applied sciences

MDPI

Article

Area Division Using Affinity Propagation for Multi-Robot Coverage Path Planning

Nikolaos Baras * and Minas Dasygenis

Department of Electrical and Computer Engineering, University of Western Macedonia, 50100 Kozani, Greece
* Correspondence: nbaras@uowm.gr

Abstract: In the wake of advancing technology, autonomous vehicles and robotic systems have burgeoned in popularity across a spectrum of applications ranging from mapping and agriculture to reconnaissance missions. These practical implementations have brought to light an array of scientific challenges, a crucial one among them being Coverage Path Planning (CPP). CPP, the strategic planning of a path that ensures comprehensive coverage of a defined area, while being widely examined in the context of a single-robot system, has found its complexity magnified in the multi-robot scenario. A prime hurdle in multi-robot CPP is the division and allocation of the operation area among the robots. Traditional methods, largely reliant on the number of robots and their initial positions to segment the space, often culminate in suboptimal area division. This deficiency can occasionally render the problem unsolvable due to the sensitivity of most area division algorithms to the robots' starting points. Addressing this predicament, our research introduced an innovative methodology that employs Affinity Propagation (AP) for area allocation in multi-robot CPP. In our approach, the area is partitioned into 'n' clusters through AP, with each cluster subsequently assigned to a robot. Although the model operates under the assumption of an unlimited robot count, it offers flexibility during execution, allowing the user to modify the AP algorithm's similarity function factor to regulate the number of generated clusters. Serving as a significant progression in multi-robot CPP, the proposed model provides an innovative approach to area division and path optimization, thereby setting a strong foundation for future exploration and practical enhancements in this field.

Keywords: affinity propagation; area allocation; coverage path planning

check for updates

Citation: Baras, N.; Dasygenis, M. Area Division Using Affinity Propagation for Multi-Robot Coverage Path Planning. *Appl. Sci.* **2023**, *13*, 8207. https://doi.org/10.3390/app13148207

Academic Editor: Jonghoek Kim

Received: 3 June 2023
Revised: 4 July 2023
Accepted: 10 July 2023
Published: 14 July 2023

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

1. Introduction

Over the past few decades, we have witnessed an exponential advancement in technology that has fundamentally reshaped the global landscape [1]. With a broad gamut ranging from telecommunications and computing to artificial intelligence and robotics, these technological breakthroughs have drastically altered the ways in which humans operate and perform tasks [1–3]. Once labor-intensive or monotonous tasks are now executed with unparalleled speed, precision, and efficiency, largely due to automation and the advent of intelligent systems.

A predominant player in this transformational journey has been the field of robotics. Robots, once confined to the realms of science fiction, now pervade numerous sectors, such as manufacturing [4], healthcare [5], agriculture [6–8], and logistics [9], serving as invaluable tools for enhancing productivity and improving the quality of services. In disaster management, robots have been instrumental in executing search-and-rescue missions in environments too hazardous for human intervention [10,11]. In agriculture, robotic systems are leveraged for tasks ranging from seeding to harvesting, contributing significantly to precision farming.

However, as with any emergent technology, the rapid rise of robotics has posed a unique set of challenges, requiring innovative solutions. A prominent challenge in the domain of robotics is Coverage Path Planning (CPP). In essence, CPP is the process of

developing a route or path that allows a robot to cover an entire operational area in an efficient manner [12,13]. A vital aspect of numerous applications, CPP is integral to tasks such as field inspection in agriculture [14–16], where a robotic system needs to cover a farm field to assess crop health.

The complexity of CPP intensifies when we move from single-robot systems to multi-robot systems. Multi-robot CPP involves devising paths for multiple robots to ensure comprehensive and efficient coverage of a larger or more complex area [17–26]. A pressing issue in multi-robot CPP is the division of the operational area among the robots, a problem known as the 'area division problem'. For example, consider a team of drones deployed for large-scale environmental monitoring or a fleet of autonomous vehicles performing a search-and-rescue mission in a disaster-stricken area. The optimal allocation of specific regions to individual robots can drastically enhance the coverage efficiency and operational coordination, reducing redundancies and saving valuable time. Addressing the area division problem effectively is paramount for the successful deployment of multi-robot systems across a range of real-world applications. As such, it remains a vibrant area of research, inviting novel solutions and methodologies that can meet the evolving demands of modern robotic systems.

Several researchers have proposed models to resolve the multi-robot CPP problem, yet they often encounter significant limitations. A prevalent limitation of multi-robot CPP algorithms lies in the arbitrary selection of the number of robots and their starting locations. Both these factors wield substantial influence over the efficacy of the algorithm. For example, an inadequate number of robots may yield incomplete coverage, resulting in overlooked regions. Conversely, deploying an excessive number of robots can render the task inefficient and unnecessarily complicated, thus elevating the cost and computational burden. Likewise, the robots' initial placement crucially impacts the effectiveness of the algorithm, dictating the distribution of robots and the thoroughness of area coverage. Another major constraint is the limited scalability of the algorithms, referring to the capability of an algorithm to perform optimally as the number of inputs or variables increases. In the context of multi-robot CPP, scalability pertains to the algorithm's ability to manage expansive environments and multiple robots cooperating to execute a task.

This paper presents a significant contribution to the field of multi-robot CPP by introducing a novel model for area division predicated on the Affinity Propagation (AP) algorithm. The fundamental premise of the proposed approach is to deconstruct the multi-robot CPP problem into a collection of single-robot CPP tasks, which may or may not be interdependent. While numerous models in the literature adopt a similar divide-and-conquer strategy, our model distinguishes itself in three vital respects:

(i) The proposed approach assigns territory to robots without arbitrarily defining the number of robots (and hence the number of areas). The AP clustering algorithm leverages a weighted similarity index (SI) function to discern similar cells and ascertain the optimal number of robots for the multi-robot CPP task. This effectively liberates the user from the responsibility of determining the appropriate number of robots for the task. To our knowledge, we are the first to incorporate an AP algorithm to address the multi-robot CPP problem.

(ii) The algorithm proposed in this paper presents a novel perspective on multi-robot area division, where the number of robots and their respective areas are not predesignated arbitrarily by the user. A common drawback of many existing algorithms is their inability to handle scenarios where the robots' initial positions are in close proximity, leading to inefficient area division and path planning. Unlike these approaches, our proposed algorithm demonstrates exceptional resilience to such situations. It intelligently circumvents the constraints of predefined robot counts and positions, thereby offering a more flexible and efficient solution for area division in multi-robot coverage path planning.

(iii) The similarity function of the AP algorithm takes into consideration several factors, such as layer type, the spatial connectivity of cells, and their "normalized distance",

before incorporating them into a cluster. This results in reduced area blending rates, meaning that the cells within the final clusters created by this method are less likely to be enveloped by cells from other clusters.

The structure of the remaining manuscript is as follows:

Section 2 of the document introduces similar works from the literature and highlights their advantages and disadvantages. Following that, in Section 3, the formal problem definition is provided. The subsequent section, Section 4, elaborates on the algorithm we propose. Section 5 presents the experimental data and techniques employed to assess the effectiveness of our algorithm. The algorithm's limitations, possible performance improvements, and the usage of parallel computing are discussed in Section 6. Finally, Section 7 contains concluding remarks.

2. Literature Review

This section will delve into research endeavors in the literature that confront the multi-robot coverage path planning problem, particularly emphasizing area division and allocation. The objective of this section is to furnish a comprehensive review of extant algorithms and methodologies employed to mitigate the problem of multi-robot coverage. An analysis, comparison, and critical evaluation of related works were conducted to highlight their respective merits and demerits, thereby establishing a foundation for assessing the scientific contribution and novelty of the model proposed in this paper and identifying lacunae in the prevailing state-of-the-art.

A widely recognized offline multi-robot CPP algorithm is the one proposed by Tang et al. [21] using the MSTC* framework. This algorithm primarily aims to generate coverage paths for multiple robots while considering realistic physical constraints such as obstacles and communication paths among robots. The MST technique, employed to divide the target environment into smaller sub-regions, forms the backbone of the algorithm. Robots are then allocated to these sub-areas according to their capabilities and workload requirements. This algorithm possesses the advantage of addressing physical constraints, a pivotal aspect in real-world scenarios where robots need to maneuver through complex and dynamic environments. However, the algorithm does not consider environmental uncertainties, leading to sub-optimal performance in volatile and dynamic situations. Further, it predetermines the number of robots, thereby deciding the number of sub-areas. It is notable that Tang et al.'s problem formulation aligns with the problem formulated in this paper in terms of area division and allocation, although the specific problem details the authors address are distinct.

Another noteworthy offline multi-robot CPP algorithm is the one developed by Rahman et al. [24] for autonomous radiation mapping using a mobile robot. The primary objective of this algorithm is to create coverage paths that a single robot can traverse to conduct radiation mapping in a designated area. The method draws from a genetic algorithm, utilized to generate a multitude of potential coverage paths, and, subsequently, the optimal path is selected based on criteria such as coverage efficiency and path length. The flexibility of this algorithm lies in its adaptability; it can be modified to cater to varying radiation mapping scenarios. Moreover, the algorithm's computational efficiency makes it suitable for extensive application. However, it uses the K-means clustering technique to partition the overall space into smaller sub-spaces, implying that the algorithm necessitates a completely arbitrary selection of the number of robots.

The DARP algorithm [18] represents a notable approach in the domain of multi-robot coverage path planning. It offers a systematic solution by dividing the total environment into distinct sub-areas, each allocated to a specific robot. The primary objective of DARP is to minimize the total coverage time, accomplished by intelligently dividing the environment based on its characteristics and the robotic fleet's capabilities. However, the algorithm makes predetermined assumptions about the number of robots and their initial positions, leading to potential limitations in more complex environments.

An interesting approach was proposed by Idir et al. [19], who suggested a multi-robot CPP algorithm based on the DARP algorithm. The authors sought to tackle the problem of weight allocation for multi-robot coverage using a weighted approach. The algorithm employs a grid-based representation of the environment and segregates the environment into a collection of cells necessitating coverage. The robots are assigned weights and the DARP algorithm is utilized to distribute these weights among the robots and determine their coverage paths. The strengths of this approach encompass handling the weight allocation problem, thereby enhancing coverage performance, and the ability to manage large-scale scenarios. Despite its improvements over the original DARP algorithm, it retains the limitation of being unable to find a solution when the initial positions of the robots are proximate.

Another DARP-based algorithm was proposed in [25]. The authors presented an A*-modified DARP algorithm. This modified version of the DARP algorithm assigns tasks to the appropriate robot and based on an Up-First approach the Spanning Trees are constructed in order to ensued full coverage of the initial area. The authors claim that, compared to the original DARP algorithm, their modifications yield higher efficiency and a higher coverage rate.

A different approach for multi-robot CPP was proposed in [26] that uses Ant Colony Optimization. The authors introduced an improved Ant Colony Optimization (ACO) algorithm for single-robot CPP, which optimizes the energy and time consumption by building the best possible Spanning Tree (ST) and, consequently, an optimal path. For multi-robot scenarios, the study employed the DARP algorithm [18], dividing the total area into smaller, equally sized sections, thus simplifying the complex computation. Each subarea then constructs a spanning tree using the improved ACO. In the final stage, the end nodes are shared among subareas to develop ideally-shaped spanning trees that minimize the number of turns in the coverage path. The algorithms are proven to be near polynomial, and simulation results highlight benefits including complete coverage, no backtracks, minimum path length, zero preparation time, and the least number of turns. However, the implementation of this approach still suffers from the drawbacks of the DARP algorithm, meaning the arbitrary choice of robot's initial positions.

Given the literature review, it is evident that multi-robot CPP has garnered considerable attention from researchers. However, a direct evaluation and comparison of these research works is challenging as each paper addresses a subtly different problem. A commonality that most offline multi-robot CPP algorithms share, including those mentioned above, is their reliance on arbitrarily defining the number of robots and sensitivity to the initial environmental conditions. Further research is imperative to devise new multi-robot CPP algorithms that are devoid of these limitations, are more efficient, and tailored for real-world applications. The Affinity Propagation algorithm proposed in this paper demonstrates the potential to address these issues, and thus presents a significant advancement in this field.

3. Problem Definition

The effective division of a given environment into distinct sub-areas for the deployment of multiple robots presents a challenging problem, particularly in the context of ensuring contiguous access within each sub-region. To lay a solid foundation for our investigation, we embarked on a mathematical formulation of this problem, succinctly capturing the essential aspects and constraints involved. This mathematical model serves as an unambiguous representation of the underlying problem, illuminating its core elements and thus guiding the development of algorithmic solutions.

Our environment is described as a binary matrix representation, defining accessible areas and obstacles. We considered robots that are assigned to different sections of this environment, each section described as a 'sub-area'. We also incorporated the distinct nature of the terrain, assigning different types to each cell in the environment. The fundamental goal was to find a valid division of the environment into sub-areas, each assigned to a

unique robot, with the sub-areas satisfying a set of constraints related to accessibility and continuity. The mathematical model elucidated below presents a precise description of the problem, facilitating further discussion and solution design.

Given:

- An environment A of size $X \times Y$ represented as a binary matrix, $A = [a_{i,j}]$, where $a_{i,j} \in \{0,1\}$ for all $1 \leq i \leq X$ and $1 \leq j \leq Y$. In the environment matrix A, $a_{i,j} = 0$ signifies an obstacle, whereas $a_{i,j} = 1$ represents an accessible cell. By definition, robots can only traverse accessible cells.
- A set of robots $R = \{r_1, r_2, \ldots, r_n\}$, where n is the number of robots.
- A matrix $T = [t_{i,j}]$, where, that describes the type of each cell $a_{i,j}$.
- A matrix $E = [e_{i,j}]$, where, that describes the elevation level of each cell $a_{i,j}$.
- An elevation weight value E_W which represents the elevation weight importance factor, and a floor type F_W which represents the floor type elevation weight importance factor.
- A matrix $T_w = [w_1, w_2, \ldots, w_n]$ where w_i represents the weight assigned to the i th floor type. The number of values in T_w is equal to the number of different floor types. Based on this information, we seek:
- A set of sub-areas $S = \{s_1, s_2, \ldots, s_n\}$, where each sub-area s_i is a contiguous partition of A assigned to robot r_i. Formally, we define the following constraints:
- Each sub-area s_i maintains 4-neighbor continuity for all its accessible cells (Figure 1), i.e., for each pair of accessible cells c_{m_1,n_1} and c_{m_2,n_2} in s_i, there exists a sequence of accessible cells $c_{m_k,n_k}, k = 1, \ldots, p$, such that $c_{m_1,n_1} = c_{m_1,n_1}, c_{m_p,n_p} = c_{m_2,n_2}$, and c_{m_k,n_k} is a 4-neighbor of $c_{m_{k+1},n_{k+1}}$ for all $k = 1, \ldots, p-1$. The 4-neighbor criterion specifies that connectivity between cells must be either horizontal or vertical—not diagonal. Thus, in essence, every cell in the sequence is horizontally or vertically adjacent to its successor, for all values of k ranging from 1 to $p-1$.
- This condition ensures a continuity or chain of 4-neighbor connections between any two accessible cells within a given sub-area, thereby preserving the rule of 4-neighbor connectivity throughout the entire grid.
- The environment A is the union of all sub-areas S, i.e., $A = \mathrm{U}_{i=1}^{n} s_i$.
- The intersection of any two distinct sub-areas s_i and s_j for $i \neq j$ is empty, i.e., $s_i \cap s_j = \varnothing$ for all $i \neq j$.
- The problem is to find a bijective function $f : R \to S$ such that $f(r_i) = s_i$ for all $1 \leq i \leq n$, fulfilling the constraints mentioned above.

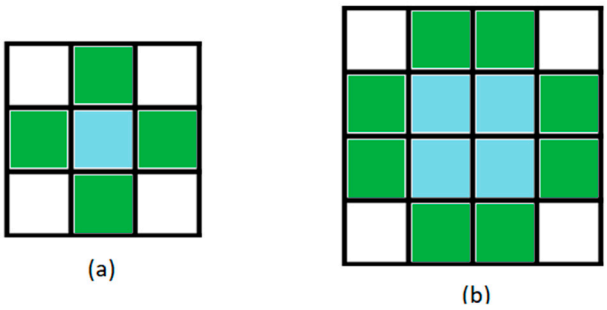

Figure 1. Image (**a**) depicts a cluster that consists of a single cell (blue cell). Green cells represent the cells that have 4-neighbor connectivity with the blue cell. Image (**b**) depicts a larger environment along with its 4-neighbors. Essentially, 4-neighbor connectivity between cells prevents a robot from moving diagonally within the environment.

4. The Proposed Algorithm

4.1. Data Initialization

To develop a strong solution for the multi-robot CPP problem, we introduced a new algorithm based on AP [27] (Figure 2). AP is a powerful clustering algorithm commonly used for dividing and assigning areas. In our approach, we rely on AP to group data points

into meaningful clusters. The algorithm operates by exchanging messages containing real values among the data points. These messages help the algorithm determine the best number of clusters and how to assign the data points to those clusters. The algorithm continues this message exchange process until a consensus is reached on the optimal cluster configuration. This methodology allows us to bypass the necessity to predetermine the number of robots or the dimensions of the respective sub-areas arbitrarily. As opposed to conventional applications, we employed a meticulous transformation process to adapt our grid-like environment, A, into a dataset suitable for AP (Figure 3).

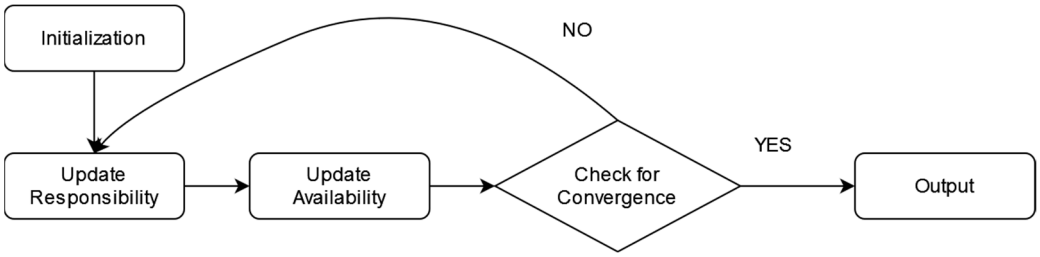

Figure 2. The initialization step prepares the data, converts the grid to a set of data points, and calculates the similarity matrix *S*. Then, the responsibility matrix *R* is updated. The responsibility matrix reflects how well suited a data point is to serve as the exemplar for the other data points. Next, the Availability matrix is updated. This matrix reflects how suitable an exemplar is for each data point to serve as its exemplar. Finally, the algorithm iterates until the maximum number of iterations set, or until there are no changes from the last iteration.

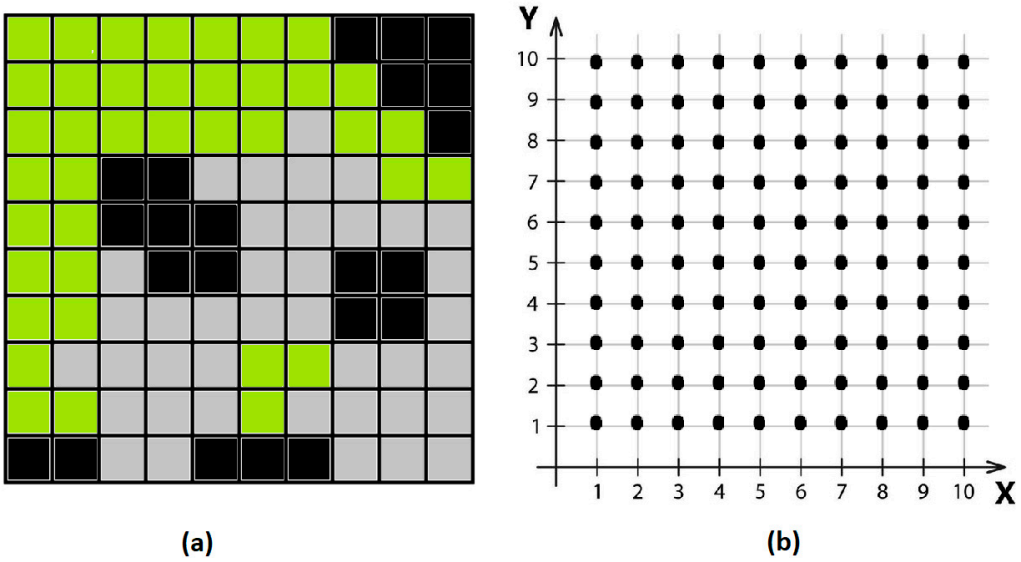

Figure 3. The initial environment (**a**) consists of obstacles (black cells), cells of type grass (green cells), and cells of type asphalt (gray cells). In order to divide the environment into multiple sub-areas, we first have to convert the environment into a set of data points. (**b**) depicts the data points in a X, Y Cartesian coordinate system.

4.2. Calculating Normalized Distance

In this process, each cell within the initial environment is mapped onto a data point in a two-dimensional coordinate system. To discern the relationship between data points, we define a similarity function. In many standard applications of AP, similarity is gauged by the negative of Euclidean distance. This inversely proportional relationship implies a smaller similarity for larger absolute distances between data points.

However, the application of Euclidean distance to our context of CPP within a two-dimensional grid, where we aim to preserve each cluster's continuity and its cell's 4-neighbor relationship, is not fitting. This metric fails to account for obstacles that may disrupt the path between two locations. This discrepancy between Euclidean distances and actual traversable distances can lead to significant errors, with the disparity being exacerbated by the presence of numerous obstacles.

Understanding the inherent constraints of the traditional approach, we adapted our similarity function to include a more tailored and nuanced model (Algorithm 1).

Algorithm 1: Calculating the normalized 4-neighbor distance between two points

1. Input: Binary matrix A with dimensions X by Y, starting point (x_1, y_1), and target point (x_2, y_2)
2. Output: minimum distance D between the starting and target points
3. Function 4_neighbor_actual_distance $(A,\ x_1, y_1,\ x_2,\ y_2)$:
4. Step 1: Initialize a distance matrix D with dimensions X by Y, set all elements to infinity
5. Step 2: Initialize a queue Q
6. Step 3: Set $D\ [x_1, y_1] = 0$ and add (x_1, y_1) to Q
7. Step 4: While Q is not empty:
8. Step 4.1: Dequeue a point (x, y) from Q
9. Step 4.2: Loop through each of its four neighbors $(x_n,\ y_n)$ in the environment A:
10. Step 4.2.1: If $(x_n,\ y_n)$ is an obstacle $(A[x_n,\ y_n] = 0)$, skip this neighbor
11. Step 4.2.2: If $D[x_n,\ y_n] > \mathrm{D}[x,\ y] + 1$:
12. Step 4.2.2.1: Update $D[x_n,\ y_n] = D[x,\ y] + 1$
13. Step 4.2.2.2: Add $(x_n,\ y_n)$ to Q
14. Step 5: Return $D[x_2,\ y_2]$ as the minimum distance D between the starting and target points
15. End Function

Our novel algorithm not only calculates the minimum 4-neighbor distance between cells, similar to a BFS approach, but also takes into account the type of floor and elevation of each cell, thereby capturing crucial information about the landscape's unique characteristics. It thereby ensures a more accurate and practical representation of the environment's traversability. Incorporating these parameters directly into the similarity function provides a more realistic framework for area division in 2D grid environments. This ultimately leads to enhanced efficiency and effectiveness in our CPP solutions, as it ensures more prudent and strategic allocation of sub-areas to robots, factoring in complex grid conditions that could impact their performance.

4.3. Calculating the Similarity Matrix

In the similarity matrix, each element holds a singular metric quantifying its resemblance to the other elements in the adjacent vicinity (Algorithm 2). The degree of similarity between any two elements augments in direct proportion with the increase in the similarity value. It warrants highlighting that, due to the intrinsic characteristics of the AP method, the similarity function $S(p_1, p_2)$ may not necessarily be identical to $S(p_2, p_1)$. Although this directional feature is not incorporated in the current implementation of our proposed methodology (Figure 4), potential future adaptations of the algorithm may consider its integration to facilitate directional clustering of cells and other specific elements. Utilizing the AP methodology, we classified data points into distinct sub-regions for each robot, following an assessment of the similarity quotient between each pair of data points. The AP algorithm operates through the transmission of messages that denote a data point's proclivity towards a particular cluster. Subsequent to the resolution on the number of clusters and the allocation of data points to respective clusters, these messages are subject to iterative refinement.

Algorithm 2: The procedure that calculates the weighted Similarity Matrix S

1. Input:
2. Environment matrices: binary A, weight W, elevation E, floor F
3. Significance multipliers: t, q, r
4. Output: similarity matrix S
5. Function similarity_calculation(A, X, Y, W):
6. Step 1: Initialize an X by Y matrix S
7. Step 2: Loop through each pair of data points (x_1, y_1) and (x_2, y_2) in the environment A:
8. Step 2.1: Calculate the distance between the data points:
9. d = 4_neighbor_actual_distance (x_1, y_1, x_2, y_2)
10. Step 2.2: Calculate the elevation difference
11. $e = |E[x_1, y_1] - E[x_2, y_2]|$
12. Step 2.3: Calculate the floor discrepancy
13. $f = |F[x_1, y_1] - F[x_2, y_2]|$
14. Step 2.2: Multiply each metric by its weight factor:
15. $d = t \times d$
16. $e = q \times e$
17. $f = r \times f$
18. Step 2.3: Store the similarity between the data points in the similarity matrix S:
19. $S[x_1, y_1, x_2, y_2] = -(d + e + f)$
20. Step 3: Return the similarity matrix S
21. End Function

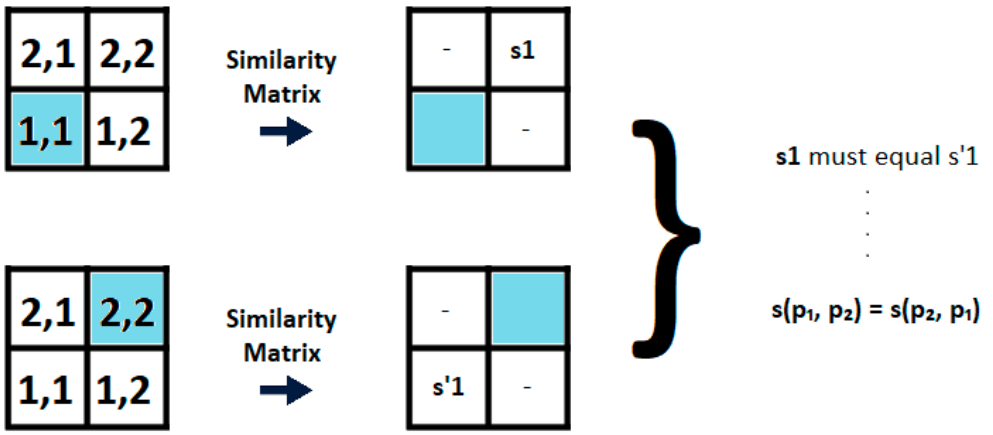

Figure 4. Two points (denoted with blue color) will always have the same similarity matrix with each other. In practice, this means that going from point p_1 to point p_2 has the same cost as going from point p_2 to p_1.

4.4. Generation of Clusters

After calculating the similarity between all pairs of data points, the AP algorithm was used to cluster the data points into sub-areas for each robot. The AP algorithm works by passing messages between data points, which indicate their preference for a particular cluster. These messages are updated iteratively until a consensus is reached on the number of clusters and which points belong to which cluster.

Upon the conclusion of the iterative message-passing phase, the AP algorithm proceeds towards the establishment of sub-areas (Algorithm 3). This crucial stage determines the most suitable exemplar for each cluster—a data point that accrues the maximum preference value when both availability and responsibility are taken into account. Consequently, each data point (cell) is assigned to the exemplar of its respective cluster.

Algorithm 3: Generating the sub-areas using AP

1. Input: grid-like area A with dimensions X by Y
2. Output: sub-areas for each robot
3. Step 1: Convert the grid-like area A into a set of data points
4. Represent each grid cell as a data point in a two-dimensional coordinate system
5. Step 2: Calculate the similarity between every pair of data points
6. Calculate the similarity based on Algorithms 1 and 2 (taking into consideration the weight factor)
7. Store the similarity in a matrix S
8. Step 3: Initialize messages between data points
9. Initialize two matrices, R and A, to store the messages between data points
10. Initialize the self-similarity matrix, S, to store the similarity between a data point and itself
11. Step 4: Iterate until convergence
12. Update the responsibility matrix, R
13. Update the availability matrix, A
14. Step 5: Identify exemplars
15. Identify the data points with the highest responsibility and availability values as exemplars
16. Step 6: Assign sub-areas to each robot
17. Assign each non-exemplar data point to the closest exemplar (only if they are spatially connected using the 4-neighbor scheme)
18. Group the data points assigned to each exemplar into a sub-area
19. Step 8: Return the sub-areas for each robot

A key attribute of the proposed algorithm lies in its capacity to autonomously ascertain the optimal number of clusters (sub-areas), without any need for user-defined inputs. This ability to self-regulate cluster formation ensures flexibility and adaptability, which is particularly beneficial in complex real-world applications.

The final output of the algorithm is a list of cluster labels for each data point in the initial environment, indicating the respective robot assignment for each cell (Figure 5). This clustered set of data points serves as the foundation for subsequent stages of the proposed model, leading to the final path planning for the robots.

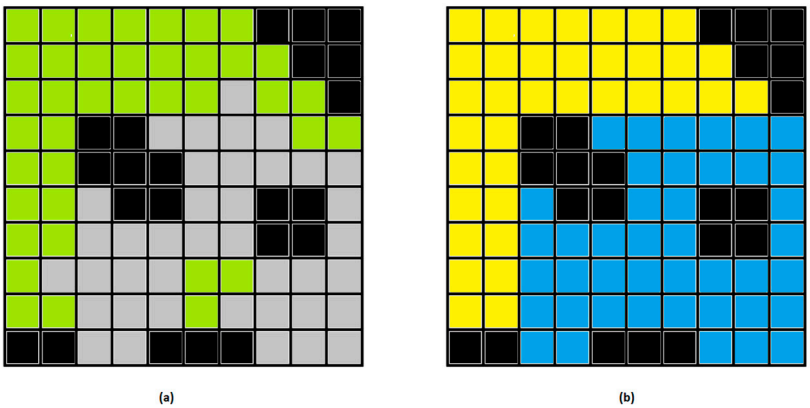

Figure 5. The initial environment as depicted in (**a**) contains two layer types (denoted by green and gray cells). The output of the algorithm in (**b**) shows the two clusters (blue and yellow), as they were generated by the algorithm. It is worth mentioning that different values of the importance value T_w may result in slightly different clusters.

5. Experimental Results

In this section, we present the findings obtained from our in-depth analysis of the proposed algorithm. The experimental results validate the efficiency and adaptability of our algorithm in effectively decomposing the environment into suitable sub-areas. The

evaluation metrics employed primarily centered on the criteria of computational time and the quality of the generated clusters.

In terms of comparison with existing algorithms, it is important to highlight that our proposed AP-based algorithm addresses a unique problem in the field of multi-robot CPP. In the literature, many algorithms attempt to solve the issue of dividing an initial area into multiple sub-areas for each robot. However, the identical problem that our algorithm solves, without arbitrarily pre-determining the number of robots and their initial positions, is lacking in the current literature. Nevertheless, we embarked on an indirect comparative analysis, contrasting our algorithm's performance with traditional algorithms for area division in CPP, which rely on pre-specified robot counts and locations. The comparative study primarily emphasized the improvements in efficiency, flexibility, and adaptability introduced by the proposed algorithm. Despite the inherent differences in the problem contexts, the comparative analysis provided a clear demonstration of the strides made by the AP-based approach, particularly in situations where the initial robot positions are in close proximity.

The conducted simulations predominantly operated in two meticulously constructed environments of distinct sizes, each featuring distinct characteristics. Both environments were generated using a pseudo-random process, offering a unique combination of accessible areas and obstacles along with varying environmental types, ensuring the robustness of the simulated scenarios.

The smaller 24 $\times$ 24 environment served as an essential proving ground for our proposed algorithm, presenting a grid with a variety of unique parameters and diverse characteristics. The total area consisted of 576 cells, with approximately 80% of the grid being accessible and approximately 20% assigned as obstacles. These obstacles were evenly scattered throughout the grid to simulate potential hindrances that robots might encounter in real-world scenarios. Beyond mere accessibility, cells in the environment were characterized by two distinct terrain types—grass and asphalt. The grass cells accounted for roughly 60% of the accessible area, while the asphalt cells constituted the remaining 40%. This bifurcation of terrain types aimed to emulate real-world environments where robots might encounter varied terrains requiring different path planning strategies and navigation capabilities. To augment the realism of the simulation, cells were assigned varying elevation levels. The elevations ranged from 0 to 10 units, with a standard deviation of three units to ensure a substantial variation in elevation across the grid. An elevation weight factor of 0.1 was applied to indicate the importance of elevation. Similarly, the type of terrain also had an associated weight factor to indicate the importance of terrain type.

Subsequently, a second, larger, and more complex environment was introduced for a more in-depth simulation. The dimensions of this environment are 100 $\times$ 100. The rest of the parameters were the same as those of the previous smaller environment. Due to its substantial size, detailed visualization was rendered impractical. Nevertheless, we present empirical data and statistics to illustrate the algorithm's performance. This larger grid serves to emulate more complex real-world scenarios, thereby demonstrating the scalability and adaptability of the proposed AP algorithm in diverse, challenging situations.

The computational experiments were conducted on a dedicated testbed configured to ensure accurate and consistent results. The hardware setup encompassed a high-performance workstation equipped with an Intel Core i7-9700K 8-core processor clocked at 3.60 GHz, bolstered with 32 GB of DDR4 RAM and enabling efficient data handling and manipulation, which were particularly critical given the size of the datasets and complexity of operations involved in Affinity Propagation.

Table 1 shows the experimental results for each setup. For each algorithm, we conducted 20 experiments using the aforementioned parameters. The table shows the average values for the number of generated clusters and the cluster quality.

Table 1. Experimental results for two environments using three clustering algorithms for CPP.

Environment Size	Algorithm	Generated Clusters	Cluster Quality
24 × 24	[18] n = 2	2	0.66
24 × 24	[19] n = 2	2	0.84
24 × 24	[25] n = 2	2	0.89
24 × 24	Proposed	2.19	0.91
100 × 100	[18] n = 2	2	0.39 (often failed to find solution)
100 × 100	[19] n = 2	2	0.85
100 × 100	[25] n = 2	2	0.92
100 × 100	Proposed	6.52	0.94

It is worth mentioning that it is difficult to evaluate the proposed algorithm by directly comparing it with others found in the literature, since these algorithms are not configurable to identify the different cell types, elevation, and importance factor. Therefore, we could only compare the results of these algorithms by taking into account the quality of the clustering. The quality of clustering was calculated using the Silhouette Coefficient (SC) [28].

The SC, also known as Silhouette Score, is a well-established and popular metric for evaluating the quality of a clustering algorithm. The essence of this method lies in its dual-faceted measurement approach, quantifying both cohesion and separation simultaneously for each individual cell. It operates by comparing the average distance of a cell to all other points within its own cluster (cohesion) against the average distance to points in the nearest cluster (separation). The coefficient thus provides an aggregate measure of how similar a given data point is to its own cluster relative to other clusters. Higher values of the SC suggest that the cell is well-clustered and lower values imply that the specific cell might have been better assigned to a neighboring cluster. It is widely regarded well due to its intuitive interpretation, its capability to work with any distance metric (in our example the normalized distance as presented in Algorithm 1), and its agnosticism towards the specific clustering algorithm used. To properly evaluate the proposed algorithm, we calculated the SC not only for the average distances but also for the similarity with regard to elevation and floor type. The total SC calculated was weighted based on the respective weights of floor type and elevation.

A more nuanced evaluation of the proposed algorithm was achieved by conducting multiple runs in the same environment but with varied importance factors attached to floor type and elevation. This exploratory approach aimed to investigate the algorithm's capability to adapt and respond to shifts in preference and the importance of environmental features. For this experiment, the main focus was on gauging the homogeneity of the resulting clusters. Homogeneity here was defined as the proportion of a cluster that exhibits uniformity in terms of either floor type or elevation. When the importance factor assigned to floor type was modified, we anticipated observing clusters that largely contain the same floor type. Consequently, the algorithm's sensitivity to floor type would be reflected by the degree of homogeneity of the resulting clusters. Analogously, when the emphasis is shifted to elevation, the homogeneity of the clusters, in terms of their elevation, becomes the pivotal measure of algorithm performance. The experimental results are presented in Table 2.

The experimental results indicate that the importance factors affect the final clusters and their cells. It is important, however, to fine tune the exact values for each multi-robot CPP task, based on the capabilities of the available robots. A visual representation of a 10 × 10 environment and the output of the algorithm with different importance factor T_W are depicted in Figure 6. A larger 24 × 24 environment is depicted in Figure 7.

Table 2. Experimental results from the execution of the proposed algorithm in the same environment using different elevation E_w and floor type F_W weight factors. These values range from 0 to 1 and describe how important the preservation of elevation or floor type within a generated sub-area is. Increasing or decreasing both values at the same time reduces the importance of both variables within the clustering process.

Environment	Importance Factors (E_w, F_W)	Height Homogeneity	Floor Homogeneity
100×100	(0, 0)	0.61	0.49
100×100	(0.8, 0)	0.75	0.45
100×100	(0.8, 0.8)	0.59	0.53
100×100	(0, 0.8)	0.53	0.78

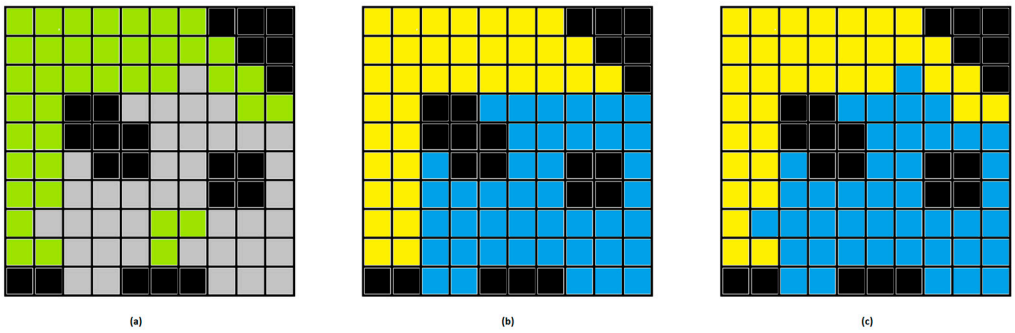

Figure 6. The initial environment as depicted in (**a**) contains two layer types (green and gray). The second image (**b**) shows the output clusters of the algorithm (blue and yellow) with an importance factor $T_W = 0$. The third image (**c**) shows the output of the algorithm for the same input environment where the importance factor T_W is equal to 0.8. Increasing the importance factor of floor type increases the likelihood that the algorithm will consider two cells of the same type to be more similar.

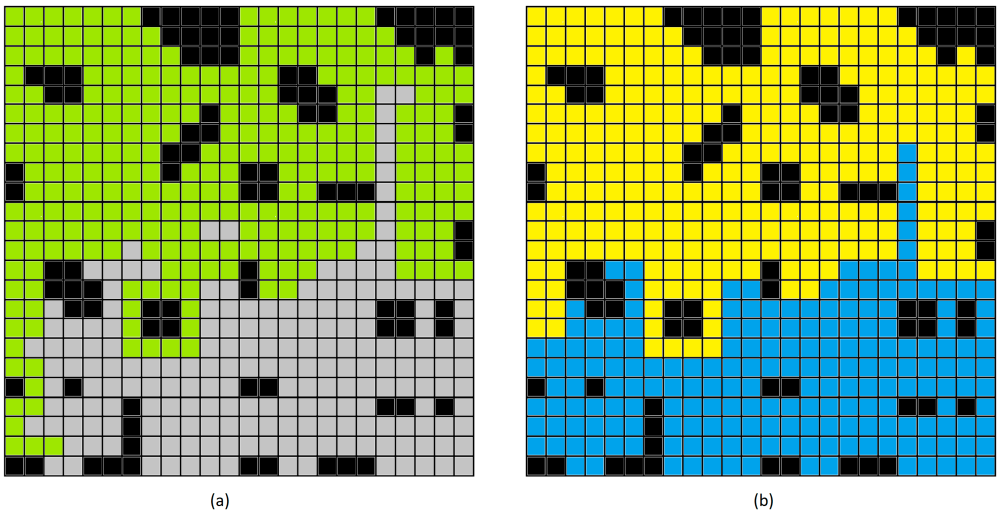

Figure 7. An example environment with dimensions 24×24. The initial environment (**a**) contains two layer types (green and gray). The second image (**b**) shows the output clusters of the algorithm (blue and yellow) with an importance factor $T_W = 0.3$.

6. Discussion

6.1. Limitations

While this paper proposed an innovative application of AP in the field of multi-robot CPP, it is not without limitations. A clear understanding of these potential constraints is essential for refining the algorithm, enhancing its applicability, and identifying areas for future research.

The primary limitation arises from the AP's inherent computational complexity. With a time complexity of $O(N^{2T})$, where N represents the number of data points and T denotes the number of iterations, AP can be computationally expensive for large-scale environments. This computational cost is primarily due to the calculation of the similarity matrix and the iterative exchange of "responsibility" and "availability" messages. This renders the algorithm less practical for real-time operations, particularly for larger environments, as it may lead to increased latency in area division and subsequent path planning.

Another important limitation is the static nature of the AP algorithm, which is very hard if not impossible to adjust for dynamic environments. The AP algorithm, as applied in this context, assumes a static environment where the position of obstacles and the type of each cell are known beforehand. In scenarios where the environment changes over time, the algorithm would need to be rerun, potentially leading to delays and inefficiencies. The ability to adapt to dynamic environments remains a significant challenge in the field of multi-robot CPP and represents a key area for future research.

6.2. Performance Improvement

The performance of the AP algorithm, as is the case with most computational procedures, is pivotal in real-world applications. The urgency for efficiency and execution speed improvements is even more pronounced when dealing with multi-robot coverage path planning, given the scale of the task and the inherent complexity associated with environment mapping and path planning. In light of this, several strategies can be considered to enhance the execution speed and overall efficiency of the AP algorithm.

One crucial step of the AP algorithm is the calculation of the similarity matrix. Given that this phase accounts for a significant portion of the computations (approximately 40%), improving its efficiency is imperative. Given its intrinsic parallelizable nature, where the similarity between each pair of data points can be calculated independently, we could potentially harness the power of parallel computing. By distributing the calculation of similarity measures across multiple cores or nodes in a parallel computing environment, we can expedite this process markedly, thereby increasing the overall efficiency of the AP algorithm.

On the other hand, the message passing phase of the AP algorithm, which is pivotal for its iterative structure, is more challenging to parallelize. Although in theory, each "responsibility" and "availability" message update could be computed in parallel, the iterative nature of the AP algorithm necessitates the results of each preceding iteration. Nonetheless, we could explore certain forms of "soft" parallelization, such as utilizing vectorized operations or parallel map functions provided by high-level languages and libraries. Even though this approach would not offer true parallelization due to each iteration still needing to await the completion of all message updates, it could still provide substantial speed improvements.

Besides parallel computing, other potential strategies for improving the efficiency of the AP algorithm could include optimization of the algorithm's parameters or the application of hardware accelerators such as Graphics Processing Units (GPUs) [29]. Fine-tuning parameters like the damping factor or the preference value could potentially reduce the number of iterations required for convergence, thereby accelerating the execution speed. Similarly, using GPUs, which are particularly suited for parallelizable tasks, could lead to substantial reductions in computation time. However, such strategies would require careful evaluation to balance efficiency gains against the potential impact on the quality of the results.

7. Conclusions

The presented research introduces a paradigm shift in the domain of multi-robot CPP by utilizing AP for optimally dividing the operational area among the robots. Instead of using traditional methods, which largely rely on the number of robots and their initial positions, this innovative methodology partitions the area into 'n' clusters using AP and

subsequently assigns each cluster to a robot. This model, while functioning under the assumption of an unlimited number of robots, provides a unique flexibility by allowing the modification of the AP algorithm's similarity function factor to control the number of generated clusters.

One field where the proposed algorithm can find profound applications is precision agriculture. This industry, already substantially automated, requires precision farming, which involves the distribution of multiple tasks, such as seeding, fertilizing, and harvesting, across a fleet of robotic entities. Identifying the optimal number of sub-areas becomes paramount to prevent overlap and redundancy in operations. The proposed algorithm, by facilitating automatic partitioning of farmland into sub-areas based on factors such as crop type and topography paves the way for improved resource management. It ensures optimal task distribution amongst autonomous agricultural machines, enhancing their overall operational efficiency, thereby contributing to a significant reduction in the time and cost associated with agricultural practices.

Additionally, this algorithm can significantly revolutionize urban search and rescue operations. Typically, these operations are time-sensitive, requiring the division of large, affected areas into smaller manageable sub-areas to enable quick and efficient search strategies. The conventional method of dividing areas based on available rescuers may not be effective, especially in scenarios where the rescuers are robotic entities. By employing this algorithm, we could efficiently partition the search area into the appropriate number of sub-areas (clusters) regardless of the number of robots, optimizing the search strategy and increasing the likelihood of successful rescue operations. Moreover, with the AP algorithm's adaptable similarity function factor, the rescue team has flexibility in regulating the number of generated clusters, facilitating a more efficient and coordinated search operation.

As a significant progression in multi-robot CPP, this methodology paves the way for novel research directions and practical enhancements in this field. The capability to deliver effective area division and path optimization, without burdening the user with the arbitrary decision of the number of robots or their initial positions, sets a new benchmark for multi-robot CPP implementations. Moreover, our work provides a strong foundation for the development of enhanced strategies that can address the existing complexities of multi-robot CPP and further expedite the deployment of autonomous systems in diverse fields ranging from agriculture to reconnaissance missions. Future work will focus on refining the proposed model, incorporating more complex environmental factors, and exploring the potential of integrating this method with different path planning algorithms for better performance.

Author Contributions: Conceptualization, N.B. and M.D.; methodology, N.B.; writing—original draft preparation, N.B.; writing—review and editing, N.B.; visualization, N.B.; supervision, M.D.; funding acquisition, N.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was carried out as part of the project «Smart Safe Navigation for Electric Bicycles and Skateboards» (Project code:KMP6-0292520) under the framework of the Action «Investment Plans of Innovation» of the Operational Program «Central Macedonia 2014 2020», that is co-funded by the European Regional Development Fund and Greece.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Mourtzis, D.; Angelopoulos, J.; Panopoulos, N. A Literature Review of the Challenges and Opportunities of the Transition from Industry 4.0 to Society 5.0. *Energies* **2022**, *15*, 6276. [CrossRef]
2. Kagermann, H.; Wahlster, W. Ten Years of Industrie 4.0. *Science* **2022**, *4*, 26. [CrossRef]
3. Demir, K.A.; Döven, G.; Sezen, B. Industry 5.0 and human-robot co-working. *Procedia Comput. Sci.* **2019**, *158*, 688–695. [CrossRef]

4. Wang, E.Z.; Lee, C.C.; Li, Y. Assessing the impact of industrial robots on manufacturing energy intensity in 38 countries. *Energy Econ.* **2022**, *105*, 105748. [CrossRef]
5. Kyrarini, M.; Lygerakis, F.; Rajavenkatanarayanan, A.; Sevastopoulos, C.; Nambiappan, H.R.; Chaitanya, K.K.; Makedon, F. A survey of robots in healthcare. *Technologies* **2021**, *9*, 8. [CrossRef]
6. Oliveira, L.F.; Moreira, A.P.; Silva, M.F. Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics* **2021**, *10*, 52. [CrossRef]
7. Aivazidou, E.; Tsolakis, N. Transitioning towards human–robot synergy in agriculture: A systems thinking perspective. *Syst. Res. Behav. Sci.* **2022**, *40*, 536–551. [CrossRef]
8. Marinoudi, V.; Sørensen, C.G.; Pearson, S.; Bochtis, D. Robotics and labour in agriculture. A context consideration. *Biosyst. Eng.* **2019**, *184*, 111–121. [CrossRef]
9. Shamout, M.; Ben-Abdallah, R.; Alshurideh, M.; Alzoubi, H.; Kurdi, B.A.; Hamadneh, S. A conceptual model for the adoption of autonomous robots in supply chain and logistics industry. *Uncertain Supply Chain. Manag.* **2022**, *10*, 577–592. [CrossRef]
10. Jorge, V.A.M.; Granada, R.; Maidana, R.G.; Jurak, D.A.; Heck, G.; Negreiros, A.P.F.; dos Santos, D.H.; Gonçalves, L.M.G.; Amory, A.M. A Survey on Unmanned Surface Vehicles for Disaster Robotics: Main Challenges and Directions. *Sensors* **2019**, *19*, 702. [CrossRef]
11. Angelopoulos, G.; Baras, N.; Dasygenis, M. Secure autonomous cloud brained humanoid robot assisting rescuers in hazardous environments. *Electronics* **2021**, *10*, 124. [CrossRef]
12. Dogru, S.; Marques, L. ECO-CPP: Energy constrained online coverage path planning. *Robot. Auton. Syst.* **2022**, *157*, 104242. [CrossRef]
13. Patle, B.K.; Pandey, A.; Parhi, D.R.K.; Jagadeesh, A.J.D.T. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
14. Moysiadis, V.; Sarigiannidis, P.; Vitsas, V.; Khelifi, A. Smart farming in Europe. *Comput. Sci. Rev.* **2021**, *39*, 100345. [CrossRef]
15. Utamima, A.; Reiners, T.; Ansaripoor, A.H. Optimisation of agricultural routing planning in field logistics with Evolutionary Hybrid Neighbourhood Search. *Biosyst. Eng.* **2019**, *184*, 166–180. [CrossRef]
16. Moysiadis, V.; Tsolakis, N.; Katikaridis, D.; Sørensen, C.G.; Pearson, S.; Bochtis, D. Mobile Robotics in Agricultural Opera-tions: A Narrative Review on Planning Aspects. *Appl. Sci.* **2020**, *10*, 3453. [CrossRef]
17. Almadhoun, R.; Taha, T.; Seneviratne, L.; Zweiri, Y. A survey on multi-robot coverage path planning for model reconstruction and mapping. *SN Appl. Sci.* **2019**, *1*, 847. [CrossRef]
18. Kapoutsis, A.C.; Chatzichristofis, S.A.; Kosmatopoulos, E.B. DARP: Divide areas algorithm for optimal multi-robot coverage path planning. *J. Intell. Robot. Syst.* **2017**, *86*, 663–680. [CrossRef]
19. Idir, O.; Renzaglia, A. Multi-Robot Weighted Coverage Path Planning: A Solution based on the DARP Algorithm. In Proceedings of the 2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, 11–13 December 2022; pp. 98–104.
20. Huang, X.; Sun, M.; Zhou, H.; Liu, S. A multi-robot coverage path planning algorithm for the environment with multiple land cover types. *IEEE Access* **2020**, *8*, 198101–198117. [CrossRef]
21. Tang, J.; Sun, C.; Zhang, X. MSTC: Multi-robot Coverage Path Planning under Physical Constrain. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 2518–2524.
22. Collins, L.; Ghassemi, P.; Esfahani, E.T.; Doermann, D.; Dantu, K.; Chowdhury, S. Scalable coverage path planning of multi-robot teams for monitoring non-convex areas. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 7393–7399.
23. Qin, Y.; Fu, L.; He, D.; Liu, Z. Improved Optimization Strategy Based on Region Division for Collaborative Multi-Agent Coverage Path Planning. *Sensors* **2023**, *23*, 3596. [CrossRef]
24. Abd Rahman, N.A.; Sahari KS, M.; Hamid, N.A.; Hou, Y.C. A coverage path planning approach for autonomous radiation mapping with a mobile robot. *Int. J. Adv. Robot. Syst.* **2022**, *19*, 17298806221116483. [CrossRef]
25. Huang, Y.; Li, M.; Zhao, T. A Multi-robot Coverage Path Planning Algorithm Based on Improved DARP Algorithm. *arXiv* **2023**, arXiv:2304.09741.
26. Gao, C.; Kou, Y.; Li, Z.; Xu, A.; Li, Y.; Chang, Y. Optimal multirobot coverage path planning: Ideal-shaped spanning tree. *Math. Probl. Eng.* **2018**, *2018*, 3436429. [CrossRef]
27. Frey, B.J.; Dueck, D. Clustering by passing messages between data points. *Science* **2007**, *315*, 972–976. [CrossRef] [PubMed]
28. Dinh, D.T.; Fujinami, T.; Huynh, V.N. Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient. In *Knowledge and Systems Sciences, Proceedings of the 20th International Symposium 2019, KSS 2019, Da Nang, Vietnam, 29 November–1 December 2019*; Springer: Singapore, 2019.
29. Hijma, P.; Heldens, S.; Sclocco, A.; Van Werkhoven, B.; Bal, H.E. Optimization Techniques for GPU Programming. *ACM Comput. Surv.* **2023**, *239*, 81. [CrossRef]

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