


Article

Autonomous Multi-UAV Path Planning in Pipe Inspection Missions Based on Booby Behavior

Faten Aljalaud ^{1,2,*}, Heba Kurdi ^{1,3}  and Kamal Youcef-Toumi ³

¹ Computer Science Department, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia; hkurdi@ksu.edu.sa

² Computer Science Department, Imam Mohammad Ibn Saud Islamic University, Riyadh 11564, Saudi Arabia

³ Mechanical Engineering Department, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA; youcef@mit.edu

* Correspondence: 438203858@student.ksu.edu.sa

Abstract: This paper presents a novel path planning heuristic for multi-UAV pipe inspection missions inspired by the booby bird's foraging behavior. The heuristic enables each UAV to find an optimal path that minimizes the detection time of defects in pipe networks while avoiding collisions with obstacles and other UAVs. The proposed method is compared with four existing path planning algorithms adapted for multi-UAV scenarios: ant colony optimization (ACO), particle swarm optimization (PSO), opportunistic coordination, and random schemes. The results show that the booby heuristic outperforms the other algorithms in terms of mean detection time and computational efficiency under different settings of defect complexity and number of UAVs.

Keywords: inspection; bio-inspired algorithms; unmanned aerial vehicle; booby; multi-UAV; path planning; pipes

MSC: 68V



Citation: Aljalaud, F.; Kurdi, H.; Youcef-Toumi, K. Autonomous Multi-UAV Path Planning in Pipe Inspection Missions Based on Booby Behavior. *Mathematics* **2023**, *11*, 2092. <https://doi.org/10.3390/math11092092>

Academic Editor: Xiaosong Du

Received: 20 March 2023

Revised: 13 April 2023

Accepted: 19 April 2023

Published: 28 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Unmanned aerial vehicles (UAVs) and multi-UAVs have opened up new possibilities for various applications in both military and civil domains. However, most UAVs still require human operators to control and guide them, which limits their autonomy and efficiency. To achieve full autonomy, UAVs need to be able to make decisions without human intervention [1,2], which requires optimizing their path planning algorithms [3,4].

One of the emerging civil applications of multi-UAVs is in construction and infrastructure inspection [5], which accounts for 45% of the UAV net market value [2]. In particular, the pipe inspection field is one of the areas with prevailing growth rates among numerous UAV applications. This is especially true because most civil defense schemes globally mandate automatic or fixed sprinklers in buildings [6]. To inspect such indoor pipe networks, human operators are usually employed. However, some companies have started automating the process using UAVs. For instance, a UK-based UAV inspection services company [7] is presenting a UAV specifically designed for indoor environments. The UAV has a camera and a protective frame surrounding the indoor drone. It is used to inspect pipes by an operator to take high-definition photos. Another example is the new fire sprinkler system at the Amazon warehouse in Ajax which started to leak randomly, requiring a building-wide shut down and evacuation. Amazon inspected the system by hiring a photography company to inspect the sprinkler network for potential damage or poor installation [8]. These cases are not autonomous and require constant human operation. As a result, real-life cases like these show the need for studies like ours, especially autonomous solutions.

In this field, corrosion detection is a critical task, as corrosion causes almost 50% of steel pipeline failures [9–11], and steel pipes are widely used in automated sprinkler systems [12,13]. Corrosion tends to form localized pits that concentrate the corrosive attack [14], so small UAVs, known as micro air vehicles (MAVs), are preferred for their reachability and easy deployment [1]. However, MAVs have limited resources and battery capacity [4], which poses a challenge for path planning optimization [2], an NP-hard problem [15]. The multi-UAV path planning problem is a particular case of the multirobot path planning problem, where robots can fly in a k -dimensional space. Multi-UAV systems outperform single-UAV systems in parallelism, robustness, simplicity, and cost [16]. While a great deal of research has been conducted on UAV applications, very few studies have considered multi-UAV applications for construction and infrastructure inspection [2].

Metaheuristics are general optimization methods that explore the problem space iteratively to find an optimal or near-optimal solution [17,18]. However, metaheuristics are often resource-intensive, as they require evaluating an objective function repeatedly with a group of search agents [18]. This poses a challenge for MAVs, which have limited resources. Therefore, metaheuristics should only be used when no problem-specific heuristics are available. A problem-specific heuristic is a customized solution that tries to find a “good” solution on the first attempt [19]. The proposed research fills a gap in the literature by developing a problem-specific heuristic for multi-UAV systems in inspection missions. Moreover, it introduces a novel technique inspired by the foraging behavior of the booby bird to overcome complex optimization problems.

The main contributions of this paper can be summarized as follows:

1. A novel booby-inspired heuristic for the path planning problem in multi-UAV systems.
2. Successful implementation of the proposed approach in the context of inspection missions.
3. A rigorous evaluation framework for the proposed approach against rival multi-UAV path planning algorithms.

The paper is structured as follows. Previous methods for multi-UAV path planning inspection using bio-inspired techniques are introduced in Section 2. The problem definition and path planning model are defined in Section 3. Section 4 explains the booby motivation for this work and the system concept, and provides the booby algorithms. Section 5 explains the evaluation process and results, while Section 6 contains the concluding remarks and identifies some future study directions.

2. Related Studies

2.1. UAV Inspection Missions

The navigation of a robot or UAV involves several stages: localization, mapping, path planning, and motion control [20,21]. These stages have been addressed differently by various studies on inspection missions using UAVs. For example, some studies have focused on path planning [22], while others have explored localization methods [23,24]. A common goal of UAV inspection mission planning is to optimize the quality of visual information captured by UAVs and create efficient mission plans for data collection. However, most of the existing studies have mainly dealt with image processing techniques [25], as reviewed by [26,27]. One exception is the study by Quenzel et al. [24], which presented an integrated chimney inspection system using a UAV. They combined laser localization and visual odometry for navigation and allowed the user to select points of interest for subsequent inspection rounds. The UAV then used a traveling salesman problem solver to find an optimal order of visiting those points for a shorter flight duration.

UAV inspection applications can be divided into two types: indoor and outdoor missions. While most of the existing research on UAV inspection missions has focused on outdoor environments [28], such as power lines [29,30], chimneys [24], bridges [31,32], railways [25], and construction sites [28], there is a lack of studies on multi-UAV applications for indoor environments. Multi-UAV inspection path planning is a challenging and important problem that deserves more attention. Bono et al. [31] proposed a trajectory

planning method for a UAV fleet based on a digital twin of a bridge inspection, where the user selects waypoints on the digital model corresponding to the points of interest. The method computes noncolliding trajectories for each UAV around the centroid trajectory using a swarm-distributed model predictive control strategy. However, this method is not suitable for inspecting pipes in an indoor environment, which is the focus of our work. We present a novel multi-UAV path planning algorithm for pipe inspection that considers constraints and collision avoidance in an indoor environment.

2.2. Bio-Inspired Approaches to the Multi-UAV Path Planning Problem

The UAV path planning problem has been addressed by various methods in the literature [33,34]. Among them, bio-inspired approaches have shown more potential to handle this complex and dynamic problem [33]. However, most of the existing studies have focused on single-UAV scenarios [34]. Multi-UAV path planning is a more challenging and realistic problem that requires further investigation. For instance, Ismail et al. [22] proposed a fruit fly optimization algorithm to find the optimal number and paths of UAVs for oilfield inspection. They assumed one depot and multiple goals for each UAV, and they optimized the initial paths using an improved fruit fly algorithm. Their results indicated that three UAVs were optimal for their problem setting. However, they only used one map and two performance measures (cost function and running time), which limits the generalization of their approach. Another related work by Pan et al. [35] formulated the multi-UAV path planning problem as a multiple traveling salesman problem (m-TSP). They developed a deep learning model trained by a genetic algorithm to collect data from distributed sensors using UAVs. Their objective was to minimize the path length and the solving time in challenging scenarios. They studied how different numbers of data nodes affected their model's performance. The performance measures were the average total distance of UAVs, the average required number of UAVs, and the average solving time. They tested their algorithm against random and genetic algorithms only.

Previous studies on multi-UAV path planning using bio-inspired approaches have some limitations in the inspection domain. They do not show how their approaches scale with different settings and numbers of UAVs, nor do they measure the inspection effectiveness with metrics such as detection time. In contrast, our work demonstrates our approach's performance under various settings and multiple UAVs, and considers several performance measures.

3. Problem Definition and Path Planning Model

3.1. Problem Definition

We considered a multi-UAV inspection mission. As shown in Figure 1, a set of UAVs should inspect a network of pipelines to locate defects and report them to a base station. The mission starts with homogenous UAVs taking off from one location to cooperatively inspect all parts of the pipes externally to detect defects. During the execution of the mission, the UAVs will fly to different locations, given that no two UAVs can be at one location simultaneously. Once a UAV has inspected a location, no other UAV will re-inspect it again. The locations and number of defects are unknown a priori, while the pipeline network's dimensions and altitude are known.



Figure 1. An operational scenario of a multi-UAV inspection mission.

3.2. Path Planning Model

Several assumptions must first be made to solve the multi-UAV cooperative inspection mission planning problem:

1. UAVs move in straight lines.
2. A network location (cell) can have a maximum of one defect which simplifies the process of detecting and localizing defects.
3. Each location can only have one UAV active at any given time to avoid collisions and ensure that each location is covered by at most one UAV [36].
4. Altitude layering [37]: Pipes and other UAVs cannot collide with one another. Thus, UAVs must fly higher or lower than pipelines. UAVs will use altitude layering once they fly. Then, the z dimension remains constant after UAV takeoff. Note that takeoff and landing times are negligible.
5. Because this is an indoor inspection mission, the weather does not affect UAVs.
6. UAVs can sense changes in the pipeline and find defects within a certain distance of the pipe.
7. The cell dimensions are smaller or equal to the UAV's detection range.
8. The battery swapping time is neglected [2].

Variables of the proposed model:

U —total number of UAVs;

u —index of a UAV;

$N_{u,w}$ —number of waypoints in a feasible trajectory for UAV u ;

$D_{i,j}$ —distance traveled by UAV between the i th waypoint and j th waypoint;

$r_{i,j}$ —energy consumption between the i th waypoint and j th waypoint;

$\sum_j r_{u,j}$ —total energy consumption of the u th UAV;

$Path_u$ —feasible trajectory for a UAV u ;

$Cost_{Path_u}$ —total distance traveled corresponding to $Path_u$;

δ_m —initial energy of every UAV;

δ_u —UAV's energy at a given time;

μ —coefficient to denote energy consumption.

In our work, the optimization algorithm generates a series of three-dimensional waypoints [38]. A feasible path $Path_u$ is stored as a vector in which an element $w_{u,i} = (x_{u,i}, y_{u,i}, z_{u,i})$ denotes the i -th waypoint of UAV u . Equation (1) defines $Path_u$:

$$Path_u = (w_{u,1}, w_{u,2}, \dots, w_{u,N_{u,w}}) \quad (1)$$

Our objective ($F_{Objective}$) is to improve the trajectory planning quality by minimizing the UAV's total traveled distance. We formulated our problem as a complicated traveling

salesman problem (TSP) [35], where there may be more than one salesman. Additionally, we added the energy constraint to consider while choosing the path. The UAV u cannot choose a path that exceeds its energy δ_u . Equation (2) describes the above model.

$$F_{Objective} = \begin{cases} \text{Min} \left(\left(\sum_{i=1}^U \text{Cost}_{Path_i} \right) \right) \\ \text{s.t.} \begin{cases} \sum_j r_{1,j} < \delta_1 \\ \dots \\ \sum_j r_{U,j} < \delta_U \end{cases} \end{cases} \quad (2)$$

The following Equation (3) computes the total distance traveled corresponding to $Traj_u$. As a result, the equation will find the cost associated with each trajectory.

$$\text{Cost}_{Traj_u} = \sum_{j=1}^{N_{u,w}-1} D_{W_{u,j}, W_{u,j+1}} \quad (3)$$

Assuming the velocity is 1 m/s in the energy consumption (EC), as denoted as r above, we computed this in our experiments as follows [35,39,40]:

$$r_{i,j} = \mu * D_{i,j} \quad (4)$$

4. Proposed Method

4.1. Booby Inspiration

Many biological studies focus on marine bird colonies and foraging. Marine birds modify their foraging techniques to maximize prey encounters in high-prey-density areas. The booby bird's foraging behavior yields many insights that may aid multi-UAV path planning [41,42]. In a booby colony, reproduction depends on breeding location and partner. Females choose partners, and males compete for nesting spots. Birds should choose the finest mates and locations (near to forage, safe from weather and predators, and easy entry and exit for breeders) [43]. There are three phases to the foraging trip. First, the booby bird leaves its colony to forage rapidly away from it at a high and constant speed along a linear path. The bird continues this behavior until it reaches a foraging zone. Second, once it reaches a foraging zone, it will change direction and speed frequently [44,45]. The foraging zone is an area with a high likelihood of encountering prey or where the bird has already detected prey. During foraging activity, the birds will land on the water or dive [44]. Hence, individual birds adjust their traveling behavior in response to prey density and maximize their prey encounters by increasing turning rates and reducing travel speeds. The term for this adaptive response to prey density is area-restricted search (ARS) behavior [42,44–46]. Finally, the booby will return to the colony at a more constant flight speed and with a relatively straight route or path parallel to the outgoing path [44,45]. Note that there can be one or more ARS zones during a trip [45]. When the booby lays eggs, it usually only lays three [43].

4.2. System Model

In our implementation, we discretize the pipes' map to enclose the pipes in small unit areas called cells, as shown in Figure 2. Each cell represents a network location (point) that should be visited exactly once. UAVs should scan all network locations (cells with circles) to find defects (cells with circles and squares). Each UAV plans its route to visit several cells and inspect them. Once the mission ends, each cell corresponding to a network location must have been visited exactly once by a UAV. Additionally, all defects should have been identified while minimizing the total travelled distance by all UAVs.

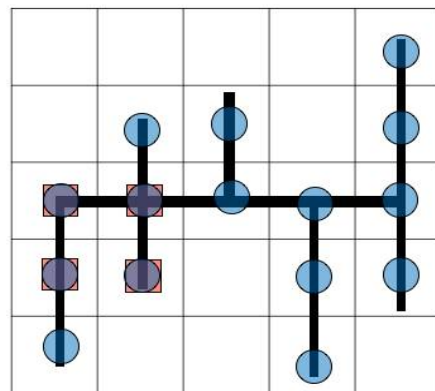


Figure 2. Grid representation of the environment (blue circles represent a location, and red squares represent defects).

The world map is clustered into different zones. The number of zones is determined during the initialization phase. Each zone can be in one of the following states: available, unavailable, inspected, and uninspected. All zones are available and uninspected states during the initialization phase.

Available UAVs are categorized into three dynamically formed sets: primary, secondary, and temporary. The UAVs have different flying modes while executing the mission: default inspection mode and ARS inspection mode. In the default mode, the UAV scans the network location by moving to the closest point from its current position. If it detects a defect in any location, it switches to the ARS mode. In this mode, the UAV examines the surrounding areas of the defect more closely for a certain period before returning to the default mode. The UAVs adjust their roles dynamically according to the inspection coverage of each zone. The mission ends when all zones are inspected.

4.3. From Inspiration to Algorithm

We began by investigating the feasibility of defining the process of assigning UAVs to a network location (looking for defects and then detecting them) in an inspection mission that is comparable to the foraging process in a booby colony. The booby system has two inspection modes: default inspection mode and ARS inspection mode; this mimics the foraging behavior of the booby once it finds prey. Moreover, the UAV has three different roles in our system: primary, secondary, and temporary, which represent the male, female, and booby egg, respectively. Table 1 shows the mapping between the booby colony and our system.

Table 1. Biological inspiration: mapping between a booby colony and the booby system.

Booby Colony	Our System
Ground	Virtual map of the network area
Nest	Zone
Male booby	Primary UAV
Female booby	Secondary UAV
Egg	Temporary UAV
Foraging	Inspection
Prey	Defect

The male booby is responsible for choosing a nest location on the ground and displaying it for potential female partners. Utilizing this behavior, we make the primary UAV choose a location in the assigned zone and send a request for a secondary UAV. Additionally, since the booby can have a limited number of eggs, the primary UAV can request a limited number of temporary UAVs to help inspect the current zone. Additionally, the booby can make longer foraging trips, so the primary UAV can step away from the assigned zone if

there is a secondary UAV in the current zone once the remaining uninspected locations are below a certain threshold.

Exploiting the method of choosing the highest quality partner for the female booby, we let the primary UAV send one request for a secondary UAV to join it in the zone. If there are many join requests for one secondary UAV, it will evaluate the requests and choose the one with the highest fitness value.

Simultaneously, the chicks of the booby stay temporarily in the nest. Therefore, we take advantage of this behavior by having temporary UAVs help inspect a specific zone for a limited duration. Once each UAV is assigned a temporary role, it will calculate the maximum number of rounds, which means the maximum number of locations to be inspected. Since the primary UAV queries the temporary UAVs if the number of uninspected locations is high, the maximum number of rounds of the temporary UAVs depends on the number of uninspected locations remaining in the zone corresponding to the primary UAV's request.

4.3.1. Central Control Algorithm

The central control algorithm has several tasks, mainly during the initialization phase of the system. First, the central control algorithm will cluster the map into a predefined number of nonoverlapping zones using the K-means clustering algorithm. Next, it will assign roles to UAVs by dividing them into two groups of primary and secondary UAVs using Equations (5) and (6).

$$\text{Primary UAVs\#} = \left\lceil \frac{U}{2} \right\rceil \quad (5)$$

$$\text{Secondary UAVs\#} = \left\lfloor \frac{U}{2} \right\rfloor \quad (6)$$

Then, the algorithm assigns primary UAVs to available zones as follows:

1. Sort zones in descending order of their number of network locations.
2. Randomly select a primary UAV and assign it to the largest available zone.
3. Keep assigning primary UAVs in the same manner until all zones are unavailable or there are no more primary UAVs.
4. If all zones are unavailable and there are still some unassigned primary UAVs, change their roles to secondary UAVs.

It is important to note that if there are still some available zones but no more primary UAVs, this case is handled by the primary UAV algorithm where once a primary UAV completes inspecting an area of current zone larger than a certain threshold, it leaves for another zone with no primary UAV (the zones that were left out in the beginning), as explained in Section 4.3.2

4.3.2. Primary UAV Algorithm

Each primary UAV will be assigned to one available zone, and the zone cannot have more than one primary UAV assigned to it. As the algorithm in Figure 3 shows, the primary UAV will choose a location in the assigned zone (at random if it is the first time inspecting this zone). In addition, each primary UAV is allowed to send one request for a secondary UAV to join it in the inspection process of the assigned zone. Even if this request has been fulfilled, the primary UAV cannot send more requests to the secondary UAV.

Once the current location of the primary UAV is inspected, the UAV will check whether the whole state of the zone has been inspected. If the zone is still uninspected, it will check if the remaining locations are more than a certain threshold X . If so, the primary UAV will request a temporary UAV. Each primary UAV is permitted to request a certain number of temporary UAVs. The number of temporary UAVs to be requested is calculated depending on the size of the zone assigned to the primary UAV. This value is calculated once for each primary UAV assignment. Intuitively, the larger the zone, the more chances there

are of getting more UAVs assigned as temporary UAVs. In Equation (7), a min–max normalization [47] is used to normalize the total number of locations in the assigned zone to be in the range [1, U].

$$Z'_{total} = \left(\frac{Z_{total} - \text{minvalue of } Z_{total}}{\text{maxvalue of } Z_{total} - \text{minvalue of } Z_{total}} \right) * (1 - U) + 1, \quad (7)$$

where Z_{total} is the total number of locations in the assigned zone; (*min value of Z_{total}*) and (*max value of Z_{total}*) are the size of the smallest and largest zones, respectively. Z'_{total} is the normalized value of the total number of locations in the assigned zone in the range [1, U]. If the primary UAV is assigned to the largest zone, it can send as many temporary UAV requests as U. However, if the primary UAV is assigned to the smallest zone, it will get a chance to send one temporary request.

If the current zone is in the inspected state, the primary UAV will withdraw any previous request the primary UAV has made. Then, the primary UAV will take what is applicable first from the below options:

1. Stay as a primary UAV if there is an available zone. In this option, our system builds a k-dimensional (k-d) tree data structure to hold the available zones' locations and queries each zone to find the nearest neighbor efficiently.
2. Change the role to secondary UAV if there is a join request.
3. Change the role to temporary UAV if there is a temporary UAV request.
4. Change the role to secondary if there is an uninspected zone.

If none of the above options apply, the primary UAV will return to the depot if all zones are inspected.

4.3.3. Secondary UAV Algorithm

The secondary UAVs will be assigned initially by the central control algorithm. Once they start execution, they will check the number of join requests from the primary UAVs, as the algorithm shows in Figure 4. If there is more than one request, the secondary UAV will evaluate these requests and choose the one that maximizes the fitness value. We followed the fitness calculations used by Teng et al. [40]. Suppose the secondary UAV is at location i , the primary UAV of the join request is at location j , the number of defects detected in the zone corresponding to the primary UAV is D_{num} , and the distance between location i and j is given by $Cost_{i,j}$. Equation (8) calculates the fitness of each available join request ($req_{fitness}$).

$$req_{fitness} = w_1(D_{num}) - w_2(Cost_{i,j}), \quad (8)$$

$w_i (i = 1, 2)$ is each component's weight, reflecting the essential differences while evaluating a candidate request. When calculating the weights of the fitness function components, the item that is more significant in the fitness computation is given a greater weight [40]. A secondary UAV seeks to select a primary UAV within a shorter distance (minimization goal) while simultaneously seeking a primary UAV with more defects detected (maximization goal). We therefore assigned a value of one to the distance weight, but with a negative sign as it should be minimized, while we gave a positive value of two to the total number of discovered defects.

Once the inspection algorithm inspects the current location of the secondary UAV, it will check whether the whole zone state has been inspected. If the zone is still uninspected, the secondary UAV will stay in the current zone for further location inspection. However, if the zone state is checked, the secondary UAV will select the first option that applies from the list below:

1. Evaluate join requests, if any, then select the request with the highest fitness value.
2. Change the role to primary UAV if there is an available zone.
3. Change the role to temporary UAV if there is a temporary UAV request.

If none of the above options apply, the secondary UAV will return to the depot if all zones are inspected.

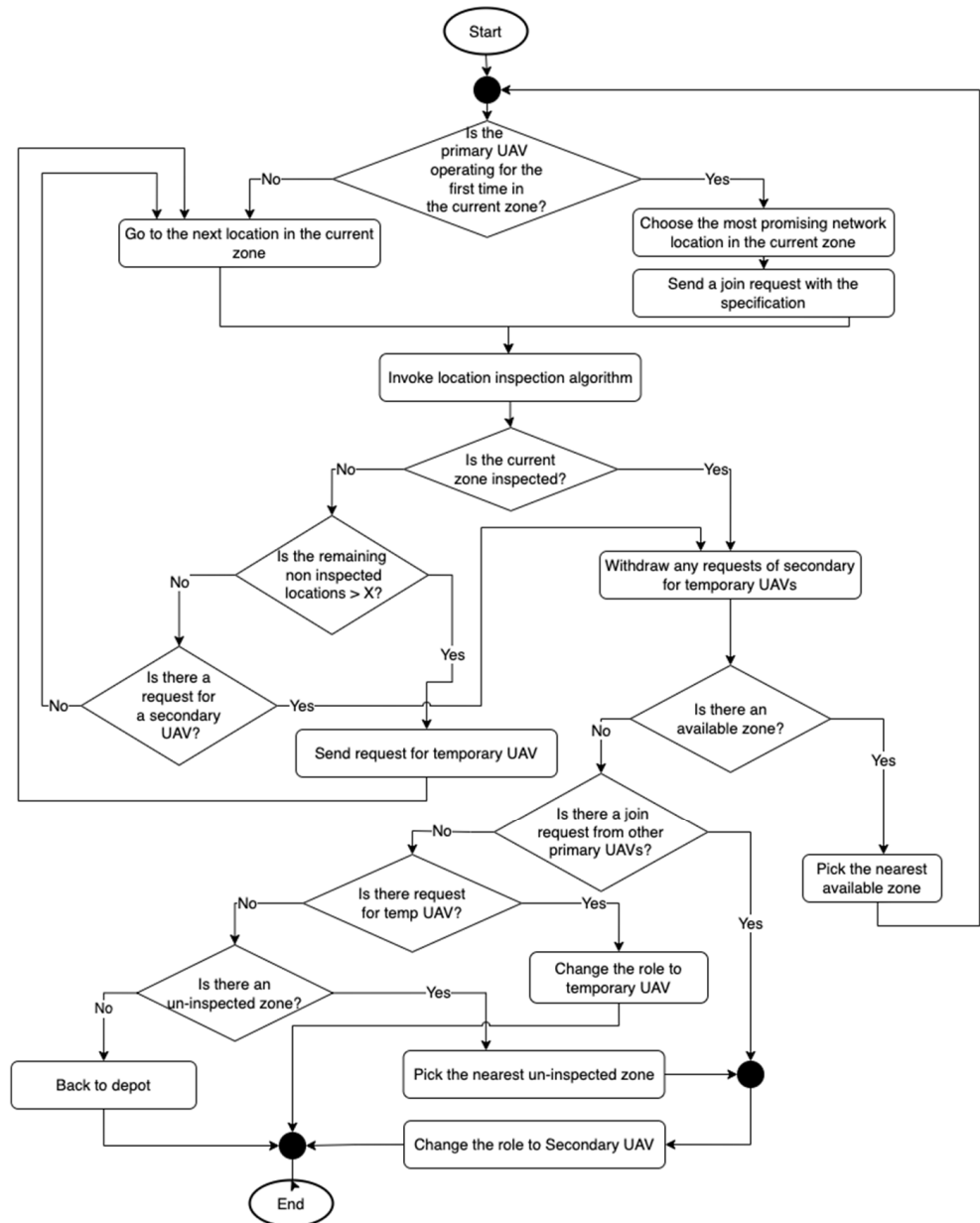


Figure 3. Flowchart of the primary unmanned aerial vehicle (UAV) algorithm.

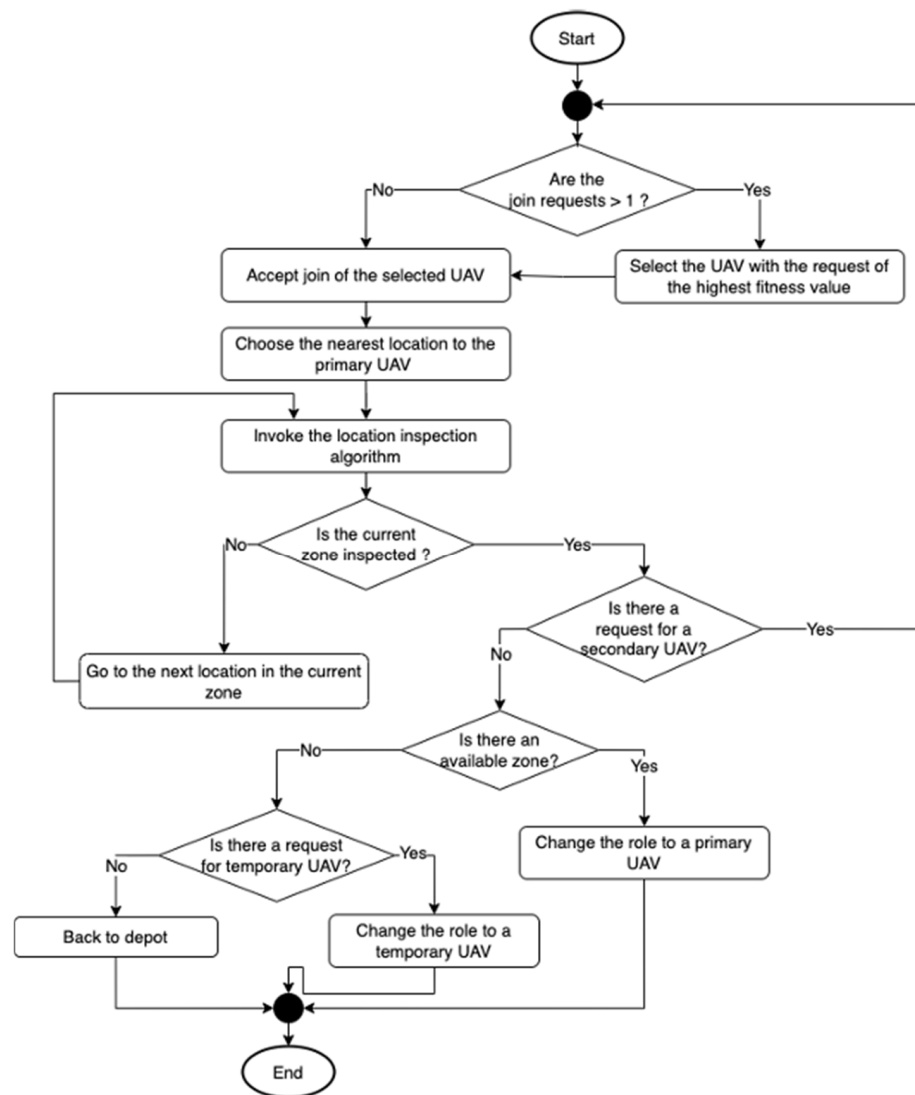


Figure 4. Flowchart of the secondary UAV algorithm.

4.3.4. Temporary UAV Algorithm

The temporary UAVs are not assigned during the initialization phase. In addition, any primary or secondary UAVs not needed at any phase of the execution can change their roles to temporary UAVs and fulfill the need for temporary UAVs.

The number of rounds the temporary UAV will perform is calculated depending on the number of uninspected locations in the assigned zone for the primary UAV. This value is calculated once for each temporary UAV assignment. Essentially, the more uninspected locations there are, the more likely it is that a higher number of rounds will be assigned as maximum rounds. In Equation (9), a min–max normalization [47] is used to normalize the number of uninspected locations in the assigned zone to be in the range [1, U].

$$Z'_{uninspected} = \left(\frac{Z_{uninspected} - \text{minvalue of } Z_{uninspected}}{\text{maxvalue of } Z_{uninspected} - \text{minvalue of } Z_{uninspected}} \right) * (1 - U) + 1, \quad (9)$$

where $Z_{uninspected}$ is the number of uninspected locations in the assigned zone; (min value of $Z_{uninspected}$) and (max value of $Z_{uninspected}$) are zero and the zone’s total number of locations, respectively. $Z'_{uninspected}$ is the normalized value of the number of uninspected locations in the assigned zone in the range [1, U]. If the temporary UAV is assigned to a zone with many uninspected locations, it will be allowed to have maximum rounds as

U. However, if the temporary UAV is assigned to a zone with a smaller percentage of uninspected locations, it will inspect fewer locations.

The algorithm of the temporary UAV is shown in Figure 5. As the algorithm shows, once the temporary UAV starts execution, it will take the first temporary request in a FIFO manner. Then, it will find the nearest location to the primary UAV that requested the temporary UAV and start inspecting that zone. Once the inspection algorithm inspects the current location of the temporary UAV, the temporary UAV will check if the maximum number of rounds has been reached. If the temporary UAV reaches its maximum range or the current zone state is inspected, it will be removed from the current zone and select the first option from the list below:

1. Stay as the temporary UAV if there is any temporary request. If this option applies, the temporary UAV will rest itself. Reassigning the temporary UAV removes it from the current zone and allows it to be assigned to another zone by selecting the available temporary UAV.
2. Change the role to primary UAV if there is an available zone.
3. If a join UAV request is received, change the role to a secondary UAV.

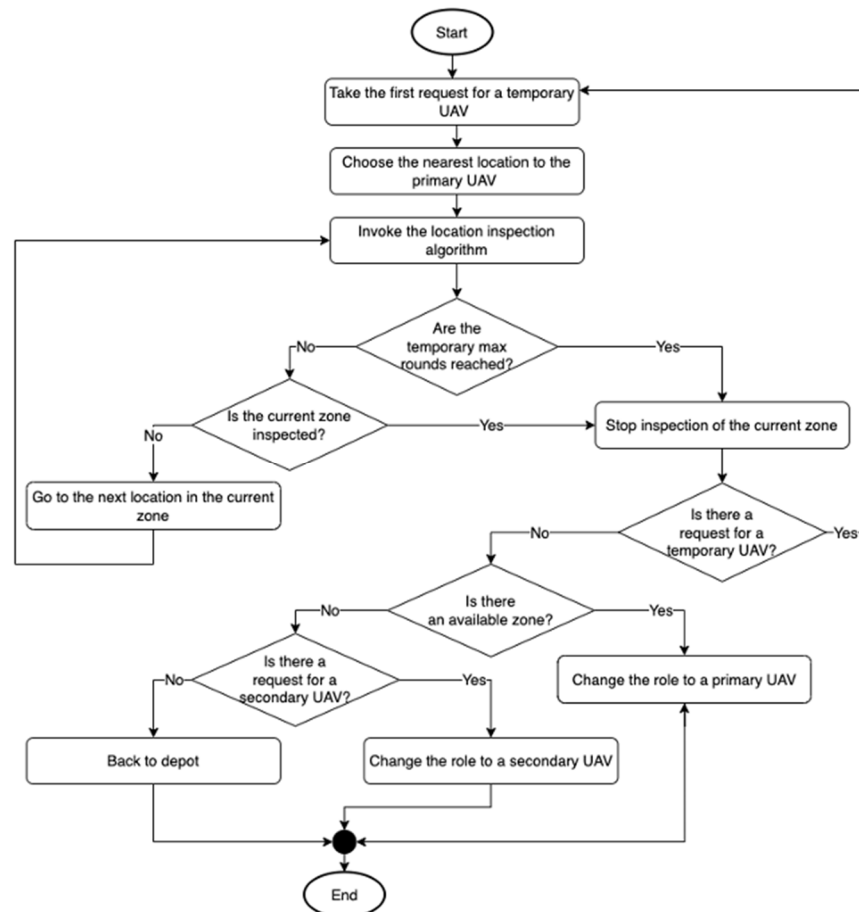


Figure 5. Flowchart of temporary UAV algorithm.

If none of the above options apply, the temporary UAV will return to the depot if all zones are inspected.

4.3.5. Inspection and Go to Next Location Algorithms

Whenever a UAV wants to inspect its current location, it will send that location to the inspection algorithm. The inspection algorithm is shown in Figure 6. The inspection algorithm is responsible for marking a location as inspected and turning on ARS inspection mode once a defect in the location is detected.

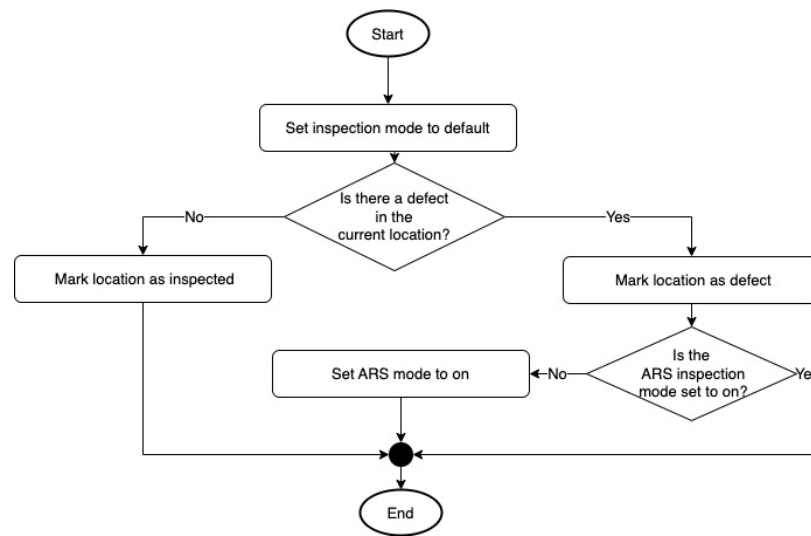


Figure 6. Flowchart of the location inspection algorithm.

After the current location is inspected, the UAV will move to another location in the current zone unless assigned to another zone. The algorithm responsible for choosing the next location is shown in Figure 7. During the initialization phase of the system, all UAV inspection modes are the default ones. In the default inspection, the UAV only inspects the four locations adjacent to the current locations. When a UAV detects a defect, the ARS inspection mode will be on. In the ARS mode, the UAV concentrates on the neighbors' locations of the defect from all eight adjacent cells (including diagonal ones).

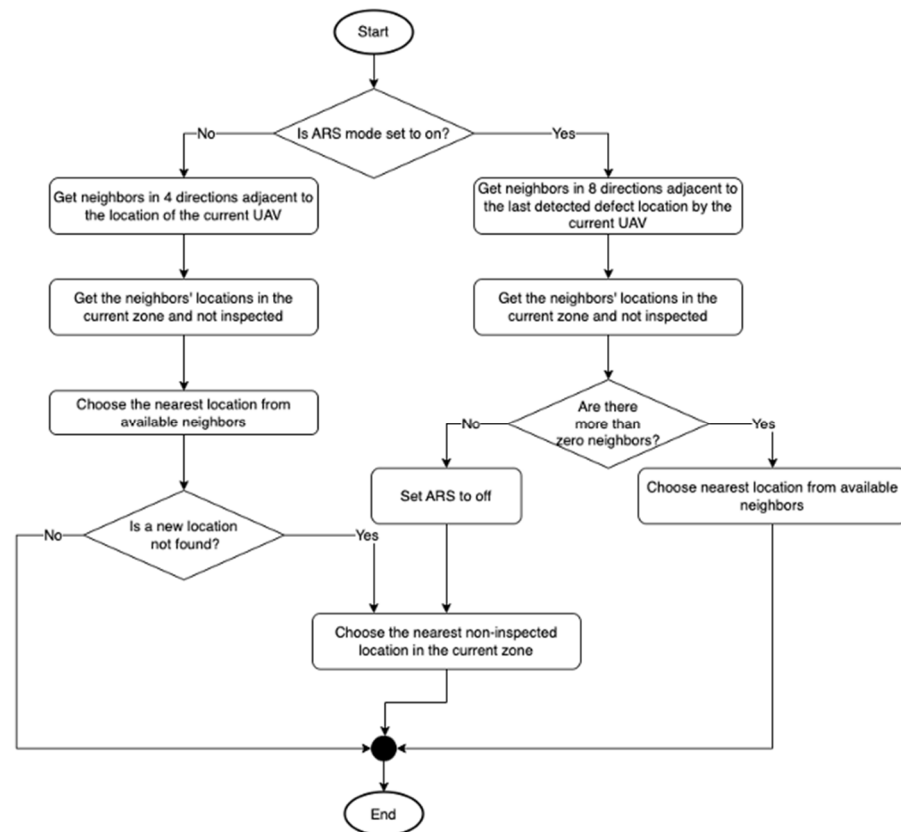


Figure 7. Flowchart of the go to the next location algorithm.

5. Experiments

5.1. Evaluation Framework

The scope of this paper is limited to detecting corrosion in indoor steel pipe networks using a system of multi-MAVs. We evaluated the performance of our system using a rigorous empirical methodology based on simulation experiments. We employed the CrazySwarm package [48], which is a comprehensive quadrotor autonomous research testbed that integrates hardware and software components. It enables seamless transition from simulation to real-world deployment [49]. We implemented all algorithms in Python 3.9 and ran all simulations on MAC studio M1 Ultra with 128 GB RAM. Our system adopted Crazyflie as the UAV model and CrazySwarm as the simulation platform; therefore, we selected an appropriate sensor for the mission and UAV and tuned the parameters to match realistic conditions. Following previous literature [50], we used Crazyflie and an ultrasonic sensor (LV-MaxSonar-EZ2), which can detect objects within a range of 0–254 inches (6.75 m). Figure 8 illustrates the Crazyflie UAV, and Figure 9 depicts the ultrasonic sensor. We also considered the specifications of the Crazyflie when estimating the energy consumption of the UAVs [51,52]. Table 2 summarizes the values of the parameters related to UAV energy.



Figure 8. Crazyflie UAV.



Figure 9. Ultrasonic sensor (LV-MaxSonar-EZ2).

Table 2. Parameters related to energy.

Parameter	Value
Speed	1 m/s
δ_m	2430 J
μ	5.8 J/s

To evaluate the inspection mission, a template of the fire sprinkler system RCP using the Edrawmax tool was used as an input map to mimic a realistic scenario as much as possible [53]. The input map is shown in Figure 10 and is handled as an occupancy matrix after preprocessing (as per Figure 11) it to form a 500×500 grid with cell size $0.5 \text{ m} \times 0.5 \text{ m}$.

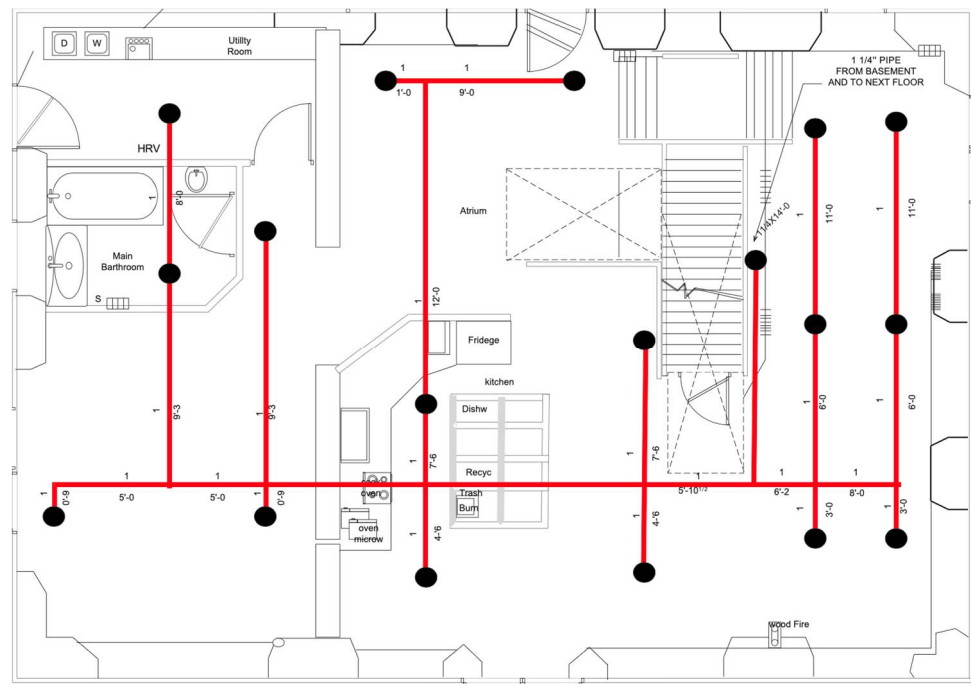


Figure 10. Input map of the pipe network.

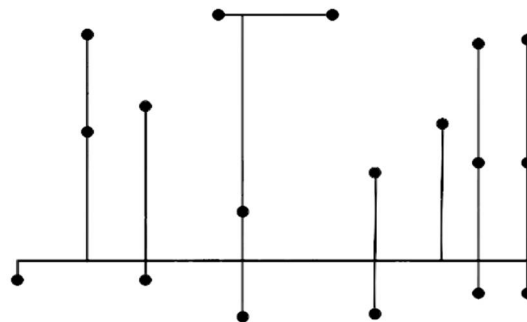


Figure 11. Input map after preprocessing.

The network locations were used as inspection points, and the defect locations were randomized within the inspection points to ensure representative samples in all the experiments. Furthermore, the defect locations were generated randomly using multivariate normal distributions to simulate hotspots within the specified radius where defects clustered together and were more likely to be found.

The controlled parameters in the simulation were as follows:

1. The number of UAVs varied between 2, 4, 6, 8, 10, 12, 14, and 16.
2. The number of defects (including hotspots and defect concentration within a single hotspot). The values of the different severity levels of the defects are shown in Table 3.

Table 3. Values of the different severity levels of the defects.

Severity Level	Number of Hotspots	Radius of Hotspot	Number of Defects
Simple	3	30	10
Average	9	30	20
Advanced	27	30	30

To evaluate the performance of our proposed approach, we adapted the performance metrics from recent studies [54], such as total travel distance (cost/fitness value), maximum

tour length, running time, mean detection time, and average consumed energy. We developed 120 scenarios by varying the number of heuristics (5), the number of UAVs (8), and the severity levels of defects (3). Each scenario was run 30 times to reduce the variability in the performance measures [54].

We used four well-established benchmark algorithms: ACO, PSO, OTA, and the random algorithm. The ACO algorithm was based on Dorigo's original paper [55] and followed the parameters listed in Table 4. The termination conditions for ACO were either no improvement in 30 iterations or reaching 1000 iterations [54].

Table 4. Parameter settings of the ACO algorithm.

Parameter	Value
α	1
β	5
ρ	0.5
Q	1
Number of ants	U
Maximum iterations	1000

For the second benchmark, we implemented a discrete version of PSO following the approach in [56] that represents the solutions (paths) as vectors. The cost is the total distance of the path of all UAVs (the vectors). We computed this value using a distance matrix saved for all locations. The number of particles is equal to the number of UAVs, according to literature studies [57]. However, in each particle, the local solution is composed of sub-vectors, each corresponding to one UAV. The sub-vectors do not have conflicts between them, and each network location is visited once in one and only one sub-vector. This ensures that if a location has a defect, it will be detected by only one UAV. We used the same suggested values for the algorithm parameters as in [56]. Furthermore, we followed the same termination conditions as the ACO. The PSO algorithm parameters are shown in Table 5.

Table 5. Parameters of PSO algorithm.

Parameter	Value
α	A random number between 0 and 1
β	$1 - \alpha$
Number of population	U
Maximum iterations	1000

The third algorithm implements an OTA strategy [58], instructing an agent to search for its nearest unexplored cell in the zone. The final algorithm implements a random choosing strategy [35], instructing an agent to search for its nearest unexplored cell in the zone. The OTA and random algorithms were utilized as baseline performance metrics.

Table 6 shows the particular parameters of the booby system. The value of the threshold X of the primary UAV algorithm was chosen empirically. Furthermore, we used k-means as the clustering algorithm. We found the optimal number of the cluster after plotting all possible k values against their internal evaluation indicators and choosing the best one [59].

Table 6. Booby parameter values.

Parameter	Value
X	70%
k	7

5.2. Results and Discussion

We conducted each experiment at least 30 times and plotted the results using a linear scale for the number of UAVs on the x-axis and a logarithmic base-2 scale for the performance metrics on the y-axis. The performance metrics included mean detection time, total traveled distance, maximum tour length, running time, and average energy used. We calculated these metrics for all benchmarks except for PSO and mean detection time. PSO does not support time inference because it defines the UAV paths as vectors during the initialization phase. Therefore, we cannot determine when a defect was found. All UAVs had a uniform battery energy level of δ_m and consumed energy at a constant rate for all movements.

5.2.1. Mean Detection Time

The mean detection time of a defect is an important metric to evaluate the performance of different algorithms for UAV-based inspection. It measures how long it takes for any UAV to detect a defect from the start of the simulation. We computed this value for each defect in each scenario and averaged it over multiple runs of each experiment. Figure 12 shows the results for our booby algorithm and three other benchmarks. The booby algorithm consistently outperformed the other algorithms in all settings, reducing the mean detection time by at least 13% compared to the random algorithm, which had the shortest running time among the benchmarks. This indicates that our approach can efficiently detect defects regardless of their number and distribution complexity.

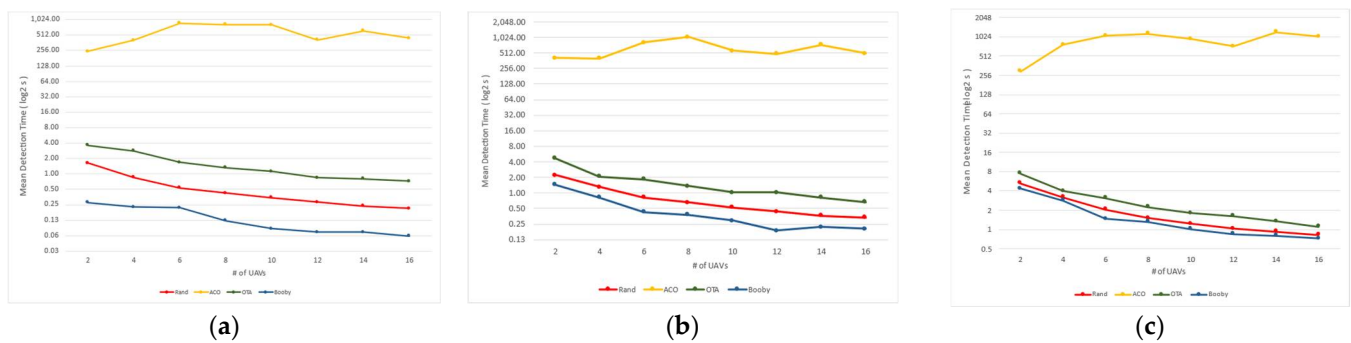


Figure 12. Mean detection time in different settings: (a) simple severity (3×10 defects); (b) average severity (9×20 defects); (c) advanced severity (27×30 defects).

According to a thorough analysis, the booby algorithm outperforms other algorithms for several reasons. First, it clusters the map into a well-chosen number of zones based on the locations of defects. Second, it assigns UAVs to the zones in a way that considers their size and priority. Specifically, it allocates more UAVs to larger zones sooner than smaller zones, as larger zones may have more defects. Third, it inspects the zones efficiently and adapts its behavior when a defect is found. For instance, the secondary UAV prioritizes joining a zone that has more defects detected by another UAV. These reasons enable the booby system to discover defects faster than other algorithms. As the number of UAVs increased, the booby, OTA, and random algorithms demonstrated a decrease in mean detection time. In contrast, the ACO performance is much worse than random and OTA. The ACO’s mean detection time increases with the number of UAVs due to its solution construction method. Each ant updates its tour with an amount inversely proportional to the distance of the tour. As a result, more ants follow shorter paths with more pheromones and neglect remote areas, which delays most defect detections. Furthermore, the ACO algorithm iterates through many paths before finding the solution, which negatively affects the mean detection time.

5.2.2. Total Traveled Distance (Cost)

Figure 13 shows that the booby algorithm produced tours with lower costs than the benchmarks, especially when more than two UAVs were involved. The random algorithm performed poorly as expected due to its naive nature. PSO also had low performance in this problem because it requires continuous solution values for its velocity update, but TSP is a discrete problem [60–63]. Clerc’s study in 2004 was one of the first to propose discrete PSO and it is still widely cited in the recent literature [64]. However, Clerc himself reported that discrete PSO was not as efficient as other algorithms [65]. Similarly, recent studies have indicated that PSO is more suitable for continuous optimization problems [63].

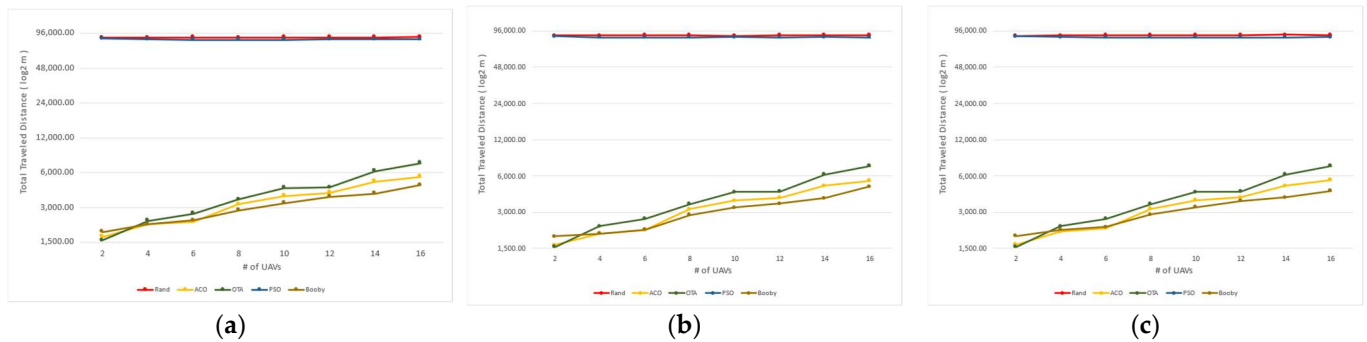


Figure 13. Total traveled distance in different settings: (a) simple severity (3×10 defects); (b) average severity (9×20 defects); (c) advanced severity (27×30 defects).

The metric cost analysis revealed that the booby algorithm had similar performance to OTA and ACO when $U = 2$. In these cases, there were fewer UAVs than clusters. This resulted in a limited number of primary UAVs initially and increased travel costs for the primary UAV later as it had to visit many available zones. Likewise, each zone was assigned a certain number of UAVs. This affected the total cost of each UAV because it had to cover most of the zone by itself.

5.2.3. Running Time

We measured the running time of each algorithm for every scenario by recording the time when the execution started and ended. Figure 14 shows the results, which indicate that booby outperformed all benchmarks in terms of speed. It solved all scenario instances faster than the other algorithms, with an average speedup of at least three times over random in simple settings (Figure 14a), three times over random in average settings (Figure 14b), and 1.2 times over random in advanced settings (Figure 14c). Moreover, booby’s running time did not increase exponentially as the number of UAVs increased, unlike PSO and ACO. This can be attributed to its efficient use of data structures such as k-d tree and dictionaries, as well as its ability to find a “good” solution within one iteration, unlike metaheuristic algorithms that require multiple iterations to converge.

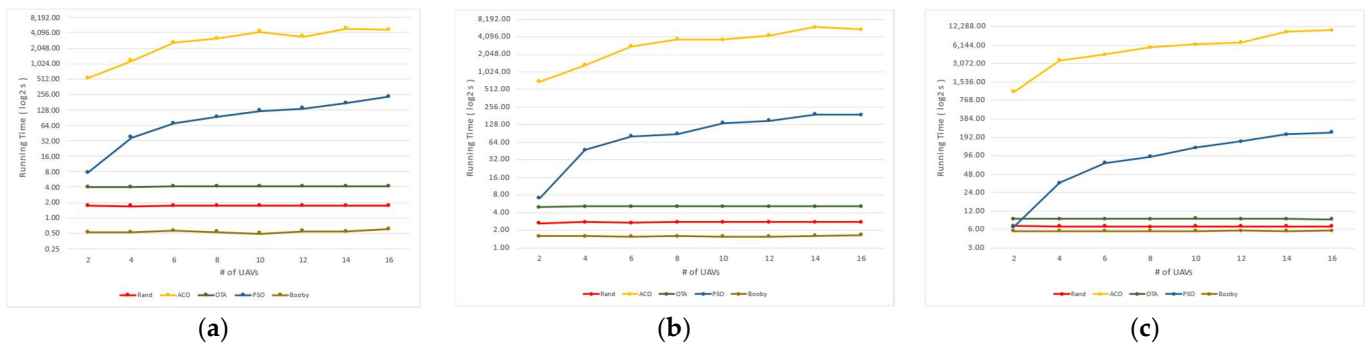


Figure 14. Running time in distance in different settings: (a) simple severity (3×10 defects); (b) average se-verity (9×20 defects); (c) complex severity (27×30 defects).

5.2.4. Maximum Tour Length

The maximum tour length is an important metric for evaluating the performance of UAV inspection algorithms. It measures the longest distance any UAV travels after completing its assigned mission. To compare our approach with OTA and ACO, we calculated the average maximum tour length for all scenarios based on each algorithm’s results. Figure 15 shows that our approach consistently achieved the lowest average maximum tour length while maintaining a low running time and minimizing the total traveled distance. To further illustrate this advantage, we also compared the maximum tour length of each scenario for our approach and OTA, which had a tradeoff between this metric and the running time. Table 7 displays the percentage difference between these two algorithms, as well as their time ratio. The data indicates that our approach not only reduced the maximum tour length by more than 10% in most scenarios (highlighted in the table), but also enhanced the runtime in all cases. The main reason for this improvement is that our approach assigns distinct roles to UAVs and uses efficient data structures to select candidate zones based on proximity and availability. Finally, our approach requests a temporary UAV to assist in inspecting a zone only if a certain percentage of the zone remains uninspected after a certain amount of time has elapsed. This way, our approach ensures that all zones are inspected thoroughly and quickly by utilizing UAVs optimally.

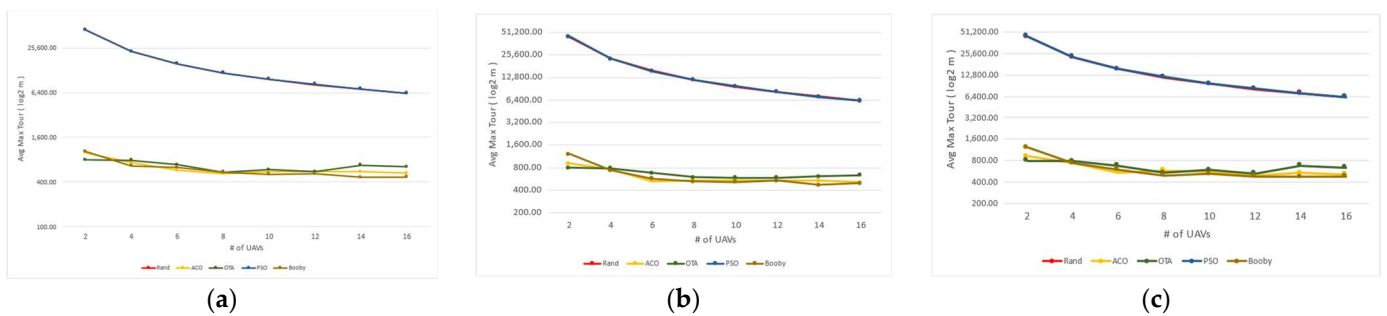


Figure 15. Average maximum tour length in different settings: (a) simple severity (3×10 defects); (b) average severity (9×20 defects); (c) complex severity (27×30 defects).

Table 7. Maximum tour length for booby versus OTA.

Defect No.	U	% Diff. in Length	Time Ratio
3 × 10	2	19.1	8
	4	−6.7	8
	6	−12.6	8
	8	−19.5	8
	10	−26.2	9
	12	−16.6	8
	14	−36.1	8
9 × 20	2	22	4
	4	−13.5	4
	6	−19.2	4
	8	−19.1	4
	10	−26.2	4
	12	−20.7	4
	14	−36.5	4
27 × 30	2	23.4	2
	4	−6.7	2
	6	−14.7	2
	8	−18.3	2
	10	−25.4	2
	12	−16.1	2
	14	−35.5	2
16	−38	2	

5.2.5. UAV’s Average Consumed Energy

Initially, the energy levels of all UAVs are identical. As each UAV navigates to a new position, it consumes some energy for this movement. The average energy consumption is the ratio of the total energy consumption to the number of UAVs at the end of the execution. Figure 16 shows how the average energy levels vary with different numbers of UAVs. There is a negative correlation between these two variables: more UAVs mean less average energy consumption. This trend is evident in the PSO and random results, but not in other benchmarks (OTA and ACO). A possible explanation is that adding more UAVs in PSO and random algorithms reduces some long tours taken by individual UAVs, which require more energy. However, such a reduction is not observed in other algorithms because they already favor shorter tours over longer ones regardless of the number of UAVs. This result confirms that PSO and random algorithms have lower average maximum tour lengths when more UAVs are added. Moreover, our approach has comparable results to OTA and ACO when there are few UAVs (U = 2, 4, and 6). However, our approach is more scalable because it minimizes the average energy consumption as more UAVs are added.

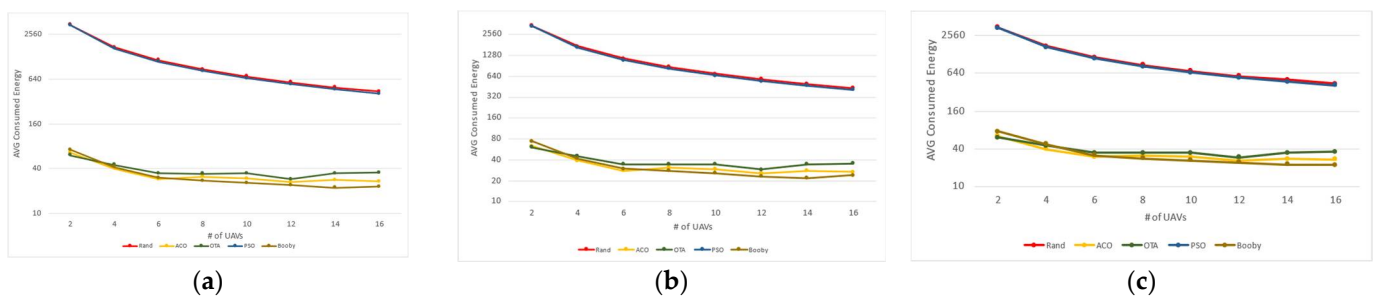


Figure 16. Average consumed energy in different settings: (a) simple severity (3 × 10 defects); (b) average se-verity (9 × 20 defects); (c) advanced severity (27 × 30 defects).

6. Conclusions

In this paper, we propose a multi-UAV path planning method that is based on the behavior of the booby. In our system, the UAVs are automatically divided into different roles and assigned to different zones on the map. Later, the UAVs can independently switch roles to maximize the system's objective by minimizing the total distance traveled by the UAVs. We used ACO, PSO, OTA, and random, which are all well-known and competing path planning algorithms, and turned them into multi-UAV path planning algorithms to compare with our approach. In our experiments, we controlled the number of UAVs and defects. In addition, we measured different performance metrics, such as total traveled distance, mean detection time, the maximum tour length, running time, and average energy consumed. Our results show that the proposed method uses less energy and can efficiently find defects with a shorter travel distance and in less time. In the simple settings of severity levels, a 59% enhancement in mean detection over all benchmarks and at least a three-fold speed increase compared to the random algorithm (which has the shortest running time of all benchmarks) were achieved. However, in the advanced settings of severity levels, a 13% enhancement in mean detection over all benchmarks and a 1.2-fold faster running time than the random algorithm (which has the shortest running time of all benchmarks) were achieved. In addition, the mean detection time of our approach decreased with the addition of UAVs. Our algorithm has been shown to be scalable for different numbers of UAVs and severity levels of defects, and it outperforms all the benchmarks significantly when increasing the number of UAVs.

In future research, it would be beneficial to consider environmental conditions in the optimization process as well as different environment maps. Additionally, defects can be classified into risk levels, and UAVs can be instructed to take appropriate actions depending on the defect's risk level. Furthermore, it would be interesting to extend the booby-inspired multi-UAV path planning method to more complex and realistic scenarios, such as dynamic environments with moving obstacles or targets, heterogeneous UAVs with different capabilities and constraints, and uncertain communication and sensing models. Another possible direction is to investigate the theoretical properties and guarantees of the proposed method, such as optimality, robustness, and scalability. Finally, a practical validation of the proposed method using real UAVs in an industrial inspection setting would be desirable to demonstrate its feasibility and effectiveness in real-world applications.

Author Contributions: Conceptualization, F.A. and H.K.; Formal analysis, F.A.; Funding acquisition, H.K.; Investigation, H.K. and K.Y.-T.; Methodology, F.A., H.K. and K.Y.-T.; Software, F.A.; Supervision, H.K. and K.Y.-T.; Validation, F.A. and H.K.; Visualization, F.A.; Writing—original draft, F.A.; Writing—review and editing, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the International Scientific Partnership Program ISPP (ISPP-119) at King Saud University.

Data Availability Statement: Not applicable.

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

References

1. Aggarwal, S.; Kumar, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Comput. Commun.* **2020**, *149*, 270–299. [[CrossRef](#)]
2. Shakhathreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634. [[CrossRef](#)]
3. Skorobogatov, G.; Barrado, C.; Salamí, E. Multiple UAV Systems: A Survey. *Unmanned Syst.* **2020**, *8*, 149–169. [[CrossRef](#)]
4. Hassanalian, M.; Abdelkefi, A. Classifications, applications, and design challenges of drones: A review. *Prog. Aerosp. Sci.* **2017**, *91*, 99–131. [[CrossRef](#)]
5. Ahmed, F.; Mohanta, J.C.; Keshari, A.; Yadav, P.S. Recent Advances in Unmanned Aerial Vehicles: A Review. *Arab. J. Sci. Eng.* **2022**, *47*, 7963–7984. [[CrossRef](#)]

6. Civil Defense Safety Conditions, Means of Prevention, and Alarm and Extinguishing Equipment That Must Be Available in Hotels, Youth Hostels and Similar Establishments. Available online: <https://gdcd.998.gov.sa/Ar/CivilDefenseLists/Documents/22.pdf> (accessed on 30 October 2021).
7. Indoor Drone Surveys for Confined and Enclosed Indoor Spaces. *Baltimore Inspection Services*. Available online: <https://baltimoreuav.co.uk/indoor-drone-confined-spaces/> (accessed on 4 January 2023).
8. AERIAL INSPECTIONS. Stature Films. Available online: <https://www.staturefilms.com/drone-inspections> (accessed on 4 January 2023).
9. Maruschak, P.; Prentkovskis, O.; Bishchak, R. Defectiveness of external and internal surfaces of the main oil and gas pipelines after long-term operation. *J. Civ. Eng. Manag.* **2016**, *22*, 279–286. [[CrossRef](#)]
10. Al-Moubaraki, A.H.; Obot, I.B. Top of the line corrosion: Causes, mechanisms, and mitigation using corrosion inhibitors. *Arab. J. Chem.* **2021**, *14*, 103116. [[CrossRef](#)]
11. Popescu, C.; Gabor, M.R. Quantitative Analysis Regarding the Incidents to the Pipelines of Petroleum Products for an Efficient Use of the Specific Transportation Infrastructure. *Processes* **2021**, *9*, 1535. [[CrossRef](#)]
12. Panossian, Z.; de Almeida, N.L.; de Sousa, R.M.F.; Pimenta, G.d.S.; Marques, L.B.S. Corrosion of carbon steel pipes and tanks by concentrated sulfuric acid: A review. *Corros. Sci.* **2012**, *58*, 1–11. [[CrossRef](#)]
13. Hassan, N.S. The Effect of Different Operating Parameters on the Corrosion Rate of Carbon Steel in Petroleum Fractions. *Eng. Technol. J.* **2013**, *31*, 1182–1193.
14. Baker, M.; Fessler, R.R. *Pipeline Corrosion*; U.S. Department of Transportation: Washington, DC, USA, 2008.
15. Raja, P. Optimal path planning of mobile robots: A review. *Int. J. Phys. Sci.* **2012**, *7*, 1314–1320. [[CrossRef](#)]
16. Bono, A.; D’Alfonso, L.; Fedele, G.; Filice, A.; Natalizio, E. Path Planning and Control of a UAV Fleet in Bridge Management Systems. *Remote Sens.* **2022**, *14*, 1858. [[CrossRef](#)]
17. Boussaïd, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [[CrossRef](#)]
18. Siddique, N.; Adeli, H. Nature Inspired Computing: An Overview and Some Future Directions. *Cogn. Comput.* **2015**, *7*, 706–714. [[CrossRef](#)]
19. Kurdi, H.A.; Aloboud, E.; Alalwan, M.; Alhassan, S.; Alotaibi, E.; Bautista, G.; How, J.P. Autonomous task allocation for multi-UAV systems based on the locust elastic behavior. *Appl. Soft Comput.* **2018**, *71*, 110–126. [[CrossRef](#)]
20. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [[CrossRef](#)]
21. Lamini, C.; Benhlime, S.; Elbekri, A. Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. *Procedia Comput. Sci.* **2018**, *127*, 180–189. [[CrossRef](#)]
22. Li, K.; Ge, F.; Han, Y.; Wang, Y.; Xu, W. Path planning of multiple UAVs with online changing tasks by an ORPFOA algorithm. *Eng. Appl. Artif. Intell.* **2020**, *94*, 103807. [[CrossRef](#)]
23. Worley, R.; Ma, K.; Sailor, G.; Schirru, M.M.; Dwyer-Joyce, R.; Boxall, J.; Dodd, T.; Collins, R.; Anderson, S. Robot Localization in Water Pipes Using Acoustic Signals and Pose Graph Optimization. *Sensors* **2020**, *20*, 5584. [[CrossRef](#)]
24. Quenzel, J.; Nieuwenhuisen, M.; Droeschel, D.; Beul, M.; Houben, S.; Behnke, S. Autonomous MAV-based Indoor Chimney Inspection with 3D Laser Localization and Textured Surface Reconstruction. *J. Intell. Robot. Syst.* **2019**, *93*, 317–335. [[CrossRef](#)]
25. Máthé, K.; Buşoniu, L. Vision and Control for UAVs: A Survey of General Methods and of Inexpensive Platforms for Infrastructure Inspection. *Sensors* **2015**, *15*, 14887–14916. [[CrossRef](#)] [[PubMed](#)]
26. Guan, S.; Zhu, Z.; Wang, G. A Review on UAV-Based Remote Sensing Technologies for Construction and Civil Applications. *Drones* **2022**, *6*, 117. [[CrossRef](#)]
27. Zhou, H.; Xu, C.; Tang, X.; Wang, S.; Zhang, Z. A Review of Vision-Laser-Based Civil Infrastructure Inspection and Monitoring. *Sensors* **2022**, *22*, 5882. [[CrossRef](#)] [[PubMed](#)]
28. Hamledari, H.; Sajedi, S.; McCabe, B.; Fischer, M. Automation of Inspection Mission Planning Using 4D BIMs and in Support of Unmanned Aerial Vehicle-Based Data Collection. *J. Constr. Eng. Manag.* **2021**, *147*, 04020179. [[CrossRef](#)]
29. Han, J.; Yang, Z.; Zhang, Q.; Chen, C.; Li, H.; Lai, S.; Hu, G.; Xu, C.; Xu, H.; Wang, D.; et al. A Method of Insulator Faults Detection in Aerial Images for High-Voltage Transmission Lines Inspection. *Appl. Sci.* **2019**, *9*, 2009. [[CrossRef](#)]
30. Liu, Y.; Shi, J.; Liu, Z.; Huang, J.; Zhou, T. Two-Layer Routing for High-Voltage Powerline Inspection by Cooperated Ground Vehicle and Drone. *Energies* **2019**, *12*, 1385. [[CrossRef](#)]
31. Bolourian, N.; Hammad, A. LiDAR-equipped UAV path planning considering potential locations of defects for bridge inspection. *Autom. Constr.* **2020**, *117*, 103250. [[CrossRef](#)]
32. Ayele, Y.Z.; Aliyari, M.; Griffiths, D.; Droguett, E.L. Automatic Crack Segmentation for UAV-Assisted Bridge Inspection. *Energies* **2020**, *13*, 6250. [[CrossRef](#)]
33. Israr, A.; Ali, Z.A.; Alkhamash, E.H.; Jussila, J.J. Optimization Methods Applied to Motion Planning of Unmanned Aerial Vehicles: A Review. *Drones* **2022**, *6*, 126. [[CrossRef](#)]
34. Ait Saadi, A.; Soukane, A.; Meraihi, Y.; Benmessaoud Gabis, A.; Mirjalili, S.; Ramdane-Cherif, A. UAV Path Planning Using Optimization Approaches: A Survey. *Arch. Comput. Methods Eng.* **2022**, *29*, 4233–4284. [[CrossRef](#)]
35. Pan, Y.; Yang, Y.; Li, W. A Deep Learning Trained by Genetic Algorithm to Improve the Efficiency of Path Planning for Data Collection with Multi-UAV. *IEEE Access* **2021**, *9*, 7994–8005. [[CrossRef](#)]

36. Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Three dimensional path planning using Grey wolf optimizer for UAVs. *Appl. Intell.* **2019**, *49*, 2201–2217. [[CrossRef](#)]
37. Yan, F.; Zhu, X.; Zhou, Z.; Chu, J. A Hierarchical Mission Planning Method for Simultaneous Arrival of Multi-UAV Coalition. *Appl. Sci.* **2019**, *9*, 1986. [[CrossRef](#)]
38. Yang, L.; Guo, J.; Liu, Y. Three-Dimensional Uav Cooperative Path Planning Based on the Mp-Cgwo Algorithm. *Int. J. Innov. Comp. Inf. Control* **2020**, *16*, 991–1006. [[CrossRef](#)]
39. Ahmed, N.; Pawase, C.J.; Chang, K. Distributed 3-D Path Planning for Multi-UAVs with Full Area Surveillance Based on Particle Swarm Optimization. *Appl. Sci.* **2021**, *11*, 3417. [[CrossRef](#)]
40. Teng, H.; Ahmad, I.; Alamgir, M.S.M.; Chang, K. 3D Optimal Surveillance Trajectory Planning for Multiple UAVs by Using Particle Swarm Optimization with Surveillance Area Priority. *IEEE Access* **2020**, *8*, 86316–86327. [[CrossRef](#)]
41. Sommerfeld, J.; Kato, A.; Ropert-Coudert, Y.; Garthe, S.; Wilcox, C.; Hindell, M.A. Flexible foraging behaviour in a marine predator, the Masked booby (*Sula dactylatra*), according to foraging locations and environmental conditions. *J. Exp. Mar. Biol. Ecol.* **2015**, *463*, 79–86. [[CrossRef](#)]
42. Sommerfeld, J.; Kato, A.; Ropert-Coudert, Y.; Garthe, S.; Hindell, M.A. Foraging Parameters Influencing the Detection and Interpretation of Area-Restricted Search Behaviour in Marine Predators: A Case Study with the Masked Booby. *PLoS ONE* **2013**, *8*, e63742. [[CrossRef](#)]
43. Schreiber, E.A.; Burger, J. (Eds.) *Biology of Marine Birds*; CRC Marine Biology Series; CRC Press: Boca Raton, FL, USA, 2002; ISBN 978-0-8493-9882-7.
44. Weimerskirch, H.; Le Corre, M.; Jaquemet, S.; Marsac, F. Foraging strategy of a tropical seabird, the red-footed booby, in a dynamic marine environment. *Mar. Ecol. Prog. Ser.* **2005**, *288*, 251–261. [[CrossRef](#)]
45. Weimerskirch, H.; Le Corre, M.; Bost, C. Foraging strategy of masked boobies from the largest colony in the world: Relationship to environmental conditions and fisheries. *Mar. Ecol. Prog. Ser.* **2008**, *362*, 291–302. [[CrossRef](#)]
46. Bairos-Novak, K.R.; Crook, K.A.; Davoren, G.K. Relative importance of local enhancement as a search strategy for breeding seabirds: An experimental approach. *Anim. Behav.* **2015**, *106*, 71–78. [[CrossRef](#)]
47. Patro, S.G.K.; Sahu, K.K. Normalization: A Preprocessing Stage. *Int. Adv. Res. J. Sci. Eng. Technol.* **2015**, 20–22. [[CrossRef](#)]
48. Preiss, J.A.; Honig, W.; Sukhatme, G.S.; Ayanian, N. CrazySwarm: A large nano-quadcopter swarm. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 3299–3304. [[CrossRef](#)]
49. Official CrazySwarm Tutorial. Available online: <https://crazyswarm.readthedocs.io/en/latest/tutorials/tutorials.html> (accessed on 12 November 2022).
50. Helland, J.; Whitaker, J.; Cowan, P.; Glass, S. *Autonomous Drone*; University of Utah Abstract: Salt Lake City, UT, USA, 2015; Available online: <https://my.ece.utah.edu/~kstevens/3992/reports/death-ray.pdf> (accessed on 23 November 2022).
51. Datasheet Crazyflie 2.1. Available online: https://www.bitcraze.io/documentation/hardware/crazyflie_2_1/crazyflie_2_1-datasheet.pdf (accessed on 22 February 2023).
52. Battery and Charger for Crazyflie 2.1 Drone. Available online: <https://www.generationrobots.com/en/403752-240-mah-battery-and-charger-for-crazyflie-21-drone.html> (accessed on 22 February 2023).
53. Anderson, L. Fire Sprinkler System Rcp. Available online: <https://www.edrawmax.com/templates/1021321/> (accessed on 15 December 2022).
54. Chen, X.; Zhang, P.; Du, G.; Li, F. Ant Colony Optimization Based Memetic Algorithm to Solve Bi-Objective Multiple Traveling Salesmen Problem for Multi-Robot Systems. *IEEE Access* **2018**, *6*, 21745–21757. [[CrossRef](#)]
55. Dorigo, M.; Maniezzo, V.; Coloni, A. The Ant System: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern.* **1992**, *26*, 1–13. [[CrossRef](#)]
56. Wang, K.-P.; Huang, L.; Zhou, C.-G.; Pang, W. Particle swarm optimization for traveling salesman problem. In Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693), Xi'an, China, 5 November 2003; IEEE: Piscataway, NJ, USA, 2003; pp. 1583–1585. [[CrossRef](#)]
57. Ziyang, Z.; Dongjing, X.; Chen, G. Cooperative search-attack mission planning for multi-UAV based on intelligent self-organized algorithm. *Aerosp. Sci. Technol.* **2018**, *76*, 402–411. [[CrossRef](#)]
58. Kurdi, H.; How, J.; Bautista, G. Bio-Inspired Algorithm for Task Allocation in Multi-UAV Search and Rescue Missions. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Diego, CA, USA, 4–8 January 2016; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2016. [[CrossRef](#)]
59. Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. Understanding of Internal Clustering Validation Measures. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 911–916. [[CrossRef](#)]
60. Strasser, S.; Goodman, R.; Sheppard, J.; Butcher, S. A New Discrete Particle Swarm Optimization Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, Denver, CO, USA, 20–24 July 2016; ACM: Rochester, NY, USA, 2016; pp. 53–60. [[CrossRef](#)]
61. Goldberg, E.F.; Goldberg, M.C.; de Souza, G.R. *Particle Swarm Optimization Algorithm for the Traveling Salesman Problem*; INTECH Open Access Publisher: London, UK, 2008; ISBN 978-953-7619-10-7.

62. Hoffmann, M.; Muhenthaler, M.; Helwig, S.; Wanka, R. Discrete Particle Swarm Optimization for TSP: Theoretical Results and Experimental Evaluations. In *Adaptive and Intelligent Systems*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 416–427.
63. Moghtadernejad, S.; Adey, B.T.; Hackl, J. Prioritizing Road Network Restorative Interventions Using a Discrete Particle Swarm Optimization. *J. Infrastruct. Syst.* **2022**, *28*, 04022039. [[CrossRef](#)]
64. Strak, Ł.; Skinderowicz, R.; Boryczka, U.; Nowakowski, A. A Self-Adaptive Discrete PSO Algorithm with Heterogeneous Parameter Values for Dynamic TSP. *Entropy* **2019**, *21*, 738. [[CrossRef](#)]
65. Clerc, M. Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem. In *New Optimization Techniques in Engineering*; Springer: Berlin/Heidelberg, Germany, 2004; p. 22.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.