Theory and Practice of Science and Technology
2025, VOL. 6, NO. 2, 96-105
DOI: 10.47297/taposatWSP2633-456920.20250602

Research on UAV Delivery Planning Based on Deep learning and Visual SLAM

Haoyang Zhou

School of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu, 210000, China

ABSTRACT
In recent years, with the rising demand for takeaways, customers' requirements for takeaway delivery efficiency have gradually increased. For this problem, a method of using UAVs as carriers for takeaway delivery is proposed, because UAVs do not need to consider traffic problems, the takeaway delivery time can be greatly shortened, and UAVs can be recycled and reused, and the distribution cost is also reduced. In this paper, we will use visual SLAM to simulate the delivery environment, and plan the path and calculate the optimal path based on deep learning reference A* algorithm and ant colony algorithm. Finally, the feasibility of the method is verified.

KEYWORDS
takeaway delivery; A* algorithm; ant colony algorithm; visual SLAM; path planning

1 Introduction

With the development of science and technology, drones have played an irreplaceable role in both military and civilian use, the advent of Internet +, and the wide application of artificial intelligence, making online ordering services a trend, and the main way of takeaway delivery today is through takeaway workers with electric vehicles, etc. Although electric vehicles can be more efficient than motor vehicles in the morning, noon and evening rush hours, when the number of takeaways is increased, it is often difficult to meet the needs of customers in terms of manpower and material resources. In order to solve this problem, a method of UAV delivery based on deep learning and visual SLAM was proposed, visual positioning and map construction were carried out through visual slam, and then an optimal distribution path was planned by using A* algorithm or ant colony algorithm through computer deep learning, so as to improve the various defects of UAV takeaway, and put forward feasible suggestions for future UAV takeaway.

2 Research status at home and abroad

2.1 Research status of visual SLAM technology

Visual SLAM (Simultaneous Localization and Mapping) is a technology used to build maps in real time and determine one's location.

● Sensor input

Visual SLAM systems use cameras to acquire image data of the environment. Common camera types include monocular cameras, binocular cameras, and RGB-D cameras[1]. The video stream or image sequence provided by the camera is the input to the entire system.

● Feature extraction and description

Extract feature points from the input image and generate corresponding descriptors. These features can be key points (e. g., SIFT, ORB) or patch-based features. Feature extraction is a key step in visual SLAM, which directly affects the performance and robustness of the system[2].

● Feature matching

Matches the feature point of the current image to the feature point in the previous frame or keyframe. The purpose of feature matching is to find correspondence in adjacent image frames for subsequent motion estimation. Commonly used matching methods include nearest neighbor search[3] and kNN[4].

● Map building

Combine the camera's motion trajectory with feature points to build a sparse or dense map of the environment. Mapping can be divided into two main types:

① Sparse map[5]: Includes only the location of feature points, typically used for positioning.

② Dense Maps[6]: Include more details about the environment, often used for navigation and obstacle avoidance.

● Loopback detection

During the SLAM process, it is recognized whether the camera has returned to the previously visited area and loopback detection is performed. Loopback detection helps to correct accumulated errors and improve the accuracy of the map. Commonly used methods are feature-based loopback detection and image-based loopback detection[7].

● Map optimization

In order to reduce the cumulative error and optimize the map, the global map is optimized. Common method is Bundle Adjustment[8]. this method improves the accuracy of the entire map by optimizing the relative pose between adjacent frames.

● System output

The final output of the visual SLAM system is a map of the environment and the current pose (position and pose) that are updated in real time. This information can be used for applications such as robot navigation, augmented reality, and more.

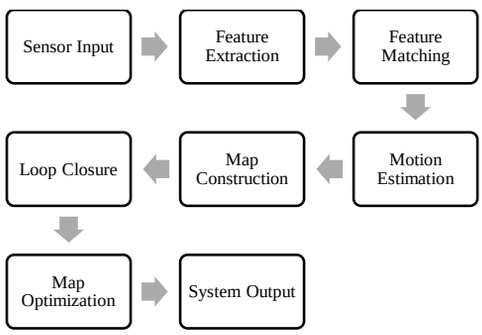

Figure 1 Visual SLAM monolithic framework

2.2 Research on deep learning

The following will introduce several algorithms that are commonly used in UAV path planning

2.2.1 A* algorithm

The A* algorithm combines the advantages of Dijkstra's algorithm[9](finding the shortest path) and the advantages of heuristic search (guided search by estimation) to find the optimal path from the starting point to the target node by comprehensively considering the actual cost and the estimated cost.

$$\mathbf{f}(\mathbf{a}) = \mathbf{g}(\mathbf{a}) + \mathbf{h}(\mathbf{a})$$

$f(a)$ is the cost of node a; $g(a)$ the cost of starting node to node a; h(a) is the cost from node a to the target node[10].

The heuristic function of the A* algorithm(h(a)) usually uses Euclidean distance($\mathbf{h}_1(\mathbf{a})$) and Manhattan distance($\mathbf{h}_2(\mathbf{a})$)[11]

$$\mathbf{h}_1(\mathbf{a}) = \sqrt{(\mathbf{x}_a - \mathbf{x}_{terget})^2 + (\mathbf{y}_a - \mathbf{y}_{terget})^2}$$

$$\mathbf{h}_2(\mathbf{a}) = \left|\mathbf{x}_a - \mathbf{x}_{terget}\right| + \left|\mathbf{y}_a - \mathbf{y}_{terget}\right|$$

$\mathbf{x}_a$ and $\mathbf{y}_a$ are the x and y coordinates of node a, respectively. $\mathbf{x}_{terget}$ and $\mathbf{y}_{terget}$ are the x and y coordinates of the target node, respectively.

Advantages:

It can accurately reflect the shortest straight-line distance between two points, and is suitable for path planning in continuous space. Suitable for a wide range of environments and applications, whether grid or non-grid maps. Euclidean distances provide consistent shortest path estimates regardless of the direction of movement.

Disadvantages:

The calculation formula involves the square sum square operation, which is relatively large and slow. In certain environments, such as meshes that can only move horizontally and vertically, the advantage of Euclidean distance is not obvious.

Advantages: only need to calculate the absolute difference between the horizontal and vertical directions, the amount of calculation is small, and the speed is fast. In a city block or grid map, the Manhattan distance is more in line with the actual walking or moving path. It is suitable for scenarios where the path can only move horizontally and vertically, such as a board, grid game, etc.

Disadvantages: Manhattan distance may underestimate the actual distance if diagonal movement is allowed. In some cases, Manhattan distances don't accurately reflect the actual shortest path, especially in non-grid-like environments.

```
function A*(start, goal)
openList = [start]
closedList = []
start.g = 0
start.h = heuristic(start, goal)
```

```
start.f = start.g + start.h
while openList is not empty
current = node in openList with
lowest f
if current is goal
return reconstruct_path(current)
remove current from openList
add current to closedList
for each neighbor of current
if neighbor in closedList
continue
tentative_g =current.g+dist_between(current, neighbor)
if neighbor not in openList
add neighbor to openList
else if tentative_g >= neighbor.g
continue
neighbor.cameFrom = current
neighbor.g = tentative_g
neighbor.h = heuristic(neighbor, goal)
neighbor.f = neighbor.g + neighbor.h
return failure
function reconstruct_path(current)
total_path = [current]
while current.cameFrom is not null
current = current.cameFrom
total_path.append(current)
return total_path
```

Advantages : The A* algorithm can guarantee that the optimal path is found when the heuristic function satisfies certain conditions, such as consistency and acceptability. Combining the advantages of the shortest path search and heuristic search of the Dijkstra algorithm, it can find high-quality paths in a short time. The A* algorithm is suitable for a variety of different graph search and path planning problems, such as robot navigation, game AI, map path planning, etc. By using heuristic functions, the A* algorithm can effectively guide the search process and reduce unnecessary node expansion. The algorithm is easily extensible and can be combined with different heuristic functions to adapt to different application scenarios.

Disadvantages :If the graph structure is complex or there are many nodes, the A* algorithm requires a large amount of computation, which may lead to a long computing time and high memory usage. The performance of the algorithm is highly dependent on the selection of heuristic functions, and improper selection of heuristic functions may lead to inefficiency or failure to find the optimal solution. The A* algorithm needs to store all nodes in both the open and closed lists, and the memory consumption can be very high in large-scale graph searches. Compared with other simple search algorithms, the implementation of A* algorithm is more complex, and it needs to deal with open lists, closed lists, and various heuristic information.

2.2.2 Ant colony algorithm

Ant colony algorithm is a heuristic optimization algorithm based on ant foraging behavior, which is widely used to solve combinatorial optimization problems. The algorithm guides path selection by simulating the pheromones left behind by ants in the search for food. Pathways with high pheromone concentrations are more attractive, resulting in a positive feedback mechanism.

Each ant constructs a path from the start to the end, updating the pheromones according to the path quality. Pheromones evaporate over time to prevent premature convergence to the local optimal solution. Through multiple iterations, the ant gradually finds the global optimal path.

The ant colony algorithm combines pheromone concentration and heuristic information to determine the movement of ants using a transfer probability formula. In the formula, the weight of pheromone concentration and heuristic information is adjusted by parameters to ensure the adaptability of the algorithm in different scenarios.[12]

Mathematical model of ant colony algorithm:

- **Transfer Probability Formula**

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\left[\tau_{ij}(t)\right]^{\alpha}*\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{s\in\ allowed_k}\left[\tau_{ij}(t)\right]^{\alpha}*\left[\eta_{ij}(t)\right]^{\beta}}, & j\in allowed_k\\ 0, & others\end{cases}$$

$$\eta_{ij}(t)=1/d_{ij}$$

$p_{ij}^{k}(t)$ indicates the probability that the k-th ant will choose to move from node i to node j at time t

Ant colony algorithm pseudocode.

$\tau_{ij}(t)$ is expressed as the concentration of information on the search path from position i to position j at time t.

$\eta_{ij}(t)$ is the heuristic factor in the algorithm, which is the derivative distance between nodes i and j.

d_{ij} means the distance from node i to node j.

α is the importance factor of pheromones.

β is the factor of importance of the heuristic information.

$allowed_k$ indicates the set of nodes that the ant has not walked[13].

● Pheromone update function

Since the pheromones left by ants have been accumulating during the iteration process, the residual pheromones generated with the increase of running time and the number of ants will cause the pheromone heuristics to fail to play their role, so the ant pheromones must be updated in time every time the path optimization is completed. At t+1, the update strategy of pheromones from position i to position j is as follows:

$$\tau_{ij}(t+1)=\tau_{ij}(t)*(1-\rho)+\Delta\tau_{ij}(t)$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t)$$

ρ is the rate of pheromone evaporation($\rho\in(0,1)$).

$\Delta\tau_{ij}(t)$ indicates the amount of pheromones released by the k-th ant on paths i to j($\Delta\tau_{ij}(0)=0$)[14].

There are many variant models of ant colony algorithm, including Ant System (AS), Ant Quantity (AQ) and Ant Density (AD). The difference between the three models is the difference in the numerical setting strategy of $\Delta\tau_{ij}^{k}(t)$.[15]

In the Ant-Cycle model,

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{L_k}, & tour(i,j)\in tour_k\\ 0, & other\end{cases}$$

Q denotes the pheromone intensity, which affects the convergence speed of the algorithm to a certain extent; L_k denotes the total length of the path traveled by the k-th ant in this cycle.

In the Ant Quantity System model:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}\dfrac{Q}{d_{ij}}, & tour(i,j)\in tour_k\\ 0, & other\end{cases}$$

In the ant-dense system model:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}Q, & tour(i,j)\in tour_k\\ 0, & other\end{cases}$$

The difference between the three is:

Ant Cycle Model (AS): Uniformly updates pheromones at the end of each iteration, which is suitable for global path optimization.

Ant Quantity Model (AQ): The ant updates pheromones at each step of movement, which is suitable for step-by-step optimization of the path.

Ant Density Model (AD): The ants update their pheromones when they reach each node, focusing more on path optimization of node density.

function ACO()
initialize pheromone levels
while not termination condition
for each ant
construct solution based on pheromone and heuristic information
end for
update pheromones based on constructed solutions
end while
return best solution found

Advantages: The ant colony algorithm is naturally parallel, and multiple ant individuals can work at the same time, which speeds up the search speed and efficiency. The algorithm can dynamically adjust the concentration of pheromones to adapt to different search spaces and problems, and has strong robustness and adaptability. Through the accumulation and evaporation of pheromone concentration, the ant colony algorithm can effectively avoid falling into the local optimal solution in the global search, so as to find the global optimal solution.The ant colony algorithm can be applied to a variety of different types of combinatorial optimization problems, such as the Traveling Salesman Problem (TSP), the Vehicle Routing Problem (VRP), etc. The ant colony algorithm combines heuristic information and pheromone mechanism to improve the search efficiency and problem solving quality of the algorithm.

Disadvantages : For some complex problems, the convergence speed of the ant colony algorithm is slow, and it may take more iterations to find a satisfactory solution. The performance of the algorithm is sensitive to parameters (such as pheromone evaporation rate, heuristic factors, etc.), and the parameter tuning process is complex and time-consuming. Since individual ants need to work at the same time, the algorithm requires a lot of computing resources in each iteration, and the amount of computation is large. In some cases, the algorithm may converge to the local optimal solution prematurely, resulting in a decrease in overall performance. The update mechanism of pheromones may lead to the imbalance of pheromone concentration on the path, which affects the search effect of the algorithm.

3 Modeling the task decomposition

Assuming that only one drone is working during the delivery process, the maximum load of the drone is set to 10kg, and the maximum takeaway load is 8kg for safety, and one takeaway is set to 750g, it can be concluded that one drone can deliver up to 12 takeaways at the same time.

Set the $\mathbf{x_i}$ to each food distribution point ($\mathbf{i} \le 12$), and define the set of food preparation locations $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ... \mathbf{x}_{12}\}$, $\mathbf{y_j}$ to each food delivery point ($\mathbf{j} \le 12$), and delivery location set $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{12}\}$, assuming that the drone flight speed is constant, so the shortest path also means the shortest flight time, and the total flight time is $\mathbf{T}$, $\mathbf{T_{ai,min}}$means that the starting point is the $\mathbf{x}_1$, the end point is the shortest path sought by the $\mathbf{x_i}$, $\mathbf{T_{bij,min}}$means the starting point is the $\mathbf{x_i}$, the end point is the shortest path sought by the $\mathbf{y_j}$, the set $\mathbf{T} = \{\mathbf{T}_{21}, \mathbf{T}_{22,}, ..., \mathbf{T_{ij}}\}$, and $\mathbf{min}(\mathbf{T})$ means the minimum value of the set.

$$\mathbf{T_{ij}} = \mathbf{T_{ai,min}} + \mathbf{T_{bij,min}}$$
$$\mathbf{T_{min}} = \mathbf{min}(\mathbf{T})$$

When the last catering mission is over, the drone mission ends and moves on to the next merchant that needs to be delivered.

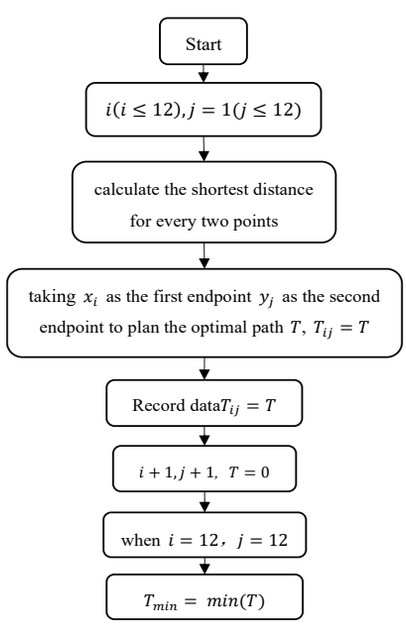

Figure 3 Modeling flowchart

First of all, the drone departs from the $\mathbf{x}_1$, and in the first stage, every food service point except for the $\mathbf{x}_1$ can become the starting point of the second stage. Therefore, taking $\mathbf{x_i(i \ne 1)}$ as the end point can always obtain an optimal path for each endpoint, which is denoted as $\mathbf{t_{ai,min}}$ and there are $(\mathbf{i} - 1)$ results. After that, $\mathbf{x_i(i \ne 1)}$ is used as the starting point, similar to the first stage, each $\mathbf{x_i(i \ne 1)}$ can plan a simplest path with each $\mathbf{y_j}$ as the end point, denoted as $\mathbf{t_{bij,min}}$. A total of

results $(\mathbf{i}-1)\times\mathbf{j}$ were generated, and $\mathbf{T}_{\mathbf{min}}$ was obtained by adding $\mathbf{t}_{\mathbf{ai,min}}$ and $\mathbf{t}_{\mathbf{bij,min}}$.

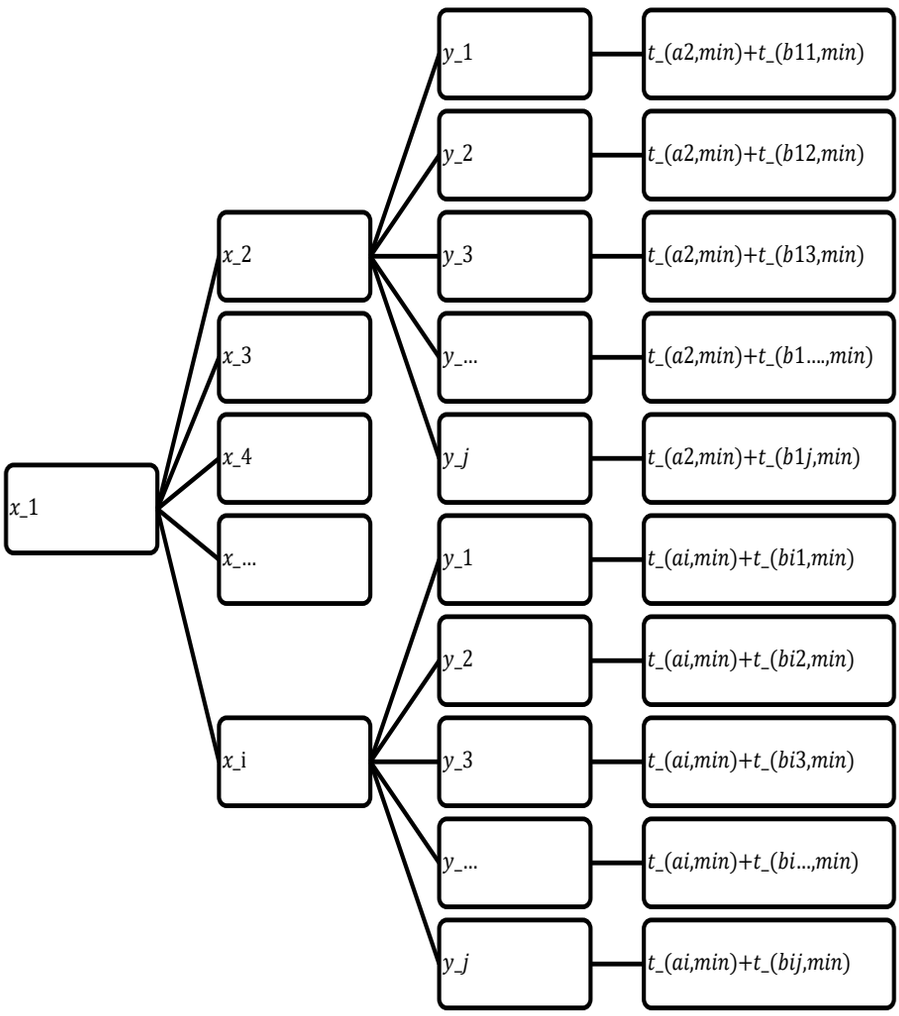

Figure 4 Result tree map

4 Simulation

4.1 Simulation requirements

Grid generation: A 51×51 contains 24 randomly distributed points, of which 12 are pick-up points (green), the first green point is (26,26), 12 are the destination (red), the obstacle point is black, and the background is white.

Obstacles: Generate random obstacles of 50 points, making sure they don't overlap with 24 points.

A* algorithm: calculate the shortest distance between every two points (including the pick-up point and the destination, Use the Manhattan distance)

Ant colony algorithm: the first part is to start with the first point, that is, (26, 26), respectively, with the remaining 11 green dots as the end point, and during the period, except for the starting point and the end point, all the other pick-up points (green dots) have to pass through and calculate the shortest total distance, and every two points of distance is replaced by the distance calculated by the A* algorithm.

（num_ants = 20; % Number of ants

alpha = 1; % Importance factor of pheromone

beta = 2; % Importance factor of heuristic

rho = 0.5; % Pheromone evaporation rate

Q = 100; % Pheromone increase strength

num_iterations = 100; % Number of iterations）

4.2 Emulate pseudocode

// Generate Grid Define grid size as 51x51 Randomly generate 24 points, with 12 as pickup points (green) and 12 as drop-off points (red) Set the first pickup point at (26, 26) Randomly generate 50 obstacle points ensuring they do not overlap with pickup and drop-off points

// Draw Grid Create a figure window Plot pickup points in green Plot drop-off points in red Plot obstacle points in black

// Calculate Shortest Distance Matrix Initialize a list of all points (pickup and drop-off points) Create a matrix to store the shortest distances, sized 24x24 For each pair of points, use A* algorithm to calculate the shortest path and save it in the matrix Save the matrix to a file named shortest_distances.txt Display the shortest distance matrix

// Ant Colony Algorithm Define parameters for the ant colony algorithm Find the index of the first pickup point (26, 26)

// Calculate t1 Collection Initialize t1 collection with size 11 For each of the remaining 11 pickup points: Use the ant colony algorithm to calculate the shortest path and save the result in t1

// Calculate t2 Collection Initialize t2 collection with size 11×12 For each endpoint in t1 as the start point: For each of the 12 drop-off points: Use the ant colony algorithm to calculate the shortest path and save the result in t2 Display t1 and t2 collections

// Create Collection T Initialize T collection For each value in t1: Add the value to the corresponding 12 values in t2 and save the results in T Display T collection

// Function Definitions Define heuristic function Define function to get neighbor nodes Define A* algorithm function Define ant

4.3 Demonstration of simulation results (using MATLAB)

Three different simulations are analyzed below：

First:

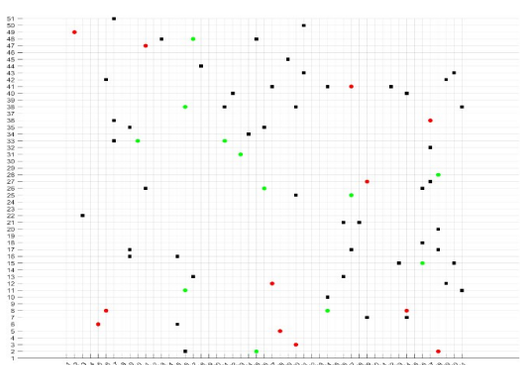

Figure 5 Generate a map

	A	B	C	D	E	F	G	H	I	J	K	L
1	452	450	436	430	430	426	424	420	418	418	408	
2	450	448	436	432	428	426	422	420	418	418	408	
3	454	448	436	430	428	426	422	422	418	418	408	
4	454	448	438	432	428	426	422	424	418	418	408	
5	452	448	436	430	428	426	422	422	418	418	410	
6	452	448	436	430	428	426	422	424	418	418	408	
7	458	450	434	432	428	426	422	420	418	418	408	
8	456	448	436	432	428	426	422	424	418	418	408	
9	452	446	436	430	428	426	422	420	420	420	408	
10	452	450	436	432	428	426	422	420	418	418	408	
11	454	450	438	432	428	426	422	420	418	418	408	
12	452	448	436	430	430	426	422	420	418	418	410	
13												
14												

Figure 6 Total distance table

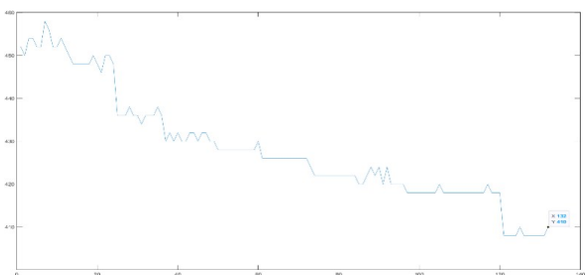

Figure 7 Total distance line plot

Second

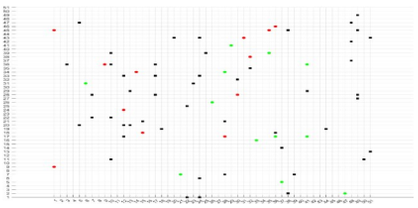

Figure 8 Generate a map

	A	B	C	D	E	F	G	H	I	J	K	L
1	452	450	436	430	430	426	424	420	418	418	408	
2	450	448	436	432	428	426	422	420	418	418	408	
3	454	448	436	430	428	426	422	422	418	418	408	
4	454	448	438	432	428	426	422	424	418	418	408	
5	452	448	436	430	428	426	422	422	418	418	410	
6	452	448	436	430	428	426	422	424	418	418	408	
7	458	450	434	432	428	426	422	420	418	418	408	
8	456	448	436	432	428	426	422	424	418	418	408	
9	452	446	436	430	428	426	422	420	420	420	408	
10	452	450	436	432	428	426	422	420	418	418	408	
11	454	450	438	432	428	426	422	420	418	418	408	
12	452	448	436	430	430	426	422	420	418	418	410	
13												
14												

Figure 9 Total distance table

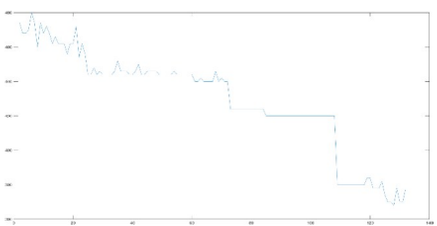

Figure 10 Total distance line plot

Third

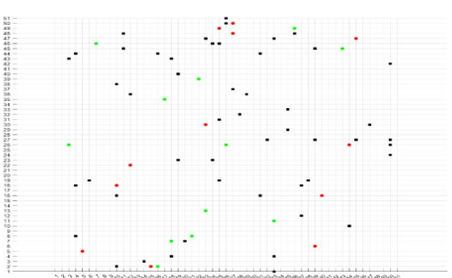

Figure 11 Generate a map

	A	B	C	D	E	F	G	H	I	J	K	L
1	458	456	460	456	454	454	454	438	418	400	390	
2	458	456	460	470	454	454	452	438	418	400	390	
3	458	456	460	456	454	454	452	438	418	400	390	
4	458	456	460	456	454	454	454	438	418	400	390	
5	458	456	460	456	454	454	452	438	418	400	390	
6	458	456	460	456	454	454	452	438	418	400	390	
7	458	456	460	456	454	454	452	438	418	400	390	
8	458	456	460	456	454	454	452	440	418	400	390	
9	458	456	460	456	454	454	452	440	418	400	390	
10	466	456	460	464	454	454	452	438	418	400	390	
11	458	456	460	456	454	454	452	438	418	400	390	
12	458	456	460	466	454	454	452	438	418	400	390	
13												
14												
15												

Figure 12 Total distance table

4.4 Simulated data analysis

1. There isn't just one shortest path.

2. Different first destinations will change the overall path planning.

3. In the case of the same first endpoint, the choice of the second endpoint has little or sometimes no effect on the overall path planning.

4. No matter how the points are randomly distributed, basically the shortest and longest distances do not vary very much.

For further analysis, we reintroduce the t1,t2 data plots of the first simulation:

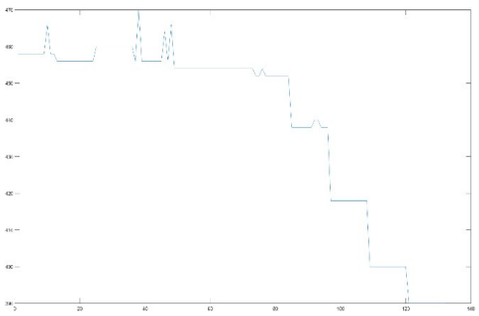

Figure 13 Total distance line plot

	A
1	176
2	176
3	174
4	174
5	174
6	174
7	176
8	176
9	176
10	176
11	174

Figure 14 t1

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	276	274	278	278	276	276	282	280	276	276	278	276	
2	274	272	272	272	272	272	274	272	270	274	274	272	
3	262	262	262	264	262	262	260	262	262	262	264	262	
4	256	258	256	258	256	256	258	258	256	258	258	256	
5	256	254	254	254	254	254	254	254	254	254	254	256	
6	252	252	252	252	252	252	252	252	252	252	252	252	
7	248	246	246	246	246	246	246	246	246	246	246	246	
8	244	244	246	248	246	248	244	248	244	244	244	244	
9	242	242	242	242	242	242	242	242	244	242	242	242	
10	242	242	242	242	242	242	242	242	244	242	242	242	
11	234	234	234	234	236	234	234	234	234	234	234	236	
12													
13													

Figure 15 t2

We can further find that although the selection of the first endpoint will not cause a large gap in the optimal path planning of the first part, the first endpoint will lead to a significant difference in the path planning of the second part. (**Note:** Unlike the previous master path planning table, the front is the same second endpoint in the transverse direction and the same first endpoint in the longitudinal direction, and the latter is the opposite).

5 Summary

In the UAV delivery method based on deep learning and visual SLAM, the A* algorithm and the ant colony algorithm have different uses in the simulation using the code. They are used for path optimization at different stages. The A* algorithm is mainly used to calculate the shortest path between each pair of points in the grid. This process includes the following steps: generating the grid and obstacles by creating a 51x51 grid and randomly generating 24 points (including pickup points and drop-off points) and 50 obstacle points. Calculating the shortest distance matrix by using the A* algorithm to calculate the shortest path between each pair of points and storing the results in a 24x24 distance matrix. The ant colony algorithm is used to further optimize the path based on the shortest distance matrix that has been calculated. This includes two stages: the first stage involves starting from the first pickup point (26, 26), passing through the other 11 pickup points to find the shortest path, and saving the results into the t1 set. The second stage involves starting from the endpoint of the first stage (the pickup points in t1), passing through the 12 drop-off points to find the shortest path, and saving the results into the t2 set. In summary, the A* algorithm is used to calculate the shortest path between points in the grid environment, which serves as the basis for path optimization. The ant colony algorithm further optimizes the delivery path between pickup points and drop-off points based on the shortest distance matrix calculated by the A* algorithm, aiming to find the global optimal solution. By combining these two algorithms, the code can efficiently calculate the optimal delivery path for the UAV from the pickup points to the drop-off points, thereby

improving delivery efficiency.

In this paper, the feasibility of the experiment is verified from the aspect of simulation, and the results show that the whole code running process is completed in less than 20 seconds, and the experimental results are good. However, whether this method is feasible still needs to be verified in real scenarios by using UAVs.

References

[1] WANG Xia, ZUO Yifan. Advances in visual SLAM research[J]. CAAI transactions on intelligent systems, 2020, 15(5): 825–834.

[2] Liu Mingzhu, Chen Rui, Chen Junyu, et al. B-Spline-ORB Feature Point Extraction Algorithm[J]. Journal of Harbin University of Science and Technology,2022,27(3):97-104.

[3] Shan Tingjia. Research on Nearest Neighbor Search Algorithm Based on Image Features[D]. Anhui: University of Science and Technology of China,2017.

[4] Wu Xueyan, Wang Shuihua, Zhang Yudong. Review of K-Nearest Neighbor Algorithm Theory and Application[J]. Computer Engineering and Applications,2017,53(21):1-7.

[5] Jiang Ming. Research on map construction method based on sparse features of visual SLAM[D]. Guangdong: Guangdong University of Technology,2023(in Chinese).

[6] ZHAO Jiaran. Three-dimensional dense map reconstruction based on visual SLAM[D]. Zhongyuan Institute of Technology, 2023.

[7] CAI Jundong. Research on visual SLAM loopback detection[D]. Inner Mongolia:Inner Mongolia University of Science and Technology, 2022.

[8] Mo Xiaorui, Zhang Weiyi, Nian Cheng, et al. A Review of Beam Adjustment Optimization Chips in Visual SLAM Robots[J]. integrated circuits and embedded systems,2024,24(11):29-40.

[9] Mo Xiaorui, Zhang Weiyi, Nian Cheng, et al. A Review of Beam Adjustment Optimization Chips in Visual SLAM Robots[J]. integrated circuits and embedded systems,1999,24(3):209-212.

[10] LIU Zhijun,SU Liang,WU Shaoxiong. Development of Local Planning Method for Obstacle Avoidance Path Based on AStar Algorithm[J]. Bus Technology and Research,2023,45(3):6-9.

[11] Geng Hongfei, Shen Jianjie. Improvement and Verification of A* Algorithm in AGV Path Planning[J]. Computer applications and software, 2022,39(1):282-286.

[12] TANG Chuanyin, ZHANG Mingli, LI Jinghong, et al. Research on Takeaway Distribution Path Planning Based on Improved Ant Colony Algorithm[J]. Journal of Nanjing University of Information Science & Technology,2024,16(2):145-154.

[13] Gu Haijiao. Research on UAV route trajectory planning algorithm[D]. Jilin: Changchun University of Technology,2021.

[14] LIU Zihou. Research on Robot Path Planning Based on Improved Ant Colony Algorithm[D]. Anhui: Anhui University of Science and Technology,2023.

[15] ZHAO Juanping. Research on ant colony optimization algorithm for path planning of mobile robot[D]. Liaoning: Northeastern University, 2012.