Hindawi
International Journal of Aerospace Engineering
Volume 2023, Article ID 8614117, 18 pages
https://doi.org/10.1155/2023/8614117

Research Article

Multi-UAV Search and Rescue with Enhanced A* Algorithm Path Planning in 3D Environment

Yuwen Du

Department of Aerospace Engineering, University of Bristol, Bristol BS8 1TR, UK

Correspondence should be addressed to Yuwen Du; hu19368@bristol.ac.uk

Received 24 September 2022; Revised 5 January 2023; Accepted 11 January 2023; Published 6 February 2023

Academic Editor: Yue Wang

The unmanned aerial vehicles (UAV) are now widely used in search and rescue (SAR) missions to locate casualties and survey terrain. To solve the problem of long calculation time and large memory usage of the UAV obstacle-avoidance path-planning algorithm in cooperative tasks, this paper proposes a method that combines the A* algorithm and the task allocation algorithm to achieve a faster and more effective path-planning method. First, the environment is displayed in the form of a grid. Then, the enhanced algorithm divides the task area for UAVs. Finally, each UAV performs SAR path planning in the mission area. The tasks of mapping the environment and searching for target points by UAV swarms are discussed in this study. Our research enhances A* algorithms for generating the shortest collision-free paths for drone swarms. Further, we evaluate the algorithm via simulating the task assignment algorithm and path-planning algorithm of a 3D map and 2D map. Compared with the traditional A* algorithm, the results demonstrate that the enhanced algorithm is effective in the scenario.

1. Introduction

In recent years, autonomous drone swarms have been an emerging technique that can be used in the field of search and rescue. Because of their high flexibility, wide adaptability, and controllable economic efficiency, autonomous drone swarms are widely applied worldwide. These days, a China Eastern Airlines jetliner carting 132 people crashed in the mountains in Guangxi province of China, according to the country's Civil Aviation Administration (CAAC). The Boeing 737 lost contact while it was enroute from Kunming to Guangzhou and search and rescue efforts began immediately at the scene of the crash. In this kind of situation, it is necessary to search and rescue as soon as possible, and also, the shortest path must be generated right away. Numerous scholars have conducted extensive research in the multi-UAV path-planning problem for search and rescue missions, to generate the shortest path and avoid multi-UAV collision. However, these studies primarily focus on the general 2D and 3D path planning of UAVs, with more attention on the obstacle-avoidance ability of UAVs when performing tasks. There is a lack of efficient and fast task assignment algorithms. The aim is to minimize the risk of collision with terrain obstacles and to identify the shortest path for all drones in the least amount of time. With the additional task allocation algorithm, UAVs are assigned with different mission areas. UAVs can parallel search the environment and perform path planning according to the allocated mission area.

This article starts with the introduction of the 2D UAV task assignment algorithm, then 2D and 3D multi-UAV path-planning algorithm basics and improvements, and finally combines UAV task assignment and an enhanced UAV path-planning algorithm in a 3D environment. Path planning is one of the most important technologies in achieving the autonomous control of UAVs [1, 2]. This report is organised as follows: in the second part, Section 2 introduces the related work on the path-planning algorithm. Sections 3.1 and 3.2 introduce the communication between multiple UAVs and their characteristics and analyse and explore the UAV swarm trajectory planning algorithm and the UAV swarm cooperative search and task assignment. Section 3.3 gives a brief overview of the different types of algorithms that have been used for path planning and then focuses on the methods used by the A* algorithm for path

planning. Section 3.4 introduces improvements to the A* algorithm to reduce the time cost and memory consumption of the path-planning algorithm and improve the efficiency of path planning. Section 3.5 details the steps to convert a 3D topographic map into grid from using MATLAB to facilitate the application of the algorithm in path planning. The fourth part simulates the task assignment algorithm and path-planning algorithm of the 3D map and 2D map and analyses and compares the simulation results. It is discussed that by implementing improvements that combine task assignment algorithms and path planning, this enhanced path-planning algorithm has obvious advantages. First, it effectively reduces the occupation of memory space and computing time. Second, the collision of UAVs during collaboration is effectively avoided. Finally, in Section 5, conclusions will be drawn and deficiencies and future directions for improvement will be proposed.

2. Related Work

As autonomous drone swarms are widely used in the field of mapping, inspection, transportation, and monitoring, a large and growing body of literature has investigated incorporating communication into the path planning of multiple drones for search and rescue missions to achieve dynamic task allocation through information dissemination. Beard and McLain [3] described the use of a group of unmanned aerial vehicles (UAV) for collaborative search, in an area of interest that includes areas of opportunity and areas of potential danger. Because the goal of the drone teams is to access as many opportunities as possible while avoiding as many dangers as possible, to achieve cooperation, the drones need to be restricted to each other's communication range and need to avoid collisions. The research by Beard and McLain [3] on teamwork search problem UAVs with communication range limitations has been considered. The authors proposed an algorithm which finds the optimal path of the team with feasibility consideration and develops a path for nearby drones, and two methods that are suboptimal but highly efficient calculation methods are the best leader and the best path coordination search algorithms.

Hayat et al. [4] has proposed two adaptive strategies, as shown in Figure 1, which are based on the centralised synchronous notification and connection (SIC) path-planning strategy, and the strategies' goal is to avoid affecting the regional coverage goals and overall mission time which can define the task as search, notify, and monitor with the best possible link quality. Figure 1 shows the two strategies as follows: (a) SIC following QoS (SIC+): first, optimize the search and notification tasks and then find the best location for monitoring; (b) SIC with QoS (SICQ): simultaneous optimization of search, notification, and monitoring tasks. Based on Hayat's research, it is shown that combining communication in the path design can result in effectively performing more tasks within a given task time and improving the quality of the resulting path in terms of connectivity. However, the research in the literature [4] pays more attention to using communication to assign the targets to each drone and plan the shortest path based on the distance between the target and the drone in 2D strategies.

Through the above task allocation method, the shortest optimal trajectory can be generated, multiple drones can collaboratively search for targets in a given area, and some prior data about the distribution of targets in these areas can be obtained. More research on autonomous unmanned aerial vehicle (UAV) decision-making and control has been carried out in autonomous drone swarms' cooperative control and collision avoidance. In the field of unmanned aerial vehicle (UAV) cooperation, Jin et al. [5] developed an extensive dynamic model that captures the randomness of cooperative search and task assignment problems and designs algorithms to achieve high levels of performance. Jin et al. [5] focused on the value of predictive task allocation as a function of the number of unknown targets and the number of drones and proposed a hybrid algorithm on this basis which is used to switch the use of predictions and balance the search and task response.

Because the autonomous drone swarm is a technology with important applications in the fields of surveying and mapping, inspection, transportation, and surveillance, each drone needs to complete a subobjective within the overall target range, while avoiding collision with obstacles and other drones in the environment being particularly important. To achieve this goal, Ganesan et al. [6] has combined task allocation with path planning to perform collision avoidance during navigation and compared the grid-based global planning algorithm with the gradient-based local planning algorithm. Ganesan et al. [6] also evaluated potential field-planning algorithms with different cost functions and proposed a method to adaptively modify the UAV's speed when using the Huber loss function to perform collision avoidance and observe its impact on the UAV's trajectory. By combining task allocation and path-planning algorithms, the avoidance environment and local obstacle plans are executed without global knowledge. For example, a situation with a very large number of environmental obstacles will lead to a poor approximation, while merging tasks can alleviate. A method of adaptive change was also proposed by Ganesan et al. [6] to ensure safe speed during obstacle avoidance.

One of the basic elements of autonomous and cooperative missions performed by unmanned aerial vehicles (UAV) is trajectory planning, which is necessary to ensure the safe and collision-free movement of different vehicles. Finding the lowest cost path through graphs is at the core of many problems. Changes in the arc cost during the execution of the path may result in the need to replan the rest of the path. During the replanning period, the drone needs to wait to calculate a new path or move in the wrong direction. Therefore, rapid replanning is essential. In order to accomplish this task, Yang et al. [7] discuss the rationale for the most successful 3D path-planning algorithms developed in recent years. According to this article, the comprehensive applicability of the A* algorithm is introduced according to its exploration mechanism and its advantages and disadvantages. The A* algorithm generates a shorter path comparing with other methods when computing in 3D path

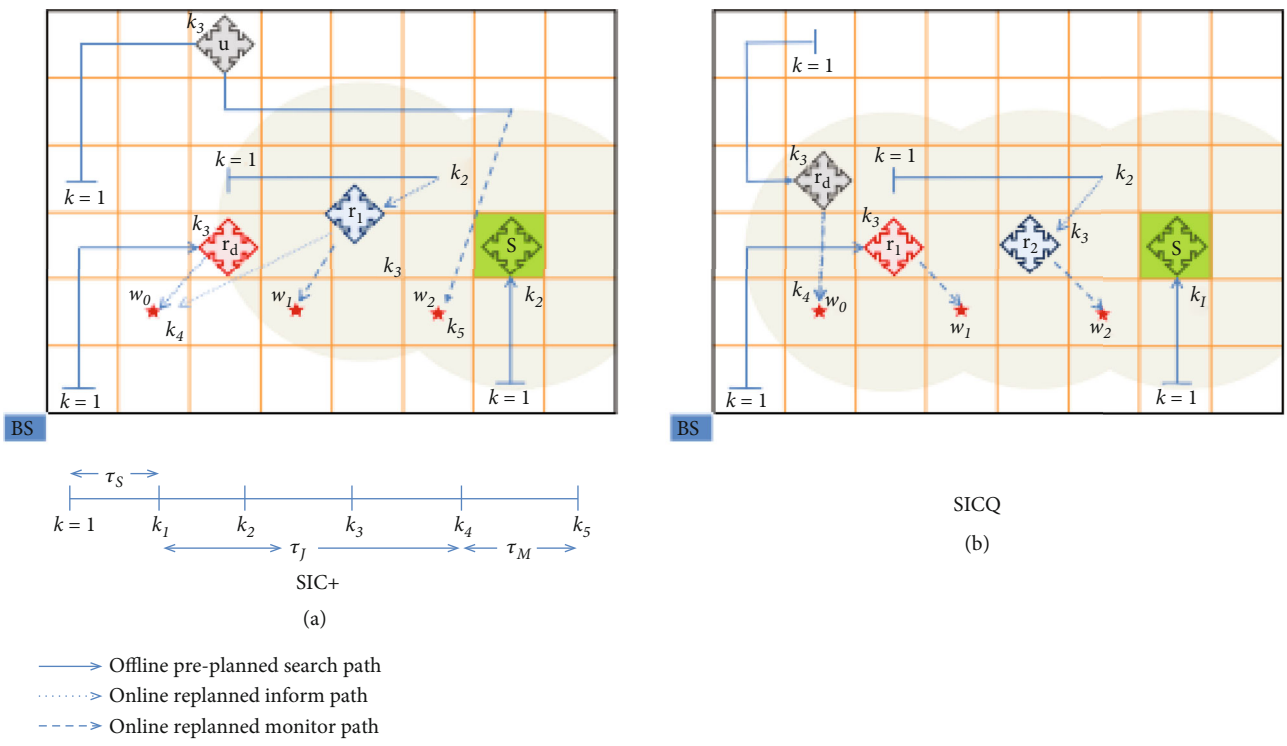

Figure 1: The structure of the two adaptive strategies [4].

planning; however, the A* algorithm has the characteristic which is a long time calculation when storing data into an open table and a closed table. Peng et al. [8] proposed an enhanced method which is a new way to store the array in the open table and the closed table. This research changes the way of array storage by accessing the number ranks of a specified element to locate the array elements and according to the experimental results. The operation efficiency of the enhanced A* algorithm was enhanced, and the advantages of the original A* algorithm was retained.

When developing and generating the shortest path with those algorithms in 3D conditions, what we must do is to optimize the path into the smoothest path based on the grid method. Samaniego et al. [9] have implemented an architecture for generating a 3D flight path for a fixed-wing unmanned aerial vehicle (UAV). The 3D flight path needs to consider the rotation and elevation constraints of the drone to generate a feasible flight path by minimizing the rotational force. Figure 2 shows the perspective view of the 3D flight problem of a fixed-wing UAV. CG represents the position vector of the UAV's centre of gravity. The global coordinate system is located at the origin, and the directions of the local body coordinate system are represented by Euler angles, pitch angles, and yaw angles. They are defined by three orthogonal vectors, aligned with the three axes of the vehicle, and centred on the CG. Finally, the angular velocities X, Y, and Z along the local axis of the UAV are denoted by p, q, and r, respectively. Samaniego et al. [9] proposed a multiobjective optimization problem (MOP), which is aimed at independently maximizing each turning radius of the path. These studies introduced application communication and path planning for UAV rotation control to generate the shortest path to accomplish more tasks in a limited time.

Additionally, path planning is crucial for ground hexapod robot search and rescue tasks. The flexibility in path planning must be taken into account on unstructured terrain. According to Chen et al. [10], the structure of the regular flexible gait planner and gait feedback is the foundation of gait transition hierarchical control. The simulation and experimental results demonstrate that through the use of this control framework, the robot alters its foot trajectory in dynamic, unstructured terrain and achieves an elastic gait for obstacle avoidance. And Chen et al. [11] also demonstrated how the realisation of the robot's path planning and wheel-leg obstacle avoidance through terrain inspection is made possible by the application of visual perception system recognition.

Overall, these studies highlight the need for improving drone swarms' path-planning algorithm for UAV search and rescue in the 3D environment and focusing on collision avoidance. Considering all this evidence, it seems that during the development of the UAV swarm path-planning technology, many studies monitored UAVs to generate the optimal route and proposed a variety of algorithms to carry out UAV cooperation and find the shortest path for the swarms while avoiding collision between swarms. However, the focus of such research still needs improvement to obtain two-dimensional shortest paths through grid-based path-planning algorithms. One of the tough challenges for all researchers in this domain is how to get the smoothest and shortest path when obstacles appear in the UAV's path. A more systematic and theoretical analysis is required for

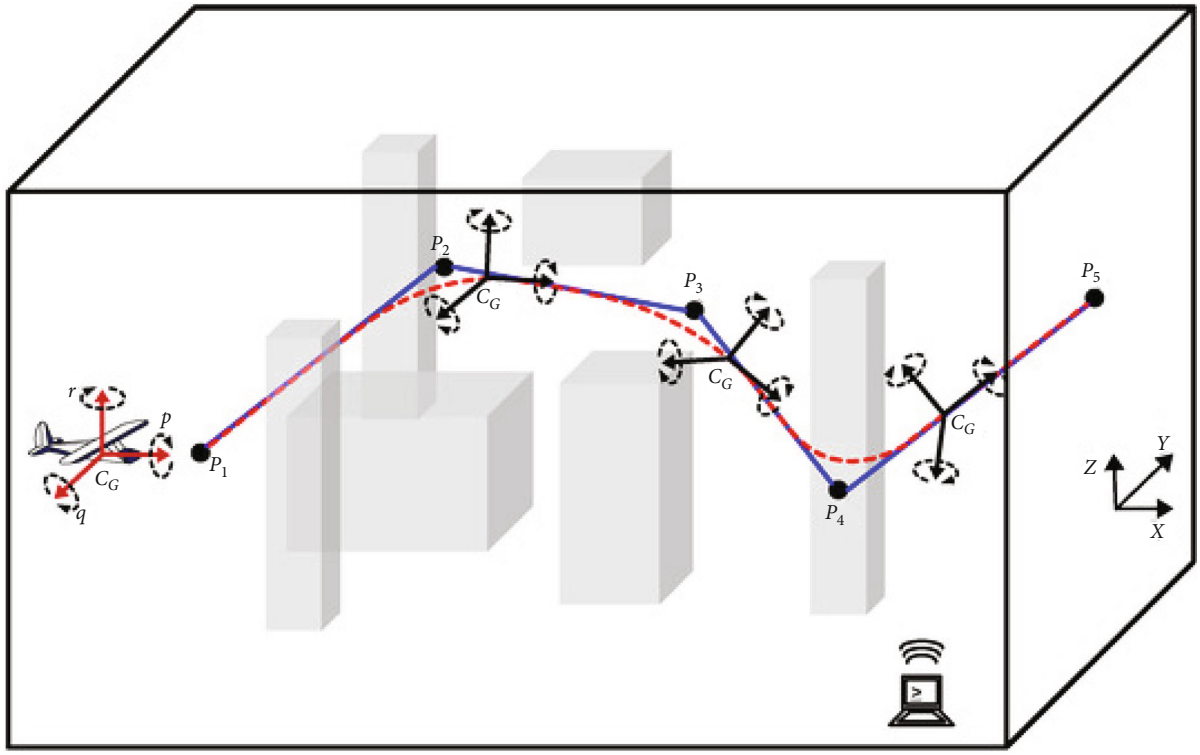

Figure 2: The collision-free point set Pi is represented by black dots; the blue line describes the discrete path constructed according to the 3D path plan; the red dashed line is the new smooth and optimized path that the UAV can follow [9].

applying an improving algorithm into UAV swarm path planning in order to generate the smoothest and shortest path while avoiding the obstacles in 3 dimensions.

Path planning is a broad research direction. How to combine the generation of optimal routes with task assignment in a 3D environment and apply it to UAV swarms has not been extensively studied. To fill this gap, this review critically evaluates the implementation of more efficient methods to generate optimal paths covering an area, minimizing distance while maintaining distance to obstacles detected by sensors. This project is aimed at improving the A* algorithm. By combining advantages of grid mapping, the shortest and smoothest path is generated for the drone swarm in the 3D environment, while avoiding the collision of the drone swarm with environmental obstacles. With this goal in mind, this paper presents the impact and application of a range of grids to algorithms to determine how swarms of drones can avoid collisions and reach targets quickly based on the grid.

3. Methodology

In this chapter, firstly, UAV swarm task assignment algorithms are introduced. Then, communication with cooperative searching is introduced. Dynamic task assignment during the SAR can be achieved with the consideration of the task assignment and communication algorithm. Path-planning algorithms are discussed in Section 3.3. The A* algorithm is chosen through comparison between multiple algorithms due to its high performance and accuracy. Then, the optimization of the A* algorithm is proposed with an additional function to solve the problem of large memory space requirements. Finally, the environment can be mapped in the form of grids. And the combination of the task assignment algorithm and the enhanced A* algorithm is applied on the grid map for path planning.

3.1. UAV Swarm Task Assignment. During the search and rescue mission of the UAV swarms, the UAVs must search for and confirm the location of the target and must cooperate as a UAV team at each target site. Jin et al. [5] discussed an extensive dynamic model that was developed to capture the randomness inherent in cooperative search and task assignment problems, designed an algorithm capable of achieving a high level of performance, and then developed a new algorithm capable of balancing the search and task response based on the analysis of the experimental results of the proposed algorithm. Chen et al. [12] developed a job assignment mechanism based on the UAV's time restriction for completing the mission. Each drone determines its own shortest route and least arrival time using coordination variables and broadcasts it to the team using an algorithm. Each drone in the team then solves the same optimization problem to calculate the competition time, enabling all drones to reach the target location and accomplish the mission in the least amount of time feasible.

To successfully accomplish a mission, each drone will be given a subgoal within the operation's general scope and navigate the environment by avoiding collisions with objects and other drones. We are tasked with the job of covering the

whole area with a swarm of UAVs, where the global position of the target is known, and the UAV is tasked with performing a search and rescue operation to the target by identifying the barriers. To achieve rapid coverage, the drone must decide which settings to visit and in what sequence. In simulations, we mimic this job by combining task assignment and route planning to ensure collision avoidance during navigation, as well as by using a grid-based A* algorithm. We need to employ the drone swarm to search and plan the route for the unknown region including the goal location in the drone search and rescue operation. The drone swarm must decide which targets to visit and in what sequence to visit them depending on the target point's importance. Additionally, the drone must navigate its surroundings in order to reach the goal location while avoiding collisions with objects and other drones.

Each drone will reduce the amount of time required to cover the surroundings by prioritizing access locations and doing route planning. A route-planning algorithm is then used to prioritize and avoid colliding with objects and other drones. Both job assignment and route-planning calculations may be decentralised, significantly reducing the need for complete global awareness of the surroundings in a real-time system [13]. When a UAV swarm undertakes a search and rescue operation, the mission may be separated into task assignment for the target locations that need to be visited and route planning for obstacle avoidance. The job assignment challenge necessitates the utilisation of a swarm of UAVs capable of mapping the surroundings in the quickest possible period. The work is not complete until all of the target points in the environment have been mapped. Due to the fact that certain target locations have a greater priority than others, the UAV must first visit the higher-priority targets before proceeding to the lower-priority targets. Each drone must forecast which target point to visit and in what sequence to visit them, as well as conduct the bare minimum step of mapping all target sites. The route-planning algorithm utilises the projected sequence to visit the target location and generates a trajectory that avoids colliding with obstacles and arrives at the destination.

To establish the sequence of visits to the target sites in a task assignment issue, the drones must utilise aggregate knowledge about the position of the target points and autonomously pick activities that allow them to map the environment as rapidly as feasible. The drone can forecast each target point using the policy network trained using the policy gradient technique described in Yang et al. [14], the exact steps being as follows:

Step 1. To expedite the route planning and job assignment, we choose to model the task assignment in a 2D environment and to depict the environment using a grid. The UAV swarm is located between the beginning and target points. Generated at random positions inside the grid, a certain number of points are randomly chosen and given a greater priority than the other points. The drone can go in eight directions: east, south, west, north, northeast, southeast, northwest, and southwest.

Step 2. We need to evaluate the performance of the reinforcement learning algorithm by recreating the required tasks; therefore, we put up a reward system to track the predictive capacity of the drones. Each drone is awarded with alpha if it successfully maps a low-priority target and with 2alpha if it successfully maps a high-priority target, where alpha is a positive real number. And if the drone delays mapping high-priority target sites, there are two situations which are as follows: (1) the drone is not penalised if all high-priority target points are mapped. (2) When there are still unmapped high-priority target locations, the drone will be punished with the size as the metric between its current location and the closest unmapped high-priority point. Then, we can calculate the final reward signal received by the drone after all target points have been mapped.

While looking for the target spot, the drone must avoid obstacles in its route. We must design the path for the course of the estimated target location by the drone to avoid colliding with obstacles. We use a route-planning technique stated in Galceran and Carreras [15] that avoids barriers without requiring complete knowledge of the environment—the wavefront planner, which is a grid-based global planner. The algorithm assumes that the drone can see the complete surroundings and that any barriers in the environment are static. As a result, each change in the obstacle location must be sent to the drone, and the planning algorithm must be rerun to adapt the trajectory. When the environment is divided into discrete grids, target cells are initialised to 0, obstacle cells to -1, drone cells to 1, and all other cells to 2. Begin with the drone location cell and choose all accessible cells based on connection and initialise their values. Continue selecting and initialising all reachable cells from the chosen cells repeatedly until the drone reaches the destination location cell. The algorithm does a breadth-first search in the grid, selecting the cell with the lowest value at each step to determine the shortest route from the drone location to the goal state.

3.2. Communication with Cooperative Searching. Communication can be introduced into the multi-UAV route-planning issue for search and rescue missions to enable dynamic job assignment via information distribution. While achieving this aim, we must avoid interfering with area coverage goals and total mission time; therefore, we must fulfil search, notification, and monitoring while maintaining ideal link quality. By combining the two communication mechanisms proposed in literature [4] in the path design and making necessary policy adjustments according to the UAV mission, it is possible to execute more tasks in the given mission time and improve the level of the generated path's connectivity quality. At various phases of a SAR operation, basic route-planning approaches can be applied. The whole environment is separated into sections inside the grid using the element decomposition approach, and the obstacles' grids are indicated. Algorithms can be used to plot a route for the drone to avoid obstructions during search and rescue operations. Path planning discretized the space before using a graph-based search to apply the A* algorithm with increased complexity, optimality, and applicability.

The UAV can identify the target by using the search algorithm to cover the complete area for route planning. However, the search job must also notify the ground base station of the target location of the search. We use information-merging communication to create search paths and enhance the chance of item discovery via collaboration. Using a method of connecting and talking with ground base stations provides for efficient exploration and flexible route planning in terms of coverage and connection characteristics. By employing all drones for coverage, the resource usage will become more efficient. When it is necessary to disseminate information, communication tasks are allocated to UAVs in order to shorten the total job completion time.

The route planner must be aware of the search area, communication technology, and quality of service requirements before beginning a search and rescue operation with multiple UAVs and several stationary targets in a bounded region. This data may be utilised to design pathways that are free of conflicts while also lowering coverage time. Each drone must be able to ensure that it will finish its job and return to the base station before the battery dies, implying that the mission has a time constraint. The full search and rescue effort may be completed in two parts. The first step is the coverage stage, during which the UAV must follow a preplanned course in order to complete the target search mission. When the target is discovered, the connection advances to the second step. The UAV must plan the mission in the current dispersed fashion based on the importance of the target and replan its course to alert the base station and construct a path between the target location and the base station till the search and rescue team arrives. The drone continues to follow and cover the route until it reaches the detection target point. The drone then relinquishes its coverage route and may move on to the next target site, revising its plan and spreading fresh target information across the network. As the drones interact with one another and gain new knowledge, they reprogram themselves. To maintain flight safety, UAVs are urged to broadcast beacons and their current mission status on a regular basis throughout the operation. The status contains the drone ID, the current time, the job completed, and the current coordinates.

In order to account for the overall time necessary to accomplish the activity, we must first account for the time required to perform the task in each of the phases. The time required for the search task to locate the target must be taken into consideration in the first stage, and the time required for the notification task and the monitoring task must be taken into consideration in the second stage, that is, the time required to notify the base station of the target location after detection and the time required to reorganise the UAV. The route design must take into consideration the importance of search tasks in relation to notification activities and monitoring duties.

3.3. Types of Algorithms. Path planning is a widely used problem. In control theory, path planning needs to consider stability, smoothness, and optimality. In this article, we will discuss the influence of algorithms on path planning and how to use algorithms to deal with different geometric models. Since a cell in a grid can represent a very small space and we need to ensure the shortest distance between the path and the obstacle, we choose to use the grid to represent the space state and pay attention to the feasibility and optimality of the route. Different 3D path-planning algorithms have different characteristics and can be applied to different robots and environments. According to Yang et al. [7], they divided 3D path-planning algorithms into five categories with their own unique properties which are sampling-based algorithms, node-based optimal algorithms, mathematical model-based algorithms, bioinspired algorithms, and multifusion-based algorithms. Focusing on node-based optimal algorithms, these are the algorithms which are used to deal with the weight information of nodes and arcs which can also be called a grid [5]. These algorithms can be used to find the optimal path as they generate the path by calculating the node exploration cost which is like generated network searching.

The node-based optimal algorithm can be divided into three elements which are Dijkstra's algorithm, A* algorithm, and D* algorithm. The Dijkstra algorithm is an algorithm used to find the shortest path between nodes in a graph. In some cases, the cost of moving between adjacent nodes in the graph is not equal. In the Dijkstra algorithm, it is necessary to calculate the total movement cost of each node from the starting point. All the nodes to be traversed should be placed in a priority queue and sorted according to the cost. In the running process of the algorithm, the node with the lowest cost is selected from the priority queue as the nest traversal node until it reaches the finished line. However, when the image is a grid and the cost of moving between nodes is equal, Dijkstra's algorithm becomes the same as the breadth-first algorithm. The A* algorithm can be considered an extension of Dijkstra's algorithm. The A* algorithm is a very common path-search and graph-traversal algorithm; it has better performance and accuracy. A* algorithms usually have better performance because they are guided by a heuristic function, which is a search in the state space, evaluates the location of each search to get the best location, and then searches from that location to the target. The heuristic function saves a lot of unnecessary search paths and raises efficiency. The A* algorithm calculates the priority of each node by using the following function:

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

where $f(n)$ is the comprehensive priority of node n. When we choose the next node to traverse, we always select the node with the highest overall priority which has the lowest value. $g(n)$ is the cost of the distance between node$n$ and the starting point. $h(n)$ is the estimated cost of the node from the end point, which is the heuristic function of the A* algorithm.

The A* algorithm selects the node with the highest priority with the lowest $f(n)$ value from the priority queue as the next node to be traversed. In programming, the A* algorithm uses two sets to represent the nodes to be traversed and the nodes that have been traversed, which are usually

called OPEN_set and CLOSED_set. D* is short for the dynamic A* algorithm. The algorithm is like that of the A* algorithm except that the calculation of the cost may change during the running of the algorithm.

In some cases, it is not necessary to find the shortest path but rather to find one as quickly as possible. By adjusting the heuristic function, we can control the speed and accuracy of the algorithm; thus, the A* algorithm ensures the optimality and fast convergence. And this is the reason why the A* algorithm shows flexibility during calculation. As the estimation cost of the node of each state infinitely approaches the real cost, the A* algorithm has a faster convergence speed then the Dijkstra algorithm.

Nash et al. [16] proposed the Theta* algorithm which is a variety of the A* algorithm. The Theta* algorithm can find an optimal path which is shorter and more realistic as it can generate the path with a flat and smooth turning angle by the choice of parent states. The Theta* algorithm can choose any vertex to be the parent states; however, the A* algorithm can only choose the successor as the parent states. In the experiment made in [13], they found that the Theta* algorithm provides the best way to balance the length of the path and the calculating time as it finds the shortest path in the shortest time than the others. In addition, the angle-propagation Theta* algorithm [16] can even find the shortest path while not constraining the ranges of the path angle which guarantees that the final length of the generated path is not affected. Based on the Theta* algorithm, the A* algorithm can be enhanced and extended to use more efficient graph searching in 3D environments [17]. Yang et al. [14] introduced the application of basic Theta* in the 3D environment path planning. By applying the Theta* algorithm to terrain obstacles and urban environments to evaluate solutions for different types of obstacles and implementing an experiment to compare the performance of the Theta* algorithm and the A* algorithm in a 3D environment, De Phillips et al. [17] demonstrated that the Theta* algorithm reduces searches in contrast to the A* algorithm. However, the Theta* algorithm takes such a long time in checking if there are unexpected nodes around when applied to a 3D environment.

The A* algorithm is theoretically most optimal with regard to time consideration, but its spatial growth is exponential. The basics of the A* algorithm is that after extending a node, its valuation function is calculated and sorted according to the evaluation function to the extended node, thus ensuring that each extended node is the node with the smallest valuation function. Thus, we can create the A* algorithm with these steps.

Set up a queue and calculate the valuation function f of the initial node, and queue the initial node, setting the head and tail pointer of the queue.

Remove the node of the queue header, which is referred to by the queue head pointer; if the node is the target node, output the path and the program ends; otherwise, the node is extended.

Check if the extended new node duplicates the node in the queue, and discard it if it repeats with a node that is before the queue head pointer and can no longer scale, and if the new node repeats with the node to be extended, which is after the queue head pointer, then compare the size of $g(n)$ in the valuation function of the two nodes to preserve the node with a smaller g value and then jump to step five.

If the extended new node does not duplicate the node in the queue, insert it into the appropriate location in the node queue that is to be extended and after the head pointer with its size of valuation function $f(n)$.

If the node at the head of the queue can also be extended, go straight back to the second step. Otherwise, point the queue header pointer to the next node and return to the second step.

3.4. Optimization of A Algorithm.* Niu and Zhuo [18] developed the "cell" and "region" concepts to improve awareness of the A* algorithm on the surroundings, allowing for flexible modelling of 3D environments. "Cell" is the basic unit of the enhanced A* algorithm, and "region" is the spatial object that includes a certain sort of cell. Because of the characteristics of the cell and region, in the 3D environment, the shape of the cell will influence the number of neighbours of the cell and hence influence how the A* algorithm searches with these neighbour cells. Following the experiment conducted by Niu and Zhuo [18], in the test region, the cell was chosen as a cube shape as the generated path can be represented by each movement between the cube-shaped cells; the following benefits of the enhanced A* algorithm were demonstrated.

Because the open list of the A* algorithm takes a large amount of memory space, although efforts have been made in the past to lower the memory space cost of the A* algorithm, the result have been insignificant. The region using notion in the modified method overcomes the A* algorithm storage consumption issue. Because of the addition of "region," the whole environment can be divided into several regions, and each region can be satisfied with a small memory storage space while processing.

The modified A* algorithm introduced parallel computing into algorithm calculations. As the whole environment has been divided into several regions as shown in Figure 3, there are separate areas for path planning, and the program will maintain calculating the ideal path for each multithreading segment.

The "cell" form allows the upgraded A* algorithm to operate more freely. As the region is built up with several cells, any modification to the related region will be reflected in the cell which means that in the test area, instead of recreating the whole issue area, we can simply make small changes to the cells which are affected by the issue in that region.

Since the A* algorithm requires a lengthy time while searching and traversing through OPEN_set and CLOSED_set, Nash et al. [16] proposed a new manner of array storage in the OPEN_set and CLOSED_set which obviously increases the efficiency of calculating the A* algorithm. As the basic A* algorithm stores the data into the OPEN_set and CLOSED_set with the binary tree form, it is simple to add or remove the information; however, calculating to find the node needed is complicated and a long time range is needed as traversing all the nodes is required when the OPEN_set and the

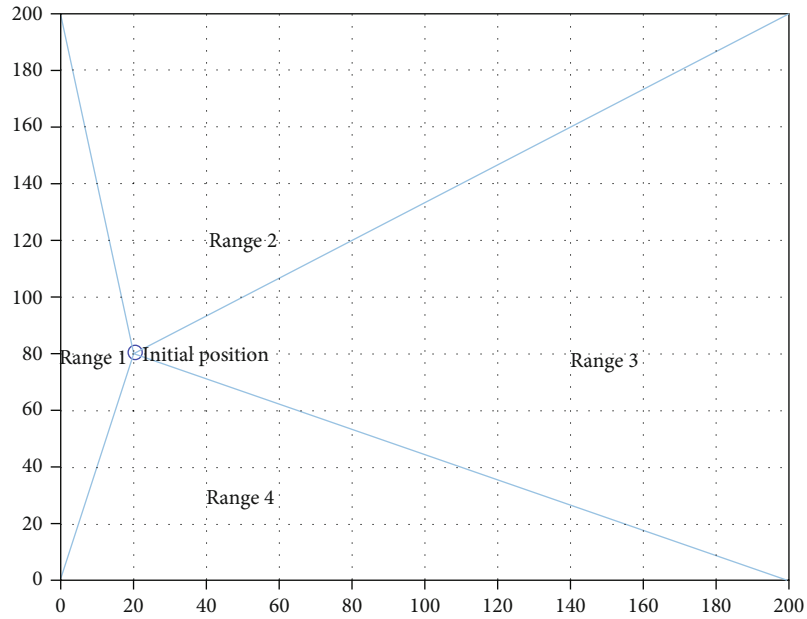

Figure 3: Example regions divided by UAV swarms.

CLOSED_set are searched. In order to measure the position of the node needed for the first time, instead of traversing each node, a new kind of data structure called the query_table (i, j) [8] was proposed to improve the OPEN_set and CLOSED_set visiting method and search for the information directly. The nodes can be located, and the state of each node may be determined by gaining access to the structured data. With this data structure, the node can be analysed with three different states which are the free state, OPEN_set state, and CLOSED_set state. After the new data structure is added, the OPEN_set remains and the CLOSED_set can be replaced by the query_table (i, j). Inside the query function, there are five members to be recorded for the A* algorithm: the $f(n)$ value of the (i, j) node, the $g(n)$ value of the (i, j) node, the $h(n)$ value of the (i, j) node, coordinates of the parent node of the (i, j) node, and the states of the (i, j) node.

In this way, when a node has been extended as the parent node, the new child node will be absolutely extended by the parent node. For the new child node, the query_table (i, j) will record the state as the OPEN_set state instead of the free state which means the next parent node; then, the query_table (i, j) will remove the old parent node from the OPEN_set and rerecord the state of the old parent node from the OPEN_set state to the CLOSED_set state. With this recording information, we can simply update the instructions for the nodes which can be found by the structure data query_table (i, j). When we take random nodes and test them, we can simply get the state information of these nodes and in addition the parent node information of these nodes.

The basic A* algorithm uses the OPEN_set and the CLOSED_set to remember all the nodes and their extended parent or child nodes. To apply this enhanced data structure into the A* algorithm, the logic of the A* algorithm will be changed based on the original A* algorithm as per the following steps.

Step 1. For the chosen parent node, select the extended child node for the parent node and apply query_table (i, j) to check the node state. Set 0 for the free state, 1 for the OPEN_set state, and 2 for the CLOSED_set state. If the state of the node is equal to 0, then set the state of the node as 1, and through the added data structure, the query function will calculate the $f(n)$, $g(n)$, and $h(n)$ of this node, record the coordinates of the parent node, and change the state of the parent node to 2.

Step 2. After the query function has changed the state of the old parent node into the CLOSED_set state and added all the child nodes into the OPEN_set, the A* algorithm will compare these nodes and select the one with the minimum $f(n)$ value which will be set as the new extended parent node. Then, based on the selected child node, the A* algorithm can then extend new child nodes which can be used to check if the selected child node is the suitable new parent node for path generation. By applying query_table (i, j), we can check the state of the new child node. If the state of the new child node is 0, then the query function will change it to 1 and the selected child node can be set as the new extended parent node; then, the query function can find the next extended parent node with the new extended parent node. However, if the state of the new child node is 1, then query function needs to recalculate the $f(n)$ value of the new child node and compare it with the selected old child node. If the $f(n)$ value of the new child node is smaller than that of the old one, the query function will remove the selected old child node and exchanged the recorded $f(n)$ value and the coordinate of the parent node for the new child node. Moreover, if the state of the new child node is 2, then it cannot be changed to the OPEN_set state. Then, we can find the new extended parent node and set the state to 2.

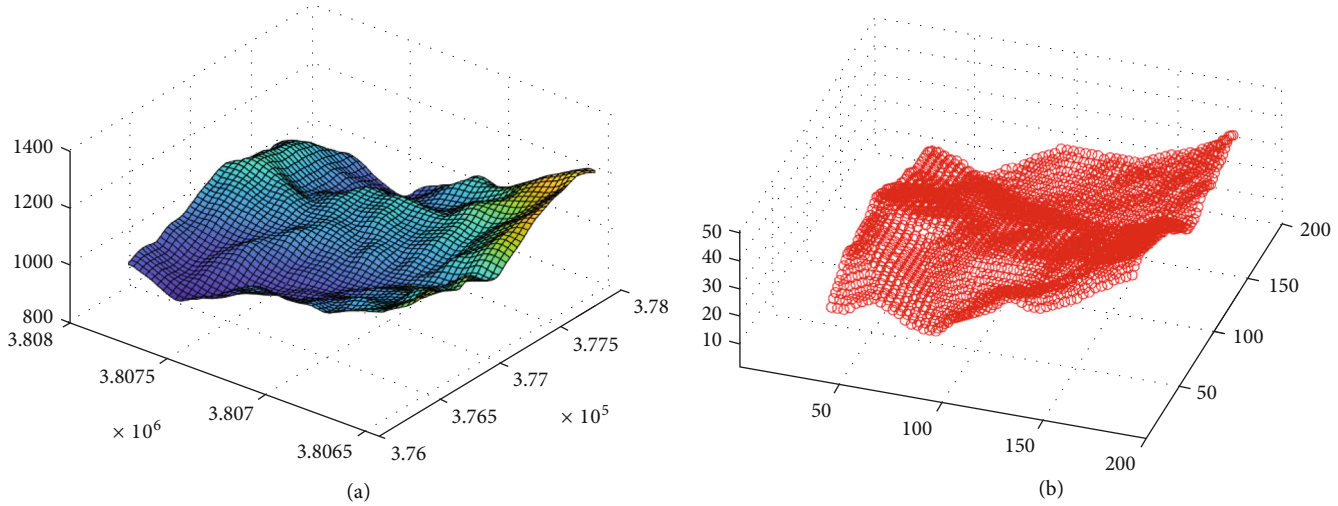

Figure 4: (a) The surface of the topographic map; (b) obstacles represented in the grid map.

Step 3. Determine if the target node is inside the new child nodes. If the target node is inside the new child nodes, then the path can be generated and the loop can be exited; otherwise, go back to step 1 and continue to find the next extended parent node until the A* algorithm finds the target node. As each child node has been recorded with its own parent node, then from the target node, go back with each parent node to the initial node; this is the optimal path generated by the A* algorithm.

Peng et al. [8] has made a comparative experiment for the basic A* algorithm and this optimal A* algorithm. The A* algorithm and the optimal A* algorithm were tested for 20 times in two distinct obstacle distributions. The test findings demonstrate that when the optimal A* algorithm is compared to the basic A* algorithm, the operational efficiency is increased by more than 40%, according to the results. By using the data structure instead of the OPEN_set and the CLOSED_set, the approach retains the benefits of both the basic A* algorithm and the optimal A* algorithm while simultaneously enhancing the operational efficiency of the A* algorithm.

3.5. Grid Mapping. Unmanned aerial vehicle (UAV) 3D path planning seeks to discover the most optimum and collision-free pathways in congested 3D environments while taking into consideration geometric, physical, and temporal restrictions. While the drone is generating a route, it is combining sensor readings to determine whether the cells are vacant. It is possible that the cells along the ray, i.e., the cells that are not occupied, will be empty, whereas the cells at the end of the ray may be filled with information. To represent a grid map, the map must be represented as a uniform array of cells in space, with each cell of the grid representing the map as an array of values and commonly a single byte that represents the number of cells that are filled by a given value and the state of the cell. In order to meet the requirements of unmanned aerial vehicle (UAV) obstacle-avoidance analyses, we have included a 3D route-planning method for unmanned aerial vehicles (UAV) in complex settings presented by Yan et al. [19] into our algorithm. An octree method is used to partition the surroundings into voxels in this approach. It is vital to guarantee that UAVs have sufficient free space in order to fulfil the safety criteria of the aircraft. When building bounding box arrays in the complete 3D space to assess open voxel connections, Yan et al. [19] demonstrated the deployment of a probabilistic roadmap method (PRM) to randomise bounding box arrays in order to guarantee that unmanned aerial vehicles (UAV) have adequate space margins to pass through. In order to provide a more efficient distribution of roadmap nodes in three-dimensional space, sampling is used.

The availability of open space in the surroundings is critical for the design of UAV flight paths, and this information is very valuable. These details, which are represented by a 3D mesh and can be used for 3D route planning, are supplied in this document. So, we must seek out available space in the surroundings. The use of the octree technique may significantly speed up the process of creating a 3D mesh structure in 3D modelling software. An octree is a hierarchical data structure used for three-dimensional spatial subdivision in computer graphics. Each of its nodes corresponds to the space enclosed inside the cubic volume, and each node is referred to as a voxel in certain circles. The volume will be broken into eight subvolumes, with each subvolume being split in turn until the voxel size reaches the required level of detail. In order to indicate the status of the environment, subvolumes holding 3D data will be employed to express the fact that the environment is occupied or idle. The idle state depicts a period when there is no activity in the surroundings. Detecting impediments and planning pathways in complicated situations necessitates the use of massive calculations. It is important to guarantee that the smallest possible free voxel may be utilised for UAV route planning to assure safety, i.e., to prevent collisions, while preforming these computations. This means that if the drone cannot travel through the subvolume of the voxel, that voxel will never

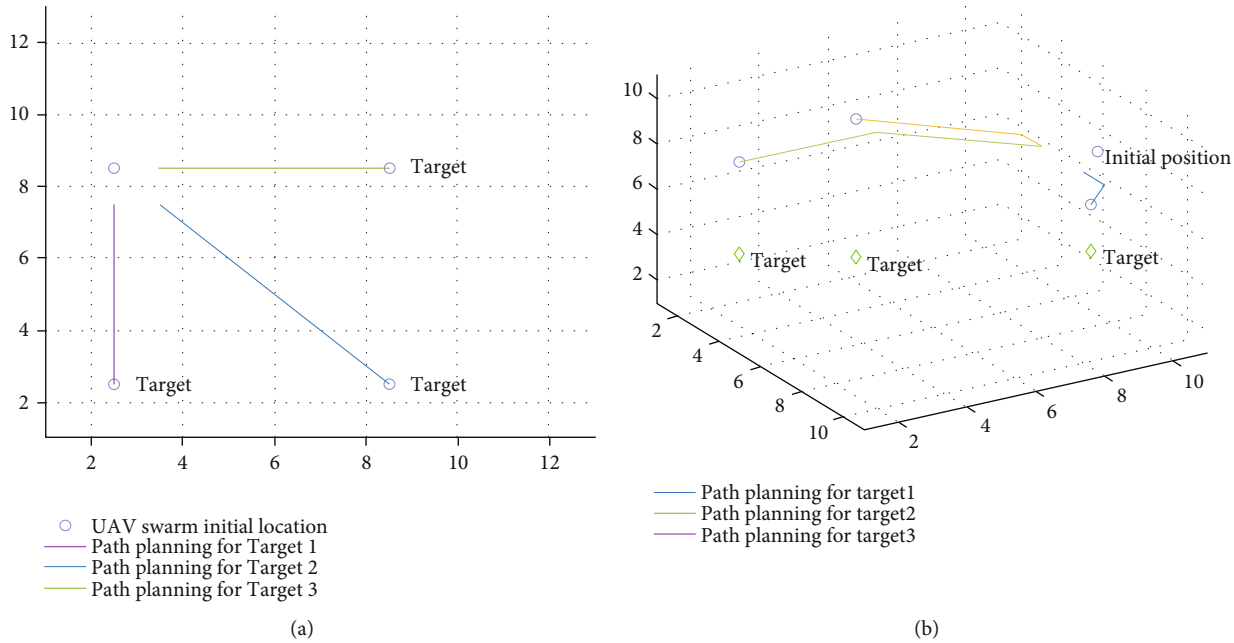

Figure 5: 2D (a) and 3D (b) task assignment examples for drone swarm.

Table 1: Result comparison.

		2D			3D		
		Time (s)	OPEN_set	Path length	Time (s)	OPEN_set	Path length
UAV without TA		4.013977	32 × 8	21 × 2	3.422817	197 × 10	52 × 3
	UAV1	1.344934	24 × 8	7 × 2	1.861286	197 × 10	38 × 3
UAV with TA	UAV2	1.290524	34 × 8	7 × 2	1.067759	78 × 10	8 × 3
	UAV3	1.322498	24 × 8	7 × 2	0.546768	42 × 10	4 × 3

be subdivided again in our calculations. It is necessary to first identify the physical size of the drone to determine the size of the resolution.

When the octree method was applied in the occupancy grid, the condition of the environment can be identified by doing a discrete grid analysis to see whether each cell is occupied or empty. When the drone is equipped with range-based sensors, the range value of each sensor will be directly utilised to update the status of each cell as either full or free by combining the range value of each sensor with the location of the drone. The size of the map in the drone's memory expands in proportion to the size of the environment, and if a lower cell size is utilised, it will rapidly become insufficient to keep up with the environment. To set aside memory for each cell in the matrix so that the occupancy grid may be satisfied, Niu and Zhuo [18] suggested a different approach known as topological decomposition. Topological approaches are more concerned with detecting UAV-related ambient elements than with directly assessing the quality of the geometric environment. Occupancy grids have been implemented using laser range-finders, stereovision sensors, or a mix of sonar and infrared sensors and sensory data from stereovision.

The experiment is carried out by using a 3D mountain environment map and replicating the mountain environment by importing a grid map of the area under consideration. First, we need to create a map that encompasses the whole landscape. The grid map representation will split the space into free-space areas, each of which is represented by a single node. The grid-based approach is used to create the map, and then, the topology-based method is used to accomplish the route finding, localization, and rectification operations. UAVs can construct maps by using scanning ultrasonic sensors and two identical sensors. In certain situations, the backtracking technique described by Kim et al. [20] is used to cover all grid cells and depict them in real time on the simulator window. When employing the backtracking technique, the drone will record the status of the node in the grid after it has crossed the path by using the enhanced A* algorithm. As a result, no sensor information is needed for the movement at this moment based on the recorded grid state. Figure 4 shows the end outcome of converting from a constructed map in the DEM format to a grid map and displaying it as a node.

Then, in order to attain the objective as quickly as possible, we apply the modified A* algorithm to compute the $f(n)$ value and choose the smallest value so that each point can find the quickest and safest path. The beginning point, passing point, and destination point are required for

Table 2: Comparison between time cost and memory space of the optimal A* algorithm with and without the task assignment algorithm.

		Time (s)	OPEN_set	Path length
UAV without TA		72.653799	5751 × 8	4626 × 2
UAV with TA	Range 1	4.762879	192 × 8	127 × 2
	Range 2	26.462905	3286 × 8	2981 × 2
	Range 3	26.172347	1582 × 8	1343 × 2
	Range 4	7.598442	290 × 8	175 × 2

the shortest trip. The findings are visually shown on the construction grid map of the simulator window. We can extract valuable information for UAV route planning from the grid map, such as the distance and direction of the UAV to obstacles. During flight, the drone may meet dynamic obstacles, most notably other drones. In this instance, the UAV must modify its course by sensing and identifying the location of further obstacles. When the drone changes direction, the current location is graphically updated in grid cells. By comparing the updated route to the original path, the UAV will avoid the dynamic obstacle successfully if the original path unit collides with the location of the dynamic obstacle. The A* algorithm can plan a new route based on the updated path, and it can also continue the original path.

4. Results and Discussion

In this chapter, the traditional A* algorithm and the enhanced A* algorithm are compared and discussed in terms of running speed, memory space, and path length. During the simulation experiments, the algorithms are applied in 2D and 3D maps.

4.1. Task Assignment Result Analysis. We simulated an algorithm study of a job assignment to test our hypothesis. In accordance with Hayat et al. [4], the time steps given by the outcomes of distinct policy networks do not correspond to one another. It was possible to notice that the policy network, although taking the most steps, needed the least number of parameters to watch when we simulated the amount of time steps required by the task assignment method to map a 12 × 12 grid. Because of this, we must choose between time step performance and computer parameter computation to get the best overall result. An UAV swarm can complete the full task by using the task assignment algorithm and separate the tasks into each single UAV, and the UAV swarm can cover the entire environment by employing the A* algorithm to each single UAV to plan the course of that environment within the mission range and carry out search and rescue operations.

We will test the influence of drone swarms on the search time for environmental coverage in the simulation trials by assigning tasks to the drones in the swarm throughout the simulation. In this case, we disregard the barriers in the surroundings and simply consider the ability of the UAV to plan a route to the target location while facing it. As shown in Figure 5, we created 2D and 3D simulation scenarios in MATLAB to test the functionality of the UAV swarms. Both the drone and the target point are regarded to be points on the grid in this scenario. A two-dimensional map of the drone's flight environment may be created by projecting the drone's flight environment onto a grid map, and certain data from the two-dimensional map can be compared and examined more clearly using the map that was created. For our first experiment, we created a basic 2D map and compared the time it took a swarm of drones to cover an area with the time it took a single drone to cover the complete area. It can be plainly seen according to Table 1 that, even though enhancing the computation of the task assignment in the fundamental route-planning algorithm needs more processing, the total route-planning time and the pressure of memory space of a single drone can be effectively reduced.

4.2. Optimization of A Algorithm Result Analysis.* Experiments to verify the enhanced 3D A* algorithm are being conducted. In contrast to the classic A* algorithm, which describes the experimental region using a grid, the upgraded 3D A* algorithm describes the experimental area using 3D object "cells." The upgraded A* algorithm, when used in conjunction with the task assignment algorithm, can split the whole environment into numerous regions, with each UAV searching and covering the area that corresponds to the division. The enhanced A* algorithm first determines the joint nodes between the starting point of the UAV swarms and an environmental boundary and then determines how much area is involved in the path planning of each UAV by traversing the joint nodes and determining the best path through the joint nodes.

In order to provide a more accurate view of the performance of the upgraded 3D A* algorithm, we use the route-planning results of the classic A* algorithm as a comparison object to illustrate its improvement. To the contrary of the upgraded A* algorithm, the optimum route solution offered by the classic A* algorithm does not take into consideration the issue of path planning according to distinct areas. Traditional A* algorithms tend to cover the whole environment at once, which is inefficient. Even though this generation approach of examining the whole environment at once is simple, it requires a significant amount of storage space and CPU time. During the simulation experiment, the amount of memory space available for storing search nodes must be equal to or more than the total number of nodes. Accordingly, to search and map the full test region using the classic A* algorithm, a grid map with dimensions of

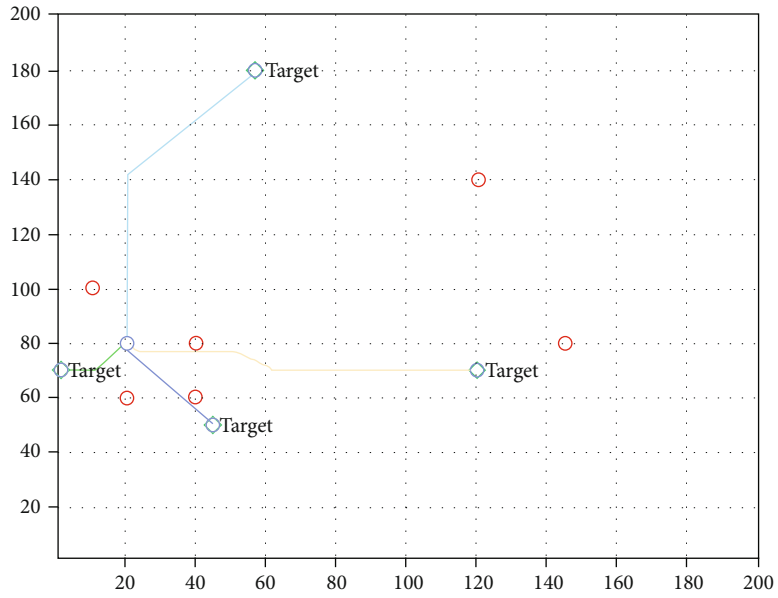

Figure 6: UAV swarms searching the environment for four ranges and four targets.

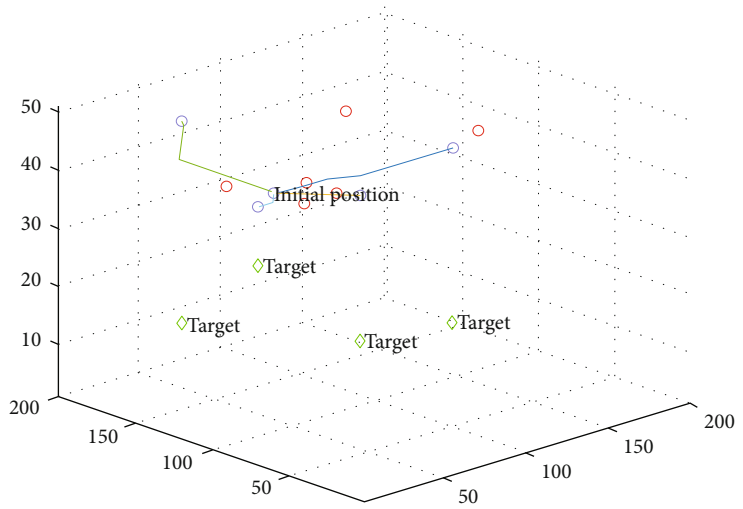

Figure 7: UAV swarm path planning via the optimal A* algorithm in the 3D environment.

Table 3: Comparison between time cost and memory space of the optimal A* algorithm and the traditional A* algorithm.

	Time (s)	OPEN_set	Path length
Traditional A* algorithm	51.53069	13,061 × 8	12,742 × 2
Optimal A* algorithm	29.64890	13,057 × 8	12,742 × 2

Table 4: Three sets of UAV starting points and target points.

	Start position	Target position
Group 1	(10, 10, 45)	(132, 128, 27)
Group 2	(15, 29, 22)	(93, 125, 15)
Group 3	(3, 35, 20)	(36, 137, 10)

200 × 200 units is required. Instead of considering the whole area, the updated A* algorithm simply must evaluate a specific region. Figure 4 describes that in the simulation experiment, the algorithm first determines the area to be explored and evaluated based on the initial position of the drone swarms.

To begin, we created a basic 2D map for simulation trials, in which the regions representing obstacles were marked with dots to indicate their locations. Although barriers of the obstacles in the actual world are often not of regular forms, we deal with these irregular obstacles by regularizing their shapes in order to imitate the smooth progression of the experiment and hence the

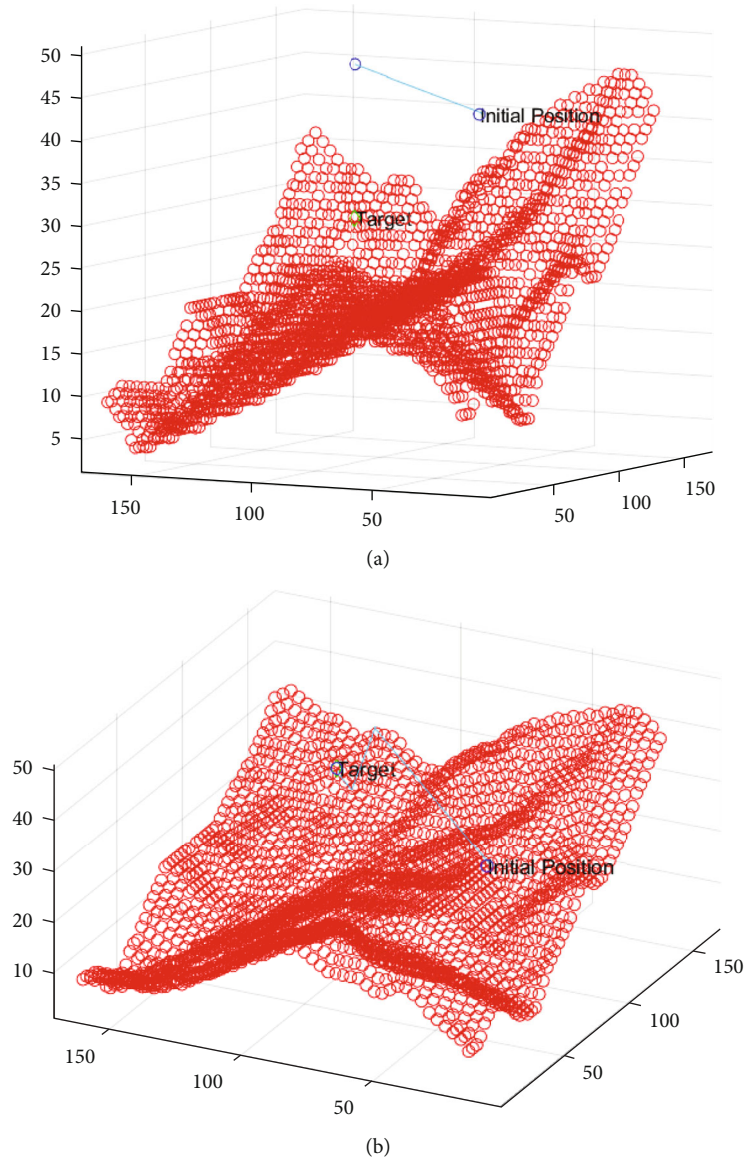

Figure 8: Path planning for simulating group 1 with the optimal A* algorithm (a) and traditional A* algorithm (b).

smooth advancement of the experiment. Areas with no impediments are represented by blank spaces. The map has been scaled to 200×200 pixels, and the target point and the beginning point of the drone swarm have been randomly placed on the map. The two algorithms are performed at the same time, and the outcomes as well as the time spent running each algorithm are logged and analysed. The results of the calculations are reported in Table 2.

It can be observed from the comparison that the upgraded algorithm running time and running efficiency are twice as fast as the old algorithm running time and efficiency. The revised A* algorithm, as a result, has greater coverage environment efficiency and therefore significantly enhances the efficiency of route planning in search and rescue operations. In order to determine whether or not the algorithm's overall route-planning capabilities can be utilised in a 3D environment, we must simulate the obstacle-avoidance flight of the UAV in a 3D environment. It is vital to guarantee that the experimental simulation settings in the three-dimensional environment are compatible with the two-dimensional environment, which has the dimensions $200 \times 200 \times 50$, as illustrated in Figure 6. In addition, the UAV swarm's beginning location and target position should be the same. Simulation of the modified A* algorithm and the classic A* algorithm is displayed, and the correctly created flight paths are indicated as blue lines. The findings of the simulation can be used to validate the viability of the modified A* algorithm in a three-dimensional space setting. The path planning of the upgraded A* algorithm in the 3D environment is shown in Figure 6.

According to our simulation experiments, the classic A* algorithm searches the whole region in 72.653799 seconds, which is a significant time investment. By using the enhanced A* algorithm in the simulation experiment, we were able to determine the amount of time necessary to search for each region. It takes 4.762879 seconds to complete

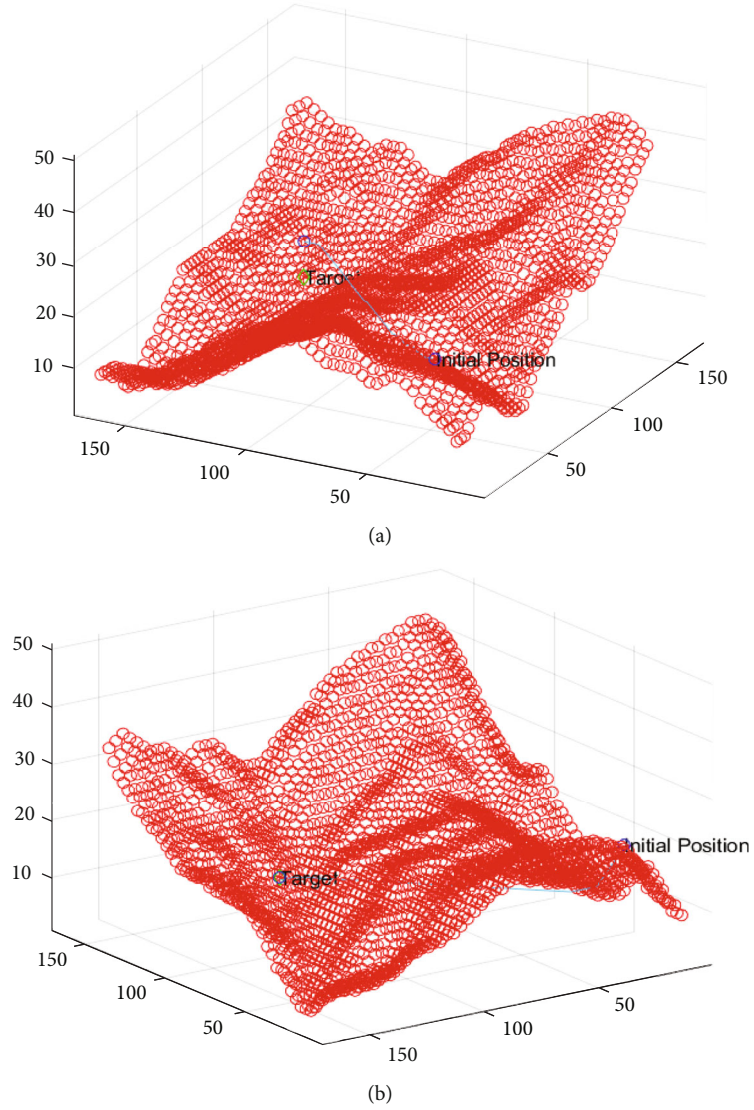

Figure 9: Path planning for simulating group 2 with the optimal A* algorithm (a) and traditional A* algorithm (b).

the search route in region 1 and 26.462905 seconds in region 2, while it takes 26.172347 seconds and 7.598442 seconds to complete the search path in region 3 and region 4, respectively. As can clearly be seen, the total time required by the enhanced A* algorithm to search in each area is less than the total time required by the classic A* algorithm to search over the whole experimental region. Furthermore, because of the enhanced A* algorithm's parallel-computing capability, which means that each UAV can search its own area at the same time, the total search time can be equal to the search time of the most time-consuming area. Consequently, it becomes reasonable to recognize that the aggregated time cost is only equal to the time cost that is the highest in each of the areas concerned, i.e., that the overall time cost is equal to the search time of 26.462905 seconds for region 2. When compared to the search time cost of the classic A* algorithm, such a search time result significantly lowers the amount of calculation time and searching cost of the computers.

Following the introduction of the enhanced A* algorithm, we discovered that the enhanced A* algorithm not only decreases the computing time but also reduces the amount of storage required. For the sake of our experiments, we can compare the cost of dynamic storage. The cache size of temporary storage which is the amount of storage space needed by the open list during the operation of the A* algorithm can be used to illustrate the dynamic storage cost of the algorithm. The classic A* algorithm has a dynamic overhead of 5751×8 double bytes for storing an element in its memory. However, 192×8 double bytes in range 1, 3286×8 double bytes in range 2, 1582×8 double bytes in range 3, and 290×8 double bytes in range 4 represent the dynamic storage overhead of the newly enhanced A* algorithm, and the total storage cost can be determined by the sum of these four zones' storage costs which is equal to 5350×8 double bytes. Moreover, in the situation of parallelism, the optimized algorithm only requires the dynamic storage space equivalent to the maximum area dynamic

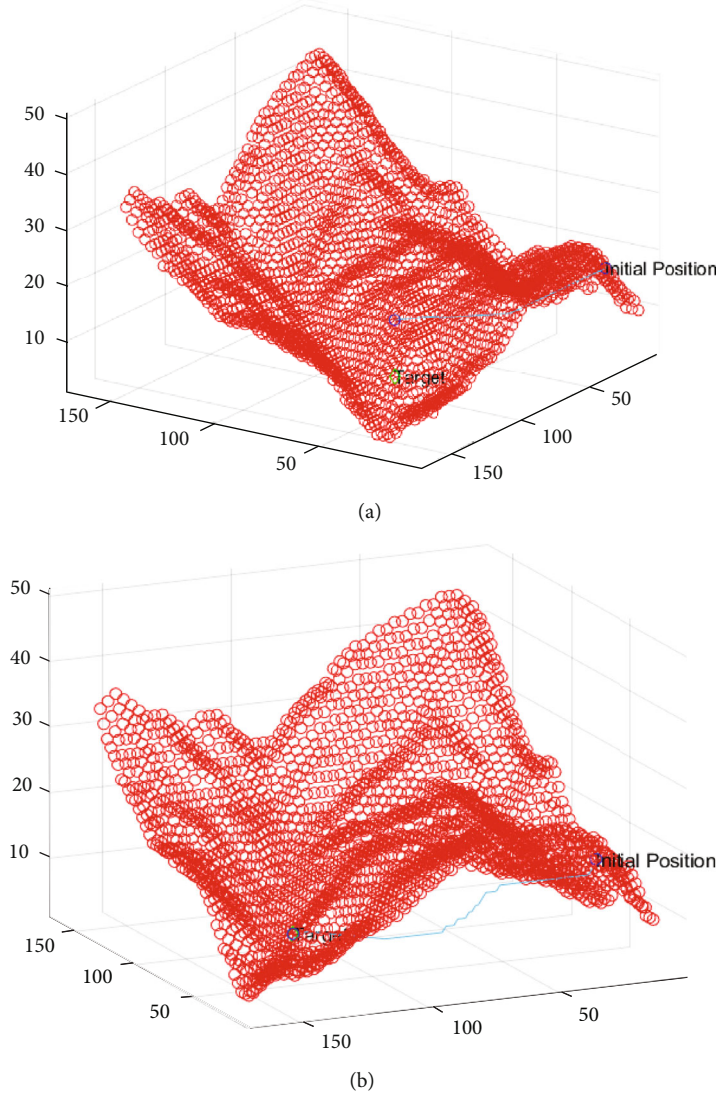

Figure 10: Path planning for simulating group 3 with the optimal A* algorithm (a) and traditional A* algorithm (b).

storage cost; i.e., the enhanced A* algorithm only requires the temporary usage of 3286×8 double bytes of storage space.

The A* algorithm running speed is expected to be enhanced further in the following steps. We have shown via simulation studies that the running speed of the A* algorithm is greatly enhanced because of the addition of the new data structure. A grid map of 200×200 pixels is used in the environment, as shown in Figure 7. The beginning point of the grid position is $(20, 80)$, the ending point of the grid position is $(145, 180)$, and the width of each side of the grid is set at 1 cm. Obstacles that cannot be traversed, drone positions, and target locations are still shown on the grid. The simulation program is still MATLAB, and it runs the A* algorithm to find the best route of each grid diagram and to generate the ideal path diagram of the final search. Figure 7 depicts the route optimization impact diagram of the simulation experiment based on the grid diagram, in which the blue line depicts the optimum path from the start point to the end point. Table 3 shows the time it took to find the best route using the standard A* algorithm and the time it took to find the ideal path using the modified A* algorithm. When the data in the table is compared, it can be observed that the upgraded A* algorithm optimum route search time is much less than that of the classic A* algorithm's ideal path-search time.

4.3. Enhanced A Algorithm in Complicated Environment Result Analysis.* Finally, for the UAV to be successful in the simulated search environment experiment, it must be capable of doing route planning in a complicated geographic context. It is our intention for the drone to go from its starting place in its memory map to the target point at the furthest end of the map. Upon obtaining complete knowledge about a given region, the robot may extend its memory map while travelling to another place, which is generally a target point location that can be searched from a distance, and the previous area is removed from the list of enlarged areas.

Table 5: Comparison between time cost and memory space of the optimal A* algorithm and the traditional A* algorithm.

		Time (s)	OPEN_set	Path length
Optimal A* algorithm	Group 1	83.302443	8682 × 10	2823 × 3
	Group 2	1397.958990	32648 × 10	22,004 × 3
	Group 3	2398.762770	53063 × 10	42,966 × 3
Traditional A* algorithm	Group 1	5327.154309	72576 × 10	56,733 × 3
	Group 2	5226.409287	58245 × 10	45,836 × 3
	Group 3	5825.127755	83178 × 10	73,759 × 3

We must first employ a bigger and more sophisticated environment to assess the efficiency of our algorithm performance and optimization in order to ensure that it is as successful as possible. The grid map technique is used to map distinct natural mountain settings in a grid, which is then processed by MATLAB. Using MATLAB, the raw data for all environment is recorded in DEM format and shown as grid maps, which are then processed. In order to make our simulation studies easier, we used mountain model data that was readily accessible on the internet. The data is acquired by drone using sensors such as laser rangefinders and other similar technologies.

As shown in Table 4, the experiment is to select three different sets of UAV starting points and mission target points in the same mountain habitat. The mountain environment faced by the first group is very straightforward, the mountain does not have a significant height difference, and the drone can have a relatively clear vision. In the second set of cases, the dip between high and low peaks might make route planning for UAVs more difficult due to the uneven terrain. It is more difficult to navigate in the final group, and the drone must traverse a mountain range to locate a target position that must be looked for and retrieved on the opposite side of the mountain range. We execute route planning using the modified A* algorithm in three distinct settings and measure the time it takes the UAV to complete the goal of covering the whole environment in each environment. Figures 8–10 depict the outcomes of the route-planning process. We then compare the results as shown in Table 5. Experiments have shown that the algorithm can determine the safest and fastest route for unmanned aerial vehicle (UAV) swarms in challenging situations.

According to these results, it is clear that it is possible to scale an algorithm over several drones and prevent repeat training by using a task assignment model which means that we can train on one drone and then duplicate the model's parameters across numerous drones while still doing calculations. This method allows us to save significant amounts of computational resources. When job assignment is done using a policy network, however, it is necessary to verify that all visited target sites are stationary since the UAV must know the sequence in which the search target points will be visited before the mission can begin. Drones may depart from their initially anticipated optimal course in real life owing to measurement mistakes or the necessity to avoid obstacles, as is the case with humans. It will be necessary to perform the task of correcting the deviated path according to the information search by the UAV during the travel process through the policy network to be able to correct the initially predicted action. This will increase the computational requirements of the system.

And following the use of communication as a task target in task assignment, the entire time taken from task start to task assignment and notification of target location is significantly reduced. This is in accordance with prior analyses and our experimental findings. With regard to the connection quality of the path obtained by the final object detection, the quality of the coverage path is highly dependent on the connectivity of the network. The use of communication can help the UAV to guarantee the quality of the coverage path to a certain extent. When the density of drones and the density of the connection network reach a specific level, the drone swarms can determine the position of the target point immediately after the drone has discovered the target point which allows for faster response times. Therefore, in future development and application, machine learning and other methods can be introduced to conduct autonomous training on UAV path planning to ensure that UAVs avoid deviating from the optimal path.

5. Conclusions

The task assignment method is combined with the A* path-planning algorithm, and we conduct obstacle avoidance and locally planned activities without having a global understanding of the obstacles in the environment. An approach that that is only focused on the obstacle-avoidance search and rescue route in the 2D environment is also proposed, and this method is then applied to the 3D environment path design. This method can shorten the time required for route planning while maintaining safety during obstacle avoidance. These assessments and outcomes will make it easier to make decisions about how robots finish task assignment and route planning, and they can be used for other systems and not only swarms of drones. The problems of task assignment and route planning are combined into our work. When confronted with a circumstance in which there are a significant number of obstacles, we may handle the task assignment and route-planning tasks concurrently under a single general framework by merging the two issues described above. Using agent learning, the UAV can attain the desired state while also avoiding obstacles at the same time in this framework.

By merging task assignment and route planning, our enhanced A* algorithm is capable of solving complicated environment-oriented problems with high efficiency. This enhanced A* algorithm not only provides a considerable increase in computing speed and a significant reduction in search time but also decreases the amount of computer memory required by the algorithm to perform its functions. UAV search and rescue operations benefit from this enhanced A* algorithm in another way as well; because drone swarms can subdivide the space, drones can react to an increased number of emergent situations with more flexibility. A significant role for this application in the process of drone search and rescue is played by the military. When an aircraft crashes, it is quite simple for it to spark additional calamities, such as forest fires, in the surrounding area. At this point, the drone will need to alter certain search and rescue paths to accommodate the current scenario. As an example, if the UAV is not allowed to visit area A during the experiment, it must merely make minor adjustments to the route plan of area A and attempt to find a new path. Using this correction approach, the modified A* algorithm does not need to search for and find the solution for the entire environment, which saves a significant amount of unneeded memory usage as well as computation time.

The modified A* algorithm can additionally take into account a variety of other optimization techniques. When developing the enhanced A* algorithm, for example, it is possible to use the idea of layers. By using the notion of layers, we are able to further subdivide the environment into cells that have the same vertical features and are contained inside a common horizontal extent. This updated A* algorithm is more focused on combining task assignment and route planning at the same level as our previous A* algorithm. While the conventional approach of searching and route planning in a 3D environment does not need as much search and mapping time as it is focused just on the horizontal plane, it does necessitate much more computer memory space and more search and mapping time to cover the complete area. We can suppose that by adding the notion of layers, the area of scanning the environment can be allotted to numerous UAVs in a layer-by-layer way, therefore increasing the efficiency of the search. This algorithm of looking for information about the surroundings can be done layer by layer. Path planning can be expanded to include more than just the x-axis and y-axis movement directions by including a layer-by-layer method into mission planning. Furthermore, this way of unlimited movement can be used to search for nodes that cannot be found in numerous horizontal planes. It is also possible to minimize the need for additional memory space in the 3D environment by using the way of searching for coverage environments with the collaboration of the drone swarms.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

No potential conflict of interest was reported by the author.

References

[1] J. Chen, D. Chenglie, Y. Zhang, P. Han, and W. Wei, "A clustering-based coverage path planning method for autonomous heterogeneous UAVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25546–25556, 2022.

[2] J. Chen, Y. Zhang, W. Lianwei, T. You, and X. Ning, "An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16842–16853, 2022.

[3] R. W. Beard and T. W. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 1, Maui, HI, USA, 2003.

[4] S. Hayat, E. Yanmaz, C. Bettstetter, and T. X. Brown, "Multi-objective drone path planning for search and rescue with quality-of-service requirements," *Autonomous Robots*, vol. 44, no. 7, pp. 1183–1198, 2020.

[5] Y. Jin, Y. Liao, A. A. Minai, and M. M. Polycarpou, "Balancing search and target response in cooperative unmanned aerial vehicle (UAV) teams," *IEEE Trans-actions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 3, pp. 571–587, 2006.

[6] R. G. Ganesan, S. Kappagoda, G. Loianno, and D. K. A. Mordecai, "Comparative analysis of agent-oriented task assignment and path planning algorithms applied to drone swarms," https://arxiv.org/abs/2101.05161.

[7] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3D path planning algorithms," *Journal of Control Science and Engineering*, vol. 2016, Article ID 7426913, 22 pages, 2016.

[8] J. Peng, Y. Huang, and G. Luo, "Robot path planning based on improved A* algorithm," *Cybernetics and Information Technologies*, vol. 15, no. 2, pp. 171–180, 2015.

[9] F. Samaniego, J. Sanchis, S. Garcia-Nieto, and R. Simarro, "Smooth 3D path planning by means of multiobjective optimization for fixed-wing UAVs," *Electronics*, vol. 9, no. 1, p. 51, 2020.

[10] Z. Chen, J. Li, S. Wang, J. Wang, and L. Ma, "Flexible gait transition for six wheel-legged robot with unstructured terrains," *Robotics and Autonomous Systems*, vol. 150, article 103989, 2022.

[11] Z. Chen, J. Li, J. Wang, S. Wang, J. Zhao, and J. Li, "Towards hybrid gait obstacle avoidance for a six wheel-legged robot with payload transportation," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 3, pp. 1–21, 2021.

[12] J. Chen, F. Ling, Y. Zhang, T. You, Y. Liu, and D. Xiaoyan, "Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system," *Swarm and Evolutionary Computation*, vol. 69, article 101005, 2022.

[13] Y. Jinchao Chen, Y. Z. He, P. Han, and D. Chenglie, "Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems," *Journal of Systems Architecture*, vol. 129, article 102598, 2022.

[14] S. Yang, Y. Wang, and X. Chu, "A survey of deep learning techniques for neural machine translation," 2020, https://arxiv.org/abs/2002.07526.

[15] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[16] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: any-angle path planning on grids," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, pp. 1177–1198, Vancouver, Canada, 2007.

[17] L. De Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for UAVS in 3D environments," *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1–4, pp. 247–264, 2012.

[18] L. Niu and G. Zhuo, "An improved real 3D A* algorithm for difficult path finding situation," in *Proceeding of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, Beijing, China, 2008.

[19] F. Yan, Y.-S. Liu, and J.-Z. Xiao, "Path planning in complex 3D environments using a probabilistic roadmap method," *International Journal of Automation and Computing*, vol. 10, no. 6, pp. 525–533, 2013.

[20] S. Kim, H. Kim, and T.-K. Yang, "Increasing SLAM performance by integrating grid and topology map," *Journal of Computers*, vol. 4, no. 7, pp. 601–609, 2009.