Frontier-led Swarming: Robust Multi-Robot Coverage of Unknown Environments

Vu Phi Tran, Matthew A. Garratt, Kathryn Kasmarik, Sreenatha G. Anavatti

Abstract—This paper proposes a novel swarm-based control algorithm for exploration and coverage of unknown environments, while maintaining a formation that permits short-range communication. The algorithm combines two elements: swarm rules for maintaining a close-knit formation and frontier search for driving exploration and coverage. Inspired by natural systems in which large numbers of simple agents (e.g., schooling fish, flocking birds, swarming insects) perform complicated collective behaviors for efficiency and safety, the first element uses three simple rules to maintain a swarm formation. The second element provides a means to select promising regions to explore (and cover) by minimising a cost function involving robots’ relative distance to frontier cells and the frontier’s size. We tested the performance of our approach on heterogeneous and homogeneous groups of mobile robots in different environments. We measure both coverage performance and swarm formation statistics as indicators of the robots’ ability to explore effectively while maintaining a formation conducive to short-range communication. Through a series of comparison experiments, we demonstrate that our proposed strategy has superior performance to recently presented map coverage methodologies and conventional swarming methods.

Index Terms—Swarm Intelligence, Frontier Search, Heterogeneous Robot Swarm, Area Coverage Algorithm

I. INTRODUCTION

A common assumption in recent work related to multi-robot coverage path planning systems [1-5], is the availability of a map (that is, a known environment) at the planning stage. Furthermore, existing multi-robot coverage solutions for unknown environments, including graph-based approaches [6] and Voronoi-based approaches [7], perform coverage assuming individual robots are independent of each other [2]. The failure of a single robot can cause incomplete area coverage. This paper relaxes the assumption that the structure of an environment is known in advance, and presents a cooperative swarming algorithm that incorporates search, coverage and self-organising formation control. Motivated by emergent behaviors of animal aggregation found in insects, birds, fish, and other organisms, the interdisciplinary fields of swarm intelligence and robotics take advantage of complex yet efficient and safe self-organized swarm formations by building a system of apparently simple interaction rules that take local observable information and limited-range communication among proximate agents as inputs [8]. The result is a system capable of robust and efficient coverage behaviour.

The contributions of this paper are:

1) A novel frontier-led swarming (FrontierSwarm) algorithm is introduced. This algorithm allows the robotic swarms to achieve fast area coverage while maintaining a close-knit formation to facilitate short-range communication.

2) A distributed architecture for implementing frontier-led swarming. In this architecture, each agent can directly exchange coverage and obstacle information with a set of neighbors without passing data through a central processing unit. Thus, the system is fully distributed and the loss of a subset of agents does not critically affect the operation of the remaining swarm members in completing the overall mission.

3) A comparative study benchmarking our frontier-led swarming algorithm against the frontier-based map coverage strategy without swarming and swarming algorithms without frontier search. The algorithms are tested on homogeneous and heterogeneous robot teams to cover a map as quickly as possible while maintaining a close, self-organized swarm formation to permit short-range communication. We introduce a combination of time, coverage and swarming metrics for the purpose of this comparative study.

The approach to swarming taken in this paper is based on Reynolds’ boids model [9]. In contrast to other models such as the Vicsek model [10], where flocking is generated by the simple combination of alignment interactions and random linear velocities, the Reynolds model [9] can be used to incorporate the collective motion behavior with other operations, such as reaching the goal and avoiding obstacles, by adding a set of external steering forces. Noticeably, although robots know their absolute locations at each step, boids methods do not require highly accurate positioning, which can be a drawback of several formation control strategies. The boids approach decomposes the complicated group motion behaviors into three simple steering rules (cohesion, alignment, or separation) at the individual agent level. Decision making is localised within neighbourhoods with only nearby agents within a relevant radius impacting the movement of any individual.

Moreover, as we will see in this paper, the boid swarming model offers a highly flexible framework that transfers between homogeneous and heterogeneous groups of robots. Recent swarming and flocking algorithms have tended to focus on homogeneous robot teams and initially assume perfect
communication channels [11-13]. This paper presents a novel approach to exploration and coverage that extends to unknown environments where only limited, local communication is possible. We anticipate that this may be useful for environments where communication has been disrupted, for example following a disaster, or where communication channels are contested.

To achieve exploration and coverage, we adopt the concept of a frontier, which is the border between unexplored and known areas [14, 15]. Frontier algorithms have previously been developed for search, and we adapt one in this paper for area coverage in an unknown environment.

The rest of this paper is organized as follows. In Section 2, a literature review of related work is presented, including our problem formulation and the preliminaries that form the underlying theory for our approach. Our frontier-led swarming algorithm and robotic architecture is introduced in Section 3. Our experimental environment, methodology and results are discussed in Section 4. Finally, concluding remarks are presented in Section 5.

II. RELATED WORK

In this section we first introduce the underlying mission we are addressing in Section A: area coverage in an unknown environment. We then review existing challenges related to coverage in Section B and existing approaches to search in an unknown environment in Section C. Finally we introduce the boid model that underlies our solution in Section D.

A. Area Coverage Problems

Area coverage requires a group of agents or robots to visit every position in an environment. Typical real-world examples where area coverage must be applied include vacuum cleaning, painting and surveillance. To formally define the coverage problem a grid discretisation of a space is generally first defined with a certain resolution. Grids can be of different shapes including cells with triangular, square, hexagonal or diamonds shapes [16]. Different definitions of coverage of the grid exist, including dynamic coverage and static coverage [17]. In static coverage, agents need to seek out optimal locations that form an appropriate geometry so that their network of sensors can cover the whole environment [18]. However, in some practical missions, the environment cannot be fully covered by a static coverage geometric network due to the limitation in sensing range or too few sensors being available. In such cases, the problem of dynamic coverage is considered, where mobile robots with limited-range sensors continuously move and sample data, normally from unit spaces within a discretized domain until a predefined coverage level is attained [19]. In this paper, we focus on the dynamic coverage problem. We further assume that the environment to be covered is not known in advance. That is, the robots do not know the boundaries of the environment, nor the locations of any potential obstacles.

In a simple formal definition of dynamic coverage, a 2D grid cell $j$ is covered by agent $A^i$ if $A^i$ was within cell $j$ at some point during the exploration [20]. In that case, we say that $A^i$ has directly covered cell $j$.

In a multi-agent system, if agent $A^i$ communicates the coverage of cell $j$ to a neighbour agent $A^k$, we say that agent $A^k$ has indirectly covered cell $j$. Synchronised coverage is achieved in a multi-agent system when each agent has directly or indirectly covered all grid cells of the map. In other words, each agent knows that each cell has been directly covered by some other agent.

Collaborative, complete exploration and coverage are two active research fields, where multi-robot systems are required to maintain situational awareness in unknown obstacle-cluttered scenarios. Solving these problems can benefit many potential applications, including environmental waste management [21], patrolling [22] and search and rescue [23].

B. Challenges Associated With Area Coverage and Existing Solutions

In mobile robotics, the exploration task involves sensing the structure of unvisited areas and constructing a map. In contrast, the complete area coverage task aims at visiting or periodically re-visiting all necessary locations within a particular workspace [24]. Single-agent space coverage has been addressed by various means, such as genetic algorithms [25], coverage path planning [26], or reinforcement learning [27]. Algorithms using a single-agent may be optimal if sufficient time is available or if agents are expensive (e.g., planetary exploration missions). However, navigating and exploring a vast area using a single robot suffers from drawbacks, including long turnaround time, depletion of robot power resources, and poor quality and quantity of information collected [16]. In particular, single-agent systems may not be an ideal choice for surveillance or disaster response when mission time is critical.

In a modern multi-agent coverage system, information about the explored space is shared with nearby agents. Each agent maintains its own map and performs its own space coverage strategy. Many popular approaches rely on centralized control [28, 29] requiring high-bandwidth, reliable communication. Performance degrades when the number of agents increases or when communication bandwidth is limited or intermittent. Thus the multi-agent coverage task is widely considered in a distributed fashion [16, 30-32]. Operating a distributed robotics framework considerably reduces the load of communication and enhances the resilience of system connectivity - important factors for large-scale networks [33]. Another upside of the distributed system is that agents can have lower computational requirements and can consequently be of lower cost and more easily replaced [34].

A body of existing work has considered coverage problems assuming a known environment. The environment may be known because the robot was given a map, or because it performed a pre-processing step to discover a map. The existence of a map permits planning based solutions to be pursued, in which way-points are identified offline. These way points are then ordered into a path plan before the robots engage in the actual activity of coverage. We consider these approaches here.
1) Viewpoint and Path Generation: Viewpoint generation describes the production of way-points and poses of a vehicle during coverage. Viewpoint generation is an essential aspect of an area coverage problem because it provides the necessary guidance for the coverage path and the behavior of the agents so that they can cover the structure or environment of interest as fast as possible.

Several previous studies propose model-based viewpoint generation approaches. The work in [35, 36] decomposed an area of interest into sub-regions and then assigns sub-regions to different robots. Many recent studies [1-3] propose a new solution for minimum-time coverage using a group of unmanned vehicles. The coverage problem is considered as a vehicle routing problem. In this method, the area of interest is decomposed into a set of sweeping line rows, as depicted in Fig. 1. The number of sweeps corresponds to the number of robots required to achieve coverage.

Although model-based techniques are quite useful, they are more suitable if one has a prior model of the environment. In the cases of unobserved or large scale environments, whose characteristics are difficult to determine in advance, model-free viewpoints generation techniques are preferred.

Frontier-based methods are a range of state-of-the-art model-free techniques aimed at leveraging the benefits of tree-based frameworks and search algorithms. These approaches dominate recent literature on environment exploration and map construction tasks due to their accuracy and efficiency [14, 37, 38]. The desirable unexplored sub-areas searched by a tree-based algorithm, so-called frontiers, are assigned as temporary destinations to visit by a single robot or group of robots. When there are no new frontiers left to detect, the entire environment is deemed to have been explored thoroughly [24]. Some other techniques based on frontiers that reduce the computational requirements and exploration turnaround time can be found in [15, 39, 40]. Frontier-based exploration can perform well not only on a single mobile platform but also on multiple robots [41, 42].

Our study proposes a partially model-based method that leverages the frontier search algorithm’s advantages to control the swarm coverage behaviors. Our approach includes a model-based component that only requires the environment’s size as prior knowledge to form a spatial coverage-obstacle matrix within each individual robot’s memory. The matrices’ coverage status is shared locally to shape the swarm behaviors, leading to efficient exploration and coverage. However, we assume there is no prior knowledge of the obstacles located in the environment. Thus, a fully model-based method is inapplicable as it is impossible to compute a trajectory beforehand. To address this, a frontier-based (model-free) component is used in this paper to guide each robot to unexplored areas while obstacle avoidance behavior is achieved using a dynamic repulsive force technique.

2) Search and Coverage: In existing work that combines aspects of the problem studied in this paper, Wang, Liang, and Guan [43] introduce a frontier search algorithm combined with particle swarm optimization (PSO) to improve the coverage capability of a swarm of mobile robots. The algorithm implements two working states: exploration and walking. In the exploration state, robots in a networked team discover their own non-overlapping sub-regions by travelling to local frontiers within such sub-areas. Once a robot explores all cells in its assigned area, it adapts a walking state to find another sub-area to explore using the PSO algorithm. The PSO algorithm aggregates three behaviours: a force representing the robot’s direction in the previous time step, an attraction force toward the closest frontier cell in the map, and a dispersion force vector toward a cell with the minimum amount of other robots on its direction. When the robot reaches another sub-area, the exploration state starts again. This algorithm encourages swarm dispersion to cover more space within a period. In contrast, our approach maintains a close-knit formation to facilitate short-range communication among robots.

In a different approach, Cheng et al. [44] designed a dynamic area coverage algorithm utilizing a swarm of autonomous robots, along with a leader-follower formation strategy. The robots possess only limited-range sensing and data processing capabilities. Due to these limitations, the authors present a hypothesis which states that using a flocking approach to create a formation of robots during the task would improve the exploration and coverage effectiveness and efficiency. The follower robots’ location in the formation is dynamically adapted based on the movement of a leader robot in order to avoid obstacles. Meanwhile, a Braitenberg-motion-based coverage strategy is applied to cover the environment. Their findings demonstrate that the formation of multiple robots created by flocking behaviors is superior to an individual coverage strategy in terms of completion time and stability.

As described earlier in this section, for the area coverage problem, the cooperative robots need to discover unknown environments in minimum time and maintain proximity to support communication. The research in this paper is motivated by these goals.

C. Reynolds’ Boids Model of Swarming

This section introduces the underlying swarming approach used in this paper: the boids model. A basic Boids swarming model [9] can be viewed as a type of rule-based reasoning to create swarming. The three fundamental rules are:

- Cohesion: An agent should move towards the average position of its neighbours.
- Alignment: An agent should steer to align itself with the average heading of its neighbours.
Separation: An agent should move to avoid collision with its neighbours.

The rules are generally implemented as forces that act on agents when a certain condition holds. Suppose we have a group of $N$ agents $A^1, A^2, A^3, \ldots A^N$. At time $t$, agent $A^i$ has a position, $p^i_t$, and a flocking velocity, $v^i_{flock(t)}$. At each time step $t$, the velocity of agent $A^i$ is updated as follows:

$$v^i_{flock(t+1)} = v^i_{flock(t)} + W_cv^i_{c(t)} + W_av^i_{a(t)} + W_s v^i_{s(t)} + (W_{av}v^i_{av(t)}) or (W_{av}v^i_{w(t)})$$

where $v^i_{c(t)}$ is the vector of cohesion force; $v^i_{a(t)}$ is the vector of alignment force; $v^i_{s(t)}$ is the vector of separation force; and $v^i_{w(t)}$ is the vector of the environment boundary (“wall”) avoidance force.

We distinguish the wall and obstacle avoidance forces in this paper because the environment boundary is known in advance, but the position of obstacles is not. Different positive weights $W_c, W_a, W_s, W_w$, and $W_{av}$, are used to characterize the strengths of corresponding forces. Once a new velocity has been computed, the position of $p^i_{t+1}$ of agent $A^i$ in the next time step $t+1$ is computed as follows:

$$p^i_{(t+1)} = p^i_t + v^i_{flock(t+1)}$$

Formally, we can define a subset $N^i$ of agents within a certain range $R$ of $A^i$ as follows:

$$N^i = \{A^k \mid k \neq i \land ||p^k_t - p^i_t|| < R\}$$

The number of agents in such a subset can be denoted by $|N^i|$. Different ranges may be used to calculate cohesion, alignment and separation forces respectively, or other factors such as the communication range of a boid. The average position $\bar{c}^i_t$ of agents in subset $(N^i)_t$ whose positions are within the cohesion range $R_c$ of $A^i$ is calculated as:

$$\bar{c}^i_t = (\sum_k p^k_t)/(|N^i|)$$

The vector of cohesion force is the vector from the current position of agent $A^i$ towards this average position:

$$v^i_{c(t)} = \bar{c}^i_t - p^i_t$$

Similarly, we can calculate the average position $\bar{s}^i_t$ of agents in subset $(N^i)_t$ whose positions are within the separation range $R_s$ of $A^i$ as:

$$\bar{s}^i_t = (\sum_k p^k_t)/(|N^i|)$$

The vector away from this position is calculated as:

$$v^i_{s(t)} = p^i_t - \bar{s}^i_t$$

The average velocity $v^i_{a(t)}$ of agents in subset $(N^i)_t$ whose positions are within the alignment range $R_a$ of $A^i$, is calculated by:

$$v^i_{a(t)} = (\sum_k p^k_{flock(t)})/(|N^i|)$$

These vectors can then be normalised and multiplied by their corresponding weights, before calculating the next velocity. The newly calculated velocity may be further normalised and then scaled by a speed value chosen in the range of $[V_{min}, V_{max}]$.

In this paper, a boundary wall collision avoidance velocity has been designed to keep the robots within a known rectangular area. The wall collision avoidance velocity $v^i_{w(t)}$ is defined by two components for a 2-D environment:

$$w^i_{x(t)} = \begin{cases} -1 & \text{if } x^i_t < x \\ 0 & \text{if } x^i \leq x^i_t < x \\ 1 & \text{if } x^i_t > x \end{cases}$$

$$w^i_{y(t)} = \begin{cases} -1 & \text{if } y^i_t < y \\ 0 & \text{if } y^i \leq y^i_t < y \\ 1 & \text{if } y^i_t > y\end{cases}$$

where $(x, y)$ and $(x, y)$ denote the left, right, bottom, and top limits of the workspace.

Similar to the force calculation formula for environment boundaries, an obstacle avoidance force is calculated for obstacles, which is defined to be inversely proportional to the relative position between the $i^{th}$ robot and an unexpected obstacle $p^o_{ro}$. It can be computed as follows when the collision distance $R_{av}$ is violated:

$$v^i_{av(t)} = \begin{cases} -p^o_{ro(t)} & \text{if } p^o_{ro(t)} \leq R_{av} \\ 0 & \text{if } p^o_{ro(t)} > R_{av} \end{cases}$$

In this paper we will demonstrate both simulated and real robots applying boid rules. Specifications of the sensors used to detect boundaries and unknown obstacles will be discussed later in the paper. We first present the formulation of our frontier-led swarming algorithm.

III. A NOVEL FRONTIER-LED SWARMING ALGORITHM

This paper addresses the problem of coverage in a partially unknown environment with communication constraints. Specifically, the boundaries of the environment are pre-set, but the positions of obstacles within those boundaries are unknown. Communication is limited to short-range communication only. Here, we propose a distributed (decentralised) solution that requires a coverage matrix, search component and formation control. Therefore, this section is split into three parts: Section A introduces the space coverage matrix, Section B describes the frontier search algorithm, and Section C describes the swarm formation controller.

A. Local Coverage and Obstacle Matrix

Unlike a centralized setting where a common, global coverage matrix may be used by all agents, in this paper, every agent stores its own memory, a 2D array $C_A$, composed of square grid cells of value $\varphi_{XY}(X, Y)$ of appropriate resolution $\varepsilon$ covering the environment whose maximum and minimum coordinates $\gamma$ and $\delta$ on the $x$ and $y$ axes are given in advance. We call this the local coverage-obstacle (CO) matrix of the agent. The map can be decomposed into dimensions as follows:

$$\text{dim}(X, Y) = \text{int}(\frac{\gamma - x \cdot \delta - y}{\varepsilon})$$
where $x$ and $y$ are positive integer numbers. For each agent, all cells are first initialised as uncovered:

$$\varphi_{A^i}(X,Y) = 0, \ \forall X,Y \in \mathbb{N}. \quad (14)$$

If the agent passes through an undiscovered or covered square cell $(X,Y)$, then set:

$$\varphi_{A^i}(X,Y) = 1. \quad (15)$$

Otherwise, if the agent senses an obstacle cell $(X,Y)$ within a certain radius, then set:

$$\varphi_{A^i}(X,Y) = 2. \quad (16)$$

When the distance between two agents $A^i$ and $A^k$ is smaller than a communication radius $R_c$, they exchange their local CO matrices. To do this, agent $A^i$ will update its internal CO matrix incorporating data from agent $A^k$’s CO matrix as follows:

$$\varphi_{A^i}(X,Y)(t+1) = \max(\varphi_{A^i}(X,Y)(t), \varphi_{A^k}(X,Y)(t)). \quad (17)$$

This localised sharing of CO matrices aims to reduce network load and reduce the impact of issues such as loss of line-of-sight between pairs of robots. Short-range agent communications are well facilitated by swarming movements since the robots are continually pulled together due to the cohesion force that controls their movement (See Section 2.C). We now explain how a frontier search force can be incorporated with this swarming movement.

### B. Frontier-led swarming

In frontier search, the environment is approximated into a 2D or 3D grid map, where each cell possesses an occupancy value of 1 for explored, or 0 for unexplored. Frontiers are regions on the boundary that separates explored cells from unexplored cells. Any cell between an explored and unexplored cell is called a frontier cell. Groups of connected frontier cells are called frontier regions. Robots attempt to navigate to the nearest frontier region. The descriptions expressed above are detailed in Fig. 2.

Let $n^t_j$ denote the frontier cell $j$ at time step $t$. First, all valid frontier cells at time $t$ are collected into a list of frontier cells $\Gamma_t = [n^0_t, ..., n^t_j]$, arranged in a queue data structure as described by Algorithm 1. A breadth-first search algorithm, illustrated in Algorithm 2, is then implemented on each frontier cell in $\Gamma_t$ to map the frontier cells and relevant unexplored cells into a corresponding frontier region and add it in a set $\Xi_t = \{\zeta^1_t, ..., \zeta^H_t\}, \ H \leq J$. Finally, the frontier regions are ranked to minimise the Euclidean distance $D^\zeta_{\Xi_t}$ from the robot’s current position to the closest frontier cell of that region and the grid size $S^\zeta_{\Xi_t}$ of the frontier area that frontier cell represents. The utility function is defined as:

$$F^\zeta_{\Xi_t} = \Psi_D D^\zeta_{\Xi_t} - \Psi_S S^\zeta_{\Xi_t}, \quad (18)$$

$$\zeta_t = \arg \min_{\zeta_t \in \Xi_t} F^\zeta_{\Xi_t}, \quad (19)$$

where $\Psi_D$ and $\Psi_S$ represent weighting parameters associated with the two terms. It can be seen that Eq. 18 is minimized when the distance is small and the frontier size is large.

After an accessible frontier $\zeta_t$ is determined as a desired destination by (19), the algorithm then steers the robot towards that region’s centroid. The search process is repeated by each robot until no more frontiers in the internal maps are found.

**Algorithm 1 Frontier cell search**

**Input:** current cell $c_t(X,Y)$

**Output:** list of frontiers $\Xi_t$

1. Initialize a queue with $c_t(X,Y)$ as element $\Omega \leftarrow c_t(X,Y)$
2. Initialize a list of frontiers $\Xi_t \leftarrow \emptyset$
3. **while** $\Omega$ is not empty **do**
4.   Dequeue a cell $c \in \Omega$
5.   Get set $\Pi$ including 4-connected cells of $c$
6.   **for** $c' \in \Pi$ **do**
7.     **if** $c'$ is a free cell **then**
8.       Add to queue $\Omega \leftarrow c'$
9.     **end if**
10.   **if** $c'$ is an unexplored frontier cell and $c'$ does not belong to any frontier **then**
11.     Find frontier region $\zeta_t(c')$ using Algorithm 2
12.     Add to frontier list $\Xi_t \leftarrow \zeta_t(c')$
13.   **end if**
14. **end for**
15. **end while**

Return $\Xi_t$

After the best frontier cell is selected, the attractive velocity toward the centre of the chosen frontier $p_\zeta$ acting on the relevant robot is obtained by:

$$v_{\text{frontier}} = \begin{cases} 
    p_\zeta - p^r, & \text{if } v^r_s = 0 \\
    0, & \text{if } v^r_s \neq 0
\end{cases} \quad (20)$$
Algorithm 2 Frontier region connectivity search

**Input:** frontier cell \( n^t \)

**Output:** a frontier region contains frontier and unexplored cells as 8-connected region \( \zeta(n^t) \)

Initialize a queue with \( n^t \) as element \( \Phi \leftarrow n^t \)

Initialize frontier region \( \zeta(n^t) \leftarrow n^t \)

while \( \Phi \) is not empty do

Pop a cell \( c \in \Phi \)

Get set \( \Pi \) including 8-connected cells to \( c \)

for \( c' \in \Pi \) do

if \( c' \) is an unexplored free cell then

Add to queue \( \Omega \leftarrow c' \)

Add to frontier region \( \zeta(n^t) \leftarrow c' \)

end if

end for

end while

Return \( \zeta(n^t) \)

---

**Algorithm 2** Frontier region connectivity search

**Input:** frontier cell \( n^t \)

**Output:** a frontier region contains frontier and unexplored cells as 8-connected region \( \zeta(n^t) \)

Initialize a queue with \( n^t \) as element \( \Phi \leftarrow n^t \)

Initialize frontier region \( \zeta(n^t) \leftarrow n^t \)

while \( \Phi \) is not empty do

Pop a cell \( c \in \Phi \)

Get set \( \Pi \) including 8-connected cells to \( c \)

for \( c' \in \Pi \) do

if \( c' \) is an unexplored free cell then

Add to queue \( \Omega \leftarrow c' \)

Add to frontier region \( \zeta(n^t) \leftarrow c' \)

end if

end for

end while

Return \( \zeta(n^t) \)

---

In case the same best cell is simultaneously selected by two robots, a potential robot-robot collision can occur. To solve this issue, when two robots are closer than the critical radius \( R_s \), they compute only the separation vector and neglect the frontier vector, as indicated in (20).

The fused velocity vector, as shown in Fig. 3, is the summation of the frontier velocity and the flocking velocities:

\[
v_r = \sum_{i=1}^{N} v_{i, \text{flock}} + v_{\text{frontier}}, \tag{21}\]

Based on the fused velocity vector, the robot’s linear and angular velocity can be computed as follows:

\[
V = \sqrt{v_{rx}^2 + v_{ry}^2}, \quad \omega = k(\tan^{-1}(v_{ry}/v_{rx}) - \theta), \tag{22}\]

where \((v_{rx}, v_{ry})\) indicate the robot velocities on the \(x\) and \(y\) axes. The robot’s angular velocity is \(\omega\). Furthermore, \(\theta\) is the robot’s heading angle and \(k\) is a constant gain.

Fig. 4 illustrates a flowchart of the whole proposed control algorithm.

---

C. Obstacle Avoidance

In an uncertain or unknown environment, the robot can collide with another robot or an unexpected obstacle when traveling to the target point without a proper obstacle avoidance strategy. In this paper, we assume LiDar sensors provide data to implement a suitable obstacle avoidance algorithm. We use a dynamic distance-based repulsive scale \(\delta\) in this work to continuously convert the cohesion and alignment velocity vectors into the collision avoidance force as in (23)-(26). By doing this, the cohesion force and the separation force acting on the robot will not be cancelled out when the robot is exploring nearby obstacles or in narrow passages.

\[
\delta^t = \frac{R_{av} - p_{ro}(t)}{R_c}, \tag{23}\]
\[
W_{av}^i = W_{av}^i \delta^t \tag{24}\]
\[
W_{a}^i = W_{a}^i \max((1 - 2\delta^t), 0) \tag{25}\]
\[
W_{c}^i = W_{c}^i \max((1 - 2\delta^t), 0) \tag{26}\]

---

IV. EXPERIMENTS

In this section we first describe the performance metrics against which we evaluate our algorithm in Section A. We
then describe the robot setup in Section B and the environment setup in Section C. Comparative algorithms are described in Section D. Results and discussion are presented in Section E.

A. Performance Metrics

Four evaluation metrics are used for the different algorithms presented in this article. The first two metrics are concerned with the exploration and coverage performance of the system. The second two metrics are concerned with the self-organising formation control of the system. The first metric we will examine is the coverage percentage \((CP)\) (27). This metric tracks the percentage of the known region that has been visited by any robot.

\[
CP = \frac{|\Theta_e|}{|\Theta|} \times 100\% \tag{27}
\]

where \(|\Theta_e|\) and \(|\Theta|\) are the number of covered cells and the total number of cells to be covered in this problem respectively.

The second metric is the turnaround time \((TT)\). This metric summarises the coverage performance in terms of the time taken for the robots to indirectly cover the known environment, including mapping of all unknown obstacles.

The third metric is the ‘group’ metric \(G\) (28), which estimates how closely a set of robots is clustered together (a lower value indicates greater grouping). The grouping metric \(G\) is defined by Equation 28 where \(N_r\) is the number of the robots, \(\bar{p}_r\) is the average position of the \(N_r\) robots at the given time, and \(T_0\) is the time that the robots start moving.

\[
G = \sum_{t=T_0}^{T_0+15dt} \frac{1}{N_G} \sum_{i=1}^{N_r} \frac{||p_i^t - \bar{p}_r||_2}{N_G}, \tag{28}
\]

where \(dt\) depicts the sample time. \(N_G = 15\) represents the sample length of the group metrics.

The fourth metric is the ‘order’ metric \(O\) (29), which computes how similarly the robots are aligned (a lower value indicates greater alignment). The order metric \(O\) is defined by Equation 29 below where \(\bar{v}_r\) is the average velocity of the \(N_r\) robots at the given time.

\[
O = \sum_{t=T_0}^{T_0+15dt} \frac{1}{N_O} \sum_{i=1}^{N_r} \frac{||v_i^t - \bar{v}_r||_2}{N_O}, \tag{29}
\]

where \(N_O = 15\) represents the sample length of the order metrics.

Five runs were conducted for each experiment, and the evaluation metrics’ mean and standard deviation are computed and shown in all result tables in Section 4.E. Video data was also collected. All video demonstrations showing spatial coverage of Robot 1 in every experiment can be viewed at the following address: https://youtu.be/4mDpP_TMjFU; https://youtu.be/XqImdQJvkjQ; https://youtu.be/2t-X7_5FeIY; and https://youtu.be/xOJviz0ee0g.

B. Experimental Setup of Robots

The experiments in this paper use TurtleBot Burger and Pioneer 3DX robots. TurtleBots are small, reliable, low-cost two-wheel differential drive unmanned ground vehicle (UGV) platforms (see Fig. 5). Pioneer 3DX is a three wheeled robot. In our experimental setup, the robots are controlled by on-board computers integrated with the Robot Operating System (ROS) and other compatible open-source software. We utilise 360-degree LiDAR sensors for obstacle detection.

To control these UGVs, ROS supports two control input signals, namely yaw rate and linear velocity. These desired speeds are computed using Eq. 22, and then used as the reference signals for inner loop PID controllers which control the servo motors driving the wheels of each robot type.

In our real-time, controlled experiments, the behavior of all robots is captured by a VICON Tracker motion capture system and transmitted to each robot every 0.02s through the User Datagram Protocol (UDP) network protocol. In future real applications, we envisage this central localisation node would be replaced with a GPS or ultra-wideband (UWB) sensor mounted on each robot to provide the global position of each robot. The CO matrices for each robot are shared with neighbouring robots at an update rate of 0.01s. The sampling time is fixed at 0.02s. All proposed algorithms and the multi-ROS Masters were implemented on each robot (see Fig. 6).

Table I shows the parameter setup for the physical experiments and the two different types of robots employed in this work. These parameters were set and tuned manually using established rules of thumb for boid swarms [9]. Specifically, the cohesion force was set at a low value, with the separation and obstacle avoidance forces set stronger, but operating in a smaller radius. The separation radius was selected to maintain the physical safety of each robot, and other parameters were tuned manually.

C. Experimental Environments

Three experimental environments were established: (1) a small, empty square arena \(3.6 \times 3.6m\) based on our VICON motion capture system laboratory; (2) a larger \(4.4 \times 4.8m\)
TABLE I: TurtleBot Burger and Pioneer 3-DX Robot Parameters

| Parameter | Description | Burger | P3-DX |
|-----------|-------------|--------|-------|
| \(W_a\)   | Alignment weight | 0.5    | 0.6   |
| \(W_c\)   | Cohesion weight  | 0.23   | 0.21  |
| \(W_s\)   | Separation weight | 1.1    | 1.2   |
| \(W_w\)   | Boundary avoidance weight | 1.1    | 0.05  |
| \(W_f\)   | Cohesion force weight | 0.08   | 0.03  |
| \(W_{av}\) | Obstacle avoidance force weight | 1.1    | 1.1   |
| \(R_a\)   | Alignment radius  | 1.5    | 1.5   |
| \(R_s\)   | Separation radius | 0.65   | 0.85  |
| \(R_c\)   | Cohesion radius   | 1.5    | 1.5   |
| \(R_{av}\) | Obstacle avoidance radius | 1.7    | 1.7   |
| \(\dot{V}\) | Maximum linear speed | 0.25   | 0.25  |
| \(\dot{\theta}\) | Maximum angular speed | 0.9    | 0.9   |
| \(\epsilon\) | CO matrix resolution | 0.3 m  | 0.3 m |

Fig. 6: Distributed Multi-ROS Master architecture for the swarming behavior of the cooperative ground vehicles. Only two robots are shown for simplicity.

square arena with four boxes representing unknown obstacles in our VICON laboratory; and (3) a simulated, cluttered, urban-like environment.

Environment 1 is shown in Fig. 7. In this picture we can see a heterogeneous swarm of six TurtleBot and two Pioneer UGVs. This environment permits us to examine the performance of our algorithm on real robots in a known, controlled setting.

An abstract representation of Environment 2 is shown in

Environment 3 is a 2D large-scale urban-like scenario 120×90m cluttered with obstacles simulated in a virtual Gazebo world (see Fig. 9). This environment enables us to examine the performance of our algorithm in a larger unknown environment. We use two scaled versions of this environment to investigate the impact of environment size on performance.

For measurement purposes in this paper, we assume that once all frontier cells in the local map of any robot are covered directly or indirectly, a message ‘completed task’ is sent to the remaining robots and the experiment ends. In future systems, such a message may also be sent to a base station or operator through a separate low-bandwidth longer range link. This message may also contain the complete coverage map, or the robot may be programmed to return to a set location to update other robots or the base station/operator with the final map.

D. Comparative Algorithms

In this section, the performance of our Frontier-led Swarming (FrontierSwarm) algorithm is examined and compared with
four alternatives. The first is a Frontier (Frontier) algorithm without swarming. This approach uses the frontier force and separation rule only (to prevent robots colliding). The purpose of this comparative algorithm is to provide benchmarking against a state-of-the-art exploration method.

The second comparative algorithm is a greedy (non-frontier) spatial coverage method with swarming (GreedySwarm). This approach uses the same boid rules and CO matrix. However, instead of using a frontier attraction force $v_{\text{frontier}}$ as described in Section 3, the final velocity is computed with an attractive force toward the nearest unexplored cell found by Algorithm 3 [20]. The purpose of this comparative algorithm is to confirm that frontier-led swarming provides an advantage over a more naive (greedy) exploratory approach.

The third is swarming (Swarm) only without any frontier or other attractive component towards unexplored cells [45]. The purpose of this comparative algorithm is to provide a baseline for understanding the impact of an exploratory force (whether frontier-led or greedy) on swarming performance.

Finally, a multi-vehicle routing-based area coverage approach [2, 3] is also considered in the comparative experiments. This provides a way to compare performance to an approach that has zero coverage redundancy (but which requires a map of the environment).

E. Results and Discussion

1) Experiment 1: Frontier-Led Swarming versus Frontier, Greedy Swarm and Swarm Algorithms: Tables II-III and the video demonstrations in the supplementary material present the coverage performance of our proposed algorithm. All algorithms are able to cover the whole region ($CPs = 100\%$). The differences in turnaround time are not significant at the 95% confidence level for the FrontierSwarm, Frontier and GreedySwarm algorithms for either the homogeneous or heterogeneous robot groups in either of the environments. However, the Swarm algorithm is significantly slower to achieve 100% coverage than either of the other three algorithms in the larger area. This is because the random movement of this algorithm has greater redundancy of coverage in the larger environment.

On the other hand, the group metric indicates that FrontierSwarm is maintaining a tighter formation than the Frontier approach in all experimental settings. The difference in performance is statistically significant at the 95% confidence level. The FrontierSwarm formation is not as tight as the Swarm or GreedySwarm approaches. This is likely due to the influence of the frontier rules pulling the swarm apart a little. The order metric is greater for the Frontier approach than for Frontier-Swarm, GreedySwarm and Swarm approaches. The difference is statistically significant in both cases. This makes sense because the swarming approaches should encourage ordering. All of these conclusions demonstrate that the FrontierSwarm algorithm can solve an exploration and coverage task while maintaining a small communication range between robots.

Moreover, from the FrontierSwarm group and order metric rows in the two tables, we can observe that the robots’ $G$ clustering locations with respect to the average robot positions are within $R_c$ (1m and 1.49m) and the velocity differences $O$ among robots maintain low values in all cases (0.3m.s$^{-1}$ and 0.11m.s$^{-1}$). This validates the swarming quality of the presented method.

2) Experiment 2: Frontier-led Swarming versus Multi-Robot Routing: The next experiment compares the performance of a multi-vehicle routing (sweep) approach to our FrontierSwarm strategy. This experiment is conducted in the large environment without obstacles, as the basic sweep algorithm does not account for obstacles. The heterogeneous, seven robot setup is used.

As can be seen in Fig. 10, both strategies achieve complete

---

**Algorithm 3 GreedySwarm cell selection using a greedy approach**

**Input:** current cell $c^0_l$; list of unexplored cells $\Theta_{u(t)}$; the map size; and the coverage matrix resolution $\varepsilon$

**Output:** Closest unexplored cell $c^*_u(t)$

1. Initialize the dimensions of coverage matrix by Equation 13.
2. Convert the robot position from the global coordinate frame into the coverage map coordinate frame.
3. For $c^j_u(t) \in \Theta_{u(t)}$
   - Find distance to all unexplored cells: $d^j_t = ||c^j_t - c^d_u(t)||_2$
4. Select the closest unexplored cell: $c^*_u(t) = \arg\min_{c^j_u(t) \in \Theta_{u(t)}} (d^j_t)$

Return $c^*_u(t)$
TABLE II: Experiment 1: Comparative performance of the proposed FrontierSwarm algorithm in a small area with known boundaries.

| Metrics    | FrontierSwarm | Frontier | GreedySwarm | Swarm |
|------------|---------------|----------|-------------|-------|
| CP (%)     | 100.00 ± 0.00 | 100.00   | 100.00      | 100.00|
| TT (s)     | 120.44 ± 17.01| 71.08 ± 5.80| 114.40 ± 26.64| 121.00 ± 18.04|
| G (m)      | 1.02 ± 0.02 | 1.72 ± 0.10 | 0.90 ± 0.00 | 0.86 ± 0.00 |
| O (m.s⁻¹)  | 0.30 ± 0.05 | 0.39 ± 0.10 | 0.11 ± 0.08 | 0.29 ± 0.10 |

TABLE III: Experiment 1: Comparative performance of the proposed FrontierSwarm algorithm in a large area with known boundaries and unknown obstacles.

| Metrics    | FrontierSwarm | Frontier | GreedySwarm | Swarm |
|------------|---------------|----------|-------------|-------|
| CP (%)     | 100.00 ± 0.00 | 100.00   | 100.00      | 100.00|
| TT (s)     | 173.68 ± 25.25| 131.86 ± 30.21| 153.82 ± 22.39| 179.17 ± 23.16|
| G (m)      | 1.39 ± 0.01 | 1.29 ± 0.01 | 0.86 ± 0.00 | 0.80 ± 0.00 |
| O (m.s⁻¹)  | 0.12 ± 0.03 | 0.19 ± 0.20 | 0.10 ± 0.07 | 0.12 ± 0.01 |

TABLE IV: Experiment 2: Comparison of FrontierSwarm to sweep routing in an known environment without obstacles.

| Metrics    | FrontierSwarm | Routing |
|------------|---------------|---------|
| CP (%)     | 100.00 ± 0.00 | 100.00 ± 0.00 |
| TT (s)     | 227.3 ± 30.00 | 142.0 ± 5.4 |

Fig. 11: Snapshots of coverage performance over time in the presence of faulty robots (or faulty communication links). The two failed ground vehicles at any time are marked with red circles.

Fig. 10: Comparison of trajectories of robots during (a) sweep routing and (b) frontier-led swarming. The frontier region, the unexplored cells, the explored cells, and the chosen frontier cell are represented by green square cells, white square cells, blue square cells, and yellow circle cells, respectively. Inter-robot communications are used by robots to validate the health status of other robots. After an interval of 3s in which a robot has been unable to obtain a reply from another robot, the offending robot is considered ‘lost,’ and the cohesion and alignment forces acting on the remaining robots are ignored. Separation forces continue to be applied based on LiDAR range information provided from the still active robots. This prevents the lost robot becoming a collision hazard to the other swarm members. However, if the failed robot’s neighbor data is recovered, it can immediately continue the area coverage task in collaboration with its team members.

3) Experiment 3: Robustness of Frontier-Led Swarming to Communication Failure or Robot Failure: In the third experiment, during the coverage mission, two robots among a team of eight heterogeneous mobile robots are randomly either shut down or turned on after every 30s. Inter-robot coverage (CPs = 100%), but the Routing approach achieves coverage in the shortest time (difference significant at the 95% level). This is because there is no redundancy of coverage in this approach. However, this algorithm is potentially less robust because if an individual robot fails during execution, completion of the coverage task will be jeopardised. Moreover, it also requires prior knowledge of the structure or environment. Finally, this method can only be applied to clear regions with no obstacles.

4) Experiment 4: Robustness of Frontier-Led Swarming against LiDar and Vicon Sensor Noise: In this experiment, the robustness of the proposed coverage algorithm is examined in a cluttered environment, where noisy LiDar and VICON measurements cause the misidentification of obstacle cells, covered cells or robot position. A measurement noise level of approximately 0.1m is introduced to examine this.

As depicted in the plots of Fig. 12(a), we establish a test environment with only one obstacle occupying a single cell. We see in Fig. 12(b) that due to the negative effect of LiDar noise, this obstacle cell is marked at many positions early in the exploration. The incorrect obstacle cells are corrected to covered cells after more robots pass close to those cells.
or robot-robot collisions were recorded, and the

The cell resolution size.

coverage or obstacle position errors will occur in multiples of

positioning error is larger than one cell resolution size, then

resolution provides some tolerance for positioning error. If the

incorrect record of coverage will occur. This is because the cell

is less than or equal to the cell resolution size, then no

cells. The impact will be as follows: If the position error

levels can theoretically result in incorrect labeling of covered

cells. The experiment was repeated 5 times for each comparative algorithm. The configuration settings are identical to those used in the physical experiments (Table I), except that the cell resolution \( \varepsilon \) is increased to 1m. At the beginning of the experiment, the robot locations are evenly distributed throughout the urban map by an operator.

As summarised in Table V, our proposed FrontierSwarm technique enables the robots to cover the whole unknown region with the smallest turnaround time (1522.3s for the FrontierSwarm approach compared to 1773.3s, 2234s, and 2325.7s of the Frontier, GreedySwarm and Swarm approaches, respectively). The results of the FrontierSwarm and Frontier approaches are not significantly different at the 95% confidence level. However, the FrontierSwarm approach has perfect coverage in a similar timeframe. Furthermore, the space coverage task cannot be accomplished by the GreedySwarm approach in a reasonable coverage period. Only 96% of cells are explored within 2325.7s. It is observed that the attractive force and the repulsive force acting on each robot are often canceled out when the robot is navigating between two facing obstacles. This can cause the robot to be stationary (indecisive) for an extended period. The conclusions consolidate our arguments that the frontier-led swarming algorithm is also well-suited for larger, unknown environments.

Despite the initial uniform distribution of robots in the large, simulated urban area, we see from Table V that robots using the FrontierSwarm, GreedySwarm and Swarm approaches move so that they group together and maintain close alignment during the experiments. Frontier swarm maintains closer grouping and alignment than Frontier, although has looser grouping and alignment than GreedySwarm and Swarm for the reasons discussed in Experiment 1.

**6) Experiment 6: Impact of number of robots and environment size on performance**: Table VI examines the impact on turnaround time when the number of robots or size of the environment is changed. We do this systematically by first increasing the number of robots and secondly re-scaling the size of the urban-like environment. Each result is computed by averaging over ten runs.

The first result we see is that the more robots are involved in the coverage task, the shorter the turnaround time that can be achieved. However, there appears to be evidence that the completion time may increase slightly if too many robots are used. Specifically, the 80 robot case in the 40m takes 45s more than the 70 robots case to entirely cover the same arena. This is caused because all robots must avoid collisions and as the density of robots increases they must take more evasive actions to do this and focus less on pursuing frontier regions. In the larger environment there is evidence that increasing the number of robots will decrease the time to achieve coverage.

---

**TABLE V: Experiment 5: Performance of FrontierSwarm in a cluttered urban environment with unknown obstacles.**

| Metrics | FrontierSwarm | Frontier | GreedySwarm | Swarm |
|---------|--------------|----------|-------------|-------|
| \( G \) (\( m \)) | 1.16 ± 0.01 | 1.39 ± 0.01 | 1.00 ± 0.00 | 0.97 ± 0.01 |
| \( O \) (\( m\alpha \)) | 0.24 ± 0.01 | 0.59 ± 0.01 | 0.33 ± 0.00 | 0.21 ± 0.00 |

---

Similarly, positioning errors caused by high VICON noise levels can theoretically result in incorrect labeling of covered cells. The impact will be as follows: If the position error is less than or equal to the cell resolution size, then no incorrect record of coverage will occur. This is because the cell resolution provides some tolerance for positioning error. If the positioning error is larger than one cell resolution size, then coverage or obstacle position errors will occur in multiples of the cell resolution size.

In this experiment with noise up to 0.1m no obstacle-robot or robot-robot collisions were recorded, and the \( G \) and \( O \) metrics remain within reasonable bounds of 1.05±0.078m and 0.15±0.015m, respectively. All frontier cells in Robot 1’s internal map were covered, and the complete coverage task was accomplished after 220.6s.

5) **Experiment 5: Performance of the FrontierSwarm Algorithm in a Simulated, Complex Environment**: This experiment examines performance of a larger heterogeneous swarm (16 robots) in the simulated urban-like environment. Again, the experiment was repeated 5 times for each comparative algorithm. The configuration settings are identical to those used in the physical experiments (Table I), except that the cell resolution \( \varepsilon \) is increased to 1m. At the beginning of the experiment, the robot locations are evenly distributed throughout the urban map by an operator.

As summarised in Table V, our proposed FrontierSwarm technique enables the robots to cover the whole unknown region with the smallest turnaround time (1522.3s for the FrontierSwarm approach compared to 1773.3s, 2234s, and 2325.7s of the Frontier, GreedySwarm and Swarm approaches, respectively). The results of the FrontierSwarm and Frontier approaches are not significantly different at the 95% confidence level. However, the FrontierSwarm approach has perfect coverage in a similar timeframe. Furthermore, the space coverage task cannot be accomplished by the GreedySwarm approach in a reasonable coverage period. Only 96% of cells are explored within 2325.7s. It is observed that the attractive force and the repulsive force acting on each robot are often canceled out when the robot is navigating between two facing obstacles. This can cause the robot to be stationary (indecisive) for an extended period. The conclusions consolidate our arguments that the frontier-led swarming algorithm is also well-suited for larger, unknown environments.

Despite the initial uniform distribution of robots in the large, simulated urban area, we see from Table V that robots using the FrontierSwarm, GreedySwarm and Swarm approaches move so that they group together and maintain close alignment during the experiments. Frontier swarm maintains closer grouping and alignment than Frontier, although has looser grouping and alignment than GreedySwarm and Swarm for the reasons discussed in Experiment 1.

6) **Experiment 6: Impact of number of robots and environment size on performance**: Table VI examines the impact on turnaround time when the number of robots or size of the environment is changed. We do this systematically by first increasing the number of robots and secondly re-scaling the size of the urban-like environment. Each result is computed by averaging over ten runs.

The first result we see is that the more robots are involved in the coverage task, the shorter the turnaround time that can be achieved. However, there appears to be evidence that the completion time may increase slightly if too many robots are used. Specifically, the 80 robot case in the 40×40m takes 45s more than the 70 robots case to entirely cover the same arena. This is caused because all robots must avoid collisions and as the density of robots increases they must take more evasive actions to do this and focus less on pursuing frontier regions. In the larger environment there is evidence that increasing the number of robots will decrease the time to achieve coverage.
of the swarming metrics studied in this paper. Such as turnaround time, or multiple objectives including some evolutionary algorithms to optimise either a single objective alignment. This could be a subject for future research using decreased by trading off communication range, grouping or explored cells to update and correct the coverage information.

First, the frontier-led swarming algorithm in this paper cannot track environmental changes (e.g., moving obstacle avoidance or new events) taking place in explored cells. One possible direction of future work is to combine the current approach with a digital pheromone to drive periodic revisits into explored cells to update and correct the coverage information. Secondly, this paper has not considered a search for ‘optimal’ swarm parameters. It may be that coverage time can be decreased by trading off communication range, grouping or alignment. This could be a subject for future research using evolutionary algorithms to optimise either a single objective such as turnaround time, or multiple objectives including some of the swarming metrics studied in this paper.

| Environment Size (m x m) | No. of Robots | Turnaround Time (s) |
|--------------------------|---------------|---------------------|
| 40 x 40                  | 40            | 467.25 ± 62.38      |
|                          | 50            | 371.79 ± 50.78      |
|                          | 60            | 327.60 ± 74.70      |
|                          | 70            | 321.00 ± 88.77      |
|                          | 80            | 366.88 ± 58.28      |
| 80 x 80                  | 40            | 1002.42 ± 171.04    |
|                          | 50            | 883.20 ± 123.33     |
|                          | 60            | 816.81 ± 161.42     |
|                          | 70            | 807.27 ± 157.07     |
|                          | 80            | 725.58 ± 155.82     |

V. CONCLUSION AND FUTURE WORK

This paper proposed a novel Frontier-led Swarming algorithm for exploration and coverage in unknown environments. Our key conclusions are:

1) Our FrontierSwarm algorithm can achieve coverage while maintaining a close-knit, self-organising formation of robots. This result was demonstrated in multiple small and larger scale environments, with different homogeneous and heterogeneous robot swarms.

2) Re-tuning of algorithm parameters is not required to transfer the system to different environments.

3) Using a multi-ROS Master implementation and local communication, an individual robot’s failure does not prevent a frontier-led swarm from completing its coverage mission, although its performance will slow. Likewise, frontier-led swarming can achieve coverage in the presence of sensor and positioning noise.

4) The frontier-led swarm algorithm can achieve coverage in cluttered environments with unknown obstacles, where comparative algorithms become trapped between obstacles.

A number of areas for future investigation exist, both in terms of the problem definition itself, and the solution strategy. First, the frontier-led swarming algorithm in this paper cannot track environmental changes (e.g., moving obstacle avoidance or new events) taking place in explored cells. One possible direction of future work is to combine the current approach with a digital pheromone to drive periodic revisits into explored cells to update and correct the coverage information. Secondly, this paper has not considered a search for ‘optimal’ swarm parameters. It may be that coverage time can be decreased by trading off communication range, grouping or alignment. This could be a subject for future research using evolutionary algorithms to optimise either a single objective such as turnaround time, or multiple objectives including some of the swarming metrics studied in this paper.

REFERENCES

[1] A. Xu, C. Viriyasuthee, and I. Rekleitis. Efficient complete coverage of a known arbitrary environment with applications to aerial operations.

[2] G. S. Avellar, G. A. Pereira, L. C. Pimenta, and P. Iscold. Multi-UAV routing for area coverage and remote sensing with minimum time. Sensors, 15(11):27783–27803, 2015.

[3] N. Karapetyan, K. Benson, C. McKinney, P. Taslakian, and I. Rekleitis. Efficient multi-robot coverage of a known environment. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1846–1852, 2017.

[4] J. Nauta, Y. Khaluf, and P. Simoens. Hybrid foraging in patchy environments using spatial memory. Journal of the Royal Society Interface, 17(166):20200026, 2020.

[5] X. Huang, M. Sun, H. Zhou, and S. Liu. A multi-robot coverage path planning algorithm for the environment with multiple land cover types. IEEE Access, 8:198101–198117, 2020.

[6] J. M. Palacios-Gasós, D. Tardioli, E. Montijano, and C. Sagüés. Equitable persistent coverage of non-convex environments with graph-based planning. The International Journal of Robotics Research, 38(14):1674–1694, 2019.

[7] O. Arslan and D. E. Koditschek. Voronoi-based coverage control of heterogeneous disk-shaped robots. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 4259–4266. IEEE, 2016.

[8] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence, 7(1):1–41, 2013.

[9] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th annual conference on Computer graphics and interactive techniques, pages 25–34, 1987.

[10] T. Vicsék, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. Physical review letters, 75(6):1226, 1995.

[11] João Paulo Lima Silva de Almeida, Renan Taizo Nakashima, Flávio Neves-Jr, and Lúcia Valéria Ramos de Arruda. Bio-inspired on-line path planner for cooperative exploration of unknown environment by a multi-robot system. Robotics and Autonomous Systems, 112:32–48, 2019.

[12] D. Yu, C. L. P. Chen, C. E. Ren, and S. Sui. Swarm control for self-organised system with fixed and switching topology. IEEE Transactions on Cybernetics, 2019.

[13] R. Konda, H. M. La, and J. Zhang. Decentralized function approximated q-learning in multi-robot systems for predator avoidance. IEEE Robotics and Automation Letters, 5(4):6342–6349, 2020.

[14] W. Gao, M. Booker, A. Adiwahono, M. Yuan, J. Wang, and Y. W. Yun. An improved frontier-based approach for autonomous exploration. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pages 292–297. IEEE, 2018.

[15] J. A. Caley, N. R. Lawrance, and G. A. Hollinger. Deep learning of structured environments for robot search. Autonomous Robots, 43(7):1695–1714, 2019.

[16] R. Alamdounh, T. Taha, L. Seneviratne, and et al. A survey on multi-robot coverage path planning for model reconstruction and mapping. SN Appl. Sci. 1, 847, 2019.

[17] Y. Kantaros, M. Thanou, and A. Tzes. Distributed coverage control for concave areas by a heterogeneous robot–swarm with visibility sensing constraints. Automatica, 53:195–207, 2015.

[18] M. Zhong and C. G. Cassandras. Distributed coverage control and data collection with mobile sensor networks. IEEE Transactions on Automatic Control, 56(10):2445–2455, 2011.

[19] Y. Wang and I. I. Hussein. Awareness coverage control over large-scale domains with intermittent communications. IEEE Transactions on Automatic Control, 55(8):1850–1859, 2010.

[20] M. Masár. A biologically inspired swarm robot coordination algorithm for exploration and surveillance. In 2013 IEEE 17th International Conference on Intelligent Engineering Systems (INES), pages 271–275, 2013.

[21] C. P. Le, A. Q. Pham, H. M. La, and D. Feil-Seifer. A multi-robotic system for environmental dirt cleaning. In 2020 IEEE/ACSE International Symposium on System Integration (SII), pages 1294–1299. IEEE, 2020.

[22] T. Kuyucu, I. Taney, and K. Shimohara. Superadditive effect of multi-robot coordination in the exploration of unknown environments via stigmergy. Neurocomputing, 148:83–90, 2015.

[23] K. Albina and S. G. Lee. Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms. IEEE Access, 7:14246–14259, 2019.

[24] Sujata Sharma and Ritu Tiwari. A survey on multi robots area planning. Intelligent Review Letters, 2:1–10, 2016.

[25] K. Albina and S. G. Lee. Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms. IEEE Access, 7:14246–14259, 2019.

[26] T. Kuyucu, I. Taney, and K. Shimohara. Superadditive effect of multi-robot coordination in the exploration of unknown environments via stigmergy. Neurocomputing, 148:83–90, 2015.

[27] K. Albina and S. G. Lee. Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms. IEEE Access, 7:14246–14259, 2019.

[28] Sujata Sharma and Ritu Tiwari. A survey on multi robots area planning.
[25] M. A. Yakoubi and M. T. Laskri. The path planning of cleaner robot for coverage region using genetic algorithms. *Journal of innovation in digital ecosystems*, 3(1):37–43, 2016.

[26] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.

[27] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. V. Le, P. Veerajagadheswar, K. Tiwari, and M. Ilyas. Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot. *Automation in Construction*, 112:103078, 2020.

[28] Y. Khaluf and P. Simoens. Collective sampling of environmental features under limited sampling budget. *Journal of Computational Science*, 31:95–110, 2019.

[29] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7:36, 2020.

[30] C. Luo, S. X. Yang, X. Li, and M. Q. Meng. Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots. *IEEE Transactions on Industrial Electronics*, 64(1):750–760, 2017.

[31] K. Albinia and S. G. Lee. Hybrid stochastic exploration using grey wolf optimizer and coordinated multi-robot exploration algorithms. *IEEE Access*, 7:14246–14255, 2019.

[32] J. Song and S. Gupta. Care: Cooperative autonomy for resilience and efficiency of robot teams for complete coverage of unknown environments under robot failures. *Autonomous Robots*, 44(3):647–671, 2020.

[33] M. Li, K. Lu, H. Zhu, M. Chen, S. Mao, and B. Prabhakaran. Robot swarm communication networks: architectures, protocols, and applications. In *2008 Third International Conference on Communications and Networking in China*, pages 162–166. IEEE, 2008.

[34] E. Sahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[35] A. Gautam, J. K. Murthy, G. Kumar, S. P. A. Ram, B. Jha, and S. Mohan. Cluster, allocate, cover: An efficient approach for multi-robot coverage. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 197–203. IEEE, 2015.

[36] Jalil Modares, Farshad Ghanei, Nicholas Mastronarde, and Karthik Dantu. Ub-anc planner: Energy efficient coverage path planning with multiple drones. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 6182–6189. IEEE, 2017.

[37] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai. A survey on inspecting structures using robotic systems. *International Journal of Advanced Robotic Systems*, 13(6):1729881416663664, 2016.

[38] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[39] I. Arvanitakis, K. Giannousakis, and A. Tzes. Mobile robot navigation in unknown environment based on exploration principles. In *2016 IEEE Conference on Control Applications (CCA)*, pages 493–498. IEEE, 2016.

[40] M. M. Almasri, A. M. Alajlan, and K. M. Elleithy. Trajectory planning and collision avoidance algorithm for mobile robotics system. *IEEE Sensors journal*, 16(12):5021–5028, 2016.

[41] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2135–2142. IEEE, 2017.

[42] M. Julià, A. Gil, and O. Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4):427–444, 2012.

[43] Y. Wang, A. Liang, and H. Guan. Frontier-based multi-robot map exploration using particle swarm optimization. In *2011 IEEE symposium on Swarm intelligence*, pages 1–6. IEEE, 2011.

[44] K. Cheng, Y. Wang, and P. Dasgupta. Distributed area coverage using robot flocks. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 678–683, 2009.

[45] C. Chibaya and S. Bangay. Flock inspired area coverage using wireless boid-like sensor agents. In *Tenth International Conference on Computer Modeling and Simulation (UKSim 2008)*, pages 144–149. IEEE, 2008.