.

[26] R. C. Holte, M. B. Perez, R. M. Zimmer, and A. J. MacDonald, "Hierarchical A*: Searching abstraction hierarchies efficiently," in *Proc. AAAI/IAAI*, vol. 1, 1996, pp. 530–535.

[27] X. Zheng, M. Ma, Z. Zhong, A. Yang, L. Chen, and N. Jing, "Two-stage path planning for long-distance off-road path planning based on terrain data," *ISPRS Int. J. Geo-Inf.*, vol. 13, no. 6, p. 184, May 2024, doi: 10.3390/ijgi13060184.

[28] D. Harabor and A. Botea, "Hierarchical path planning for multi-size agents in heterogeneous environments," in *Proc. IEEE Symp. Comput. Intell. Games*, Dec. 2008, pp. 258–265.

[29] N. Sturtevant, J. Traish, J. Tulip, T. Uras, S. Koenig, B. Strasser, A. Botea, D. Harabor, and S. Rabin, "The grid-based path planning competition: 2014 entries and results," in *Proc. Symp. Combinat. Search*, 2015, pp. 241–250.

[30] N. Pelechano and C. Fuentes, "Hierarchical path-finding for navigation meshes (HNA*)," *Comput. Graph.*, vol. 59, pp. 68–78, Oct. 2016.

[31] S. Ting, S. Yuqi, Y. Jianping, Y. Haiyue, and W. Xiande, "Path planning for lunar surface robots based on improved ant colony algorithm," *Trans. Nanjing Univ. Aeronaut. Astronaut.*, vol. 39, no. 6, pp. 672–683, 2022, doi: 10.16356/j.1005-1120.2022.06.004.

[32] C. Chagas, E. Zacarias, L. A. de Lima Silva, and E. Pignaton de Freitas, "Hierarchical and smoothed topographic path planning for large-scale virtual simulation environments," *Exp. Syst. Appl.*, vol. 189, Mar. 2022, Art. no. 116061, doi: 10.1016/j.eswa.2021.116061.

[33] S. K. Moghadam, M. Ebrahimi, and D. D. Harabor, "Guards: Benchmarks for weighted grid-based pathfinding," *Exp. Syst. Appl.*, vol. 249, Sep. 2024, Art. no. 123719, doi: 10.1016/j.eswa.2024.123719.

[34] C.-Y. Kim and S. Sull, "Grid graph reduction for efficient shortest pathfinding," *IEEE Access*, vol. 11, pp. 74263–74276, 2023, doi: 10.1109/ACCESS.2023.3293125.

[35] A. Pershutkin, A. Dukhanov, and P. Gladilin, "An approach to terrain trafficability evaluation based on a neural network for emergency decision-support systems," in *Proc. IEEE 13th Int. Conf. Appl. Inf. Commun. Technol. (AICT)*, Oct. 2019, pp. 1–6.

[36] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta: Any-angle path planning on grids," *J. Artif. Intell. Res.*, vol. 39, pp. 533–579, Oct. 2010, doi: 10.1613/jair.2994.

[37] N. Rivera, C. Hernández, and J. Baier, "Grid pathfinding on the 2K neighborhoods," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1, pp. 891–897, doi: 10.1609/aaai.v31i1.10666.

[38] N. Rivera, C. Hernández, N. Hormazábal, and J. A. Baier, "The 2K neighborhoods for grid path planning," *J. Artif. Intell. Res.*, vol. 67, pp. 81–113, Jan. 2020.

[39] B. Kramm, N. Rivera, C. Hernández, and J. A. Baier, "A suboptimality bound for 2K grid path planning," in *Proc. Symp. Combinat. Search*, 2021, pp. 63–71.

[40] J. Chen, C. Tan, R. Mo, H. Zhang, G. Cai, and H. Li, "Research on path planning of three-neighbor search A* algorithm combined with artificial potential field," *Int. J. Adv. Robotic Syst.*, vol. 18, no. 3, May 2021, Art. no. 172988142110264, doi: 10.1177/17298814211026449.

DUOHANG SUN received the B.S. degree in software engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2022. He is currently pursuing the M.S. degree with The 32nd Research Institute of China Electronics Technology Corporation, Shanghai, China. His main research interests include agent path planning and software engineering.

ZHE SUN received the B.S. degree in computer science and technology from the Ocean University of China (OUC), in 2013, and the M.S. degree in computer science and technology from the East China Institute of Computing Technology, in 2016. His main research interests include computer software and theory, mapping navigation, and digitization. He was awarded the Society Best Symposium Paper Award, in 2011.

PEINAN SHAO received the B.S. degree in computer science from Nanjing University and the M.S. degree in computer science from the East China Institute of Computer Research. He is currently the Chief Expert of China Electronic Technology Group Corporation (CETC) and the Vice-Chief Engineer of The 32nd Research Institute of CETC. His research interests include software engineering and artificial intelligence.