mathematics

MDPI

Article

Hybrid Vulture-Coordinated Multi-Robot Exploration: A Novel Algorithm for Optimization of Multi-Robot Exploration

Ali El Romeh [1], Seyedali Mirjalili [1,2,3,*] and Faiza Gul [4]

[1] Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Brisbane 4006, Australia; ali.romeh@student.torrens.edu.au
[2] Yonsei Frontier Lab, Yonsei University, Seoul 03722, Republic of Korea
[3] University Research and Innovation Center, Obuda University, 1034 Budapest, Hungary
[4] Department of Electrical Engineering, Air University, Aerospace & Aviation Campus KAMRA, Islamabad 43600, Pakistan
* Correspondence: ali.mirjalili@torrens.edu.au

Abstract: Exploring unknown environments using multiple robots has numerous applications in various fields but remains a challenging task. This study proposes a novel hybrid optimization method called Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME), which combines Coordinated Multi-Robot Exploration (CME) and African Vultures Optimization Algorithm (AVOA) to optimize the construction of a finite map in multi-robot exploration. We compared HVCME with four other similar algorithms using three performance measures: run time, percentage of the explored area, and the number of times the method failed to complete a run. The experimental results show that HVCME outperforms the other four methods, demonstrating its effectiveness in optimizing the construction of a finite map in an unknown indoor environment.

Keywords: Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME); optimization; African Vulture Optimization Algorithm (AVOA); multi-robot exploration; finite map; Coordinated Multi-Robot Exploration (CME); unknown environments; path planning

MSC: 68T20

Citation: Romeh, A.E.; Mirjalili, S.; Gul, F. Hybrid Vulture-Coordinated Multi-Robot Exploration: A Novel Algorithm for Optimization of Multi-Robot Exploration. *Mathematics* **2023**, *11*, 2474. https://doi.org/10.3390/math11112474

Academic Editor: Simeon Reich

Received: 19 April 2023
Revised: 18 May 2023
Accepted: 22 May 2023
Published: 27 May 2023

1. Introduction

The exploration of unknown environments by a group of robots is a challenging task that has numerous applications in various fields, including search and rescue operations [1,2], surveillance [3], agriculture [4], environmental monitoring [5,6], mining [7,8], manufacturing [9], and space exploration [10,11]. Multi-robot exploration can significantly improve the efficiency and accuracy of exploring large and complex environments [12,13]. Prior methods for multi-robot exploration were based on either deterministic or meta-heuristic algorithms [14,15]. However, there is limited research that combines the benefits of both techniques.

In the current research, we introduce an innovative hybrid optimization technique, Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME), which seamlessly integrates the Coordinated Multi-Robot Exploration (CME) methodology [16] and the African Vulture Optimization Algorithm (AVOA) [17]. This novel approach aims to optimize the construction of finite maps during multi-robot exploration missions. HVCME effectively addresses the limitations of previous methods [18–22], such as inefficiency and incomplete exploration, by prioritizing adjacent cells around a robot and employing the foraging behavior of African vultures for search space optimization. Consequently, this leads to a substantial enhancement in exploration efficiency and accuracy, proving beneficial for applications in search and rescue operations, surveillance, and environmental monitoring. To evaluate the performance of HVCME, we conducted a comparative analysis with four other

algorithms integrated with CME, including Grey Wolf Optimization (GWO) [23], Salp Swarm Algorithm (SSA) [24], Mountain Gazelle Optimizer (MGO) [25], and Sine Cosine Algorithm (SCA) [26]. The assessment employed three key performance metrics: execution time, the proportion of the explored region, and method failure frequency to conclude a run, further highlighting HVCME's advantages in multi-robot exploration scenarios.

HVCME can be employed in various space exploration applications. In search and rescue operations, a group of robots can explore an area more efficiently and quickly than a single robot, increasing the chances of finding missing persons or identifying hazardous areas. In surveillance, robots can be used to monitor large areas and provide real-time feedback to a control center, reducing the need for human intervention. In environmental monitoring, robots can be used to collect data from remote or hazardous locations, providing valuable information for scientific research and resource management, and many more.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work in the field of multi-robot exploration and optimization. Section 3 explains the proposed HVCME algorithm in detail. Section 4 presents the experimental setup and results. Finally, Section 5 concludes the paper and discusses future research directions.

2. Related Work

Multi-robot exploration has garnered significant attention in contemporary times owing to its practical relevance across diverse domains [2–5]. In real-world deployment scenarios, such as exploring uncharted locations, human–robot teams provide numerous advantages. Multi-robot systems operated by a human operator may reach places that humans cannot, such as other planets or underwater, and can cover larger regions more effectively. As a result, how to efficiently develop and implement such systems is swiftly becoming a topic of study.

In such situations, swarm robots outperform single-robot systems, which are incapable of covering wide areas and present a key single point of failure for the mission. Although centralized multi-robot management is one option, robotic swarms with decentralized control have been found to be more efficient. The difficulties in programming swarm behaviors have already been discussed in several studies. Swarm control algorithms, also known as "behaviors", are widely used, utilizing the processing power of all units together and significantly reducing the load on each robot. Furthermore, swarm robots rely on local interactions, both with their swarm neighbors and with their surroundings in the environment, which makes them more resilient to fluctuating mission circumstances [27].

Machine learning has recognized bio-inspired optimization algorithms in recent years as a way to address optimal solutions to complicated scientific and technical challenges with time and space constraints. These challenges are fundamentally nonlinear and are frequently constrained by path or terminal limitations [28]. The new trend is to use bio-inspired optimization algorithms, which offer a promising way to solve problems that standard optimization algorithms cannot handle efficiently.

Research in this field can be generally categorized into two main streams: deterministic algorithms and meta-heuristic algorithms. Deterministic algorithms are based on predefined rules that dictate the actions of robots, while meta-heuristic algorithms utilize search-based strategies to efficiently explore the environment. Unlike deterministic approaches, meta-heuristic algorithms draw inspiration from natural processes to optimize their search strategy.

2.1. Deterministic Methods

The field of robotics has recently shown considerable interest in multi-robot exploration due to its relevance in various practical applications. To address this, a method has been proposed for exploring such environments with multiple robots, which takes into account the trade-off between the cost of reaching a target location and its corresponding usefulness. This approach facilitates the allocation of appropriate targets to robots, allowing

them to simultaneously explore different regions of the environment, especially in cases where communication ranges are limited. The effectiveness of the algorithm was assessed through experiments and simulations, which demonstrated its efficacy in efficiently distributing the robots throughout the environment and accomplishing their mission [16]. The paper contributes to the growing body of research on multi-robot coordination for exploration tasks.

One of the earliest deterministic algorithms for multi-robot exploration is the coverage algorithm proposed by Galceran and Carreras [29]. The algorithm divides the environment into cells and assigns each robot to a cell. The robots explore their respective cells, and once they have completed their task, they move on to an adjacent unexplored cell. The algorithm was shown to be effective in small-scale environments, but its performance deteriorates in larger and more complex environments. Another deterministic algorithm for multi-robot exploration is the sweep algorithm proposed by Wang and Syrmos [30]. The algorithm assigns robots to different areas of the environment, and the robots explore their respective areas in a sweep-like manner. The algorithm was shown to be effective in environments with few obstacles, but its performance deteriorates in environments with more obstacles.

Andries and Charpillet [31] propose a new taboo-list approach for the multi-robot exploration of unknown structured environments that utilizes a distributed exploration algorithm, without being guided by frontiers, to guide agents on a globally shared map. The algorithm incorporates features such as robot perspective vision, variable vision range, and optimization to prevent agents from prematurely gathering at the rendezvous point. The performance of the algorithm is assessed via simulation using standardized maps.

Overall, deterministic approaches have proven to be effective in certain environments for exploration tasks. However, these approaches have a tendency to become trapped in local optima and repeat the same patterns, which can limit the efficiency of the exploration process. Unfortunately, changing the environment, such as the map, is not always a viable solution. Therefore, researchers have explored alternative approaches that involve randomized decision-making and distributed coordination among multiple robots, which can enhance the adaptability and scalability of the exploration process. These approaches have shown promising results in various scenarios, highlighting the importance of considering both deterministic and stochastic approaches in exploring unknown environments with multiple robots.

2.2. Metaheuristic Methods

Meta-heuristic algorithms have become increasingly popular in recent years due to their effectiveness in solving complex optimization problems. In this literature review, we explore several meta-heuristic algorithms: Grey Wolf Optimizer (GWO), Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), Mountain Gazelle Optimizer (MGO), Sine Cosine Algorithm (SCA), Practical Swarm Algorithm (PSO), Genetic Algorithm (GA), and African Vulture Optimization Algorithm (AVOA).

The Grey Wolf Optimizer Algorithm proposed by Mirjalili and Lewis [23] is inspired by the social hierarchy and hunting behavior of grey wolves. The algorithm uses four types of grey wolves to model a hierarchical organization and the three primary stages of a search process to optimize problems. The algorithm has been tested on various optimization functions and classical engineering design problems, and the results show that the GWO algorithm is highly competitive compared to other well-known meta-heuristics. This algorithm has also been applied in the field of optical engineering.

The Salp Swarm Algorithm [24] is a meta-heuristic algorithm inspired by the foraging behavior of salps in the ocean, designed for solving optimization problems with both single and multiple objectives. Its effectiveness has been tested on various mathematical optimization functions, which have demonstrated that the algorithm can converge effectively toward the optimal solution. Additionally, the Salp Swarm Algorithm has been utilized to solve complex engineering design problems that require significant computational resources.

The Mountain Gazelle Optimizer [25] is a novel algorithm that takes cues from the social organization and hierarchy of wild mountain gazelles. Rigorous evaluations and tests have been carried out on this method using diverse benchmark functions and engineering problems. The results of these analyses demonstrate that the MGO outperforms comparable algorithms on most benchmark functions, indicating the algorithm's excellent performance. Furthermore, the MGO's search capabilities remain robust, even when faced with optimization problems of higher dimensions, thus cementing its effectiveness and versatility.

The Sine Cosine Algorithm [26] is a recent optimization method that leverages a mathematical model rooted in sine and cosine functions to generate a variety of initial random candidate solutions. The algorithm has undergone testing on several established test cases, which demonstrate its ability to explore diverse regions of a search space, avoid local optima, converge towards the global optimum, and capitalize on promising areas of a search space during optimization. Additionally, the Sine Cosine Algorithm has been employed to optimize the cross-section of an aircraft's wing, highlighting its versatility and potential for real-world applications.

The Genetic Algorithm [32] is inspired by the process of natural selection and genetics. In this algorithm, a population of candidate solutions is evolved over successive generations through selection, crossover, and mutation. The algorithm has been applied to various optimization problems, such as function optimization, machine learning, and control system design. The GA has been found to be effective in finding optimal or near-optimal solutions in complex search spaces.

The Practical Swarm Algorithm [33] is a variant of the traditional Particle Swarm Optimization Algorithm that aims to improve the convergence and robustness of the algorithm. The PSO algorithm is inspired by the social behavior of bird flocking or fish schooling. The algorithm has been applied to several optimization problems, such as function optimization, parameter identification, and image processing. The Particle Swarm Optimization (PSO) Algorithm has demonstrated its efficacy in addressing complex optimization problems characterized by high dimensionality and non-linearity.

The Whale Optimization Algorithm [34] emulates the social behavior of humpback whales and their bubble-net hunting strategy. Extensive testing has been conducted on this algorithm using a range of optimization problems and structural design challenges, demonstrating its impressive competitiveness relative to both state-of-the-art meta-heuristic algorithms and conventional methods. Notably, the WOA algorithm has proven effective in tackling diverse real-world problems across multiple domains, including mechanical engineering, electrical engineering, and economics.

The African Vultures Optimization Algorithm proposed by Abdollahzadeh et al. [17], takes inspiration from the foraging and navigation behaviors of African vultures. This algorithm has undergone comprehensive testing on various benchmark functions and has been rigorously compared to several existing algorithms. The results of these tests reveal AVOA's superiority in identifying optimal solutions for a wide range of optimization problems, including both single and multiple-objective optimization. AVOA has been successfully applied to problems in various fields, such as mechanical engineering, electrical engineering, and economics, demonstrating its versatility and efficacy. Notably, the Wilcoxon rank sum test was used for statistical evaluation, revealing the AVOA's significant superiority at a 95% confidence interval.

Overall, a variety of meta-heuristic algorithms such as GA, PSO, GWO, SSA, MGO, SCA, WOA, and AVOA have shown great potential in addressing complex optimization problems. These algorithms have been applied to various applications, including robot exploration in challenging environments, to locate optimal or near-optimal solutions. The efficacy of these algorithms can be attributed to their ability to efficiently explore the search space, evade local optima, and converge towards the global optimum. Furthermore, these algorithms can handle diverse optimization problems, such as continuous, discrete, and mixed-integer optimization problems.

2.3. Hybrid Method

Several studies have explored the use of multi-robot systems for exploring unknown and cluttered spaces with the primary objective of efficient mapping and navigation. Previous studies have predominantly employed deterministic or meta-heuristic algorithms to optimize robot trajectories and minimize uncertainties. However, there is a lack of research on combining these techniques to consolidate their advantages and overcome their limitations.

Albina and Lee [35] proposed a hybrid algorithm that combines the Coordinated Multi-Robot Exploration Algorithm with the Grey Wolf Optimizer to optimize robot trajectories for exploration and mapping of the environment. Simulation results demonstrated that the hybrid algorithm outperformed the Coordinated Multi-Robot Exploration algorithm by enhancing the deterministic approach and achieving complete exploration and mapping of the environment. Another study by Gul et al. [36] proposed a new framework that combines the Coordinated Multi-Robot Exploration Algorithm with the Frequency Modified Hybrid Whale Optimization Algorithm to achieve optimal exploration and mapping of the environment. The proposed algorithm was found to outperform other contemporary optimization techniques.

Gul et al. [37] proposed a novel Aquila Optimization Algorithm for Multi-Robot space exploration in a barrier-filled environment. The proposed Coordinated Multi-Robot Exploration Aquila Optimizer (CME-AO) Algorithm demonstrated superior performance compared to contemporary algorithms such as conventional CME, CME Arithmetic Optimization Algorithm (CME-AOA), and Frequency Modified Hybrid Whale Optimization Algorithm (FMH-WOA). In another study, Gul et al. [38] introduced a Hybrid Stochastic Optimizer (HSO) that employs both deterministic CME and stochastic Arithmetic Optimization (AO) techniques for efficient multi-robot space exploration. The proposed algorithm is capable of enhancing the explored area and reducing the search time, leading to significant improvements in the exploration process.

Finally, Romeh and Mirjalili [39] introduced an innovative hybrid algorithm that merges the deterministic Coordinated Multi-Robot Exploration (CME) with the meta-heuristic Salp Swarm Algorithm (SSA) to enhance space search performance. The authors demonstrated through experimental results that the novel CME-SSA algorithm surpassed four other cutting-edge methods concerning exploration efficiency, encompassing metrics such as total area coverage, successful exploration rate, and time required to finish the exploration task. While the study's strengths lie in the successful integration of CME and SSA, resulting in enhanced performance measures, it also faces limitations. These include the generalizability of the findings to various exploration scenarios and potential challenges related to scalability or computational constraints when implementing the method with larger robot teams or more expansive search spaces. Furthermore, the study's experimental maps were limited to 20 m $\times$ 20 m, which raises concerns about the CME-SSA method's performance in more complex and larger environments.

In our study, we propose a novel hybrid optimization method called Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME), which combines CME and AVOA to optimize the construction of a finite map in multi-robot exploration. To the best of our knowledge, this is the first study that combines CME and AVOA for multi-robot exploration.

Overall, deterministic, and meta-heuristic algorithms have their respective strengths and weaknesses. Deterministic algorithms are simple and easy to implement, but their performance deteriorates in larger and more complex environments. Meta-heuristic algorithms, on the other hand, can handle complex environments, but they can be computationally expensive and require tuning of multiple parameters. The proposed HVCME algorithm combines the strengths of both techniques and has the potential to significantly improve the efficiency and accuracy of multi-robot exploration.

3. Problem Formulation and Proposed Method

The limitations of prior methods in multi-robot exploration include inefficiency, inability to complete exploration, suboptimal construction of a finite map, and getting stuck in local optima. This study proposes a novel hybrid optimization approach that combines CME and AVOA for optimizing finite map construction in the exploration of multiple agents. The proposed algorithm, named Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME), has the potential to enhance the effectiveness and precision of multi-agent exploration across a range of applications.

3.1. Deterministic CME

Multi-robot exploration involves using multiple mobile robots to explore an environment, starting with complete uncertainty, and concluding with a well-defined map. In the context of constructing a map using robot communication, there are two approaches: centralized exploration and decentralized exploration. In centralized exploration, all robots have access to the same map, which enables them to monitor each other's progress simultaneously. This approach enhances communication among robots and ensures that they are exploring the environment efficiently. In contrast, in decentralized exploration, robots construct their own maps, and data exchange is only coordinated when robot positions overlap. While this approach reduces the complexity of the coordination process, it may lead to less efficient exploration and less sharing of information among robots [40]. In this paper, the centralized exploration approach is employed due to its ability to increase coordination and enhance communication among robots. The proposed method calculates utility values updated by all robots through iterations and real-time costs of travelling for each robot, which leads to a more efficient and accurate multi-robot exploration process.

In the process of Coordinated Multi-Robot Exploration, the map is represented using an occupancy grid map [41]. The robot is initially situated in an indoor environment, with no knowledge of its surroundings, and equipped with a sensor view that covers a limited range. To construct a finite map in an unknown space, the robot uses its sensor view to sense frontier cells [42,43], which are essential. Numerical values indicating the probability of obstruction occupying a grid cell are stored in the occupancy grid map, along with the utility and cost of travelling in each cell. However, due to the limited sensor view, only nine cells surrounding the robot are covered on the occupancy grid map [44]. Figure 1 provides a visual representation of the sensor view on the occupancy grid map.

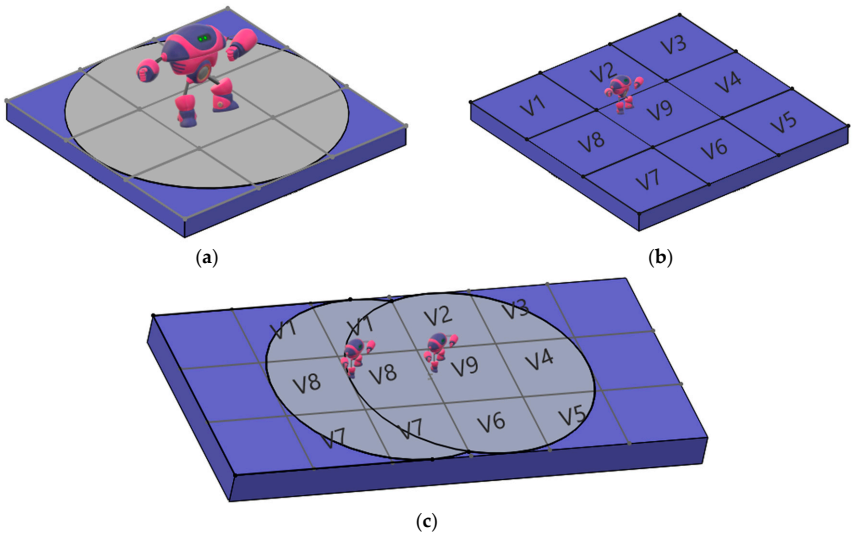


Figure 1. This figure displays the range of the sensor view in grid cells. Panel (**a**) shows the limited sensor range (V1 to V8) around the robot. Panel (**b**) shows the eight cells surrounding the robot, where cell 9 represents the robot's position. Panel (**c**) illustrates a scenario in which the robot moves from right to left, highlighting that the sensor range does not cover the cost values of V3, V4, and V5.

3.1.1. Computation of Cost Function

To determine the optimal route from the robot's current position to all frontier cells, a deterministic version of the value iteration algorithm is utilized. The algorithm calculates the cost of reaching each frontier cell, taking into account factors such as occupancy grid probability, sensor view, and Euclidean distance. The cost function is initialized with Equation (1), which considers these factors. If a cell has already been explored, the prior step's cost for that cell is added to the current position's cost. However, if the cell is classified as a frontier cell, it does not have any backward costs from the earlier stages of the ray beams that primarily opened it, as per Equation (3). The tuple $(x,\ y,\ z)$ represents the x-th cell in the direction of the *x-axis*, the y-th cell in the direction of the *y-axis*, and the z-th cell in the direction of the *z-axis* where z is considered to be zero since the utility and cost values are stored in the x-y plane of the 3D occupancy grid map. This grid map is used to represent the environment where the robot navigates. The cost of traversing a grid cell $(x,\ y,\ z)$ is inversely proportional to its occupancy probability value, $P(occ_{xyz})$. To determine the minimum cost path, the algorithm performs two steps outlined in Equations (1) and (2) [16].

1. Initialization

$$V(x,\ y,\ z) = \begin{cases} 0, \text{ if } (x,\ y,\ z) \text{ is the robot position} \\ \infty, \text{ Otherwise} \end{cases} \tag{1}$$

2. A loop will be executed to update every grid cell's status at $(x,\ y,\ z)$ coordinates.

$$V(x,\ y,\ z) = min\left\{ V_{x+\Delta x,\ y+\Delta y,\ z+\Delta z} + (\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}) \times P(occ_{x+\Delta x,\ y+\Delta y,\ z+\Delta z}) \right\} \tag{2}$$

$$V(x,\ y,\ z) = min\left\{ (\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}) \times P(occ_{x+\Delta x,\ y+\Delta y,\ z+\Delta z}) \right\} \tag{3}$$

where $\Delta x,\ \Delta y,\ \Delta z \in [-1, 0, 1]$, $P(occ_{x+\Delta x,\ y+\Delta y,\ z+\Delta z}) \in [0,\ max(occ)]$ and $max(occ)$ is the maximum occupancy probability. The occupancy probability of a cell in a grid map can take one of three values: $[0,\ 0.5,\ 1]$. A value of 0 indicates that the cell is assumed to be empty and obstacle-free, while a value of 0.5 represents an uncertain or unknown cell probability, which is approximately equal to that of a cell occupied by an obstacle. A value of 1 denotes a cell that is considered fully occupied by an obstacle or object. The selection of the next robot position is based on choosing the lowest value across neighboring cells. Multi-robot systems require a collective organization to explore, unlike single mobile robot systems, which only require a low-cost search to locate themselves. In this regard, the CME technique introduced in this study enables the distribution of tasks among multiple robots [16].

3.1.2. Utility Value

In this section, we discuss the utility value, which is a metric used to determine whether a cell in the grid map has been explored or not. Initially, all grid cells are assigned the same utility value, as seen in Equation (4). The values of utility assigned to the frontier cells decrease as the robots move toward new positions. Robots prioritize exploring new positions by selecting grid cells with higher utility values. The cost of each grid cell is determined by its distance from the robot. The utility of a frontier cell is influenced not only by its surroundings but also by the number of robots moving toward it. In order to maximize the utility values, the robots actively search for new locations that have not yet been explored. By doing so, they can uncover new information and increase their knowledge of the environment, which in turn can lead to more efficient and effective exploration strategies. As shown in Equation (4).

$$U_i^{cell} = U_{i-1}^{cell} - \sum_{i=1}^{n-1} P(\|occ_{x,\,y,\,z}^{c} - occ_{x,\,y,\,z}^{r}\|) \tag{4}$$

The current utility value of a cell, denoted as U_i^{cell}, is used to determine its usefulness or importance. Meanwhile, U_{i-1}^{cell} denotes the utility value of the corresponding cell in the previous state of the exploration process. The probability of the current cell i is symbolized by P. To optimize the exploration process, the maximum utility value at iteration i is determined using Equation (5), which takes into account the probability and previous utility value of the cell. Maximizing the utility values of cells enhances the efficiency and effectiveness of the exploration process.

$$(i, cell) = argmax\left\{U_i^{cell} - V_{x,\,y,\,z}\right\} \tag{5}$$

For effective collaborative exploration, it is necessary for the robots to be situated in close proximity to each other, allowing their sensors to scan each other. This initial positioning strategy enables the robots to disperse in different directions, leading to the exploration of various target locations and a subsequent decrease in utility values. The maps used in the experiment were fixed at a dimension of 50 m $\times$ 50 m, and the sensor rays were limited in length. The explored area is denoted by the color blue, while dark grey regions represent obstacles.

3.2. Metaheuristic African Vultures Optimization Algorithm (AVOA)

Abdollahzadeh et al. [17], proposed the African Vultures Optimization Algorithm (AVOA), a metaheuristic algorithm inspired by African vultures' foraging and navigation behaviors, with promising results in solving optimization problems. The AVOA algorithm is designed to accurately replicate vultures' behavior in the wild and achieves this through four key features. These features include the algorithm's ability to be easily adaptable to different optimization problems by specifying a maximum of N vultures, grouping vultures based on a fitness function, and assuming the weakest solution to be the worst while keeping distance from it and seeking the best solution. The AVOA also converges on the best solution by treating the two best solutions as the strongest vultures and having the other vultures approach them.

3.2.1. First Phase

The first step involves identifying the best vulture in each group. Initially, the fitness of all solutions in the population is calculated, and the best and second best solutions are designated as the best vultures of the first and second groups, respectively. The remaining solutions converge towards the best solutions of the first and second groups using Equation (6). The entire population is recalculated in each fitness iteration to improve the group living and foraging capabilities of vultures, which is considered their most crucial natural function. This approach aims to optimize solutions to problems more effectively.

$$BV(i) = \begin{cases} V_1 \; if \; p_i \; = \; L_1 \\ V_2 \; if \; p_i = L_2 \end{cases} \tag{6}$$

where BV is the best vulture, $V1$ and $V2$ are the first best vulture and the sconed best vulture, respectively. The probability calculation for the artificial vulture's optimization algorithm is an essential aspect of the algorithm's implementation. The probability of selecting certain vultures and moving them toward the best solution is determined by Equation (6). Two parameters, $L1$ and $L2$, are used to calculate the probability before the search operation. These parameters are constrained between 0 and 1, with their sum being equal to 1. To select the best solution for each group, Equation (7) uses the Roulette wheel

$$p_i = \frac{F_i}{\sum_{i=1}^{n} F_i} \tag{7}$$

The parameters in AVOA have a significant effect on the algorithm's behavior. If the $\alpha - numeric$ parameter is in close proximity to 1 and the $\beta - numeric$ parameter is in close proximity to 0, the algorithm intensifies the search operation. Alternatively, if the $\beta - numeric$ parameter is close to 1 and the $\alpha - numeric$ parameter is close to 0, the algorithm increases diversity. Thus, selecting appropriate values for these parameters is crucial to achieving optimal results in AVOA.

3.2.2. Second Phase

During phase two of the AVOA, the rate of starvation among vultures is taken into consideration. Vultures with higher energy levels can fly longer distances to search for food. However, when vultures are hungry, they experience a drop in their energy levels, causing them to become more assertive in their quest for food and sometimes even competitive with vultures that are more dominant. For this behavior, Equation (8) has been utilized, this allows for a transition from the phase of discovering new options to the phase of utilizing and maximizing the potential of the discovered options based on the hunger or satiation levels of the vultures. The rate of satiation declines over time, and to accurately model this trend, Equation (9) is employed.

$$t = h \times (sin^w \left(\frac{\pi}{2} \times \frac{Iter_i}{maxIter_i} \right) + cos \left(\frac{\pi}{2} \times \frac{Iter_i}{maxIter_i} \right) - 1) \tag{8}$$

$$R = (2 \times rand_1 + 1) \times z \times (sin^w \left(1 - \frac{Iter_i}{maxIter_i} \right) + \mathrm{t} \tag{9}$$

Equations (8) and (9), play a significant role in the AVOA, which is used to tackle intricate optimization problems. The variable F represents the vulture's satiety level, which has a direct impact on the bird's search behavior. Additionally, iteration refers to the current iteration number, while z is a random number that changes between $[-1,\ 1]$ with each iteration. The parameter h is a random number between $[-2,\ 2]$, and $rand_1$ has a value that varies randomly between $[0,\ 1]$. The value of z determines the vulture's hunger level. If it is less than 0, the bird is hungry, and if it increases to 0, it indicates that the vulture is full. During the AVOA algorithm's final stages, the exploration and exploitation phases are executed. The parameter w, described in Equation (8), controls these two phases. If the value of w increases, the likelihood of entering the exploration phase also increases, and the probability of entering this phase decreases if the parameter w is reduced.

When $|R| > 1$, the vulture's total rate decreases, causing the AVOA algorithm to enter the exploration phase. This happens because when vultures become hungry, they become more aggressive in their search for food and look for it in various areas. Conversely, if $|R| < 1$, AVOA enters the exploitation phase, and vultures focus on searching for food in the solutions' neighborhood.

3.2.3. Third Phase

During the exploration phase of the AVOA algorithm, vultures search for various random areas using two distinct strategies. A parameter, referred to as the "Exploration Probability" (EP), is assigned a value within the range of $[0,\ 1]$ before the search operation, which determines how each of the two strategies is utilized. To choose between the strategies during the exploration phase, a random number $rand_{EP1}$ is generated within the range of $[0,\ 1]$. If the generated number $rand_{EP1}$ is greater than or equal to the EP_1 parameter, the algorithm employs Equation (11). However, if the generated number is less than the EP_1 parameter, the algorithm uses Equation (13). In this case, each vulture explores the environment randomly, based on its level of satiation, as shown in Equation (10).

$$EP(i+1) = \begin{cases} \text{Equation (11)} \ if \ EP_1 \geq rand_{EP1} \\ \text{Equation (13)} \ if \ EP_1 < rand_{EP1} \end{cases} \tag{10}$$

$$EP(i+1) = BV(i) - D(i) \times R \tag{11}$$

$$D(i) = |X \times BV(i) - EP(i)| \tag{12}$$

The AVOA algorithm includes two distinct methods that vultures use to locate food in their immediate surroundings. The first approach, as described by Equation (11), entails the vultures randomly exploring for food at a particular distance from one of the best vultures in the two groups. In the next iteration, the vulture's position vector is represented by $EP(i+1)$, while the current iteration uses Equation (9) to obtain the rate of vulture satiety, R. The second method, depicted in Equation (12), involves selecting one of the best vultures, denoted as $BV(i)$, in the current iteration using Equation (6). To increase the randomness of vulture movement and prevent other vultures from accessing the food, the coefficient vector X is utilized.

$$EP(i+1) = BV(i) - R + rand_2 \times ((ub - lb) \times rand_3 + lb) \tag{13}$$

Equation (13) represents the second strategy where $BV(i)$ indicates the best vulture selected in the current iteration using Equation (6). Equation (9) calculates R, the rate of vulture satiety, during the same iteration. The random value $rand_2$ is generated within the range of $[0,\ 1]$. The lower and upper bounds of the variables in the solution space are lb and ub, respectively. Equation (13) is used by the AVOA to generate random solutions within the range of $[lb, ub]$. The randomness coefficient of the generated solutions is increased by the parameter $rand_3$. When $rand_3$ approaches 1, the algorithm adds a high level of random motion along with lb, distributing solutions with similar patterns, which enhances diversity and enables the algorithm to explore different areas of the search space.

One of the notable strengths of AVOA is its black box nature, which makes it applicable to various optimization problems without requiring much prior knowledge of the problem domain. The performance of AVOA was statistically evaluated using the Wilcoxon rank sum test, which showed significant superiority over other algorithms at a 95% confidence interval. Thus, AVOA presents a promising solution to a variety of optimization problems, and its effectiveness has been demonstrated in multiple case studies.

3.2.4. Forth Phase

The exploitation phase is an important component of the AVOA algorithm that focuses on increasing its efficiency. This phase consists of two internal phases that have unique strategies. The determination of the strategy to use is based on the values of EP_2 and EP_3, which must be within the range of $[0,\ 1]$ prior to the search operation. The algorithm translates the foraging movements of vultures into mathematical problems to improve its performance.

If $|R|$ falls within the range of $[0.5,\ 1]$, the AVOA enters the first phase of the exploitation stage where two strategies, rotating flight and siege-fight, are implemented. The decision on which strategy to employ is based on EP_2, which is set between 0 and 1 before the search operation. The algorithm generates a random number, $rand_{P2}$ within the range of $[0,\ 1]$ to implement the strategies. If $rand_{P2}$ is greater than or equal to EP_2, the siege-fight strategy is gradually implemented, while if it is less than EP_2, the rotating flight strategy is employed. This selection process is represented by Equation (14).

$$EP(i+1) = \begin{cases} \text{Equation } (15)\ if\ EP_2 \geq rand_{EP2} \\ \text{Equation } (18)\ if\ EP_2 < rand_{EP2} \end{cases} \tag{14}$$

When the level of food availability is high, vultures are considered to be satiated and energized. However, when multiple vultures congregate around a single food source, it often leads to intense competition over food acquisition. In such scenarios, stronger vultures typically opt not to share food with their counterparts. Meanwhile, weaker vultures attempt

to tire out the healthier ones by creating small conflicts and gathering around them. These foraging behaviors are modeled using Equations (15) and (16).

$$EP(i+1) = D(i) \times (R + rand_4) - d(t) \tag{15}$$

$$d(t) = BV(i) - EP(i) \tag{16}$$

Equation (12) is used to calculate $D(i)$, while Equation (9) is used to obtain R, which represents the satiation rate of vultures. Additionally, a random number between $[0, 1]$, referred to as $rand_4$, is utilized to increase the random coefficient. In Equation (16), $BV(i)$ represents one of the best vultures selected from the two groups using Equation (6) in the current iteration, whereas $EP(i)$ refers to the current vector position of the vulture. This position is used to calculate the distance between the vulture and one of the best vultures in the two groups.

AVOA utilizes rotating flights to model spiral motion, based on the spiral motion model. The mathematical model of the spiral equation represents the rotational flight between all vultures and one of the two best vultures. Equations (17) and (18) express the rotating flight (18).

$$\begin{aligned} S_1 &= BV(i) \times \left(\frac{rand_5 \times EP(i)}{2\pi}\right) \times cos(EP(i)) \\ S_2 &= BV(i) \times \left(\frac{rand_6 \times EP(i)}{2\pi}\right) \times sin(EP(i)) \end{aligned} \tag{17}$$

$$EP(i+1) = BV(i) \times (S_1 + S_2) \tag{18}$$

In the current iteration, $BV(i)$ is obtained using Equation (6) and is then used in Equations (17) and (18) to update the vulture's location. The sine and cosine functions are denoted by "sin" and "cos", respectively. Moreover, random numbers $rand_5$ and $rand_6$ within range [0, 1] are included in the calculation, and $S1$ and $S2$ are determined using Equation (17). Finally, Equation (18) is utilized to update the location of the vultures based on the obtained parameters.

During the second phase of exploitation, vultures gather at the food location and engage in competitive fights to obtain food. This phase is only executed when the $|R|$ number is less than 0.5. At the start of this phase, a random number $rand_{EP3}$ is generated. If $rand_{EP3}$ is greater than or equal to the parameter EP_3, vultures are accumulated over the food source using a specific strategy. Conversely, if $rand_{EP3}$ is less than EP_3, an aggressive siege-fight strategy is employed. The process of selecting the strategy is illustrated in Equation (19).

$$EP(i+1) = \begin{cases} \text{Equation (21)} \; if \; EP_3 \geq rand_{EP3} \\ \text{Equation (22)} \; if \; EP_3 < rand_{EP3} \end{cases} \tag{19}$$

Equations (20) and (21) model the behavior of vultures in the AVOA algorithm when they are hungry and there is high competition for food. The algorithm observes how all vultures move towards the food source, and in situations where multiple vulture species are competing, they may gather on a single food source.

$$\begin{aligned} A_1 &= V_1(i) - \frac{V_1(i) \times EP(i)}{V_1(i) - EP(i)^2} \times R \\ A_2 &= V_2(i) - \frac{V_2(i) \times EP(i)}{V_2(i) - EP(i)^2} \times R \end{aligned} \tag{20}$$

In Equation (20), the best vulture in the first and second groups in the current iteration are represented by $V_1(i)$ and $V_2(i)$, respectively. Additionally, the level of vulture satiety, denoted by R, which is computed using Equation (9), and the present vector location of a vulture, represented by $EP(i)$, are used.

$$EP(i+1) = \frac{A_1 + A_2}{2} \tag{21}$$

The next iteration's position vector of the vulture is updated using Equation (21), where $A1$ and $A2$ are determined by applying Equation (20), and $EP(i+1)$ represents the position vector of the vulture in the next iteration. When the number of available food items is less than 0.5, the leading vultures become weak and hungry, unable to compete with other vultures for food. Consequently, the other vultures become aggressive and move towards the leading vulture in pursuit of food. This movement is modeled using Equation (22).

$$EP(i+1) = BV(i) - |d(t)| \times R \times Levy(d) \tag{22}$$

The computation of vulture distance to the best vulture from two groups is crucial in Equation (21), which involves the use of Equation (16). To enhance the efficiency of the AVOA algorithm in Equation (22), Levy flight (LF) patterns have been integrated, as they are commonly utilized in various metaheuristic algorithms. The LFs were calculated based on Equation (23).

$$LF(x) = 0.01 \times \frac{u \times \delta}{|v|^{\frac{1}{\beta}}}, \delta \tag{23}$$

Equation (22) involves d, which indicates the number of problem dimensions, u and v, which are random numbers that range within $[0,\ 1]$. Additionally, β is a fixed value set to 1.5 in this equation.

The initialization process of the AVOA algorithm (Algorithm 1) involves creating N vultures, and its computational complexity is proportional to N, making it $O(N)$. After the initialization process, the update mechanism begins, which involves evaluating the fitness of each vulture and updating its location vector. This process is critical because it helps the algorithm find the optimal solution. The update process has a computational complexity of $O(I \times N) + O(I \times N \times dim)$, where I is the maximum number of iterations, and dim represents the dimension of the problem. The computational complexity of the update process can be broken down into two parts: the first part has a complexity of $O(I \times N)$, which represents the search for the optimal location, while the second part, $O(I \times N \times dim)$, represents the updating of the location vector of all formed vultures.

Algorithm 1 Pseudocode of AVOA

1. Take inputs for the population size (N) and maximum number of iterations (T).
2. The algorithm outputs the location of the vulture and its corresponding fitness value.
3. Randomly initialize the population Pi (i = 1, 2, . . . , N).
4. While the stopping condition is not met, calculate the fitness values of the vulture.
5. Set PBestVulture1 as the location of vulture, which represents the first-best location of the best vulture category 1.
6. Set PBestVulture2 as the location of vulture, which represents the second-best location of the best vulture category 2.
7. For each vulture (Pi), select $\mathbf{BV(i)}$ using Equation (6).
8. Update the value of F using Equation (9).
9. If $|R| \geq 1$, then update the location of the vulture using Equation (11) if $\boldsymbol{EP_1} \geq \boldsymbol{rand_{EP1}}$, otherwise use Equation (13).
10. If $|R| < 1$ and $|R| \geq 0.5$, then update the location of the vulture using Equation (15) if $\boldsymbol{EP_2} \geq \boldsymbol{rand_{EP2}}$, otherwise use Equation (18).
11. If $|R| < 0.5$, then update the location of the vulture using Equation (21) if $\boldsymbol{EP_3} \geq \boldsymbol{rand_{EP3}}$, otherwise use Equation (22).
12. Returns PBestVulture1 at the end.

Overall, the computational complexity of the AVOA algorithm can be estimated as $O(N \times (I + I \times dim))$, which indicates that the algorithm's running time increases linearly with the number of vultures, maximum iterations, and problem dimension. It is essential to note that AVOA, like other metaheuristics, follows a problem-solving paradigm that starts with a random population of solutions and iteratively improves them until a specified

termination condition is met. However, the AVOA algorithm's unique feature lies in its inspiration and modeling process, which are based on the characteristics of vultures' foraging behavior.

3.3. Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME)

In this paper, we present a hybrid method that combines a stochastic approach based on African Vulture Optimization Algorithm (AVOA) with cost and utility computations to efficiently explore unknown spaces using mobile robot sensor systems and create a finite map with a coordinated multi-robot system.

To start, our method initializes the grid map by assigning a value of 1 to all cells, and then identifies eight candidate cells around the robot that are covered by the sensor. These eight cells form the "vultures" population" represented by V1 to V8, as depicted in Figure 1. Next, we compute the cost of each candidate cell and derive the utilities from the cost of the surrounding eight grid cells using Equation (5). The stochastic method then identifies the two maximum utility values as the two best vultures and assigns them. The priority between the two vultures is determined based on R in Equation (9)—which is the utility in our method—as well as the random values $rand_{P1}$, $rand_{P2}$, and $rand_{P3}$ in Equations (10), (14), and (19), respectively. The values of occupancy probability of the two dominated cells also affect the priorities.

$$X_{volture,i} = \begin{cases} X_{RandV,i} - |2 \times rand_{P1} \times X_{CurrentV,i}| \times P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z}) & rand_{P1} < p_1 \\ X_{RandV,i} - P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z}) + rand_{P1} \times ((ub_i - lb_i) \times rand_{P1} + lb_i) & rand_{P1} \geq p_1 \end{cases} \tag{24}$$

We present a novel approach for determining the position of a random vulture based on Equations (11) and (13) from the AVOA framework. Specifically, Equation (24) in the exploration phase where it utilizes $X_{volture,i}$, the position of a vulture in iteration i, $X_{RandV,i}$, which represents the position of a randomly selected vulture out of a total of eight cells, $X_{CurrentV,i}$ is the position of the current vulture in the current iteration. By utilizing these values, our approach determines the optimal position of the vulture in iteration i.

To determine the highest likelihood of occupancy for a grid cell that the robot is capable of traversing, we use $P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z})$. This value is dependent on the robot's location and its intended direction of travel, as represented by the values of Δx and Δy, which are constrained to the set $[-1, 0, 1]$. Moreover, $P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z})$ is bounded by the range $[0,\ max(occ)]$, where $max(occ)$ represents the highest likelihood of occupancy for a grid cell that the robot is capable of traversing.

$$\begin{aligned} A &= X_{BestV,1} - ((X_{BestV,1} \times X_{CurrentV,i}) / (X_{BestV,1} - X^2_{CurrentV,i})) \times P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z}) \; rand_{P2} < p_2 \\ B &= X_{BestV,2} - ((X_{BestV,2} \times X_{CurrentV,i}) / (X_{BestV,2} - X^2_{CurrentV,i})) \times P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z}) \; rand_{P2} < p_2 \end{aligned} \tag{25}$$

$$X_{volture,i} = \begin{cases} (A+B)/2 \; |F| < 0.5 \;\&\; rand_{P2} < p_2 \\ X_{RandV,i} - |X_{RandV,i} - X_{CurrentV,i}| \times P_i(occ_{x+\Delta x,\, y+\Delta y,\, z+\Delta z}) \times Levy(\text{variables\_no}) & |F| \geq 0.5 \;\&\; rand_{P2} \geq p_2 \end{cases} \tag{26}$$

Equation (26) based on Equations (21) and (22) from the AVOA framework. The next iteration's position vector of the vulture is updated using Equation (25), where A and B are determined by applying Equation (20), and $X_{volture,i}$ represents the vulture's location vector in the subsequent iteration, when the number of available utility items is less than 0.5.

$$\begin{aligned} S_1 &= X_{RandV,i} \times (rand_{P3} \times X_{CurrentV,i}/2 \times \pi) \times cos(X_{CurrentV,i}) \; rand_{P3} \geq p_3 \\ S_2 &= X_{RandV,i} \times (rand_{P3} \times X_{CurrentV,i}/2 \times \pi) \times sin(X_{CurrentV,i}) \; rand_{P3} \geq p_3 \end{aligned} \tag{27}$$

$$X_{volture,i} = \begin{cases} X_{RandV,i} - (S_1 + S_2) Equation(27) & |F| < 0.5 \;\&\; rand_{P3} \geq p_3 \\ |2 \times rand_{P3} \times X_{RandV,i} - X_{CurrentV,i}| \times (P_i(occ_{x+\Delta x, y+\Delta y, z+\Delta z}) + rand_{P3}) - (X_{RandV,i} - X_{CurrentV,i}) & |F| \geq 0.5 \;\&\; rand_{P3} < p_3 \end{cases} \tag{28}$$

With a high prevalence of utility values, cells become more prone to selection. Nonetheless, intense competition for food resources arises when numerous robots congregate

around a single cell. Under these circumstances, robots typically refrain from sharing the same cell with their counterparts. Equation (28) encapsulates this behavior, drawing from Equations (15) and (18) of the AVOA algorithm.

Equation (6) is employed to refresh the positions of the most favorable vultures, where $X_{volture,i}$ signifies the positions of both the vulture and the food source in the ith dimension. Vultures adjust their locations based on the highest occupancy probability for a grid cell that a robot can navigate. The upper and lower boundaries of each dimension are taken into account, and random values $rand_{P1}$, $rand_{P2}$, and $rand_{P3}$ are used to modify the vultures' positions.

Figure 2 illustrates the application of AVOA's stochastic method to determine the ideal position for the robot's movement among the eight candidate cells, which constitutes the premier initial vulture. This process persists until a stopping criterion is satisfied.

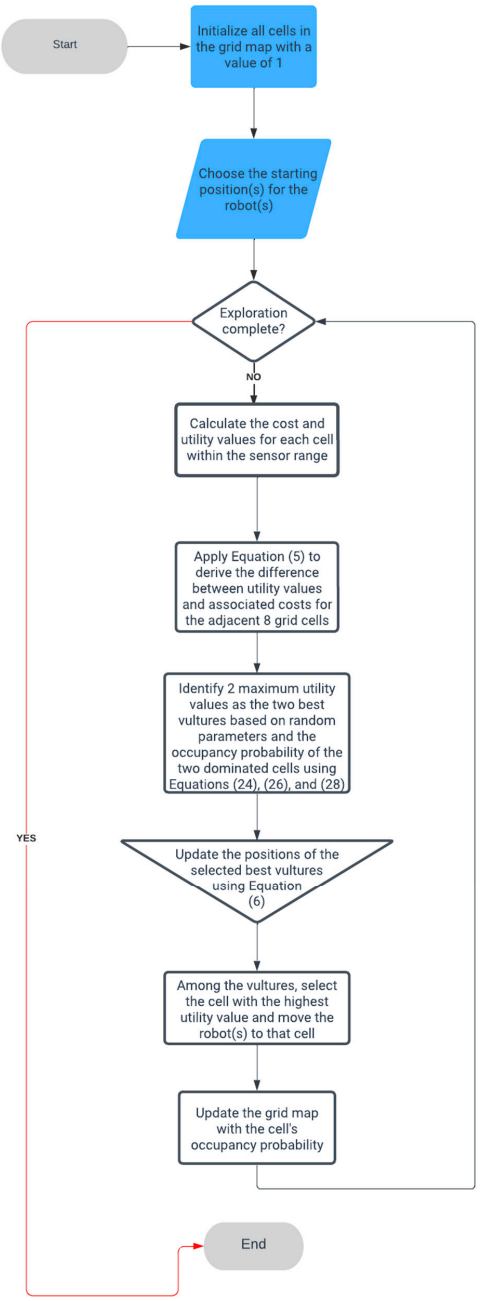


Figure 2. The hybrid optimization algorithm called Hybrid Vulture-Coordinated Multi-Robot Exploration HVCME.

The proposed hybrid approach facilitates effective exploration of uncharted territories by mobile robot sensor systems. Compared to conventional methods, this strategy substantially diminishes the count of candidate cells for the following move, thereby enhancing the exploration process's efficiency. Furthermore, the stochastic method incorporated within this approach infuses randomness, enabling the detection of areas that might have been otherwise overlooked. Consequently, the hybrid method shows promise in optimizing the process of constructing a finite map with a coordinated multi-robot system.

In this paper, a new method called HVCME is proposed for multi-robot exploration. The method uses the African Vulture Optimization Algorithm (AVOA) with a selection process based on selecting the maximum values for the top two vultures, which symbolize the vulture's positions, the method strives to efficiently explore uncharted regions within a grid map. After the robot moves according to the best vulture positions, the utility value of adjacent grid cells is reduced using Equation (4), enhancing the exploration process. The proposed method has several advantages that make it an effective optimization technique. One of the main advantages is its ability to avoid local optima due to the AVOA's vulture food search strategy, which balances exploration and exploitation. This ensures that the method can find the best possible solution without getting stuck in local optima. Additionally, the method performs well on both unimodal and multimodal optimization problems, preserves an optimal equilibrium between exploration and exploitation, and also shows rapid convergence with a decreasing trend in fitness changes. Additionally, it outperforms other optimization algorithms in extensive test functions and attains statistically significant distinctions in nearly all outcomes. Finally, the method continues to produce better solutions even with increasing dimensions of the problem space. Overall, these advantages make the AVOA a promising technique for a wide range of optimization problems.

4. Results and Discussion

Optimizing multi-robot exploration through the use of meta-heuristic algorithms requires rigorous testing due to the random behavior of these algorithms. Proper evaluation of algorithmic performance necessitates a suitable and adequate set of test maps, including simple and complex environments, to test exploration under varying conditions.

In this study, we present a simulation of the Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME) method, a novel hybrid optimization approach that combines CME and AVOA to construct a finite map in multi-robot exploration. We utilized two environments of varying complexity to evaluate the algorithm's performance, and map complexity was altered by introducing barriers and modifying their direction.

To evaluate HVCME's effectiveness, we compare its performance against four other algorithms—Coordinated Multi-Robot Exploration and Grey Wolf Optimizer ($CME - GWO$), Coordinated Multi-Robot Exploration and Salp Swarm Algorithm ($CME - SSA$), Coordinated Multi-Robot Exploration and Sine Cosine Algorithm ($CME - SCA$), and Coordinated Multi-Robot Exploration and Mountain Gazelle Optimization ($CME - MGO$). We scrutinized their performance employing three metrics: execution time, the proportion of the explored area, and the count of unsuccessful runs. To ensure a fair comparison, we maintained a consistent map size of 50 m $\times$ 50 m for all simulations. In our simulation, the blue regions represent the examined area, while dark grey regions indicate obstacles. We positioned the robots close to each other initially to facilitate cooperative exploration, which resulted in divergent directions and decreased utility of selected targets.

In this study, the simulation analysis aimed to determine the percentage of the explored area, given the robots' ability to move freely in any direction. An alternative approach was utilized to address this objective, and the resulting percentage was calculated using Equation (29).

$$E = \frac{U_u - U_e}{U_u} \times 100 \tag{29}$$

A new method was employed to assess the entire explored area in an unknown environment. The proportion of the explored area was computed based on the difference

between unexplored and explored utility values in the obstacle-free zone. The total explored grid cells, denoted as "E", were then calculated using Equation (29), where U_u is the unexplored utility value in the obstacle-free zone, and U_e is the explored utility values. To guarantee a just comparison among various approaches, identical parameters were applied to all algorithms and simulations. These parameters encompass the number of iterations, obstacle locations, the number of robots and their starting positions, map dimensions, and sensor range.

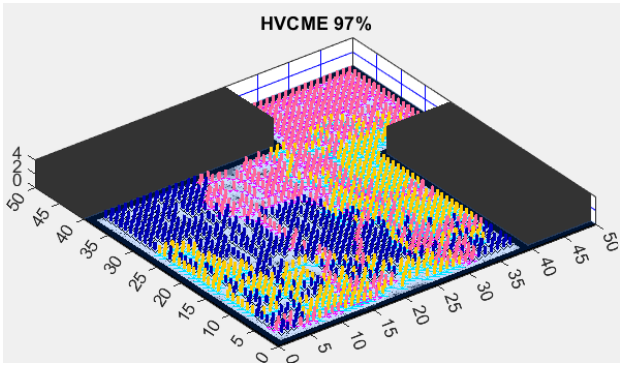
In order to ensure reliable results, a strategy based on the central limit theorem was employed [45]. This approach suggests that sample sizes of approximately 30 to 50 randomly picked individuals are adequate for achieving a fairly normal distribution. In line with this, a sample size of 30 was used in this simulation, with each algorithm being run for 500 iterations 30 times to collect the required samples. It is important to note that AVOA is stochastic, meaning that each run generates different results. In all the maps environments, every color on the map signifies an individual robot. As three robots were utilized in this simulation, three distinct colors were employed to distinguish between them, each color represents a different robot.

Upon completing the simulation, the performance of the proposed HVCME method and other methods was assessed by considering three crucial factors: the total number of explored grid cells, the time taken for map exploration, and the instances where a method failed to complete a full run.

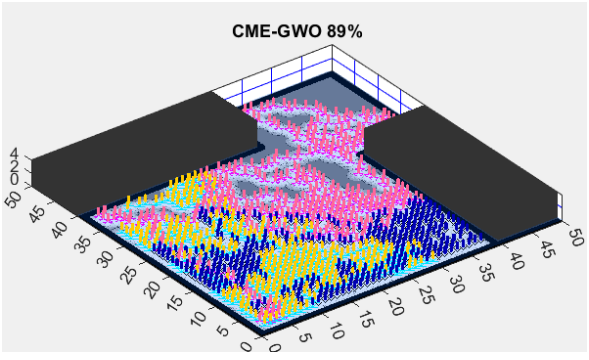
In the subsequent section, we present our experimental results, compare the performance of HVCME with the other algorithms in a simple and complex unknown environment, and discuss the implications of our findings.

4.1. Simple MAP

In this study, two types of simple maps were used to create a relatively free space environment with fewer obstacles for the robots to navigate. As depicted in Figures 3 and 4, the maps had fewer obstacles and more open spaces, which made it easier for the robots to move around and avoid collisions. The exploration results of the five algorithms were presented in the figures, with each color representing a different robot. Notably, the HVCME algorithm performed exceptionally well, achieving over 90% exploration in 29 iterations and around 98% in 4 of them. Furthermore, increasing the number of iterations by 10 to 20 led to a 99% exploration rate. These findings demonstrate the effectiveness of the HVCME algorithm in exploring relatively simple environments with few obstacles.

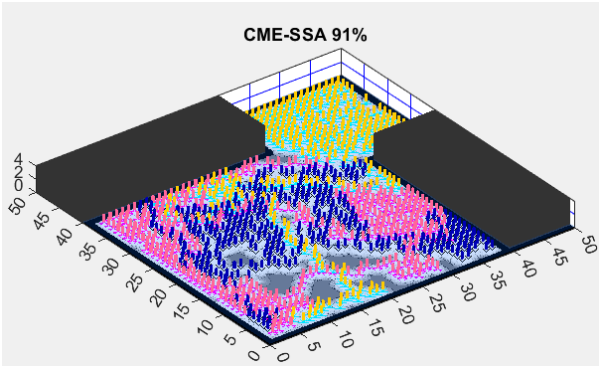


(**a**)

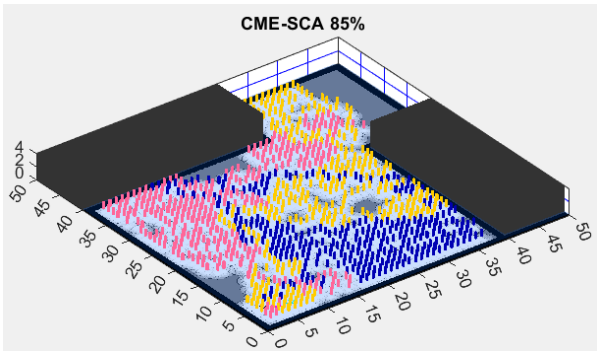


(**b**)

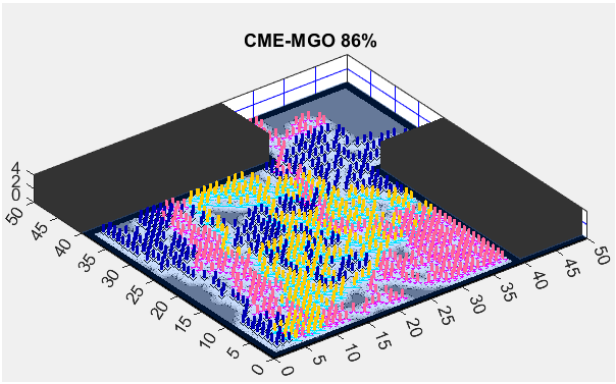
Figure 3. *Cont.*



(**c**)

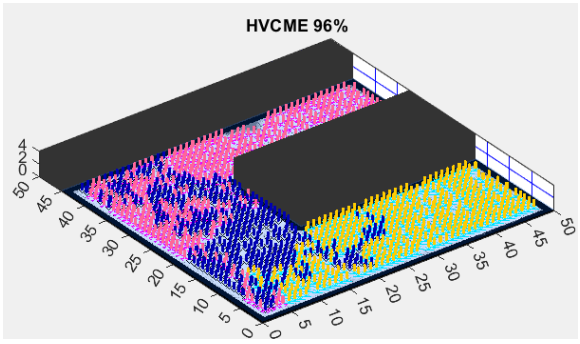


(**d**)

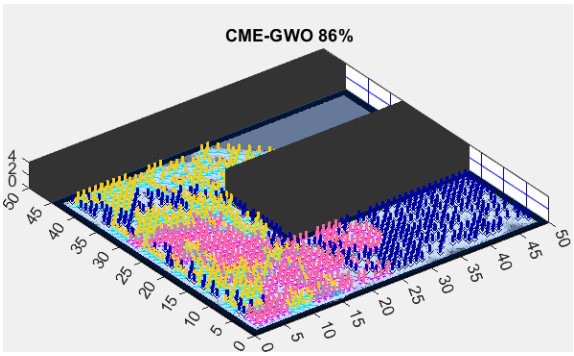


(**e**)

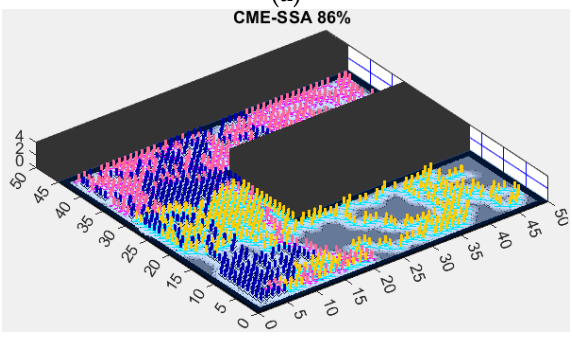
Figure 3. Comparing exploration efficiency of different algorithms on Simple Map 1 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.



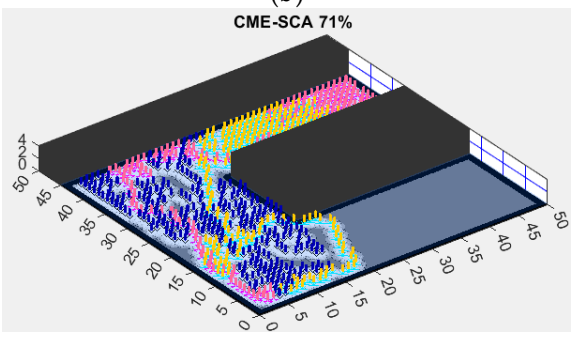
(**a**)



(**b**)

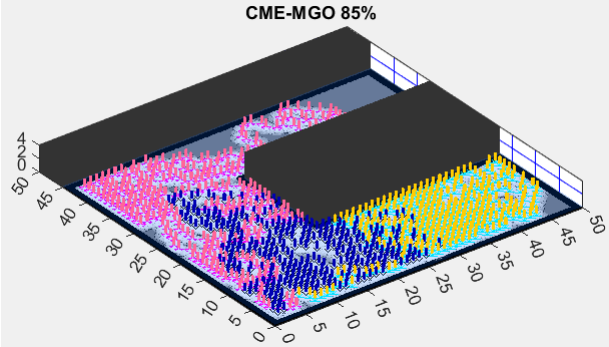


(**c**)



(**d**)

Figure 4. *Cont*.



(**e**)

Figure 4. Comparing exploration efficiency of different algorithms on Simple Map 1 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.

Table 1 displays the average and standard deviation indicators for the proposed HVCME algorithm and four other optimization methods across different test functions. These indicators measure the overall performance of the algorithms across multiple runs. The average indicator shows that the HVCME algorithm has the highest explored area's average, indicating its superior performance overall. Conversely, the standard deviation indicator reveals that the HVCME algorithm is more stable compared to the other techniques, implying its ability to consistently explore a considerable part of the search space.

Table 1. Average and standard deviation indicators of optimization algorithms across exploration in simple environment.

	HVCME		***CME−GWO***		***CME−SSA***		***CME−SCA***		***CME−MGO***	
Map No	***ave***	***std***	***ave***	***std***	***ave***	***std***	***ave***	***std***	***ave***	***std***
Map 1	93.48789747	2.394276646	88.10766242	3.107866611	89.8972499	3.166780379	83.06727879	10.47223842	87.87226601	5.240005
Map 2	95.39844996	2.484151142	91.12047224	3.520321483	91.84252617	3.100894958	85.69176361	7.003067214	87.53942155	7.241871

Table 2 displays the average time taken in seconds for each algorithm to complete one full run consisting of 500 iterations. The experiments were conducted on Simple Maps 1 and 2, and the average time per run was recorded for the HVCME algorithm and four other optimization methods, namely $CME - GWO$, $CME - SSA$, $CME - SCA$, and $CME - MGO$. The results show that the HVCME algorithm was the fastest among the five methods for both *Simple Map* 1 and *Simple Map* 2. For Simple Map 1, the HVCME algorithm spent an average of 92.56 s per run, while the other methods spent an average of 95.86 to 97.82 s per run. For Simple Map 2, the HVCME algorithm spent an average of 92.80 s per run, while the other methods spent an average of 96.17 to 98.47 s per run. These findings indicate that the HVCME algorithm is not only efficient in terms of exploration but also in terms of time consumption, making it a promising approach for multi-robot exploration tasks.

Table 2. Average time (in seconds) taken by each optimization approach to complete one full run (500 iterations) in Simple Map 1 and Simple Map 2.

Map No	**HVCME**		***CME−GWO***		***CME−SSA***		***CME−SCA***		**CME-MGO**	
	ave	***std***	***ave***	***std***	***ave***	***std***	***ave***	***std***	***ave***	***std***
Map 1	92.5615699	0.297715034	97.39291327	0.999453403	95.85982903	0.181438103	97.82159427	1.230208498	96.77494	1.015549
Map 2	92.79769	0.401503478	96.47183573	1.006076789	96.1734938	0.290667074	98.07270377	0.89840887	98.46685	1.142403

Table 3 provides information on the count of unsuccessful simulations spanning two simple maps for each optimization algorithm. A simulation is considered failed if it cannot complete 500 iterations successfully due to obstacles or other robots blocking neighbor cells. For Map 1, the HVCME algorithm and $CME - SSA$ had no failed simulations,

while $CME - GWO$, and CME-SCA had six failed simulations each, and $CME - MGO$ had five. For Map 2, the HVCME algorithm had no failed simulations, while $CME - GWO$ had two, $CME - SSA$ had one, $CME - SCA$ had nine, and $CME - MGO$ had eight. The findings suggest that the HVCME algorithm performed the best in preventing unsuccessful simulations in the HVCME algorithm had the best performance in terms of avoiding failed simulations in both simple maps, while $CME - SCA$ and $CME - MGO$ had the worst performance, with the highest number of failed simulations in Map 2.

Table 3. Number of failed simulations across two simple maps for each optimization algorithm.

Map No	HVCME	*CME*−*GWO*	*CME*−*SSA*	*CME*−*SCA*	*CME*−*MGO*
Map 1	0	6	0	6	5
Map 2	0	2	1	9	8

In this section, we performed experiments on two simple maps to compare the performance of the HVCME algorithm with other methods regarding exploration area, time, and failures. The results revealed that the HVCME algorithm demonstrated superior performance than other techniques on all five complex maps in terms of the percentage of explored area, less time, and fewer failures. To support our claim, we presented statistical tables including the average and standard deviation for exploration area and time, and a table illustrating the number of failed simulations for each method Tables 1–3. Moreover, we provided visual representations of the simulation environments indicating the explored area for each algorithm in Figures 3 and 4. Overall, the study findings suggest that the HVCME algorithm shows promise as an effective solution for complex mapping tasks in challenging environments.

4.2. Complex Map

The outcomes illustrated in Figures 5–9 clearly establish the superiority of the proposed hybrid stochastic HVCME approach over other meta-heuristic hybrid techniques including $CME - GWO$, $CME - SSA$, $CME - SCA$, and $CME - MGO$. The five complex maps used in the experiments contained various types of narrow tunnels and corridors, with Figures 8 and 9 being the most challenging maps due to the high number of obstacles and small zones they contained. Nevertheless, the HVCME algorithm demonstrated remarkable performance by achieving a coverage rate of 95% in Figure 5 and 97% in Figure 6. Notably, the HVCME algorithm successfully explored four tight paths in map 4 (Figure 7) that remained unexplored by any other algorithms, resulting in a total coverage rate of 98%. Similarly, in Figure 8, where the map had narrow corridors, the HVCME algorithm exhibited efficient exploration of the zone, achieving a coverage rate of 93%. In the most challenging environment, map 5 (Figure 9), the HVCME algorithm successfully covered 96% of the area, while the other algorithms ($CME - GWO$, $CME - SSA$, $CME - SCA$, and $CME - MGO$) encountered numerous failures to complete the 500 iterations successfully due to the robots' initial positions being in close proximity to the barriers and the presence of tiny paths. The findings of the simulation experiments thus demonstrate the HVCME algorithm's remarkable performance in exploring complex environments with narrow tunnels and paths.

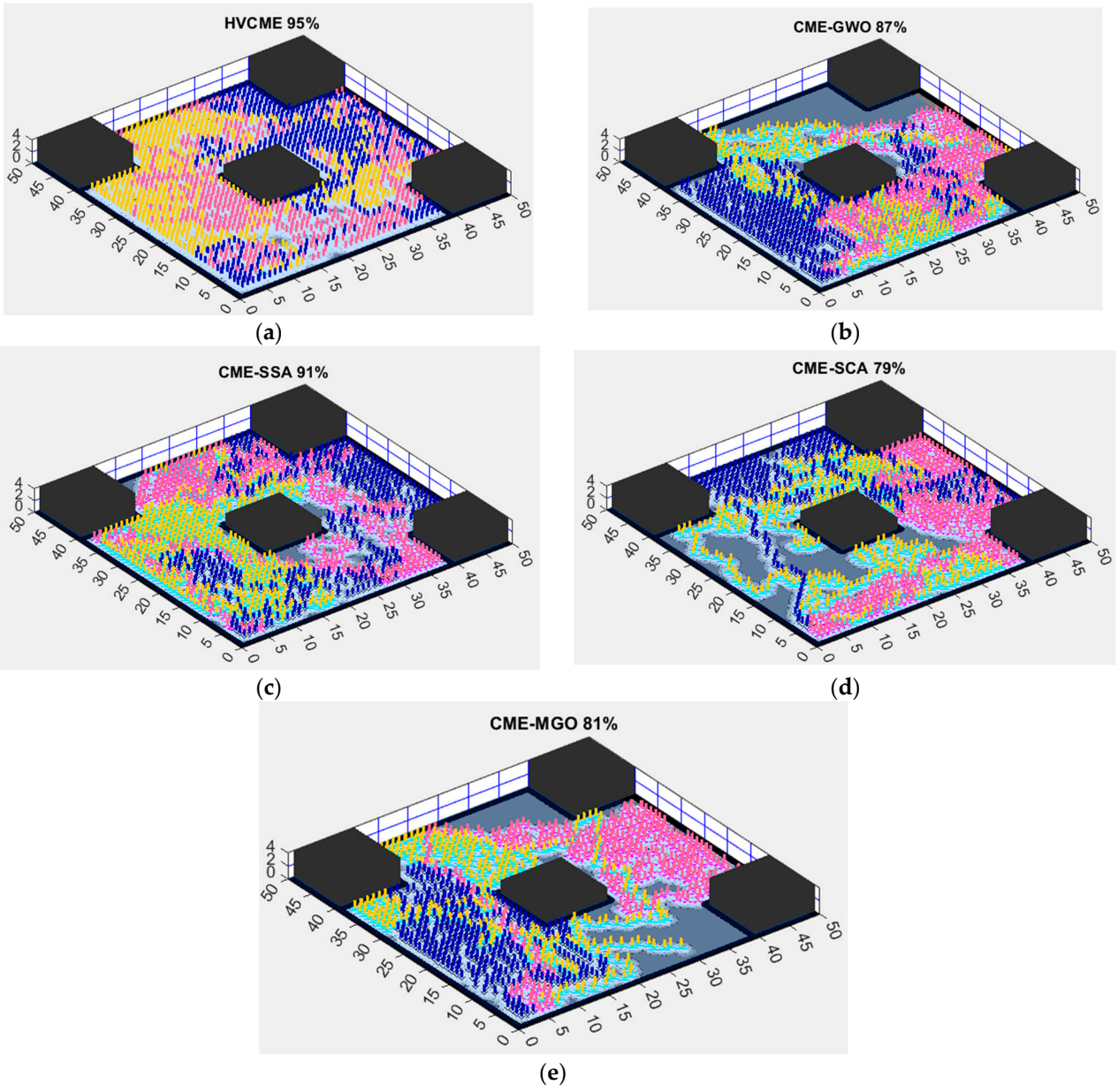


Figure 5. Comparing exploration efficiency of different algorithms on Complex Map 1 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.

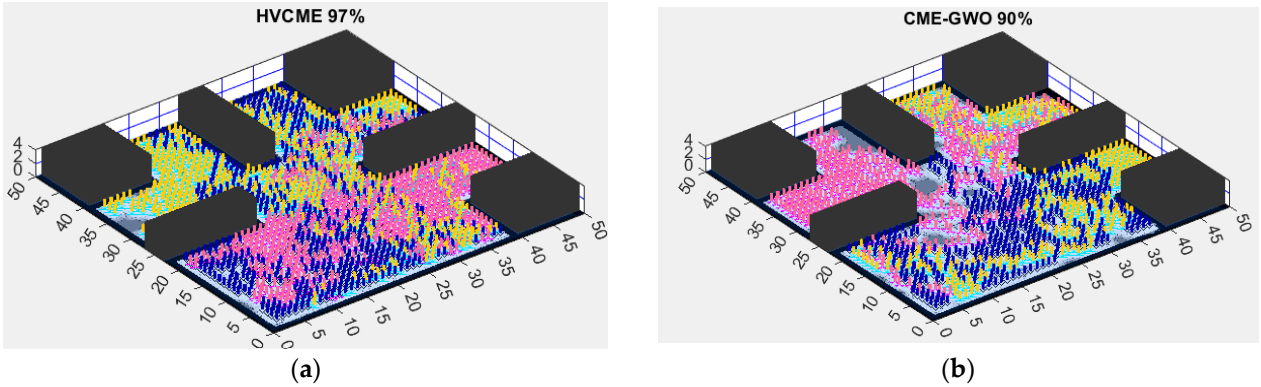
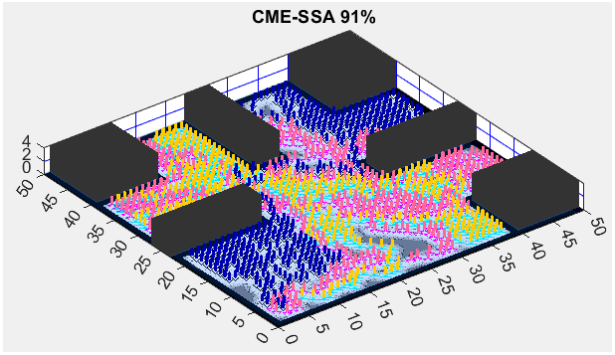
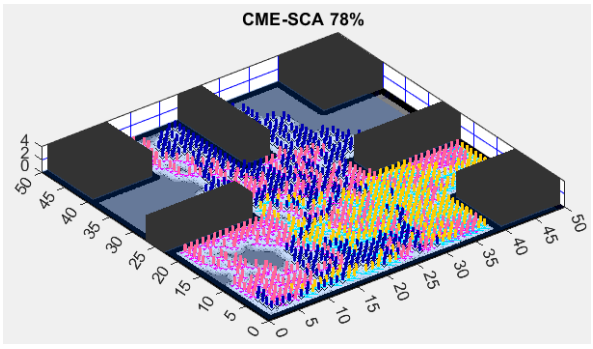


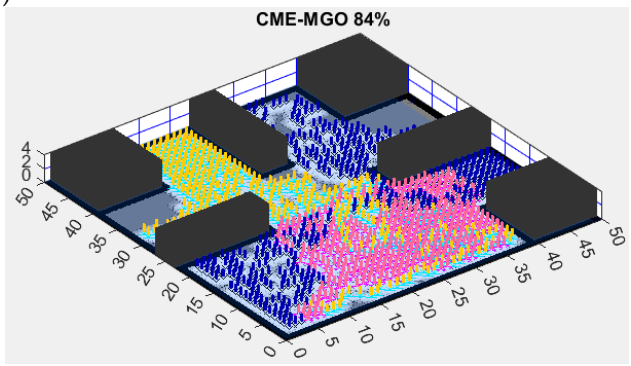
Figure 6. *Cont.*



(**c**)

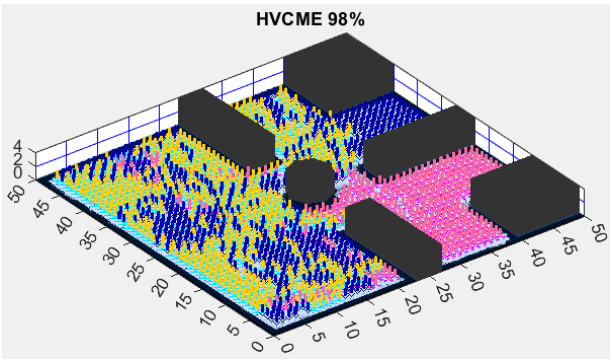


(**d**)

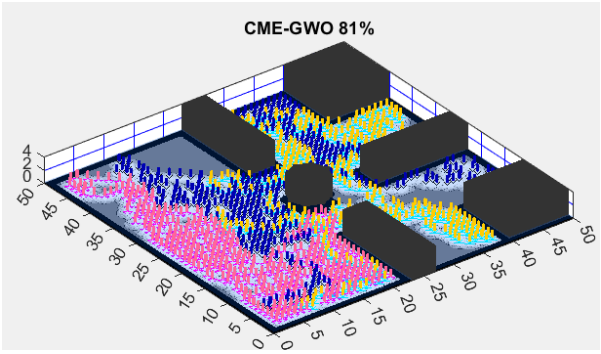


(**e**)

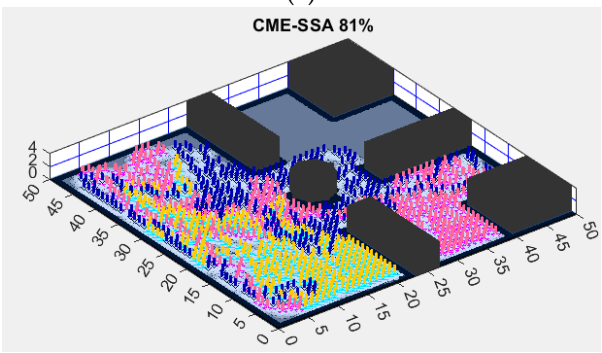
Figure 6. Comparing exploration efficiency of different algorithms on Complex Map 2 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.



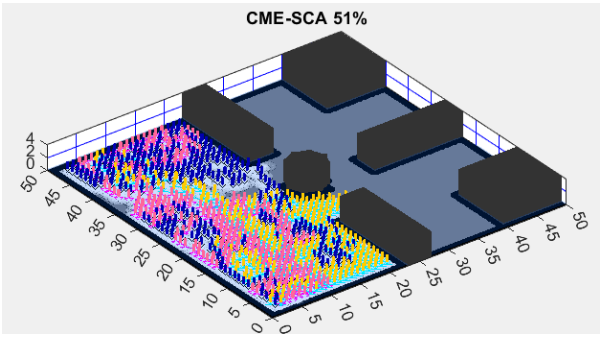
(**a**)



(**b**)

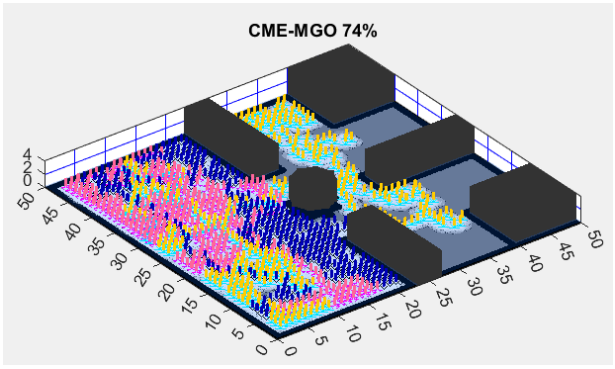


(**c**)



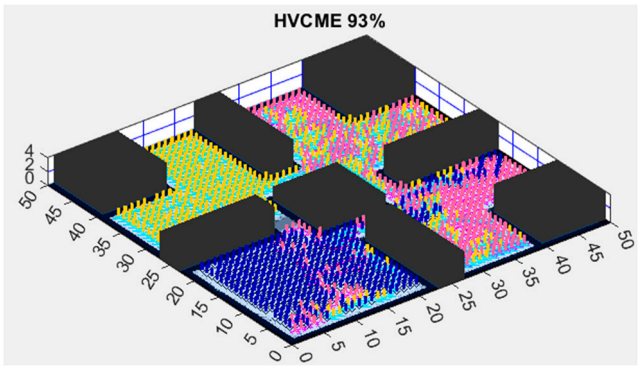
(**d**)

Figure 7. *Cont.*

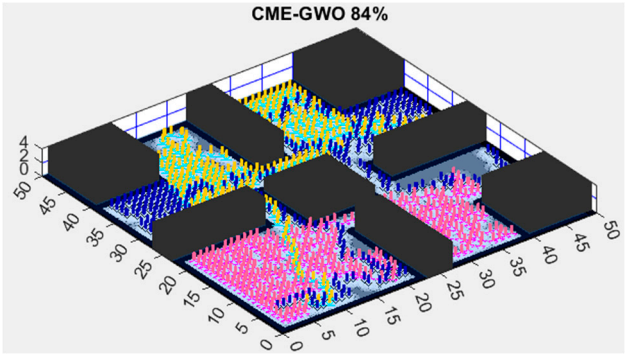


(**e**)

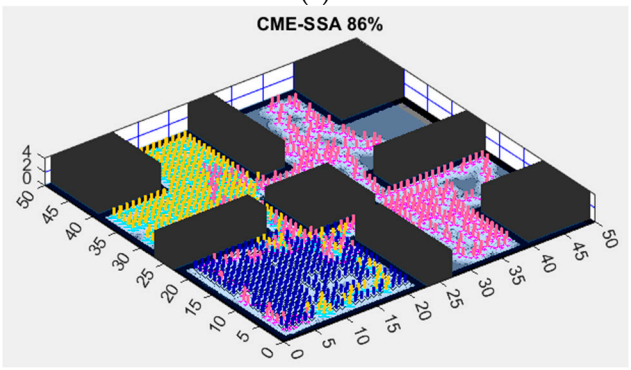
Figure 7. Comparing exploration efficiency of different algorithms on Complex Map 3 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.



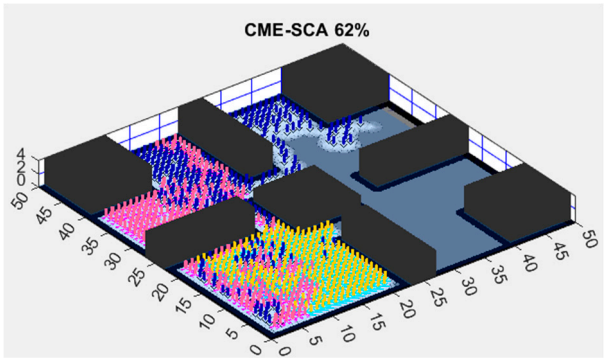
(**a**)



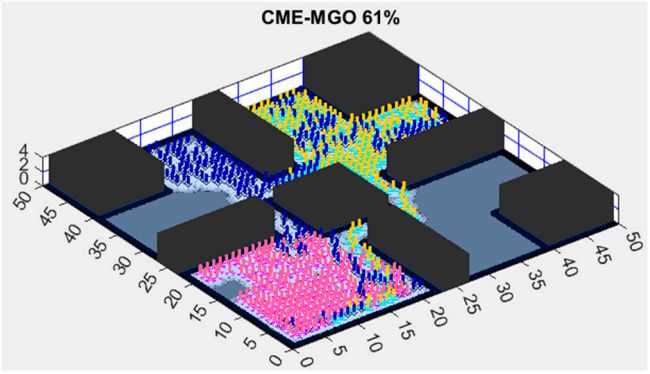
(**b**)



(**c**)



(**d**)



(**e**)

Figure 8. Comparing exploration efficiency of different algorithms on Complex Map 4 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.

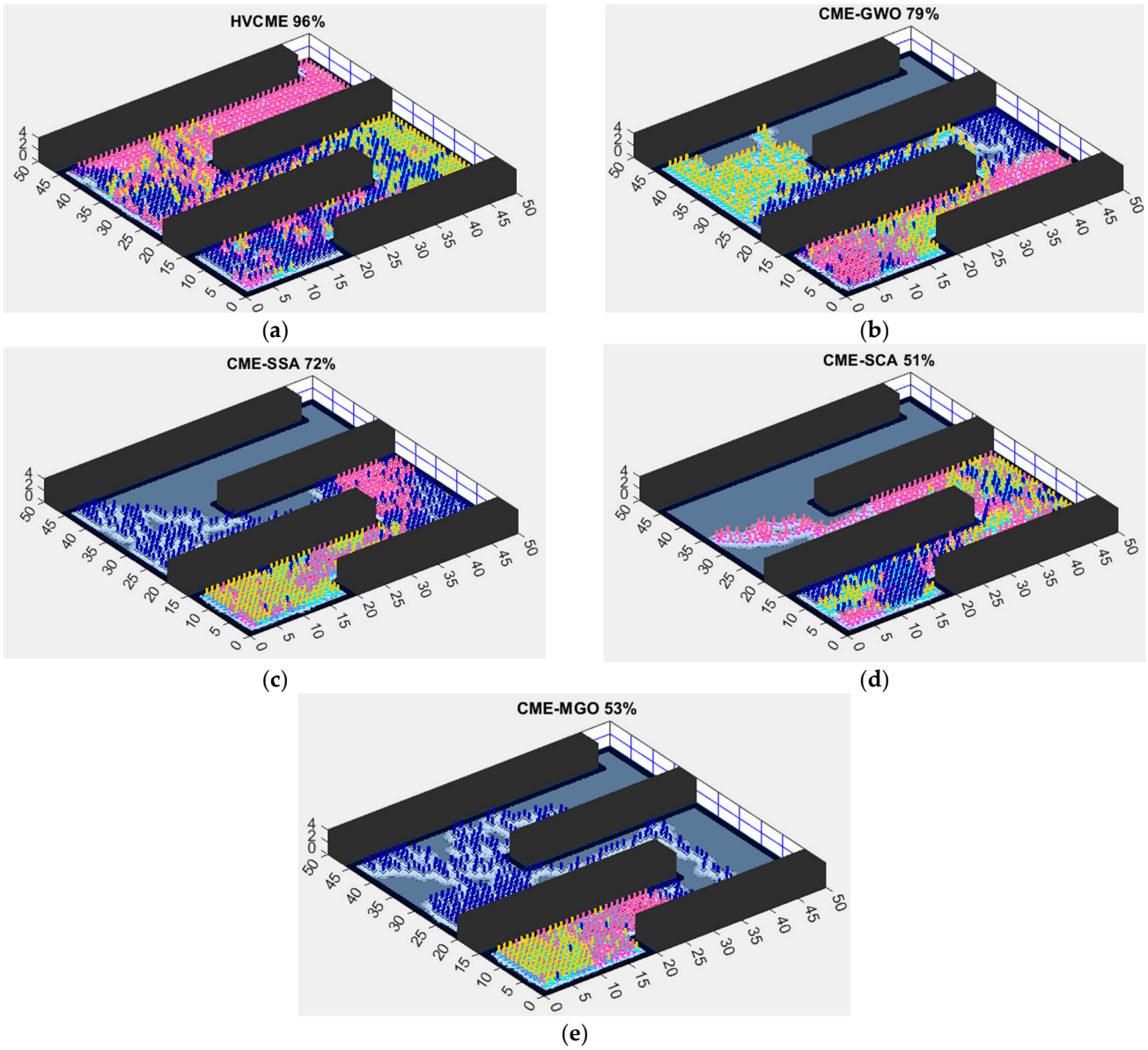


Figure 9. Comparing exploration efficiency of different algorithms on Complex Map 5 (**a**) HVCME; (**b**) $CME - GWO$; (**c**) $CME - SSA$; (**d**) $CME - SCA$; (**e**) $CME - MGO$.

Table 4 shows the percentage of the explored area for each algorithm to complete a full run consisting of 500 iterations on the 5 complex maps. The experiments were conducted on the HVCME algorithm and four other optimization methods. The table presents the average and standard deviation for each algorithm's explored area percentage on each map. The results show that the HVCME algorithm explored a higher percentage of the area on all five complex maps compared to the other methods. On average, the HVCME algorithm explored 94.51% of the area on Map 1, 96.01% on Map 2, 95.58% on Map 3, 92.96% on Map 4, and 94.35% on Map 5. In contrast, the other methods explored an average of 86.95% to 92.25% of the area on Map 1, 87.76% to 92.24% on Map 2, 86.79% to 95.76% on Map 3, 78.67% to 87.07% on Map 4, and 64.95% to 84.40% on Map 5. These findings indicate that the HVCME algorithm is effective in exploring a higher percentage of the area in complex environments compared to the other hybrid meta-heuristic methods.

Table 4. Percentage of explored area by each algorithm on five complex maps.

Complex Map	HVCME		*CME−GWO*		*CME−SSA*		*CME−SCA*		*CME−MGO*	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
Map 1	94.50890454	2.382410734	89.55102994	3.852037282	91.25102994	3.570840307	87.09321198	9.052147289	86.94423174	9.256605
Map 2	96.01468484	1.891439467	91.91375336	5.585904571	92.24708669	4.754568025	87.76453495	10.41421993	88.71989181	8.220601
Map 3	95.58057606	2.517066142	92.02424676	4.470404936	95.7575801	2.9997413	86.79440577	8.049828891	88.56805714	7.178603
Map 4	92.96318879	3.251433378	86.90087334	7.827569791	87.06754001	8.015598253	79.43808633	16.91861299	78.67348931	16.90669
Map 5	94.35310345	3.08846343	83.1999704	8.679611475	84.3999704	9.22311462	64.94504341	16.38479256	71.85112614	16.75737

Table 5 shows the duration, measured in seconds, for each algorithm to finish a complete run, consisting of 500 iterations on the five complex maps. The HVCME algorithm has the lowest average time taken for all the maps, ranging from 92.79 s for Map 1 to 92.94 s for Map 5. The other algorithms, $CME-GWO$, $CME-SSA$, $CME-SCA$, and $CME-MGO$, all have higher average times taken, ranging from 96.44 s to 99.05 s for Map 2 and from 100.16 s to 100.91 s for Map 4. The HVCME algorithm also has the lowest average and standard deviation for the time taken on all the maps, indicating that it is the fastest and more consistent in its performance than the other algorithms. Overall, the HVCME algorithm seems to be the most efficient algorithm when it comes to time taken to complete a full run on these complex maps.

Table 5. Time taken (in seconds) by different algorithms to complete a full run consisting of 500 iterations on five complex maps.

Complex Map	HVCME		*CME−GWO*		*CME−SSA*		*CME−SCA*		*CME−MGO*	
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*
***Map* 1**	92.79204967	0.275904459	97.261465	0.582357351	96.43628347	0.348885944	97.7564766	1.075883657	98.55007553	1.041942
***Map* 2**	93.2828055	0.15315418	97.35219077	1.01303978	96.50676813	0.256538034	99.6986837	1.182198858	99.04586097	1.836164
***Map* 3**	93.25613333	0.202724196	97.50659197	2.052379901	96.6835376	0.508972425	100.4113115	1.464349696	100.3491859	1.526962
***Map* 4**	93.25031283	0.226763813	97.3608672	1.238485844	96.9642087	0.576151537	100.9094861	1.247809351	100.1620379	1.581998
***Map* 5**	92.9428172	0.275405003	97.1399253	0.752556205	96.96096007	0.580188703	100.5302785	1.258368772	100.6206503	1.675212

Table 6 presents the number of unsuccessful simulations for various algorithms across five intricate maps. A simulation is considered unsuccessful if it cannot complete 500 iterations due to the presence of obstacles or other robots in neighboring cells. The HVCME algorithm had no failed simulations on the first two maps and the lowest number of failures on the remaining maps, ranging from one to three times compared to other algorithms. In contrast, $CME-GWO$ had failures on all maps, ranging from 5 to 101, while $CME-SSA$ had failures on maps 3, 4, and 5. $CME-SCA$ had failures on all maps, ranging from 13 to 417, and CME-MGO had failures on all five maps, ranging from 28 to 103. These results suggest that HVCME is the most robust algorithm, followed by $CME-GWO$ and $CME-SSA$, while $CME-SCA$ and $CME-MGO$ are less robust.

Table 6. Number of failed simulations for each algorithm across complex maps.

Complex Map	HVCME	*CME−GWO*	*CME−SSA*	*CME−SCA*	*CME−MGO*
Map 1	0	55	0	48	74
Map 2	0	5	0	15	88
Map 3	2	18	2	13	28
Map 4	1	101	5	133	54
Map 5	3	99	4	417	103

In this section, we conducted experiments on five complex maps that contained various types of narrow tunnels and corridors, with some of them being the most challenging due

to the high number of obstacles and small zones they contained. The objective was to compare the performance of the HVCME algorithm against other methods in terms of exploration area, time, and failures. The findings revealed that the HVCME algorithm surpassed the other techniques in the exploration percentage of all five intricate maps, while also demanding less time and encountering a reduced number of failures. This was demonstrated using statistical tables, including the average and standard deviation for exploration area and time, as well as a table displaying the number of failed simulations for each method Tables 4–6. In addition to the statistical analysis, we also presented visual representations of the simulation environments showing the explored area for each algorithm Figures 5–9. Overall, our findings indicate that the HVCME algorithm is a promising solution for complex mapping tasks in challenging environments.

4.3. Results, Analysis, and Discussion

In this research, we introduce the HVCME technique for independent multi-robot exploration in environments of varying complexity. The study employed qualitative findings to demonstrate extensive spatial exploration and quantitative assessments using two performance metrics, specifically average and standard deviation, to evaluate efficiency. For a comprehensive evaluation, these metrics were also utilized on four other algorithms, each completing 500 iterations independently, with 30 repetitions. The average metric evaluates the average area explored, while the standard deviation metric indicates the proposed method's stability compared to similar approaches (Tables 1–6). To further scrutinize individual runs, the Wilcoxon rank sum test, a statistical test, was employed for comparing and analyzing outcomes. Two hypotheses, $H0$ and $H1$, were established, where $H0$ posits that the proposed algorithm's exploration rate and time usage are inferior to the other four techniques, and $H1$ suggests that the proposed algorithm surpasses the other methods. A p-value of 0.05 or lower is considered statistically significant for null hypothesis rejection. To contrast the proposed algorithm with other techniques, the best-performing method's results were selected for each test function and compared in pairs with other methods. A consistent methodology was followed throughout the paper to ensure the dependability of the findings.

Table 7 shows the p-values obtained from the rank-sum test for exploration data results for different map types using HVCME, $CME-GWO$, $CME-SSA$, $CME-SCA$, and $CME-MGO$ algorithms. For simple maps, HVCME was not statistically significant compared to $CME-SSA$ on $Map\,1$ ($p = 0.085$), but it was statistically significant compared to $CME-GWO(p = 0.0013)$, $CME-SCA$ ($p = 0.0015$) and $CME-MGO$ ($p = 0.0184$) algorithms. For complex maps, HVCME was statistically significant compared to all other algorithms, with p-values ranging from 1.62×10^{-4} to 0.017. In particular, HVCME outperformed CME-SSA with a p-value of 1.62×10^{-4}, $CME-GWO$ with a p-value of 7.09×10^{-8}, $CME-SCA$ with a p-value of 0.0012, and $CME-MGO$ with a p-value of 2.57×10^{-7}. Therefore, HVCME outperformed the other algorithms in several exploration cases, as evidenced by the lower p-values.

Table 8 presents the p-values obtained from the Wilcoxon rank sum test for the time taken by each algorithm to complete a full run consisting of 500 iterations on 7 different maps. The results show that the p-values for HVCME are significantly lower than the other four algorithms, indicating that HVCME is the fastest algorithm among all. For instance, for Map 1, the p-values for HVCME range from 2.92×10^{-11} to 3.16×10^{-10}, while the p-values for the other algorithms range from 3.02×10^{-11} to 3.02×10^{-10}, which indicates that HVCME is significantly faster. Similar trends can be observed for the other maps as well. In conclusion, Table 8 indicates that HVCME outperforms the other four algorithms in terms of the time taken to complete a full run consisting of 500 iterations on 7 different maps.

Table 7. *p*-values for Wilcoxon rank sum test results on exploration data of different map types.

Map Type	*Map No*	**HVCME**	***CME−GWO***	***CME−SSA***	***CME−SCA***	***CME−MGO***
Simple	*Map* 1	N/A	0.001257099	0.085	0.0015	0.0184
	Map 2	N/A	2.69×10^{-6}	2.77×10^{-5}	1.31×10^{-8}	1.85×10^{-8}
Complex	*Map* 1	N/A	0.000396843	1.62×10^{-4}	5.27×10^{-5}	3.37×10^{-5}
	Map 2	N/A	7.09×10^{-8}	4.61×10^{-7}	0.0012	2.57×10^{-7}
	Map 3	0.017	1.78×10^{-4}	N/A	7.74×10^{-6}	3.32×10^{-6}
	Map 4	N/A	1.89×10^{-4}	0.001	0.008	9.79×10^{-5}
	Map 5	N/A	1.69×10^{-9}	9.01×10^{-7}	3.50×10^{-9}	8.35×10^{-8}

Table 8. *p*-values for time taken by different algorithms to complete 500 iterations on 7 different maps.

Map Type	*Map No*	**HVCME**	***CME−GWO***	***CME−SSA***	***CME−SCA***	***CME−MGO***
Simple	*Map* 1	N/A	3.02×10^{-11}	3.02×10^{-11}	3.16×10^{-10}	2.92×10^{-11}
	Map 2	N/A	2.62×10^{-11}	2.22×10^{-11}	2.87×10^{-10}	3.02×10^{-11}
Complex	*Map* 1	N/A	3.12×10^{-11}	3.62×10^{-11}	3.02×10^{-11}	3.69×10^{-11}
	Map 2	N/A	4.02×10^{-11}	3.52×10^{-11}	5.57×10^{-10}	3.02×10^{-11}
	Map 3	N/A	3.32×10^{-11}	2.92×10^{-11}	5.57×10^{-10}	5.57×10^{-10}
	Map 4	N/A	2.72×10^{-11}	3.99×10^{-11}	3.34×10^{-11}	3.51×10^{-11}
	Map 5	N/A	3.42×10^{-11}	3.86×10^{-11}	3.69×10^{-11}	3.82×10^{-10}

4.4. Usage Parameters of the Algorithms

In this subsection, we provide the usage parameters for the algorithms compared in our experiments: HVCME, CME-GWO, CME-SSA, CME-SCA, and CME-MGO. These parameters are crucial for understanding the performance of each algorithm and replicating our results.

As shown in Table 9, we have specified the parameters for each algorithm that were used in our experiments. These parameters were chosen based on the original algorithms AVOA [17], GWO [23], SSA [24], MGO [25], SCA [26], and our preliminary experiments to ensure a fair comparison between the algorithms. The population size represents the eight cells surrounds the robot Figure 1.

Table 9. Usage Parameters of Comparative Algorithms in Experimental Analysis.

Algorithms	**Parameter and Value**
General Parameters	Map dimensions: 50 m × 50 m Sensor ray length: 1.5 m Number of robots: 3 Number of iterations: 500 Number of runs: 30
HVCME (AVOA-based)	Population size: 8 α β γ parameters: [0, 1] Lévy flight step size (λ): 1 P1 probability: [0, 1] P2 probability: [0, 1] P3 probability: [0, 1]
CME-GWO	Population size: 8 α, β, and δ wolves a parameter [0–2]

Table 9. *Cont.*

Algorithms	Parameter and Value
CME-SSA	Population size: 8 C1 coefficient: [0, 1] C2 coefficient: [0, 1]
CME-SCA	Population size: 8 a parameter: [2, 0] r1 and r2 random numbers: [0, 1] A, B, C, and D updating strategies
CME-MGO	Population size : 8 ri_{1-6} random 1 or 2 6 random numbers r: [0, 1]

4.5. Analysis Results Summary

The present study summarizes the findings of Section 4 in a succinct and informative manner for readers. Tables 1–6 serve as a comprehensive collection of results, depicting various metrics that compare the performance of the proposed hybrid method HVCME with four other approaches, namely $CME - GWO$, $CME - SSA$, $CME - SCA$, and $CME - MGO$. Tables 1 and 4 illustrate the exploration effectiveness of these approaches, while Tables 2 and 5 provide an overview of their time consumption results. Furthermore, Tables 3 and 6 contain information on the number of unsuccessful simulations for each algorithm. The accompanying visuals depict the seven maps of varying complexities, and the tables reference these figures to provide additional details regarding the simulations.

The results show that HVCME offers superior exploration and speed averages compared to other methods, as indicated by the lower mean indicator values. Additionally, HVCME demonstrates stability in exploration and time consumption, evident from the lower standard deviations. The Wilcoxon rank sum test confirms the statistical significance of HVCME$'s$ superiority over other methods in terms of exploration and time consumption. It is worth noting that HVCME can complete a full run with fewer attempts compared to the other four methods, which require multiple trials.

Moreover, time consumption is an essential metric for evaluating algorithm effectiveness, with the primary goal of completing a task in the shortest possible time. HVCME$'s$ time consumption is computed and compared with the other four methods in Tables 2 and 5, with the results indicating that HVCME is computationally efficient, taking less time to maximize the entire space explored. In contrast, the other techniques take more time and explore lesser. Overall, these findings support the potential of HVCME as a superior approach to exploring complex search spaces while optimizing computational efficiency.

4.6. Implementation and Deployment of a Laser-Based Navigation System Using MATLAB and ROS for Turtlebot Robots

The technique presented in this study was implemented using MATLAB's Robotic System Toolbox and Navigation Toolbox, and the simulation was performed in a virtual environment. If the approach were to be employed in the physical world, a Turtlebot [46] could be utilized along with a Hokuyo laser range scanner [47] also a Tablet or laptop equipped with the Robotic System Toolbox to establish a connection between MATLAB and the robot operating system (ROS) [48]. The sensor data, ranging from 240 to 360 vision degrees, would be transmitted to MATLAB, where HVCME would calculate the next move based on the sensor input. The system does not employ any external filter, and there may be some unknown measurement noise. To ensure connectivity between the robot and the PC, strong Wi-Fi signals can be utilized, depending on the size of the indoor space that requires exploration. Several new frameworks have been designed to limit the robot's observation error uniformly, achieve finite time convergence, and reconstruct external disturbances and uncertainties [49,50].

5. Conclusions

In this paper, we proposed a novel approach for optimizing the construction of a map in an unknown indoor environment using multiple robots, called Hybrid Vulture-Coordinated Multi-Robot Exploration (HVCME). Our experimental results demonstrate that the proposed HVCME surpassed the other techniques in the exploration percentage of all five intricate maps, while also demanding less time and encountering a reduced number of failures. By combining CME and AVOA, HVCME shows promising potential for optimizing the exploration of unknown environments using multiple robots.

Our study provides a valuable contribution to the field of multi-robot exploration, as it suggests that the HVCME algorithm could be applied to various fields where exploring unknown environments is a challenging task. The proposed HVCME algorithm not only resulted in better performance in terms of the percentage of the explored area and run time, but it also showed better robustness by encountering a reduced number of failures when it comes to completing a successful run. Therefore, we conclude that the proposed HVCME is a robust and efficient technique for optimizing the construction of a finite map in an unknown indoor environment using multiple robots.

As for future work, we plan to conduct research on multi-robot exploration using a multi-objective meta-heuristic algorithm. The objectives of this study are twofold: firstly, to explore novel locations and secondly, to enhance map accuracy by ensuring that the robots do not revisit previously explored cells repeatedly. This future work will build upon the findings of our study and further enhance the efficiency and effectiveness of multi-robot exploration. We believe that our proposed HVCME algorithm and future work will contribute significantly to the advancement of the field of multi-robot exploration.

Author Contributions: Conceptualization, A.E.R.; methodology, A.E.R.; validation, A.E.R. and S.M.; formal analysis, A.E.R. and S.M.; investigation, A.E.R. and S.M.; resources, A.E.R. and S.M.; data curation, A.E.R. and S.M.; writing, A.E.R. and S.M.; original draft preparation, A.E.R. and S.M.; writing—review and editing, A.E.R., S.M. and F.G.; visualization, A.E.R. and S.M.; supervision, S.M.; project administration, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data and code used in the research may be obtained from the corresponding author upon request.

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

References

1. Queralta, J.P.; Taipalmaa, J.; Pullinen, B.C.; Sarker, V.K.; Gia, T.N.; Tenhunen, H.; Gabbouj, M.; Raitoharju, J.; Westerlund, T. Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision. *IEEE Access* **2020**, *8*, 191617–191643. [CrossRef]
2. Chang, Y.; Ebadi, K.; Denniston, C.E.; Ginting, M.F.; Rosinol, A.; Reinke, A.; Palieri, M.; Shi, J.; Chatterjee, A.; Morrell, B.; et al. LAMP 2.0: A Robust Multi-Robot SLAM System for Operation in Challenging Large-Scale Underground Environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 9175–9182. [CrossRef]
3. Alitappeh, R.J.; Jeddisaravi, K. Multi-robot exploration in task allocation problem. *Appl. Intell.* **2021**, *52*, 2189–2211. [CrossRef]
4. Habibian, S.; Dadvar, M.; Peykari, B.; Hosseini, A.; Salehzadeh, M.H.; Najafi, F. Design and implementation of a maxi-sized mobile robot (Karo) for rescue missions. *ROBOMECH J.* **2021**, *8*, 1. [CrossRef]
5. Dutta, A.; Roy, S.; Kreidl, O.P.; Boloni, L. Multi-Robot Information Gathering for Precision Agriculture: Current State, Scope, and Challenges. *IEEE Access* **2021**, *9*, 161416–161430. [CrossRef]
6. Liu, J.; Zhou, L.; Tokekar, P.; Williams, R.K. Distributed Resilient Submodular Action Selection in Adversarial Environments. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5832–5839. [CrossRef]
7. Zhang, H.; Cheng, J.; Zhang, L.; Li, Y.; Zhang, W. H2GNN: Hierarchical-Hops Graph Neural Networks for Multi-Robot Exploration in Unknown Environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3435–3442. [CrossRef]
8. Papachristos, C.; Khattak, S.; Mascarich, F.; Alexis, K. Autonomous Navigation and Mapping in Underground Mines Using Aerial Robots. In Proceedings of the 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2019; pp. 1–8. [CrossRef]

9. Wang, M.; Du, L.; Yuan, J.; Ma, S.; Bao, S. A bio-inspired continuum robot for out-pipe climbing and confined space navigating. In Proceedings of the 2021 IEEE International Conference on Robotics and Biomimetics (ROBIO), Sanya, China, 27–31 December 2021; pp. 74–79. [CrossRef]
10. Schuster, M.J.; Muller, M.G.; Brunner, S.G.; Lehner, H.; Lehner, P.; Sakagami, R.; Domel, A.; Meyer, L.; Vodermayer, B.; Giubilato, R.; et al. The ARCHES Space-Analogue Demonstration Mission: Towards Heterogeneous Teams of Autonomous Robots for Collaborative Scientific Sampling in Planetary Exploration. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5315–5322. [CrossRef]
11. Huang, Y.; Wu, S.; Mu, Z.; Long, X.; Chu, S.; Zhao, G. A Multi-agent Reinforcement Learning Method for Swarm Robots in Space Collaborative Exploration. In Proceedings of the 2020 6th International Conference on Control, Automation and Robotics (ICCAR), Singapore, 20–23 April 2020; pp. 139–144. [CrossRef]
12. Honkote, V.; Ghosh, D.; Narayanan, K.; Gupta, A.; Srinivasan, A. Design and Integration of a Distributed, Autonomous and Collaborative Multi-Robot System for Exploration in Unknown Environments. In Proceedings of the 2020 IEEE/SICE International Symposium on System Integration (SII), Honolulu, HI, USA, 12–15 January 2020; pp. 1232–1237. [CrossRef]
13. Bandyopadhyay, S.; Chung, S.-J.; Hadaegh, F.Y. Probabilistic and Distributed Control of a Large-Scale Swarm of Autonomous Agents. *IEEE Trans. Robot.* **2017**, *33*, 1103–1123. [CrossRef]
14. Darmanin, R.N.; Bugeja, M.K. A review on multi-robot systems categorised by application domain. In Proceedings of the 2017 25th Mediterranean Conference on Control and Automation (MED), Valletta, Malta, 3–6 July 2017; pp. 701–706. [CrossRef]
15. Raibail, M.; Rahman, A.H.A.; Al-Anizy, G.J.; Nasrudin, M.F.; Nadzir, M.S.M.; Noraini, N.M.R.; Yee, T.S. Decentralized Multi-Robot Collision Avoidance: A Systematic Review from 2015 to 2021. *Symmetry* **2022**, *14*, 610. [CrossRef]
16. Burgard, W.; Moors, M.; Stachniss, C.; Schneider, F. Coordinated multi-robot exploration. *IEEE Trans. Robot.* **2005**, *21*, 376–386. [CrossRef]
17. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
18. Ji, K.; Zhang, Q.; Yuan, Z.; Cheng, H.; Yu, D. A virtual force interaction scheme for multi-robot environment monitoring. *Robot. Auton. Syst.* **2022**, *149*, 103967. [CrossRef]
19. Su, Q.; Yu, W.; Liu, J. Mobile Robot Path Planning Based on Improved Ant Colony Algorithm. In Proceedings of the 2021 Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), Shenyang, China, 22–24 January 2021; pp. 220–224. [CrossRef]
20. Pires, A.G.; Rezeck, P.A.F.; Chaves, R.A.; Macharet, D.G.; Chaimowicz, L. Cooperative Localization and Mapping with Robotic Swarms. *J. Intell. Robot. Syst.* **2021**, *102*, 47. [CrossRef]
21. Mendonça, M.; Palácios, R.H.C.; Papageorgiou, E.I.; de Souza, L.B. Multi-robot exploration using Dynamic Fuzzy Cognitive Maps and Ant Colony Optimization. In Proceedings of the 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
22. Das, P.K.; Behera, H.S.; Panigrahi, B.K. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm Evol. Comput.* **2016**, *28*, 14–28. [CrossRef]
23. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
24. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp swarm algorithm: A bio-inspired opti-mizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
25. Abdollahzadeh, B.; Gharehchopogh, F.S.; Khodadadi, N.; Mirjalili, S. Mountain Gazelle Optimizer: A new Nature-inspired Metaheuristic Algorithm for Global Optimization Problems. *Adv. Eng. Softw.* **2022**, *174*, 103282. [CrossRef]
26. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
27. St-Onge, D.; Levillain, F.; Zibetti, E.; Beltrame, G. Collective expression: How robotic swarms convey information with group motion. *Paladyn, J. Behav. Robot.* **2019**, *10*, 418–435. [CrossRef]
28. Mir, I.; Taha, H.; Eisa, S.A.; Maqsood, A. A controllability perspective of dynamic soaring. *Nonlinear Dyn.* **2018**, *94*, 2347–2362. [CrossRef]
29. Galceran, E.; Carreras, M. A survey on coverage path planning for robotics. *Robot. Auton. Syst.* **2013**, *61*, 1258–1276. [CrossRef]
30. Wang, X.; Syrmos, V.L. Coverage path planning for multiple robotic agent-based inspection of an unknown 2D environment. In Proceedings of the 2009 17th Mediterranean Conference on Control and Automation, Thessaloniki, Greece, 24–26 June 2009; pp. 1295–1300. [CrossRef]
31. Andries, M.; Charpillet, F. Multi-robot taboo-list exploration of unknown structured environments. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 5195–5201. [CrossRef]
32. Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm Intelligence—From Natural to Artificial Systems*; Oxford University Press: Oxford, UK, 1999.
33. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
34. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
35. Albina, K.; Lee, S.G. Hybrid Stochastic Exploration Using Grey Wolf Optimizer and Coordinated Multi-Robot Exploration Algorithms. *IEEE Access* **2019**, *7*, 14246–14255. [CrossRef]

36. Gul, F.; Mir, I.; Rahiman, W.; Islam, T.U. Novel Implementation of Multi-Robot Space Exploration Utilizing Coordinated Multi-Robot Exploration and Frequency Modified Whale Optimization Algorithm. *IEEE Access* **2021**, *9*, 22774–22787. [CrossRef]
37. Gul, F.; Mir, I.; Mir, S. Aquila Optimizer with parallel computing strategy for efficient environment exploration. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 4175–4190. [CrossRef]
38. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P. Multi-Robot Space Exploration: An Augmented Arithmetic Approach. *IEEE Access* **2021**, *9*, 107738–107750. [CrossRef]
39. El Romeh, A.; Mirjalili, S. Multi-Robot Exploration of Unknown Space Using Combined Meta-Heuristic Salp Swarm Algorithm and Deterministic Coordinated Multi-Robot Exploration. *Sensors* **2023**, *23*, 2156. [CrossRef]
40. Smith, A.J.; Hollinger, G.A. Distributed inference-based multi-robot exploration. *Auton. Robot.* **2018**, *42*, 1651–1668. [CrossRef]
41. Lumelsky, V.; Mukhopadhyay, S.; Sun, K. Dynamic path planning in sensor-based terrain acquisition. *IEEE Trans. Robot. Autom.* **1990**, *6*, 462–472. [CrossRef]
42. Rajesh, M.; Jose, G.R.; Sudarshan, T.S.B. Multi-robot exploration and mapping using frontier cell concept. In Proceedings of the 2014 Annual IEEE India Conference (INDICON), Pune, India, 11–13 December 2014; pp. 1–6. [CrossRef]
43. Gao, S.; Ding, Y.; Chen, B.M. A Frontier-Based Coverage Path Planning Algorithm for Robot Exploration in Unknown Environment. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 3920–3925. [CrossRef]
44. Yamauchi, B. Frontier-based exploration using multiple robots. In Proceedings of the second international conference on Au-tonomous agents—AGENTS'98, Minneapolis, MN, USA, 10–13 May 1998. [CrossRef]
45. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
46. Koubâa, A.; Sriti, M.-F.; Javed, Y.; Alajlan, M.; Qureshi, B.; Ellouze, F.; Mahmoud, A. Turtlebot at Office: A Service-Oriented Software Architecture for Personal Assistant Robots Using ROS. In Proceedings of the 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), Braganca, Portugal, 4–6 May 2016; pp. 270–276. [CrossRef]
47. Kneip, L.; Tache, F.; Caprari, G.; Siegwart, R. Characterization of the compact Hokuyo URG-04LX 2D laser range scanner. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 1447–1454. [CrossRef]
48. Galli, M.; Barber, R.; Garrido, S.; Moreno, L. Path planning using Matlab-ROS integration applied to mobile robots. In Proceedings of the 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Coimbra, Portugal, 26–28 April 2017; pp. 98–103. [CrossRef]
49. Razmjooei, H.; Palli, G.; Abdi, E.; Terzo, M.; Strano, S. Design and experimental validation of an adaptive fast-finite-time observer on uncertain electro-hydraulic systems. *Control. Eng. Pract.* **2023**, *131*, 105391. [CrossRef]
50. Razmjooei, H.; Palli, G.; Abdi, E. Continuous finite-time extended state observer design for electro-hydraulic systems. *J. Frankl. Inst.* **2022**, *359*, 5036–5055. [CrossRef]