International Journal of Geo-Information

MDPI

Article

Improved A* Navigation Path-Planning Algorithm Based on Hexagonal Grid

Zehua An, Xiaoping Rui * and Chaojie Gao

School of Earth Sciences and Engineering, Hohai University, Nanjing 210098, China; anzehua@hhu.edu.cn (Z.A.); 221309020004@hhu.edu.cn (C.G.)

* Correspondence: ruixp@hhu.edu.cn; Tel.: +86-136-7110-6220

Abstract: Navigation systems are extensively used in everyday life, but the conventional A* algorithm has several limitations in path planning applications within these systems, such as low degrees of freedom in path planning, inadequate consideration of the effects of special regions, and excessive nodes and turns. Addressing these limitations, an enhanced A* algorithm was proposed using regular hexagonal grid mapping. First, the approach to map modeling using hexagonal grids was described. Subsequently, the A* algorithm was refined by optimizing the calculation of movement costs, thus allowing the algorithm to integrate environmental data more effectively and flexibly adjust node costs while ensuring path optimality. A quantitative method was also introduced to assess map complexity and adaptive heuristics that decrease the number of traversed nodes and increase the search speed. Moreover, a turning penalty measure was implemented to minimize unnecessary turns on the planned paths. Simulation results confirmed that the improved A* algorithm exhibits superior performance, which can dynamically adjust movement costs, enhance search efficiency, reduce turns, improve overall path planning quality, and solve critical path planning issues in navigation systems, greatly aiding the development and design of these systems and making them better suited to meet modern navigation requirements.

Keywords: navigation system; path planning; grid map; A* algorithm; regular hexagonal

Citation: An, Z.; Rui, X.; Gao, C. Improved A* Navigation Path-Planning Algorithm Based on Hexagonal Grid. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 166. https://doi.org/10.3390/ijgi13050166

Academic Editor: Wolfgang Kainz

Received: 28 February 2024
Revised: 1 May 2024
Accepted: 11 May 2024
Published: 16 May 2024

1. Introduction

Navigation systems are crucial tools for travel, and path planning is a fundamental aspect of their functionality. The primary objective of path planning is to identify a feasible and optimized route from the starting point to the endpoint [1]. In practice, complex terrain and traffic conditions often lead to disparities between planned paths and actual road conditions. Consequently, the development of advanced path-planning methods has emerged as a prominent topic in academic research.

Existing path-planning methods are primarily composed of two main components: (1) the construction of abstract maps and (2) the path-search algorithm [2]. Map construction encompasses various techniques, including graphical representation [3], navigation mesh representation [4,5], and the grid method [6,7]. Currently, most navigation systems use graphical representations that rely on vector-based road networks for path planning. However, this approach involves pre-set traversable paths with limited degrees of freedom. Notably, certain passable areas, such as grasslands, squares, and mountains, were excluded from the planning scope despite applying to various scenarios.

In contrast, the navigation mesh abstracts the original map into a collection of triangular meshes, utilizing the connection relationships between the triangular meshes for path planning [8]. Although this method provides a more accurate representation of a map, its structure is complex and challenging to implement and modify, and the construction process consumes a considerable amount of memory. Consequently, it is primarily used in 3D scenes, such as video games [9].

The grid method divides a scene evenly into a series of grids by considering each grid as a node in an abstract graph. Subsequently, the search algorithm navigates through the abstract graph [10]. This approach provides significant flexibility, covers the entire traffic area effectively, and addresses all aspects within the realm of path planning. Its principle is straightforward, and its implementation is relatively simple.

In a two-dimensional plane, squares, equilateral triangles, and regular hexagons uniformly cover all areas [11]. Conventional grid-based methods rely on square grids. Figure 1 illustrates the movement pattern in different grid maps, where the blue nodes indicate the current node, the white nodes indicate the neighboring nodes and the black arrows indicate the directions of movement. As illustrated in Figure 1, in a square grid, the target can move from the current node to a neighboring node in either four or eight directions. If a diagonal movement is not allowed, there are only four directions: forward, backward, left, and right, each with a movement cost of one unit. The 8-directional square grid map includes four diagonal directions, resulting in two different movement costs: 1 unit and $\sqrt{2}$ units. Therefore, when calculating the paths based on this map, it is necessary to determine the category of the forward direction, which makes the calculation extremely complex. However, the hexagonal grids feature only six target directions, each with a movement cost of one unit.

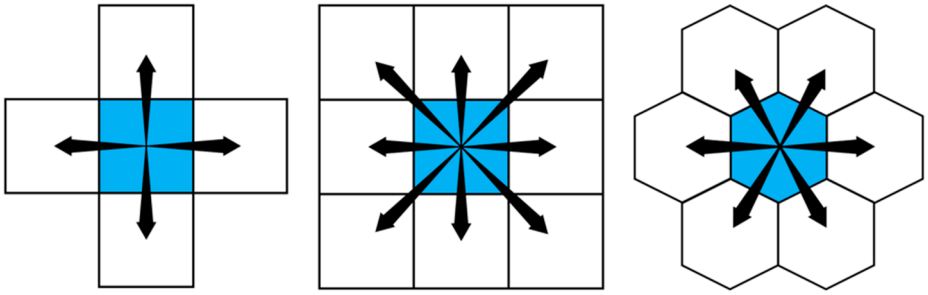

Figure 1. Movement pattern in different grid maps.

Figure 2 illustrates the path on different grid maps, where blue indicates the path nodes, black indicates the obstacle nodes and the black arrows indicate the routes. As depicted in Figure 2, the path exhibits a zigzag pattern with lower smoothness in a 4-neighbor square grid map. In an 8-neighbor square grid map, although the path is smoother, diagonal crossings may cause the map to approach or even contact obstacles, potentially leading to collisions in practical scenarios. However, hexagonal grid maps do not have these limitations. The paths on the hexagonal grid maps demonstrated smoother trajectories devoid of any rigid sensations. Compared to the 8-neighbor square map, the planned path in the hexagonal grid maintained a greater distance from the obstacles. Additionally, each node possesses a single adjacency type with neighboring nodes, which simplifies the algorithm's implementation and enhances search efficiency. Therefore, this study investigates the adoption of hexagonal grid maps over conventional square-grid maps to enhance the A* algorithm.

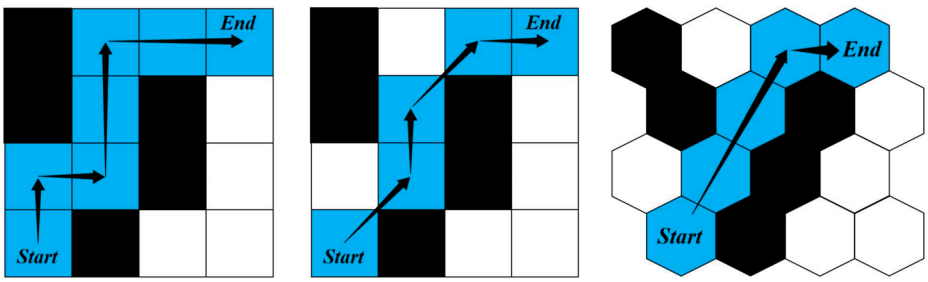

Figure 2. Path on different grid maps.

Various algorithms have been applied in the field of pathfinding, including breadth-first search (BFS) [12], depth-first search (DFS) [13], the Dijkstra algorithm [14], genetic algorithms [15], ant colony algorithms [16], and the A* algorithm [17]. Among the algorithms described above, the A* algorithm is a heuristic pathfinding method that merges the best features of Dijkstra's and greedy algorithms. Renowned for its effectiveness in solving optimal path problems in static environments [18], as early as 1984, Pearl discussed the completeness and optimality of the A* algorithm in his book *Intelligent Search Strategies for Computer Problem Solving*. However, issues such as a fixed heuristic function and the limited impact of heuristics often limit the adaptability of the A* algorithm to the influence of special areas on the movement costs during path searches. These issues also result in the traversal of many unnecessary nodes, accompanied by the creation of numerous turning points.

Since the introduction of the A* algorithm in 1968, scholars worldwide have continuously researched and optimized it. Current improvements to the A* algorithm primarily focus on optimizing distance calculation methods, enhancing heuristic functions, and secondary path optimization. The optimization of distance calculation methods [19,20] involves refining the estimation methods for the distance between the current point and the destination. By fully and rationally using the destination's location information, this approach reduces the path length and enhances the efficiency of the algorithm. However, this method prioritizes distance cost as the main criterion and overlooks the impact of special terrain on robot mobility. Enhancements to the heuristic function [19–24] involve weighting the components of the heuristic or incorporating new guiding metrics that address issues such as slow search speeds and excessive node traversals. However, these improvements lack adaptability to different maps and often result in suboptimal solutions in complex environments. Secondary path optimization [25] involves the additional processing of paths planned by the A* algorithm, which shortens the path length and improves smoothness. However, this approach requires the design of additional path-smoothing methods, thereby increasing the complexity of the algorithm and reducing the pathfinding efficiency. Moreover, the aforementioned improvements did not consider the impact of special terrain on the algorithm but rather focused more on obstacles.

In practical applications, maps often encompass unique terrains, rendering the aforementioned algorithmic enhancements inadequate for complex environments. To address this issue, the current study emphasizes the use of regular hexagonal grid maps and integrates terrain information to guide navigation pathfinding. This integration aims to enhance algorithmic efficiency and improve the smoothness of the planned paths. Thus, conventional A* algorithms, constrained by their inability to accommodate special terrain, excessive node traversals, and numerous inflection points on raster maps, were adapted. An improved A* algorithm was proposed, featuring an enhanced heuristic function with newly designed elements and the incorporation of environmental information. This refinement not only preserves the optimality of the planned path but also enhances efficiency, environmental adaptability, and path quality.

2. Conventional A* Algorithm

Pathfinding algorithms can be classified into blind and heuristic categories based on whether they use heuristics. Blind algorithms conduct searches based on predetermined strategies without considering the characteristics of the map or destination. Examples include the breadth-first search (BFS) and Dijkstra's algorithm. However, heuristic algorithms utilize estimated costs to reach the destination, calculating heuristic values to expedite the search. Representative examples include greedy and A* algorithms. The most typical one is the greedy algorithm, which selects the node with the minimum estimated cost to reach the destination at each expansion step. The A* algorithm combines the features of both Dijkstra's and Greedy's algorithms with its heuristic function, defined as

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

where n represents the node currently traversed during the search process; $g(n)$ represents the cost from the starting point to the current node; $h(n)$ represents the estimated cost from the current node to the destination, which serves as the heuristic value in the A* algorithm; and $f(n)$ is the heuristic function of the A* algorithm.

$$g(n) = \sum_{i=0}^{n} S(i) \tag{2}$$

$S(i)$ represents the distance movement for the corresponding node. For example, $S(n)$ represents the actual distance of a single step from the parent node $(n-1)$ to the current node n. Therefore, $g(n)$ is the sum of the distances for each segment of the planned path. In a hexagonal grid where the distance between adjacent nodes is uniform, you can set $S(i) = 1$ to facilitate the operation of the algorithm.

Heuristic algorithms require the calculation of distances between nodes, and the distance calculation involves the assistance of a coordinate system. Common hexagonal grid coordinate systems include cubic, offset, and axial coordinates. Among these, the cubic coordinate system is easy to understand, and its distance calculation formula is simple and clear. Therefore, a cubic coordinate system was adopted for the distance calculation. In contrast to the x- and y-axes in the Cartesian coordinates for square grid maps (as shown in Figure 3), the cubic coordinate system has three axes: q, r, and s, representing the grid coordinates as (q, r, s).

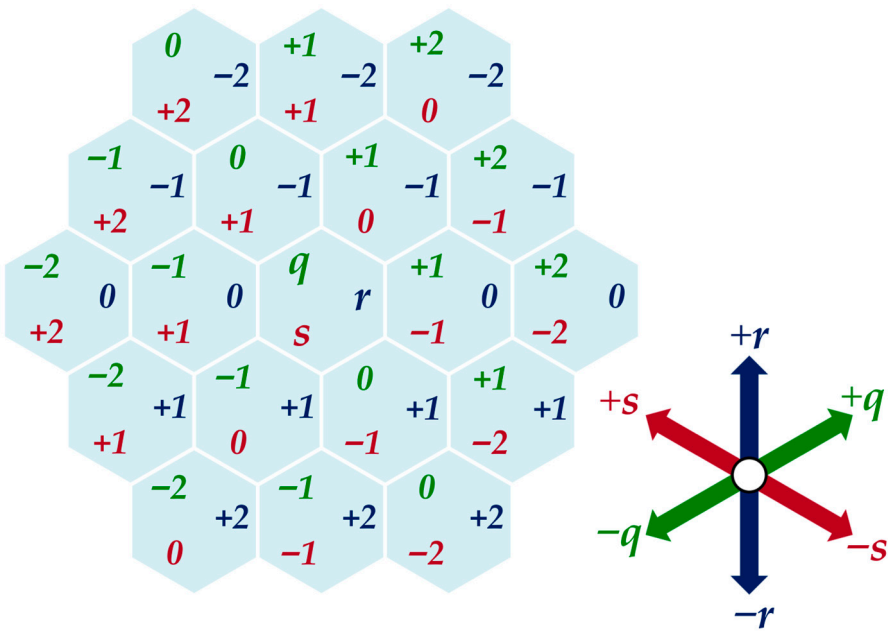

Figure 3. Diagram of cubic coordinate system.

Common distance calculation methods include the Euclidean, Manhattan, and diagonal distances, among which the Manhattan distance, also known as a taxicab or city block distance, measures the total absolute difference along each axis between two points in a grid-based environment. In hexagonal grids, it is intuitive to measure the distance between two nodes by using the number of steps per unit. Therefore, the Manhattan distance was used to estimate the movement costs and calculate the distance from the current node to the destination.

For grid $\boldsymbol{A}$ with coordinates (q_1, r_1, s_1) and grid $\boldsymbol{B}$ with coordinates (q_2, r_2, s_2), the Manhattan distance between $\boldsymbol{A}$ and $\boldsymbol{B}$ is expressed as:

$$D(\boldsymbol{A}, \boldsymbol{B}) = \frac{|\boldsymbol{A}.q - \boldsymbol{B}.q| + |\boldsymbol{A}.r - \boldsymbol{B}.r| + |\boldsymbol{A}.s - \boldsymbol{B}.s|}{2} \tag{3}$$

During the search process of the A* algorithm, each time it finds the node with the smallest heuristic function value for expansion, the specific steps of the algorithm are as follows:

(1) Create *OpenList* and *ClosedList* collections; add the starting point to the *OpenList*, and set the starting point's $g(n) = 0$.
(2) Find the node with the smallest $f(n)$ value in the *OpenList*. If the *OpenList* is empty, the search fails, and the search ends.
(3) Add the node to the *ClosedList* and traverse all neighbor nodes of the current node that are not in the *ClosedList*.
    a If the neighbor node is not on *OpenList*, let the neighbor node's movement cost $g(n + 1)$ equal $g(n) + 1$, and set the current node as the parent node of this neighbor node.
    b If the neighbor node is in *OpenList* and the calculated $g(n)$ from the current node to this neighbor node is smaller than the previously calculated $g(n)$ from the parent node, then update the value of $g(n)$, and the current node is set as the parent node of this neighbor node; if it is greater, then it is not updated.
(4) Repeat Steps 2 and 3 until the node found in Step 2 is the endpoint.

The parent node was traced sequentially from the endpoint. If the parent node is the starting point, it ends, and the path planning result is obtained.

A flowchart of the A* algorithm is shown in Figure 4.

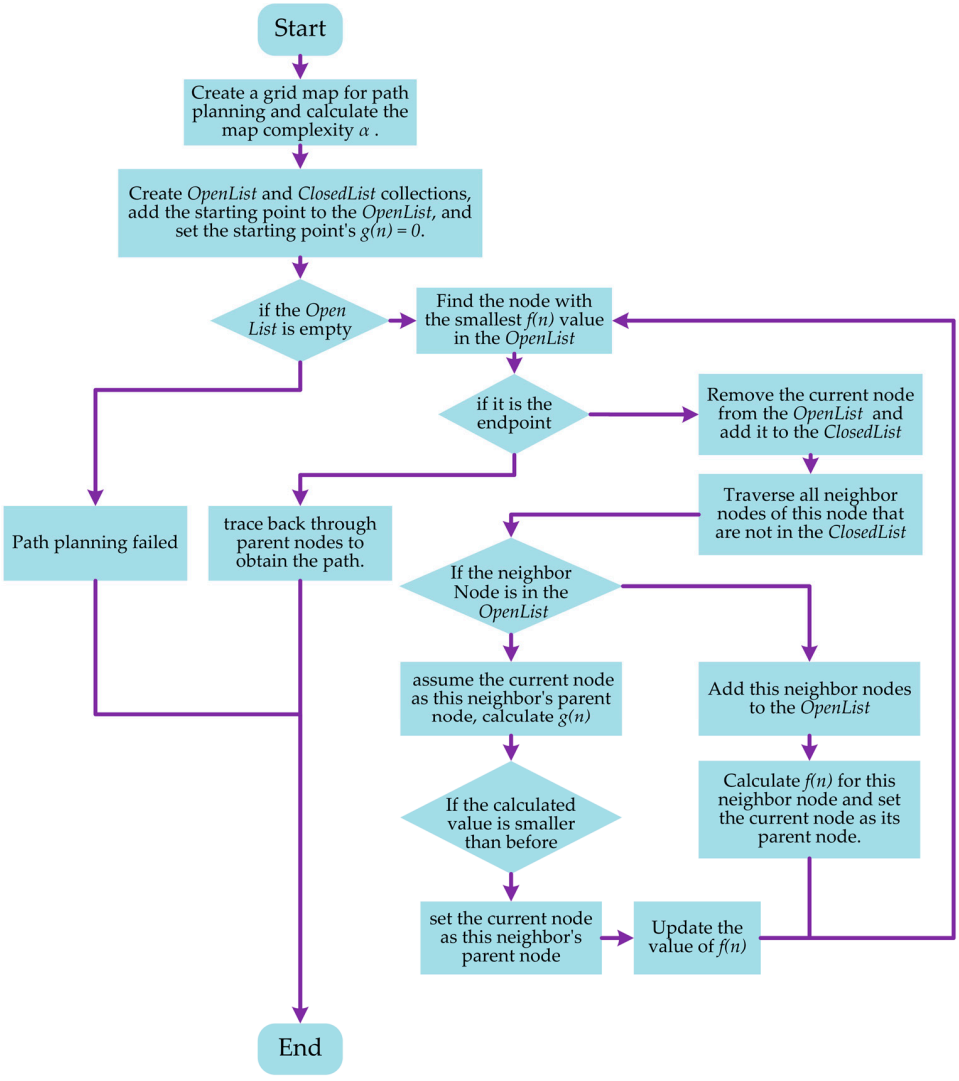

Figure 4. Flowchart of the A* algorithm.

3. Improved A* Algorithm

The conventional A* algorithm is not influenced by special terrain. During the path search process, problems such as a high number of traversed nodes, numerous turning points, and a large total turning angle occur. The algorithm is enhanced by implementing a flexible cost strategy, an adaptive heuristic function, turning the penalty function, and incorporating the impact of special terrain into the path search. While ensuring path optimality, this approach reduces the number of traversed nodes, enhances algorithm efficiency, and enables quick and stable planning of inflection points and total turning angles, resulting in a more reasonable navigation path.

3.1. Flexible Setting of Costs

The conventional grid method typically defines two attributes: passability and inaccessibility. In practical applications, path planning maps contain some special areas, such as grasslands and hills, which are often included. These areas are traversable; however, because of their specific characteristics, distance cannot be used directly as the cost of movement in these areas. Therefore, we define the grid nodes in three states: (1) Ordinary nodes, (2) nodes in special areas, and (3) obstacle nodes. The node states are stored when abstracting maps. As the algorithm traverses a new node, if the node is a regular node, the movement cost for that node is determined by the distance, specifically set to one. If the node is located within a special area, the transit cost is determined by considering both the distance and the effects of the special area. If a node is an obstacle node, it is not expanded. Further improvements include enhancing the calculation method for $g(n)$ and introducing the influencing factor K into the heuristic function. K represents the difficulty of passing through a special area. If node i is situated in a region influenced by factor K, then the single-step movement cost $S'(i)$ from the node i's parent node to node i is defined as follows:

$$S'(i) = 1 + K \quad (K > -1) \tag{4}$$

The constant term 1 represents the basic distance cost of moving to node i; the overall movement cost, or the difficulty of passage, is directly proportional to K. When $K = 0$, the node is considered ordinary, and when $K = 1$, the cost of moving to node i is twice that of moving to an ordinary node. When the K value is sufficiently large, node i can be considered located in an impassable obstacle.

Moreover, a path node may be located at the intersection of multiple areas, and the impact of these areas on the cost calculation of the node must be comprehensively considered. If m special areas are superimposed at the location of node i, the expression for $S(i)$ is

$$S'(i) = 1 + \sum_{j=1}^{m} K_j \tag{5}$$

where K_j represents the influencing factor for the special area j, and this formula is used to calculate the cost $g'(n)$ of moving to the current node after being affected.

3.2. Adaptive Heuristics

By incorporating the influence of the parent node $(n - 1)$, the heuristic value from the parent node is added to the current node's heuristic function, increasing the overall proportion of the heuristic value within the heuristic function. This enhancement can accelerate search speed, traverse fewer nodes during the search, and reduce computational load. However, excessively high search speeds may lead to the emergence of locally optimal solutions. To enable the algorithm to accelerate the search speed on simpler maps and adopt a more cautious approach for complex maps, it is necessary to adjust the heuristic weight based on map information, allowing it to adaptively control the search speed. If the number of nodes in the current map is x, the number of obstacle nodes is y, there are m types of special areas, and the number of nodes with corresponding area properties is ε,

with corresponding special area influence factors represented by K, all these parameters determined by map information, then the improved heuristic expression $h'(n)$ is as follows:

$$h'(n) = (1 - \alpha) \times [h(n) + h(n - 1)] \tag{6}$$

$$\alpha = \frac{x}{y + \sum_{i=0}^{m} \frac{\varepsilon_i K_i}{1 + K_i}} \tag{7}$$

In the above equation, $1 - \alpha$ represents the adaptive weight multiplier for the heuristic, and α denotes the map complexity, with a value range of [0, 1]. When the extent of obstacles and special areas is large, the complexity of the map increases, reducing the heuristic value and consequently slowing the search speed.

To verify the impact of the adaptive heuristic values on the search speed, an experiment was conducted on a 30 × 30 hexagonal grid map, and the results are displayed in Figures 2 and 5. Figure 5a shows the search results using the conventional A* algorithm; Figure 5b shows the search results when only the heuristic value from the parent node is introduced; and Figure 5c presents the search results after introducing the adaptive heuristic proposed in this study. In the figures, gray nodes represent ordinary nodes, black nodes represent obstacle nodes, light blue nodes denote path nodes, and orange nodes indicate nodes traversed during the search. The black triangle marks the starting point of the path, and the black pentagon marks the endpoint of the path. And the dark blue line is the planned path. The calculated value of α is 0.276. A comparison of the search results is presented in Table 1.

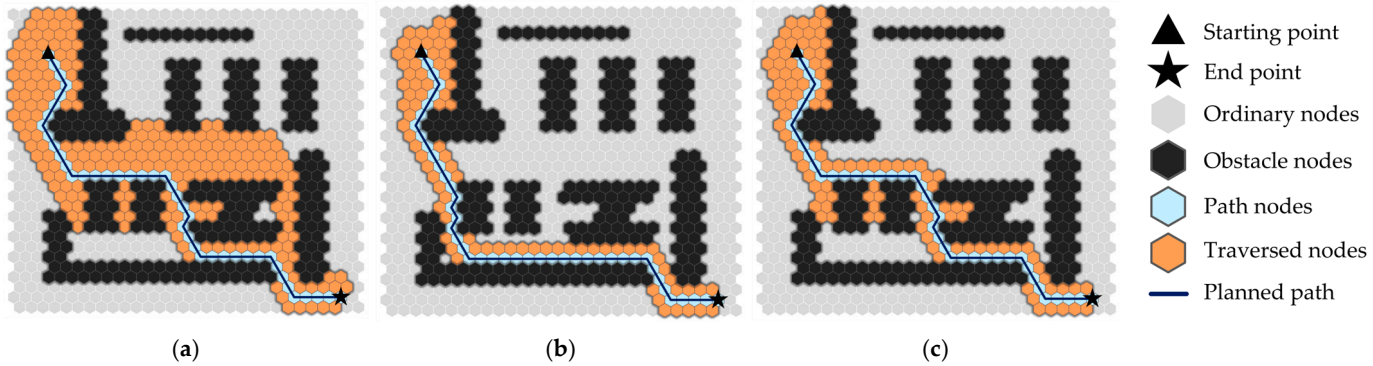

Figure 5. Adaptive heuristic test results. (**a**) Conventional A* algorithm. (**b**) Introducing the influence of the parent node. (**c**) Introducing adaptive heuristics.

Table 1. Comparison of search results.

Test Items	Conventional A* Algorithm	Introducing the Influence of Parent Node	Introducing Adaptive Heuristics
Number of path nodes/number	43	44	43
Number of traversed nodes/node	261	125	149
Number of turning points/piece	9	9	9
Total turning angle/°	540	540	540

The results indicate that, after introducing the heuristic value from the parent node, there is a significant reduction in the number of nodes traversed. As the time required to traverse each node remains the same, the search speed is also improved. However, the planned path loses optimality. However, after incorporating adaptive heuristic values into the heuristic function, the algorithm maintains path optimality, and the number of traversed nodes is significantly reduced, thereby demonstrating the feasibility of this

approach. However, there is still redundancy in the turning points across the various algorithms, necessitating further optimization and improvement.

3.3. Turn Penalty

The formation of redundant turning was due to the low stability of the algorithm during the search process. When faced with multiple paths having the same movement cost, the algorithm fails to select a route with a smaller number of turning points and a smaller total turning angle. Therefore, this study proposes the incorporation of a penalty term to suppress the formation of turning points, thus enabling the algorithm to maintain the direction of path progression and avoid turning.

The direction of the path progression is shown in Figure 6, where the blue node represents the current node, the orange node represents the parent node of the current node, and the gray nodes are the neighboring nodes of the current node, which will be traversed in the next iteration. The black arrow indicates the direction of progression from the parent node to the current node, represented by vector $\boldsymbol{a}$; the red arrow indicates the possible direction of progression from the current node to a new node, represented by vector $\boldsymbol{b}$.

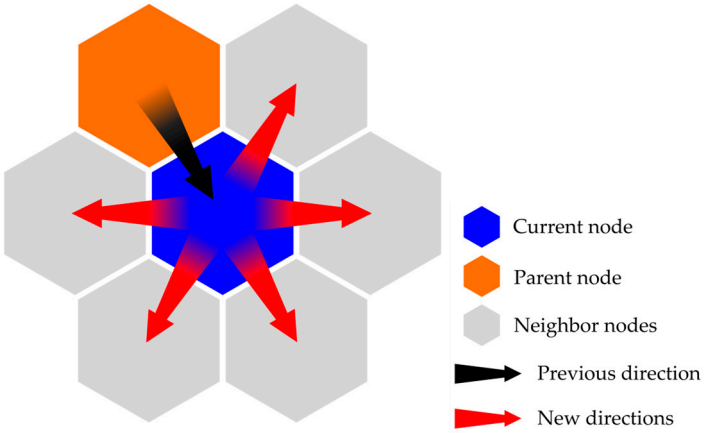

Figure 6. Schematic of path progression direction.

When traversing a new node, there are six directions for expansion, which, compared to the previously planned route, yields four outcomes: 1. Continue along the previous route; 2. The new node direction differed from the previous path direction by 60°; and 3. The new node direction differed from the previous path direction by 120°. It can be observed that the turning angle of the route is related to the angle θ between the previous path direction and the new node direction. Assume that the current node is n, its parent node is $(n-1)$, and the new node is $(n+1)$, then the angle θ is calculated as:

$$cos\ \theta = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}| \cdot |\boldsymbol{b}|} \tag{8}$$

where $\boldsymbol{a}$ is the vector from the current node to the new node, and $\boldsymbol{b}$ is the vector from the parent node to the current node.

Add a penalty value of $punish(n+1)$ to each new node; $punish(n+1) = \beta(1 - cos\,\theta)$; the value range of the penalty value is [0, 2a]. The factor β serves as a penalty intensity coefficient, with the penalty intensity being directly proportional to the value of β. Accordingly, the algorithm prioritizes the expansion of new nodes, where the direction of progression remains unchanged, to minimize the number of turns during the search and reduce the number of turning points.

3.4. Final Improved Algorithm

To enable the conventional A* algorithm to adapt to the influence of complex maps, prioritize planning paths in areas with low traffic difficulty, enhance search speed, reduce the number of traversed nodes, and minimize turning points and angles, this study proposes a flexible setting of movement costs, adaptive heuristics, and a turning penalty strategy, ultimately resulting in an improved A* algorithm. First, a regular hexagonal grid map is used for abstraction; the special terrain influence factor information is stored in the map, and α is calculated. The expression for the improved A* algorithm's heuristic function $f'(n)$ is as follows:

$$f'(n) = g'(n) + (1 - \alpha)\,[h(n) + h(n-1)] + punish(n) \tag{9}$$

4. Algorithm Simulation

To verify the performance of the improved A* algorithm proposed in this study, we simulated it using the Unreal Engine 5 (UE5), version 5.0.3. UE5 is a powerful game development engine increasingly used by researchers for developing navigation systems, making it an excellent software tool for the experiments conducted in this study. We constructed an experimental map of size 30 × 30, which included 191 obstacle nodes and 248 grassland nodes, with the influence factor K set to 1. Through calculations using Equation (7), the complexity α of this map was determined to be 0.35. The map and selection of the start and end points are shown in Figure 7. In the figure, gray nodes represent ordinary nodes, light green nodes represent grass nodes, and black nodes represent obstacle nodes. The black triangle marks the starting point of the path, and the black pentagon marks the endpoint of the path.

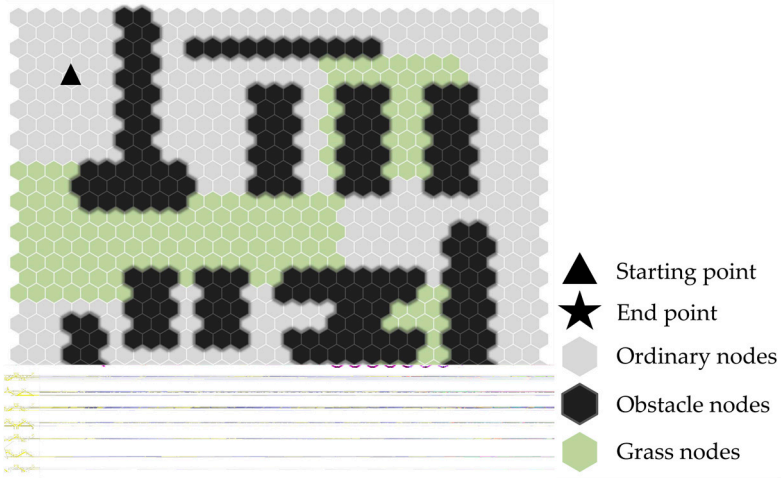

Figure 7. Experimental map.

First, we conducted experiments on the magnitude of the penalty intensity coefficient β under the premise of introducing flexible cost settings and adaptive heuristics. The search results for the algorithms with different β values are listed in Table 2.

Table 2. Performance comparison of different β value algorithms.

β Value	0	0.25	0.5	0.75	1	1.5	2
Number of path nodes/number	44	44	44	44	44	44	44
Path movement cost/step	60	60	60	60	60	60	60
Number of traversed nodes/node	263	261	261	267	267	276	280
Number of turning points/piece	17	12	10	10	10	12	12
Total turning angle/°	1020	720	600	600	600	720	720

From the analysis of the experimental results, the introduction of a turning penalty reduces both the number of turning points and the total turning angle. When β exceeds 0.5, the number of nodes traversed increases because of the excessive penalty, causing the algorithm to traverse more nodes where the direction of progression has not changed rather than nodes closer to the destination. When β is between 0.5 and 1, the path exhibits the fewest turning points and the smallest total turning angle, with a 41% reduction in the number of turning points and a 54% reduction in the number of redundant turning points, demonstrating significant optimization effects. Therefore, we select a value of 0.5 for β in the improved algorithm described in this study.

To evaluate the effectiveness of the proposed algorithm, we conducted simulations for comparison with the conventional A* algorithm, an algorithm featuring flexible cost settings, an algorithm with an adaptive heuristic, and the algorithms described in [19,20]. These simulations were performed on the map designed in the previous section. The resultant paths generated by each algorithm are depicted in Figure 8, where gray indicates ordinary nodes, light green denotes grassland nodes, light blue highlights the paths identified by the algorithms, orange marks ordinary nodes traversed during the search, and green signifies grassland nodes traversed. A comparative analysis of these algorithms is presented in Table 3, which lists their performance metrics.

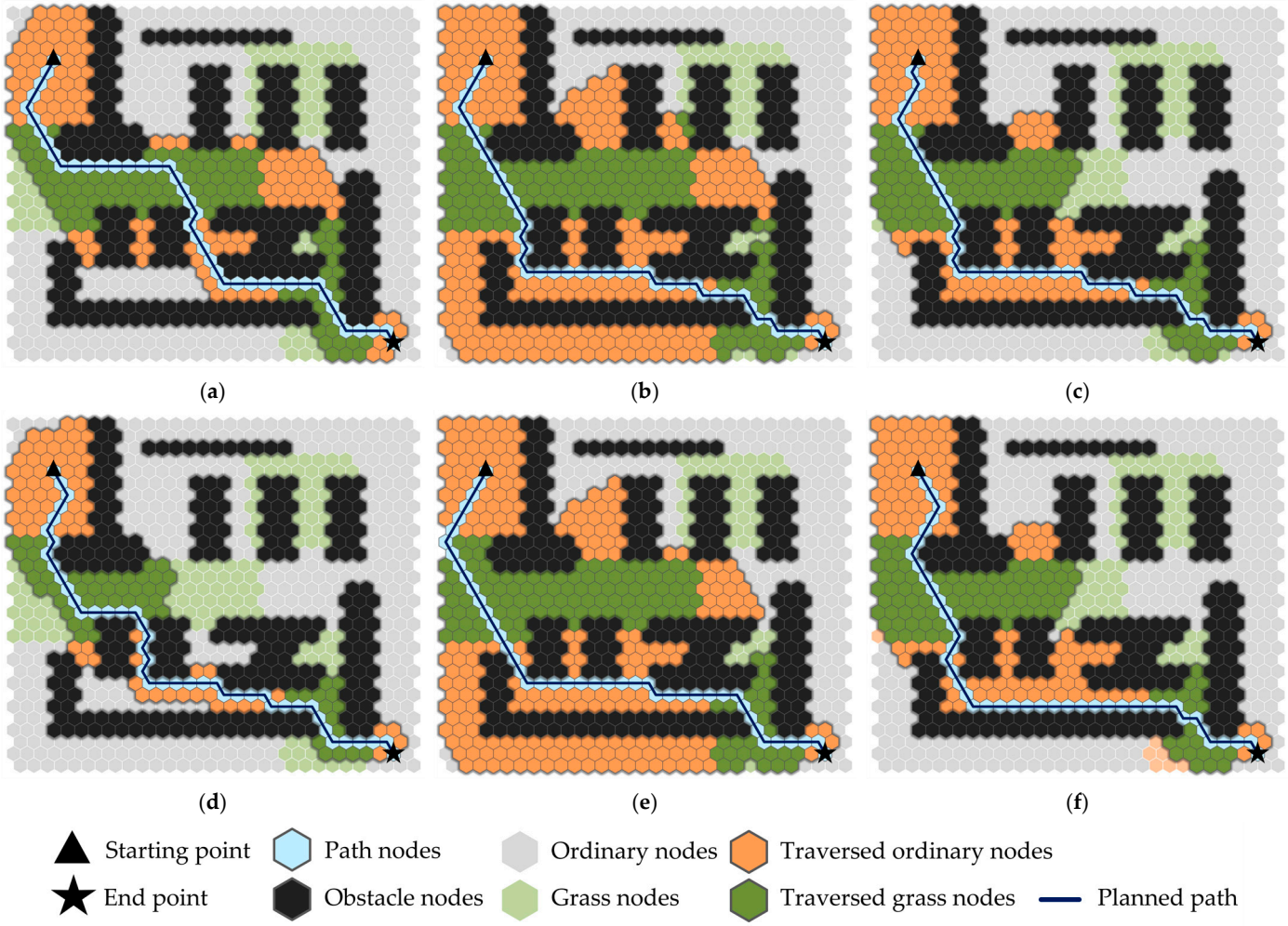

Figure 8. Results from different algorithms. (**a**) Conventional A* algorithm. (**b**) Improved algorithm for flexible cost setting. (**c**) Introduce adaptive heuristics. (**d**) The algorithm mentioned in reference [19]. (**e**) The algorithm mentioned in reference [20]. (**f**) Improved A* algorithm of this study.

Table 3. Performance comparison of different algorithms.

Index	Conventional A* Algorithm	Flexible Setting of Costs A* Algorithm	Introduce Adaptive Amount of Inspiration	The Algorithm Mentioned in Reference [19]	The Algorithm Mentioned in Reference [20]	Improved A* Algorithm of This Study
Number of path nodes/step	43	44	44	44	44	44
Path movement cost/step	66	60	60	63	60	60
Number of traversed nodes/number	261	420	263	172	409	261
Number of turning points/piece	9	15	17	18	7	10
Total turning angle/°	540	900	1020	1080	420	600

It can be observed that the conventional A* algorithm only determines the shortest distance path between the start and end points, traversing long distances through grassy areas, which increases the overall movement cost and loses optimality. After adopting a flexible cost-setting method, the algorithm identifies the path with the lowest cost and achieves optimality; however, owing to the influence of special terrains, the number of traversed nodes and turning points significantly increases. The introduction of adaptive heuristic values maintains path optimality while substantially reducing the number of traversed nodes. However, the A* algorithm already exhibits redundancy in the number of turning points and angles, and the introduction of adaptive heuristic values adds three more turning points, necessitating further optimization.

The algorithm mentioned in [19] uses the average of the Manhattan and Euclidean distances for distance calculation. Although this visually accelerates the approach to the destination, it also results in excessive turning points. Additionally, by incorporating parent node information into the heuristic calculation, a significant reduction in the number of traversed nodes was observed, which was 59.0% lower than that of the A* algorithm with flexible cost settings, significantly enhancing the search speed. However, this was achieved at the expense of sacrificing the optimality and stability of the algorithm. From Figure 8d, it is evident that the algorithm chooses to move across grassy areas in the early part of the path, increasing the movement cost by three and adding three additional turning points, thereby losing path optimality.

The algorithm in [20] introduced the orientation information of the target node into the heuristic function. It first calculates the cosine of the angle between the direction of advance and the target direction, using this cosine value to either increase or decrease the cost based on the presence of obstacles in the forward direction, thus quickly circumventing them. The algorithm in reference [18] is shown to find the optimal path, reducing the number of turning points and the total turning angle by more than half, although the decrease in the number of traversed nodes is minimal, resulting in lower search efficiency.

The improved A* algorithm proposed in this study determines the optimal path on maps with special regions, showing a substantial improvement over the flexible cost-setting A* algorithm in terms of the number of traversed nodes, turning points, and turning angles. Compared to the algorithm in [19], although it traverses more nodes, it plans an optimal path with fewer turning points. Relative to the algorithm in [20], it has more turning points, but the number of nodes traversed differs significantly, demonstrating a significant improvement in the pathfinding efficiency. The analysis in Table 3 shows that the improved A* algorithm outperforms the conventional A* algorithm. Specifically, the improved A* algorithm reduces the overall movement cost by six, a decrease of 9.1% while maintaining path optimality, and reduces the average number of nodes traversed by 21.8%, with an average decrease in the number of turning points and the total turning angle of 25.0%.

To validate the performance of the improved algorithm for different obstacles and grassland areas, we conducted a simulation using a 30 × 30 map. On this map, obstacles

and grass nodes are generated at random locations based on different probabilities. The conventional A*, flexible cost setting, and improved algorithms presented in this study were simulated under the same starting conditions. The results are summarized in Table 4. Experiments where the obstacle generation probability was set to 0.2 and 0.4, with a grass generation probability of 0, did not include results for the flexible cost-setting algorithm. This is because the flexible cost-setting algorithm is identical to the A* algorithm when no grass is present. Therefore, those results were omitted. Conventional A* algorithm plans shorter paths but ignores the impact of grassy areas, resulting in higher movement costs. By contrast, the improved algorithm maintained the minimum movement cost even when facing changes in obstacles and grassy areas.

Table 4. Comparison of experimental results and data under different maps.

Obstacle Generation Probability		**0.2**	**0.4**	**0**	**0**	**0.2**	**0.2**	**0.4**	**0.4**
Grass Generation Probability		**0**	**0**	**0.2**	**0.4**	**0.2**	**0.4**	**0.2**	**0.4**
Map Complexity α		**0.211**	**0.401**	**0.089**	**0.202**	**0.283**	**0.376**	**0.438**	**0.531**
Conventional A* algorithm	Number of path nodes/step	45	54	41	40	44	40	48	46
	Path movement cost/step	44	53	47	59	48	57	59	60
	Number of traversed nodes/number	191	368	204	198	175	127	166	114
	Number of turning points/piece	18	27	8	8	12	12	24	20
Flexible setting of costs A* algorithm	Number of path nodes/step	—	—	41	41	44	40	50	47
	Path movement cost/step	—	—	41	44	46	47	57	56
	Number of traversed nodes/number	—	—	204	533	165	173	343	182
	Number of turning points/piece	—	—	14	15	14	20	22	26
Improved A* algorithm of this paper	Number of path nodes/step	45	54	41	41	44	40	50	47
	Path movement cost/step	44	53	41	44	46	47	57	56
	Number of traversed nodes/number	106	183	128	220	114	110	172	198
	Number of turning points/piece	6	23	4	15	8	16	22	26

When the number of obstacles and grass nodes was low, the proposed algorithm discussed in this paper traversed fewer nodes and planned paths with significantly fewer turning points than the flexible cost-setting algorithm. This is because the map was not particularly complex at this stage, with paths less obstructed by obstacles and grassy areas. The adaptive heuristic and turning penalty strategies enabled the algorithm to rapidly approach the endpoint, effectively enhancing search efficiency. As the probability of generating obstacles and grass increased, making the map more complex, the difference in the number of nodes traversed and the turning points between the improved and flexible cost-setting algorithms decreased. When the obstacle and grass probabilities reached 0.4, the map became sufficiently complex (complexity exceeded 0.5), making the algorithm more cautious than the conventional A* algorithm during path searches. It traverses more nodes than the flexible cost-setting algorithm to ensure that it determines the path with the lowest movement cost. When the map complexity is high, the space for path selection decreases, making turning points unavoidable. However, the paths planned by the improved algorithm consistently had fewer or equal turning points than those planned by the flexible cost-setting algorithm, enabling smoother path planning.

These experimental analyses demonstrated that the improved A* algorithm can adapt to complex mapping environments. It exhibited good adaptability and stability when faced with maps containing different numbers of obstacles and special terrain. It can plan optimal paths with higher search efficiency and fewer turning points and angles, making it more suitable for path planning in navigation systems.

5. Conclusions

This study focused on path planning within navigation systems and conducted a comparative analysis between square and regular hexagonal grid maps. This study proposed

an improved A* algorithm tailored to hexagonal grids. The following conclusions were drawn from the experimental comparative analysis:

- To address the issues of conventional A* algorithms on grid maps that do not respond to special terrain, feature a large number of traversed nodes, and have many turning points, the movement cost calculation method was improved. By introducing an influencing factor, the movement costs were set flexibly, allowing the path search to fully consider the effects of the special terrain. Subsequently, adaptive heuristics were introduced to adjust the heuristics affected by adding parent nodes using map information, which accelerated the search speed while maintaining optimal path conditions. Finally, a turning penalty was designed to reduce the number of turns in the path.
- The improved algorithm exhibited good adaptability when facing maps with varying numbers of obstacles and special terrains. Under the premise of minimizing path movement costs, it can adjust the search speed based on the map conditions, reducing the number of nodes traversed, turning points, and turning angles, thereby enhancing the algorithm efficiency and path quality and demonstrating remarkable stability.
- The improved A* algorithm proposed in this study optimizes path planning on complex maps with special areas such as grasslands and hills, outperforming conventional A* algorithms and other improved algorithms from the literature. Overall, the path movement costs of the improved algorithm presented in this study decreased by 9.1%. While maintaining path optimality, the average number of traversed nodes was reduced by 21.8%, and the number of turning points and the total turning angle both decreased by an average of 25.0%.

In summary, the improved A* algorithm demonstrated significant superiority and adaptability, providing an effective solution for path planning in navigation systems and producing paths that are more realistic and consistent with common travel practices.

The algorithm primarily targets static path planning with complete environmental information and cannot incorporate dynamically appearing or disappearing obstacles and special terrain into path planning. Moreover, pathfinding cannot be performed when the environmental information is not fully known. Our future research will focus on dynamic and partial path planning to ensure accurate and efficient pathfinding in dynamic environments and situations with limited environmental information.

Author Contributions: Conceptualization, Zehua An and Xiaoping Rui; methodology, Zehua An and Xiaoping Rui; software, Zehua An; validation, Zehua An and Chaojie Gao; formal analysis, Zehua An and Chaojie Gao; investigation, Zehua An; resources, Zehua An and Chaojie Gao; data curation, Zehua An; writing—original draft preparation, Zehua An; writing—review and editing, Zehua An; visualization, Xiaoping Rui and Chaojie Gao; supervision, Xiaoping Rui; project administration, Zehua An; funding acquisition, Xiaoping Rui. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 42376180; No. 41771478).

Data Availability Statement: Data are contained within the article.

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

References

1. Tomás, L.; Wesley, M. An Algorithm for Planning Collision Free Paths Among Polyhedral Obstacles. *Commun. ACM* **1979**, *22*, 560–570. [CrossRef]
2. Gao, T.; Liu, W. Summarization of intelligent pathfinding-oriented 3D scene. *Comput. Eng. Appl.* **2017**, *53*, 16–22.
3. Teo, T.; Cho, K. BIM-oriented indoor network model for indoor and outdoor combined route planning. *Adv. Eng. Inform.* **2016**, *30*, 268–282. [CrossRef]
4. Lou, X.; Sun, M.; Yang, S. A fine-grained navigation network construction method for urban environments. *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *113*, 102994. [CrossRef]

5. Zhu, J.; She, P.; Li, W.; Cao, Y.; Qi, H.; Wang, B.; Wang, Y. Dynamic Planning Method for Indoor-Fire Escape Path Based on Navigation Grid. *J. Southwest Jiaotong Univ.* **2020**, *55*, 1103–1110.
6. Liu, Z.; Zhang, L.; Wei, N.; Kuang, X. Research on gridding and path planning of environmental map. *Ship Sci. Technol.* **2021**, *43*, 141–145.
7. Wang, W.; Ai, T.; Yan, X.; Lu, W. Indoor Route Planning Under Regular Hexagonal Grid Considering. *Geomat. Inf. Sci. Wuhan Univ.* **2020**, *45*, 111–118.
8. Meysami, A.; Cuillière, J.-C.; François, V.; Kelouwani, S. Investigating the Impact of Triangle and Quadrangle Mesh Representations on AGV Path Planning for Various Indoor Environments: With or without Inflation. *Robotics* **2022**, *11*, 50. [CrossRef]
9. Xiao, C.; Hao, S. An Overview of Pathfinding in Navigation Mesh. *Int. J. Comput. Sci. Netw. Secur.* **2012**, *12*, 48–51.
10. Zhu, Q.; Zhang, Y. An Ant Colony Algorithm Based on Grid Methodfor Mobile Robot Path Planning. *Robot* **2005**, *27*, 132–136.
11. Liu, Y.; Zhou, X.; Yan, Y. Research on image processing algorithms on hexagonal grid. *Comput. Eng. Des.* **2001**, *5*, 71–75.
12. Tripathy, H.; Mishra, S.; Thakkar, H.; Rai, D. A Collision-Aware Mobile Robot Navigation in Grid Environment using Improved Breadth First Search. *Comput. Electr. Eng.* **2021**, *94*, 107327. [CrossRef]
13. Gang, T. R-DFS: A Coverage Path Planning Approach Based on Region Optimal Decomposition. *Remote Sens.* **2021**, *13*, 1525. [CrossRef]
14. Wang, J.; Wei, G.; Dong, X. A dynamic fire escape path planning method with BIM. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 10253–10265. [CrossRef]
15. Tu, Y.; Zhao, Y.; Liu, L.; Nie, L. Travel route planning of core scenic spots based on best-worst method and genetic algorithm: A case study. *Manag. Syst. Eng.* **2022**, *1*, 4. [CrossRef]
16. Baker, L. Path Planning of Mobile Robot based on Improved Ant Colony Algorithm. *Sci. Technol. Eng.* **2022**, *22*, 12484–12490. [CrossRef]
17. Hart, P.; Nilsson, N.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
18. Su, L.; Jiang, H. Building Internal Path Planning Algorithm and its Application Reviewed Research. *Geomat. Spat. Inf. Technol.* **2014**, *10*, 105–109.
19. Chen, C. Research on Robot Shortest Path Planning with Improved A* Algorithm. *Comput. Digit. Eng.* **2023**, *51*, 1697–1701.
20. Wu, S.; Huang, Y.; Chen, T. Static Route Planning of Surface Ships Based on Improved A* Algorithm. *Comput. Eng. Appl.* **2022**, *58*, 307–315.
21. Zhang, Y.; Tang, G.; Chen, L. Improved A* Algorithm for Time—Dependent Vehicle Routing Problem. *Control. Eng. China* **2012**, *19*, 750–752, 756.
22. Li, X.; Hu, X.; Wang, Z.; Du, Z. Path Planning Based on Combination of Improved A-STAR Algorithm and DWA Algorithm. In Proceedings of the 2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM), Manchester, UK, 15–17 October 2020; pp. 99–103. [CrossRef]
23. Ju, C.; Luo, Q.; Yan, X. Path Planning Using an Improved A-star Algorithm. In Proceedings of the 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), Jinan, China, 23–25 October 2020; pp. 23–26.
24. Zhang, H.; Li, M.; Yang, L. Safe Path Planning of Mobile Robot Based on Improved A* Algorithm in Complex Terrains. *Algorithms* **2018**, *11*, 44. [CrossRef]
25. Zhou, J.; Yang, L.; Zhang, C. Indoor robot path planning based on improved A* algorithm. *Mod. Electron. Tech.* **2022**, *45*, 181–186.