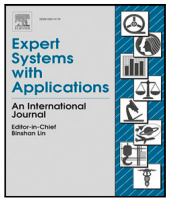
Expert Systems With Applications 227 (2023) 120183

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



A novel potential field model for perimeter and agent density control in multiagent swarms☆

Neil Eliot [a,*], David Kendall [b,1], Michael Brockway [b,2], Paul Oman [c], Ahmed Bouridane [d]

[a] School of Computer Science, University of Sunderland, Chester Road, Sunderland, SR1 3SD, UK
[b] Department of Computer and Information Sciences, Northumbria University, Ellison Place, Newcastle upon Tyne, NE1 8ST, UK
[c] Department of Mathematics, Physics and Electrical Engineering, Northumbria University, Ellison Place, Newcastle upon Tyne, NE1 8ST, UK
[d] Center for Data Analytics and Cybersecurity, University of Sharjah, United Arab Emirates

ARTICLE INFO

Keywords:
Swarms
Multiagent systems
Potential field
Perimeter control
Simulation

ABSTRACT

Currently, most potential field models for decentralised control of multiagent swarms use only single-valued parameters for the computation of control vectors. This restriction often limits the structures that can evolve, since agents are unable to modify their behaviour based on their structural role. This paper proposes an enhanced model that uses the perimeter status of agents in selecting control parameters. This allows a wider variety of emergent behaviours, many of which result in much improved swarm structures. The model is based upon equivalence classes of agent pairs, defined by their perimeter status. Array-valued parameters are introduced to allow each equivalence class to be given its own parameter values. The model also introduces a new control vector to 'flatten' reflex angles between neighbouring agents on the swarm perimeter, often leading to significantly improved swarm structure. Extensive experiments have been conducted that demonstrate how the new model causes a variety of useful behaviours to emerge from random swarm deployments. The results show that several important behaviours, such as shape control, void removal, perimeter packing and expansion, and perimeter rotation, can be produced without the need for explicit inter-agent communication. The approach is applicable to a variety of applications, including reconnaissance, area-coverage, and containment.

1. Introduction

This paper introduces a new model for agent coordination in swarm formations. The model follows the principles of earlier potential field models (Barnes et al., 2006b; Eliot et al., 2018; Gazi, 2005; Liang et al., 2019; Schneider & Wildermuth, 2003; Son et al., 2017) but is developed to allow greater control over the perimeter and agent density of emerging swarm structures. A great advantage of potential field models, particularly those that follow the approach of Reynolds (1987), is that they allow for decentralised control of large swarms of homogeneous agents, with low computation and communication requirements. This makes possible the deployment of large, robust swarms at low cost.

A disadvantage of such simple models is that the stable structures that emerge are limited to either straight lines or partial lattices (Eliot, 2017; Räz, 2013). Occasionally, the structures that emerge have anomalies such as concave 'dents' or convex 'peaks' in their perimeter, or voids in their internal core, which may arise as a result of agent failure or environmental disturbance. Such anomalies contribute to the disruption of otherwise well-structured swarms. The ability to create more regular structures, with greater control over the nature of their perimeters, is beneficial for effective deployment in applications such as reconnaissance and area coverage, where 'blind spots' are best eliminated (Elamvazhuthi & Berman, 2015), or containment, where a swarm is used to surround an object or region (Cao et al., 2012).

To develop new behaviours, relying on greater control over perimeter agents, requires that it is possible to identify the perimeter status of each agent, i.e. to determine for each agent whether it is on a perimeter or not. This is discussed by Eliot et al. in Eliot (2017), Eliot et al. (2018), Eliot et al. (2019) and the details are explained in Section 4.1.

☆ *Additional material:* A Jupyter notebook to run the simulations and generate the figures in the paper has been published as reproducible research at https://doi.org/10.24433/CO.9511427.v1.

* Corresponding author.
E-mail addresses: neil.eliot@sunderland.ac.uk (N. Eliot), kendall.d.j@gmail.com (D. Kendall), michael.brockway.99@gmail.com (M. Brockway), paul.oman@northumbria.ac.uk (P. Oman), abouridane@sharjah.ac.ae (A. Bouridane).

[1] Retired 2019.
[2] Retired 2018.

https://doi.org/10.1016/j.eswa.2023.120183
Received 22 February 2022; Received in revised form 1 April 2023; Accepted 15 April 2023
Available online 23 April 2023

The new model presented in this paper builds on this capability by using the perimeter status of agents to distinguish equivalence classes of agent pairs and to assign different parameter values to each class, allowing new emergent behaviours to be created. In addition, a novel 'convexity removal' vector is included in the new model and it is shown how this can achieve a significant improvement in swarm structure. The simplicity of the model is such that large swarms of homogeneous agents can be controlled without the need for explicit inter-agent communication, and the swarm structures that evolve can be tailored to the requirements of specific swarming applications.

Extensive experiments, using a discrete-time simulator that implements the new model, demonstrate the existence of model parameters that can be used to control the properties of swarms that emerge from random deployments of agents. Such properties include agent density, circularity, self-healing, goal-seeking, and perimeter structure and rotation. Heuristic guidelines are offered for the 'tuning' of model parameters. A new k-NN distance metric is introduced and applied to the analysis of the experimental results.

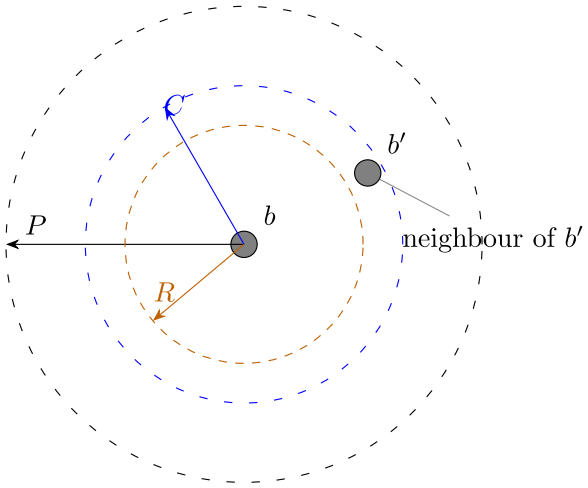


Fig. 1. Agent fields.

2. Related work

The use of potential fields to coordinate agents in swarm evolution was introduced by Reynolds for the graphical simulation of flocks of birds (Reynolds, 1987). The approach has been researched extensively since then in a variety of multiagent swarms in an attempt to improve their structure, manage obstacle avoidance, and enhance navigation (Barnes et al., 2006a, 2006b, 2007; Eliot et al., 2018; Gazi, 2005; Son et al., 2017). Improvements to swarm structure were achieved through a prototype framework for self-healing swarms, developed by Dai et al. who considered how to manage agent failure in hostile environments (Dai et al., 2006). This was similar to work by Vassev and Hinchey, who modelled swarm movement using the ASSL (Autonomic System Specification Language) (Vassev & Hinchey, 2009). This technique was employed by NASA (US National Aeronautics and Space Administration) for use in asteroid belt exploration as part of their ANTS (Autonomous Nano Technology Swarm) project. However, this work was focused on internal systems failure of agents, rather than on the removal of anomalies in agent distribution.

In the context of swarm structure maintenance, the need for formation control has been discussed by Speck and Bucci with respect to the diverse applications of swarms and the need to control swarm structure (Speck & Bucci, 2018). Roach et al. focused on the effects of sensor failure, and the impact that has on agent distribution (Roach et al., 2015). Lee and Chong identified the issue of concave edges within swarms in an attempt to create regular lattice formations (Lee & Chong, 2008). The main focus of their work is the dynamic restructuring of inter-agent formations. Ismail and Timmis demonstrated the use of *bio-inspired* healing using *granuloma formation*, a biological method for encapsulating an antigen (Ismail & Timmis, 2010). They also considered the effect that failed agents can have on a swarm when traversing a terrain (Timmis et al., 2016). Jung et al. proposed a mediator-based approach using monolithic agents (Jung & Goodrich, 2013), however, the formations are 'controlled' by a human and not truly emergent behaviours. Karthikeyan and Ali proposed a communications-based technique to create specific shapes, such as lines, circles and triangles, in a defined Euclidean space (Karthikeyan & Ali, 2006). Bruemmer et al. also proposed shape-forming swarms, again using a communications-based methodology (Bruemmer, 2002). López-González et al. demonstrated a parallel genetic algorithm for distance control in multi-agent swarms comprising a small number of agents (López-González et al., 2020). Johnson and Brown have developed computation-free techniques to allow swarms of simple robots to form a perimeter around a target. Their approach requires external control over the environment in order to control the behaviour of the swarm (Johnson & Brown, 2016). He et al. also proposed a formation control mechanism which is communications-based (He et al., 2018). Fedele and D'Alfonso proposed a model for swarm structure control but it assumes 'without delay' communications-based architecture (Fedele & D'Alfonso, 2017). Fedele and D'Alfonso also proposed a matrix-based coordination algorithm to modify the movement of a swarm's agents, allowing shape formation in a fixed 3D environment (Fedele & D'Alfonso, 2021). Extensions to the potential field model have been proposed by Hao Fang et al. for connectivity preservation in flocks of multiagent systems (Fang et al., 2017) and by Ivić et al. for area coverage (Ivić et al., 2017).

This paper proposes a new extension to the potential field model that can be used to induce, among other behaviours, perimeter expansion, perimeter packing and void removal. This is an extension of the work presented by Eliot et al. (2019) and Ismail and Timmis (2010), Timmis et al. (2016), and builds on the work of McLurkin and Demaine on the detection of perimeter types (McLurkin & Demaine, 2009). A key benefit of the approach proposed here is that it is entirely decentralised and can be implemented using only sensors, without requiring explicit communication between agents, so facilitating the control of larger, emergent swarms than would otherwise be possible.

3. Basic swarming model

This section introduces a basic model of swarm evolution that is in the tradition of potential field models introduced by Reynolds (1987) and since adopted and adapted by many others. The model abstracts the physical characteristics of agents and their environment. It is a summary of the model used in our earlier work (Eliot et al., 2018; Eliot et al., 2019) and serves as an introduction to the approach and as the starting point for the development of a new, more powerful, model.

A swarm is a finite set of agents where each agent is uniquely defined by its position vector in 2-D Euclidean space. The movement of each agent, b, is determined by a vector that is the sum of a number of weighted component vectors, each derived from interactions with agents located in one or more potential fields associated with b. Fig. 1 shows an agent and its fields. P is the perception field, i.e. the range of the sensor array. C is the cohesion field, and R is the repulsion field. Typical component vectors include: cohesion, v_c, tending to move b towards its neighbours; repulsion, v_r, tending to move b away from its neighbours; and direction, v_d, tending to move b towards a goal, if there is one. The final resultant vector for b is computed as the sum (1).

$$v(b) = v_c(b) + v_r(b) + v_d(b) \tag{1}$$

The cohesion vector for agent b is calculated based on the proximity of its cohesion neighbours, $n_c(b)$, defined in (2), where S is the set of all

agents in the swarm, C is the radius of the cohesion field for all agents, and $\|b' - b\|$ is the distance from b to b'.

$$n_c(b) = \{b' \in S \ : \ b' \neq b \wedge \|b' - b\| \leq C\} \tag{2}$$

We assume a vector from b to b' for each individual agent b' in the set of cohesion neighbours of b. These vectors are scaled by a constant weighting factor, k_c, and the overall cohesion vector for b, $v_c(b)$, is calculated as the average of these vectors, as shown in (3).

$$v_c(b) = \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} k_c(b' - b) \tag{3}$$

where $|n_c(b)|$ is the size of the set of cohesion neighbours of b.

The repulsion vector for agent b is calculated based on the proximity of its repulsion neighbours, $n_r(b)$, defined in (4).

$$n_r(b) = \{b' \in S \ : \ b' \neq b \wedge \|b' - b\| \leq R\} \tag{4}$$

Informally, $n_r(b)$ is the set of agents in the swarm, S, that are positioned inside the repulsion field of b. We assume a vector from b away from b' for each individual agent b' in the set of repulsion neighbours of b. The magnitude of each vector is inversely proportional to the distance between b and b'. These vectors are scaled by a constant weighting factor, k_r, and the overall repulsion vector for b, $v_r(b)$, is calculated as the average of these vectors, as shown in (5).

$$v_r(b) = \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} k_r \left(1 - \frac{R}{\|b' - b\|}\right)(b' - b) \tag{5}$$

The direction vector for an agent is calculated based on the position of the goal relative to the agent. If there is no goal then the direction vector is the null vector. A goal is modelled simply as a point in the same 2-D space inhabited by agents. For a goal, g, the direction vector for agent b, $v_d(b)$, is defined by (6).

$$v_d(b) = k_d(g - b) \tag{6}$$

where k_d is a constant weighting factor for direction vectors.

A more usual presentation of this kind of model is to lift the scalars, k_c, k_r, and k_d out of (3), (5) and (6), and apply them appropriately in (1), giving an equivalent model, as below:

$$\begin{aligned} v(b) &= k_c v_c(b) + k_r v_r(b) + k_d v_d(b) \\ v_c(b) &= \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} (b' - b) \\ v_r(b) &= \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} \left(1 - \frac{R}{\|b' - b\|}\right)(b' - b) \\ v_d(b) &= g - b \end{aligned} \tag{7}$$

The slightly unorthodox presentation of (1), (3), (5) and (6) helps to make clearer the relationship between the basic model and the new model presented in Section 4.

4. A new model of swarm evolution

In this paper, we propose three main extensions to the basic model:

1. the addition of a 'gap-filling' vector to reduce both concavity and convexity in swarm perimeters,
2. the use of array-valued parameters so that each agent's behaviour can be modified depending on its perimeter status and the perimeter status of its neighbours, and
3. the addition of a rotation vector for further control of the movement of agents.

4.1. Perimeter detection

Fig. 2(a) shows a simple swarm. Perimeter agents are highlighted in red and can form part of an inner or outer boundary. A swarm usually also contains non-perimeter (internal) agents, which are shown in black. Each agent's perimeter status is identified using a cyclic analysis of its cohesion neighbours (Fig. 2(b)). Ghrist et al. discuss a similar technique using sweep angles (Ghrist et al., 2008) as do McLurkin and Demaine (2009).

We order the cohesion neighbours of an agent b by their *polar angle* (α) with respect to b and the positive x-axis (Fig. 2(b)).

The polar angle with respect to b of a neighbour, b', $\alpha(b, b')$, is the counter-clockwise angle that vector $\vec{bb'}$ makes with the positive x axis, shown in Fig. 2(b) as α_i and defined by (8).

$$\begin{aligned} &\alpha(b, b') = \theta \text{ where} \\ &\quad \wedge \ 0 \leq \theta < 2\pi \\ &\quad \wedge \ \|b' - b\|(\cos\theta, \sin\theta) = b' - b \end{aligned} \tag{8}$$

We denote by $\langle b_0, b_1, .., b_{n-1}\rangle_b$ a permutation of the set of neighbours, $n_c(b)$, that is sorted in non-decreasing order of *polar angle*, i.e. $\alpha(b, b_0) \leq \alpha(b, b_1) \leq \cdots \leq \alpha(b, b_{n-2}) \leq \alpha(b, b_{n-1})$. Given such a list of length at least 2, we say that neighbours b' and b'' are *consecutive* if and only if, for some $i \in \{0, \ldots, n-1\}$, $b' = b_i$ and $b'' = b_{(i+1)\%n}$, where $\%n$ indicates the modulus with respect to integer n.

An agent b is on a perimeter if it satisfies any one of three conditions:

1. the agent has fewer than 3 neighbours, or
2. consecutive neighbours are not within each other's cohesion field, or
3. consecutive neighbours subtend a reflex angle (shown in Fig. 2(b) as δ_3).

A function, $\mathsf{prm}(b)$, specifies these conditions formally. Let b be the agent of interest and b', b'' any pair of consecutive neighbours of b, then $\mathsf{prm}(b)$ is true if any one of the following conditions is satisfied:

1. $|n_c(b)| < 3$, or
2. $b' \notin n_c(b'')$, or
3. $\delta > \pi$, where $\delta = \alpha(b, b'') - \alpha(b, b')$ if $\alpha(b, b'') - \alpha(b, b') \geq 0$ else $\delta = \alpha(b, b'') - \alpha(b, b') + 2\pi$.

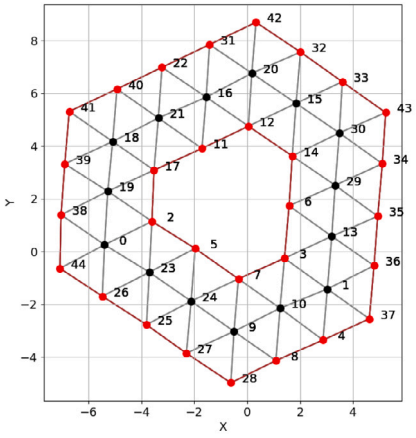
4.2. Gap-filling

A beneficial side-effect of the perimeter detection algorithm is that it becomes possible to identify an additional vector to influence the movement of perimeter agents. If an agent has been identified as a perimeter agent because it satisfies condition (2) above, namely that there is a consecutive pair in its angle-sorted list of cohesion neighbours that are not cohesion neighbours of each other, then we say that there is a *gap* between that pair of agents, and a vector is created to tend to move the perimeter agent towards the midpoint of that gap, i.e. a gap-filling vector for b contributes a motion of b towards the midpoint of a gap identified in the perimeter test for b.

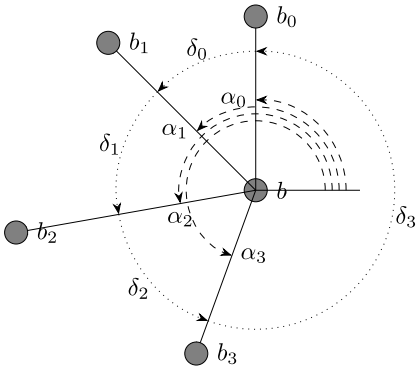
More formally, let $\langle b_0, b_1, .., b_{n-1}\rangle_b$ be the cohesion neighbours of b in polar angle order, and let b' and b'' be the first pair of consecutive neighbours that satisfy condition (2) of the perimeter function $\mathsf{prm}()$, then a gap-filling vector, $v_g(b)$, for agent b is defined by (9).

$$v_g(b) = k_g \left(\frac{b' + b''}{2} - b\right) \tag{9}$$

where k_g is a constant weighting factor for the gap-filling vector, allowing the combination of it with other motion vectors (cohesion, repulsion, ...) to be "tuned". Note that if there is no pair of consecutive neighbours of b, satisfying the condition then $v_g(b)$ is defined to be the zero vector, $\vec{0}$.



(a) External and internal perimeters: perimeter agents shown in red, internal agents shown in black.



(b) Agent neighbour angles: $\alpha_0 = \alpha(b, b_0)$, $\alpha_1 = \alpha(b, b_1)\ldots$ and $\delta_0 = \alpha(b, b_1) - \alpha(b, b_0)$, $\delta_1 = \alpha(b, b_2) - \alpha(b, b_1)\ldots$ Note δ_3 is an example of a *reflex angle*.

Fig. 2. Perimeter detection.

The use of gap-filling vectors was first introduced in Eliot et al. (2019) where it was shown to be effective in quickly reducing internal voids. In this paper, we introduce the idea that a gap-filling vector can also be created, in a similar way, for perimeter agents that satisfy condition (3) of the perimeter function prm(). The use of such a vector is referred to as 'flattening reflex angles'. See Fig. 2(b), where a gap-filling vector of this type would contribute to a movement of agent b towards the midpoint of a straight line between agents b_0 and b_3, so tending to flatten the reflex angle δ_3. This seemingly small extension to the idea of gap-filling is shown in Section 5 to have a very significant effect in controlling the shape of the external perimeter during swarm evolution.

4.3. Agent pair perimeter classes and array-valued parameters

The main contribution of this paper is the recognition that the identification of agents as perimeter agents or non-perimeter (internal) agents allows the use of different parameter values for each agent, depending on its own perimeter status and the perimeter status of its neighbours. This idea is explained in more detail in the following.

The perimeter and internal agents of a swarm S are denoted S_p and S_i, where

$$S_p = \{b \in S \ : \ \mathsf{prm}(b)\}, \text{and}$$
$$S_i = \{b \in S \ : \neg\mathsf{prm}(b)\}$$

Each pair of agents in $S \times S$ is in one of the sets $S_{ii} = S_i \times S_i$, $S_{ip} = S_i \times S_p$, $S_{pi} = S_p \times S_i$, or $S_{pp} = S_p \times S_p$, depending on the perimeter status of each agent in the pair. The sets S_{ii}, S_{ip}, S_{pi} and S_{pp} partition the set of agent pairs, $S \times S$, and are the equivalence classes of an equivalence relation on $S \times S$. Therefore, we call these sets the *agent pair perimeter classes* of S or, more briefly, the perimeter classes of S.

The perimeter class of a pair of agents, $b, b' \in S$, is determined by the perimeter status (i — internal, p — perimeter) of each, as shown in the array:

$$\begin{array}{cc} & \begin{array}{cc} i & p \end{array} \\ \begin{array}{c} b \\ \end{array} & \begin{array}{c} i \\ p \end{array} \left[\begin{array}{cc} S_{ii} & S_{ip} \\ S_{pi} & S_{pp} \end{array}\right] \end{array} \quad b'$$

If we consider Fig. 2(a) then we can see the following examples of perimeter classes containing agent pairs: $(18, 21) \in S_{ii}$, $(18, 39) \in S_{ip}$, $(39, 19) \in S_{pi}$, and $(41, 40) \in S_{pp}$.

Now the idea is to replace the single-valued parameters, R, k_c, and k_r in (3) and (5) with array-valued parameters, so that each perimeter class can be given its own specific value. For example, instead of a single, constant value for k_c, we can have an array of values, such as:

$$\begin{array}{cc} & \begin{array}{cc} i & p \end{array} \\ \begin{array}{c} k_c \\ \end{array} & \begin{array}{c} i \\ p \end{array} \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array}\right] \end{array} \quad b'$$

where $k_c = 10$ if $(b, b') \in S_{ii}$, $k_c = 20$ if $(b, b') \in S_{ip}$, etc.

Similarly, a 1-D array of values can be used instead of a scalar for parameters associated with vectors that depend only on a single agent. For example, in computing a weighted direction vector, instead of a single, constant value for k_d, we can have an array of values, such as $[10, 20]$, where $k_d = 10$ if $b \in S_i$ and $k_d = 20$ if $b \in S_p$.

Two dimensional array-valued parameters are normally written linearly as, for example, $k_c = [[10, 20], [30, 40]]$, and are indexed using any of $0/\mathsf{False}/i$ for internal agents and $1/\mathsf{True}/p$ for perimeter agents, so, for the example above, $k_c[\mathsf{True}, \mathsf{False}] = 30$ and is used for any agent pair in the perimeter class S_{pi}.

4.4. The new model

At this point, most features of the new model can be presented as a straightforward revision of the basic model of Section 3, in which the single-valued parameters, R, k_c, k_r, and k_d used in (3), (5), and (6) are replaced by array-valued parameters. It should now be clear that the k_c, k_r and k_d parameters were placed inside the summation in these equations in order to emphasise the relationship of the new model to the basic model.

In addition, a gap-filling vector, as defined in Section 4.2, is included in the resultant vector to promote self-healing, perimeter control and better swarm structure.

Finally, rotation vectors, $v_a(b)$, are introduced that allow additional control of agent behaviour. Rotation vectors are described in more detail below.

The definition of the vectors, $v(b)$, is now given by (10).

$$v(b) = v_c(b) + v_r(b) + v_d(b) + v_g(b) + v_a(b) \tag{10}$$

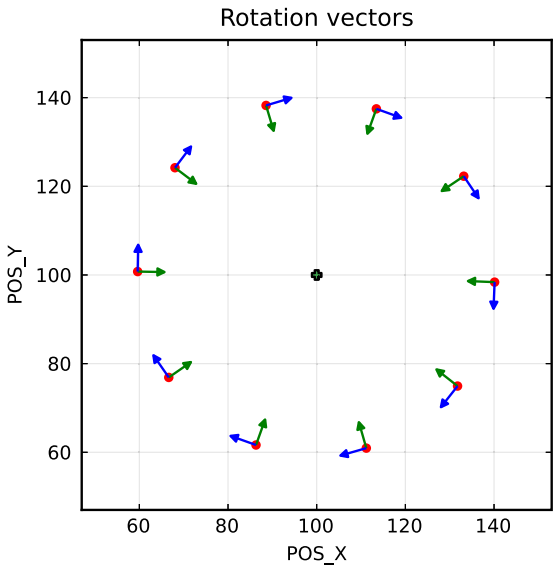


Fig. 3. Rotation vectors: the red circles represent perimeter agents, green arrows represent direction vectors, blue arrows represent rotation vectors, the cross at (100, 100) represents the goal.

The cohesion vectors, $v_c(b)$, in the new model are defined by (11),

$$v_c(b) = \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} k_c[s, s'](b' - b) \tag{11}$$

where $s = \text{prm}(b)$ is the perimeter status of b and $s' = \text{prm}(b')$ is the perimeter status of b'. k_c is a 2×2 boolean-indexed, array-valued parameter, as discussed in Section 4.3, that determines the weight of each component of the cohesion vector according to the perimeter class of the agent pair (b, b'). The definition of cohesion neighbours, $n_c(b)$, is unchanged from (2).

The definition of repulsion vectors requires a minor modification to the notion of repulsion neighbours, to allow the radius of the repulsion field to be varied according to the perimeter status of the agents involved, as shown in (12).

$$n_r(b) = \{b' \in \mathcal{S} : b \neq b' \wedge \|b' - b\| \leq R[s, s']\} \tag{12}$$

where $s = \text{prm}(b)$, $s' = \text{prm}(b')$, and R is a 2×2 boolean-indexed array of constants that determine the radius of the repulsion field.

Now, the repulsion vectors, $v_r(b)$, are defined by (13).

$$v_r(b) = \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} k_r[s, s'] \left(1 - \frac{R[s, s']}{\|b' - b\|}\right)(b' - b) \tag{13}$$

where $s = \text{prm}(b)$, $s' = \text{prm}(b')$, and k_r is a 2×2 boolean-indexed array of constants that determine the weight of a component of the repulsion vector according to the perimeter class of the agent pair (b, b').

The direction vectors, $v_d(b)$, are defined by (14).

$$v_d(b) = k_d[s](g - b) \tag{14}$$

where $s = \text{prm}(b)$, k_d is a 1-D, boolean-index array of constants giving the weights of the direction vector for internal and perimeter agents, and g denotes the position of the goal.

The gap-filling vectors, $v_g(b)$, have a non-null value only for perimeter agents, and are defined as in (9).

We also extend the basic model with rotation vectors, $v_a(b)$, which, for the purposes of this paper, we restrict to simple rotations of the direction vectors.

Let $A(\alpha)$ denote the matrix that rotates a point by angle α by pre-multiplication of its column position vector, i.e.

$$A(\alpha) = \begin{bmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{bmatrix}$$

Table 1
Baseline parameters.

Parameter	Value
C	3.0
R	[[2.0, 2.0], [2.0, 2.0]]
k_c	[[0.15, 0.15], [0.15, 0.15]]
k_r	[[50.0, 50.0], [50.0, 50.0]]
k_d	[0.0, 0.0]
k_g	0.0
k_a	[0.0, 0.0]
r_a	[0.0, 0.0]
rgf	False

Then, the rotation vectors, $v_a(b)$, are defined by (15).

$$v_a(b) = k_a[s]A(r_a[s])\frac{v_d(b)}{\|v_d(b)\|} \tag{15}$$

where $s = \text{prm}(b)$, and k_a and r_a are 1-D, boolean-indexed arrays of constants giving the weights and rotation angles, respectively, for internal and perimeter agents.

As a simple example of rotation vectors, consider Fig. 3 which shows a small swarm of 10 agents that has evolved to a stable state in which all agents are perimeter agents and the rotation vectors have been derived from the direction vectors, according to (15), using the parameters $k_a = [0, 1]$ and $r_a = [0, \pi/2]$.

Notice that a counter-clockwise rotation of the direction vectors about each agent will produce rotation vectors that tend to rotate the agents clockwise around the goal, and vice-versa.

These changes to a basic model, of the sort that has been used for many years, may appear simple but their effect on swarm evolution can be very significant, as is demonstrated in Section 5.

5. Experimental results

The results in this section are derived from a discrete time simulator that implements directly the model outlined in Section 4. This new model allows for highly configurable swarms, with good control over perimeter and overall swarm structure. We illustrate this with a number of examples and analyse the resulting structures using a variety of metrics.

5.1. Experiments

5.1.1. Baseline

The baseline parameters for all the experiments in this section are shown in Table 1, where C is the cohesion field radius, R is the repulsion field radius, k_c, k_r, k_d, k_g and k_a are the weightings of the cohesion, repulsion, direction, gap-filling and rotation vectors that contribute to the resultant vector for each agent, r_a specifies the angles of rotation for the rotation vectors, and rgf determines whether the flattening of reflex angles in the gap-filling algorithm is turned on or off. It can be assumed that any parameter values that are not mentioned explicitly in each experiment take their values from this table. These parameters are a typical set that would give a reasonable swarm structure if translated into the framework of the basic model of Section 3.

The baseline swarm consists of 400 agents which are distributed randomly over an area of 20×20 units ($-10 \rightarrow +10$), as shown in Fig. 4(a). One can imagine that this simulates a randomised drop of agents into an area of interest, with an average density of 1 agent per square unit. Hundreds of other random distributions have been simulated and the results presented in this section are typical. The resultant position of each agent after 2000 simulation steps is shown in Fig. 4(b). This is the structure that the baseline swarm reaches under the control of the basic model and serves as a useful point of comparison with the structures that can be achieved using the new model. Note that a simulation step consists of calculating the resultant

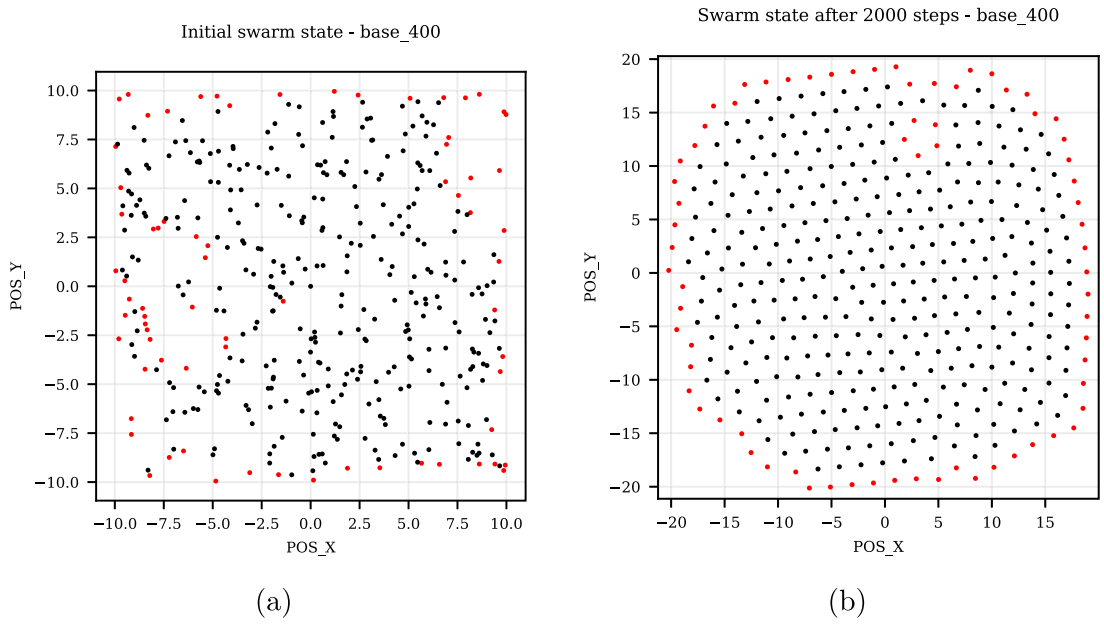


Fig. 4. Baseline swarm configurations.

vector for each agent, as described in Section 4, deriving a unit vector from the resultant by normalisation, and then scaling this vector by a 'speed' factor to represent the number of distance units moved per step. All of the examples in this section use a speed factor of 0.05. The choice of 2000 steps for each simulation allows each example experiment to reach a 'stable' state.

5.1.2. Void reduction

This set of simulations demonstrates the effectiveness of the new model in creating self-healing, evolutionary swarms that exhibit significantly improved control over swarm perimeter and overall structure than any other comparable approach. The method is an extension of the one described in Eliot et al. (2019). The importance of the extension of the method to include the flattening of reflex angles is illustrated.

We begin by taking the expanded baseline swarm of Fig. 4(b) and creating a 'hole' in it by removing 30 agents. One can imagine that this represents the simultaneous, catastrophic failure of a cluster of agents. The resulting swarm is shown in Fig. 5(a). The ineffectiveness of the basic model in removing the void is illustrated in Fig. 5(b) which shows the resulting swarm structure after 2000 simulation steps using the baseline parameters.

By contrast, Fig. 5(c) shows the structure of the swarm after 2000 steps with $k_g = 150$, i.e. applying the method of Eliot et al. (2019). We can see that the void has been removed successfully but the swarm has become distended.

Finally, Fig. 5(d) shows the structure of the swarm after 2000 steps with $k_g = 150$ and the flattening of reflex angles turned on (rgf = True). The improvement in the swarm structure at this point is striking and illustrates the effectiveness of the new approach. The swarm is almost perfectly circular, agents are evenly spaced and the perimeter is stable.

5.1.3. Compact perimeter

The utility of the array-valued parameters, introduced in the new model, is illustrated with the creation of a swarm that has a densely packed perimeter and by default exhibits a self-healing behaviour. The key idea here is to use the perimeter status of agents to select appropriate values for the cohesion and repulsion weights, k_c and k_r. By reducing the repulsion weight of agent pairs in the S_{ip} class, more internal agents are able to join the perimeter, and by increasing the cohesion weight of the agent pairs in the S_{pp} class, the perimeter agents become more densely packed.

Fig. 6(a) shows the effect of using the array-valued parameter values:

$k_c = [[0.15, 0.15], [0.15, 150.0]]$, and

$k_r = [[50.0, 10.0], [50.0, 50.0]]$

The cohesion weight for agent pairs in S_{pp} is given by $k_c[1, 1] = 150.0$ and the repulsion weight for agent pairs in S_{ip} is given by $k_r[0, 1] = 10.0$. In addition, gap-filling is applied with weight $k_g = 150$ and reflex angles are flattened (rgf = True) to smooth the perimeter. This also induces a compression effect on the swarm, ensuring that any voids are filled. The use of a high k_g value helps to create a stable, circular swarm structure. Starting from the agent positions shown in Fig. 4(a), these parameters lead to the structure of Fig. 6(a) within 2000 steps. In fact, a similar regular structure, with a tightly packed perimeter, is obtained from the baseline swarm with a void (Fig. 5(a)) using exactly the same parameters, illustrating the self-healing properties of the model. This is a significant advance over other comparable approaches.

5.1.4. Expanded perimeter

This simulation shows a different example of the use of array-valued parameters in the new model. It illustrates how a significantly different swarm can be made to evolve simply by changing the values of the parameters k_c, k_r, and k_g. In this case, we generate a swarm that expands from the compact configuration of the base state (Fig. 4(a)), *increasing* the distance between perimeter agents, while maintaining a dense core. This kind of evolution may be useful for a swarm moving into a hostile environment where the perimeter agents can act as an early warning system, while a sufficient number of internal agents is preserved for mission completion. As in the previous example, the swarm exhibits a self-healing property due to internal compression and, so, remains robust in the presence of agent failure.

The parameters shown below achieve this effect by creating a high degree of repulsion between agent pairs in S_{ip} ($k_r[0, 1] = 1000.0$), i.e. the internal agents nearest to the perimeter strongly repel the perimeter agents. The cohesion weight between perimeter agents is set just high enough to maintain a stable perimeter ($k_c[1, 1] = 15.0$).

$k_c = [[0.15, 0.15], [0.15, 15.0]]$, and

$k_r = [[50.0, 1000.0], [50.0, 50.0]]$

Gap-filling ($k_g = 50$), with flattening of reflex angles (rgf = True), controls the perimeter to create a circular shape, as shown in Fig. 6(b).

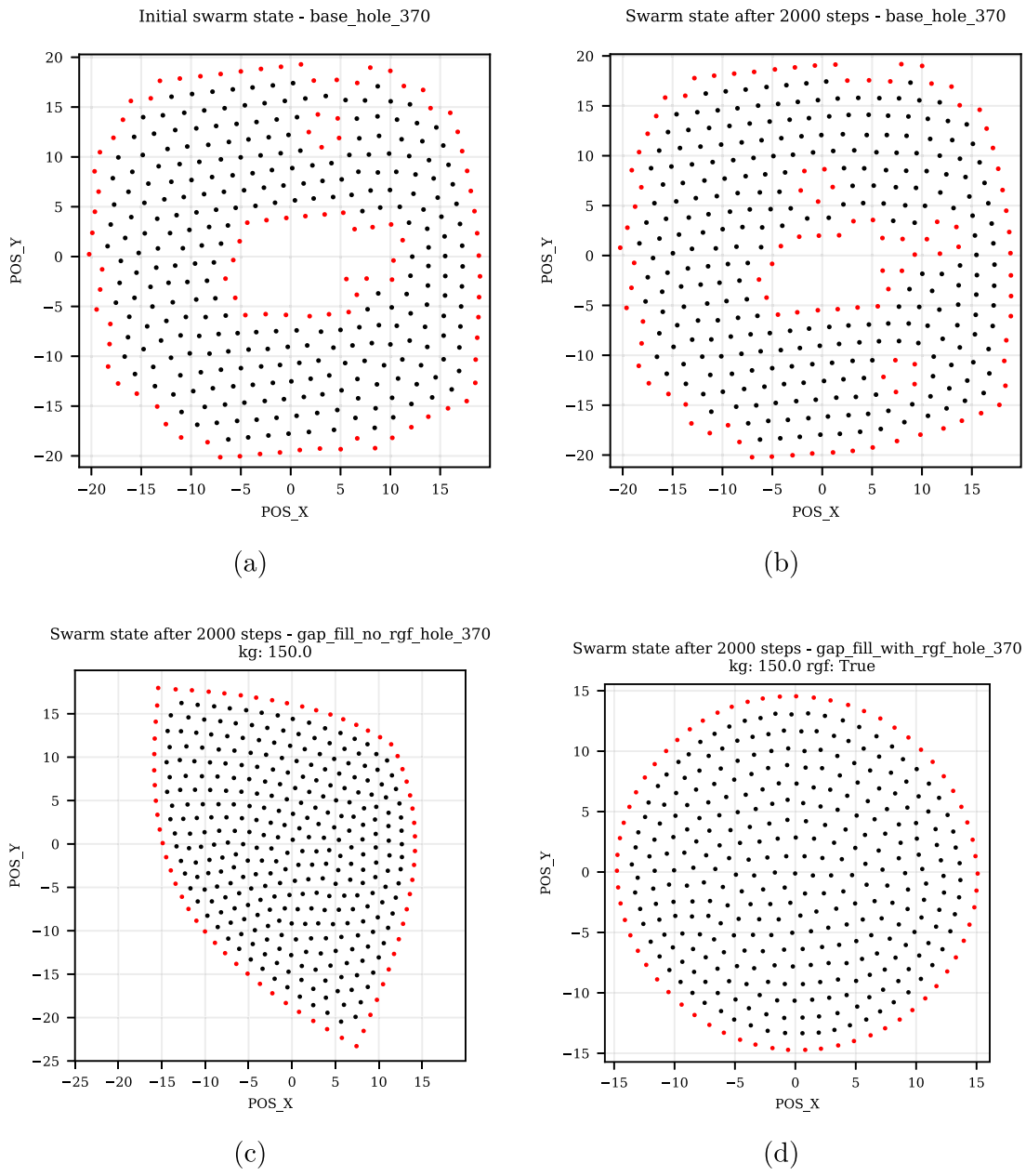


Fig. 5. Void reduction: (a) Expanded baseline swarm with void (b) No self-healing with baseline parameters (c) Swarm structure after 2000 steps with gap-filling (d) Swarm structure after 2000 steps with gap-filling and flattening of reflex angles.

Table 2
Rotating perimeter parameters.

Parameter	Small, goal-seeking	Large, low density
C	144	20
R	[[96, 96], [96, 108]]	[[16, 16], [16, 16]]
k_c	[[0.15, 0.15], [0.15, 0.15]]	[[0.25, 0.075], [0.15, 0.15]]
k_r	[[150, 50], [150.0, 50.0]]	[[120, 50], [120, 50]]
k_d	[0.15, 0.15]	[0.2, 0.2]
k_g	200	150
k_a	[0, 5]	[0, 15]
r_a	[0, π/2]	[0, π/2]
rgf	True	True

5.1.5. Rotating perimeter

The final two simulations illustrate the use of direction and rotation vectors to control the movement of agents: the first shows a small swarm of agents seeking a goal and forming a rotating perimeter around it; the second shows a much larger swarm expanding to a relatively low density compared to its initial state, again with a rotating perimeter. Fedele et al. (2022) discuss swarm rotation behaviours. This section shows the capabilities of a much simpler model. Table 2 gives the parameters for both simulations.

The diagram seen earlier in Fig. 3 shows the state of a swarm of 10 agents after evolving for 10,000 steps from an initial state in which the agents are distributed uniformly at random in a square of 10 sq. units, centred on position (0,0). Within 10,000 steps the agents have dispersed to cover an area of 5095 sq. units, the centroid of the swarm is directly over the goal, at (100,100), and the agents are rotating clockwise about the goal. A video animation showing the evolution of the swarm is available here in the online supplementary material. The potential of this behaviour to offer a low-cost solution to the basic border control problem (Marino et al., 2013; Pan et al., 2021) is clear.

Fig. 7(a) shows the evolution for 10,000 steps from the initial state of the baseline swarm, shown in Fig. 4(a), using the parameters in the

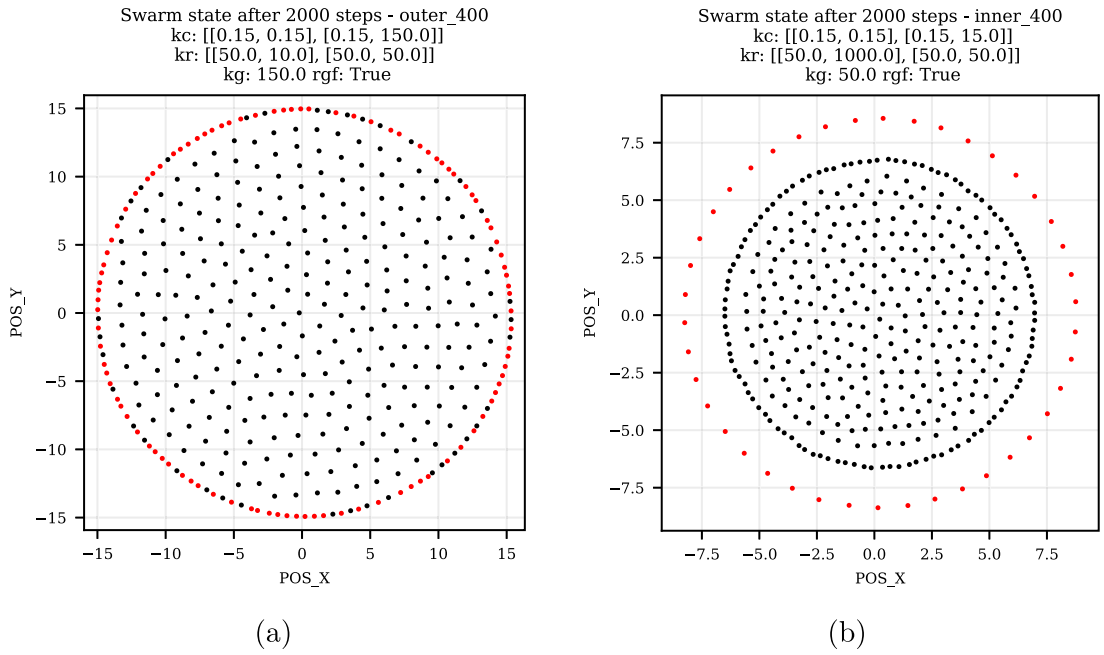


Fig. 6. Swarm configurations: Compact and expanded perimeters.

rightmost column of Table 2. The initial state covers an area of 400 sq. units. The state shown in Fig. 7(a) covers an area of 54,344 sq. units, an expansion factor of about 136. The agents are evenly dispersed and the swarm maintains an essentially circular shape. The perimeter agents rotate clockwise around the centroid of the swarm. Fig. 7(b) simulates a total failure of 10% of the agents of the swarm and Fig. 7(c) shows the recovery of the swarm structure within 5000 steps. The perimeter is repaired and continues to rotate as before. A video animation showing the evolution of the swarm is available here in the online supplementary material.

5.2. Analysis

Distance metrics of various sorts have been used by many researchers for swarm structure analysis (Barnes et al., 2006a, 2006b; Elamvazhuthi & Berman, 2015; Gazi, 2005; Schneider & Wildermuth, 2003). In this paper, we use a k-nearest neighbours metric on each of the perimeter relationship classes. This approach gives a good overview of the evolution of a swarm in respect of its perimeter and internal core.

A set of k-nearest neighbours of an agent b, taken from any set T of agents, is a set of size k such that any agent in that set is a cohesion neighbour of b and is closer to b than any other cohesion neighbour of b that is also in T but is not one of the k-nearest neighbours. If the size of the set of cohesion neighbours of b that are also in T is smaller than k, then we just take the whole set of cohesion neighbours as the k-nearest neighbours.

Formally, for any natural number $k > 0$, a set of k-nearest neighbours (k-NN) of an agent $b \in S$, with respect to any set $T \subseteq S$, is a set $n_k(b,T)$ that satisfies:

$$
\begin{aligned}
&n_k(b,T) \subseteq K \text{ where}\\
&\quad \wedge\ K = n_c(b) \cap T\\
&\quad \wedge\ (|K| \le k \implies n_k(b,T) = K)\\
&\quad \wedge\ (|K| > k \implies (|n_k(b,T)| = k\\
&\qquad \wedge \forall b' \in n_k(b,T)\ :\\
&\qquad\quad \|b' - b\| \le \min_{b'' \in (K \setminus n_k(b,T))} \|b'' - b\|))
\end{aligned}
\tag{16}
$$

Our k-NN metric computes the mean and standard deviation of agent distances for each of the perimeter classes: S_{ii}, S_{pi} and S_{pp}. The mean is calculated as shown in (17) where $\mu_d(S_1, S_2, k)$ is the mean distance between specified agent pairs in $S_1 \times S_2$, based on k-nearest neighbours.

$$
\mu_d(S_1, S_2, k) = \frac{\sum_{b \in S_1} \ \sum_{b' \in n_k(b,S_2)} \|b' - b\|}{\sum_{b \in S_1} |n_k(b, S_2)|} \tag{17}
$$

The standard deviation is calculated as shown in (18), where $\sigma_d(S_1, S_2, k)$ is the standard deviation from the mean of the distance between specified agent pairs in $S_1 \times S_2$, based on k-nearest neighbours.

$$
\sigma_d(S_1, S_2, k) = \sqrt{\frac{\sum_{b \in S_1} \ \sum_{b' \in n_k(b,S_2)} \Big(\|b' - b\| - \mu_d(S_1, S_2, k) \Big)^2}{\sum_{b \in S_1} |n_k(b, S_2)|}} \tag{18}
$$

Note, $\mu_d(S_1, S_2, k)$ and $\sigma_d(S_1, S_2, k)$ are defined only if $\sum_{b \in S_1} |n_k(b, S_2)| \neq 0$.

Now, the k-NN mean distances for the S_{ii}, S_{pi} and S_{pp} classes are given simply by $\mu_d(S_i, S_i, k)$, $\mu_d(S_p, S_i, k)$, and $\mu_d(S_p, S_p, k)$. The standard deviations are given similarly in the obvious way. In our experience, $\mu_d(S_i, S_p, k)$ and $\sigma_d(S_i, S_p, k)$ are less helpful indicators of swarm structure than $\mu_d(S_p, S_i, k)$ and $\sigma_d(S_p, S_i, k)$ and are disregarded in the rest of this paper.

We use the notation $\psi_d(S_1, S_2, k)$ to refer to the mean plus or minus one standard deviation, where these are defined.

$$
\begin{aligned}
\psi_d(S_1, S_2, k) = [&\mu_d(S_1, S_2, k) - \sigma_d(S_1, S_2, k),\\
&\mu_d(S_1, S_2, k) + \sigma_d(S_1, S_2, k)]
\end{aligned}
\tag{19}
$$

A plot of the distance metric over the evolution of a swarm gives a useful overview of the swarm's behaviour. Fig. 8 shows plots of the means of each perimeter class for some of the simulations considered above. Typically, a simulation run would be analysed using a variety of values of k, in order to get a clear understanding of the swarm's behaviour. In this paper, for reasons of space, the results of only one choice of k value for each perimeter class are presented. Fig. 5(d) shows a swarm structure in which inter-agent distances for all perimeter classes are roughly equal. Fig. 8(d) shows that a choice of $k = 2$ for the S_{ii} and S_{pp} classes and $k = 1$ for the S_{pi} class gives results that correspond with this observation, and these values of k are used throughout.

Fig. 8(a) shows the distance plot for the evolution of the baseline swarm. It can be seen that the swarm expands within 200–300 steps

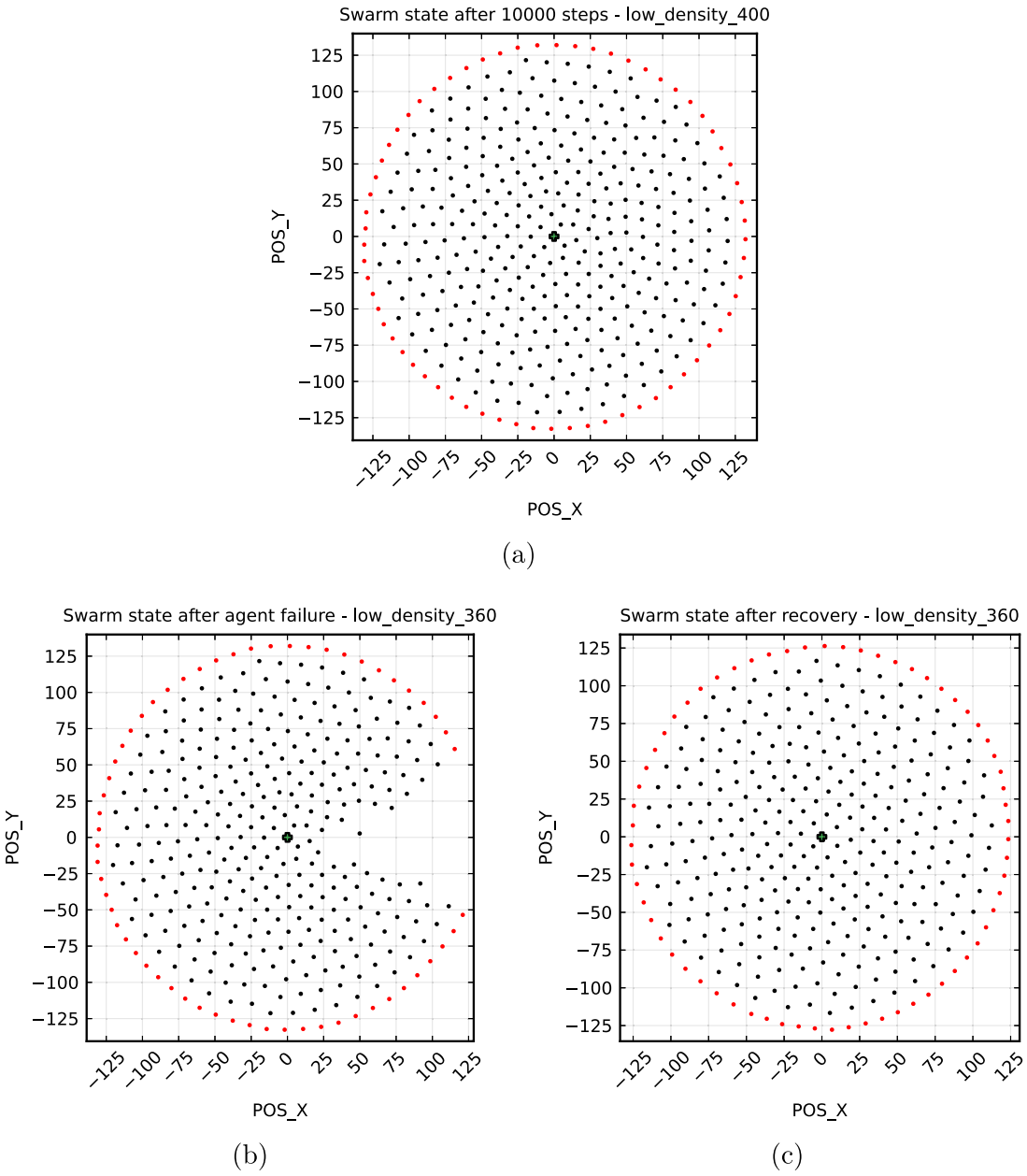


Fig. 7. Low density swarm with rotating perimeter: (a) swarm state after 10000 steps (b) swarm state after 10% of agents lost following step 10000 (c) swarm state after 5000 more steps showing recovery.

from its initial compact configuration to a stable expanded configuration in which agents have settled at a distance that is roughly equal to the radius of the repulsion field. There is a small difference in the mean distances for the different perimeter classes. The behaviour of the baseline swarm is typical of that produced by conventional swarming algorithms using single-valued potential fields and weights.

Fig. 8(b) shows the evolution of the expanded base swarm with a 'hole' near the centre. It shows a run of the swarm that exhibits very little change over the course of 2000 steps, as we can confirm by comparing the start and end structures in Fig. 5(a) and 5(b), respectively. The roughly constant higher distances between agents in the S_{pp} and S_{pi} classes are caused by the presence of the holes in the swarm and the failure of the use of the base parameters to close them.

Fig. 8(c) shows a run of a swarm starting from the same initial configuration but with gap-filling turned on (k_g = 150). The swarm is more compressed and still exhibits small differences in the mean distances between agents in different perimeter classes — between approximately 1.64 and 1.75 units. This causes the hole in the swarm to be closed but the degree of compression is not sufficient to create a circular swarm. In contrast, Fig. 8(d) shows a run of the swarm again starting from the same initial configuration but with both gap-filling (k_g = 150) and flattening of reflex angles (rgf = True). This produces very similar mean distances between agent pairs in different classes and the degree of compression is now sufficiently high to produce a remarkably regular, circular swarm structure (Fig. 5(d)).

A striking feature of the new potential field model with array-valued parameters, gap-filling, and flattening of reflex angles, is the degree of control over both the internal swarm structure and the perimeter that can be achieved simply by varying these parameter values. For example, the distance graph of Fig. 9 shows the run of a swarm, starting from the initial distribution of Fig. 4(a), that uses a set of parameters to produce a swarm with a compact internal core and a relatively expanded perimeter (Fig. 6(b)). The mean distance between agent pairs in the S_{ii} class settles at about 0.56 units, whereas the

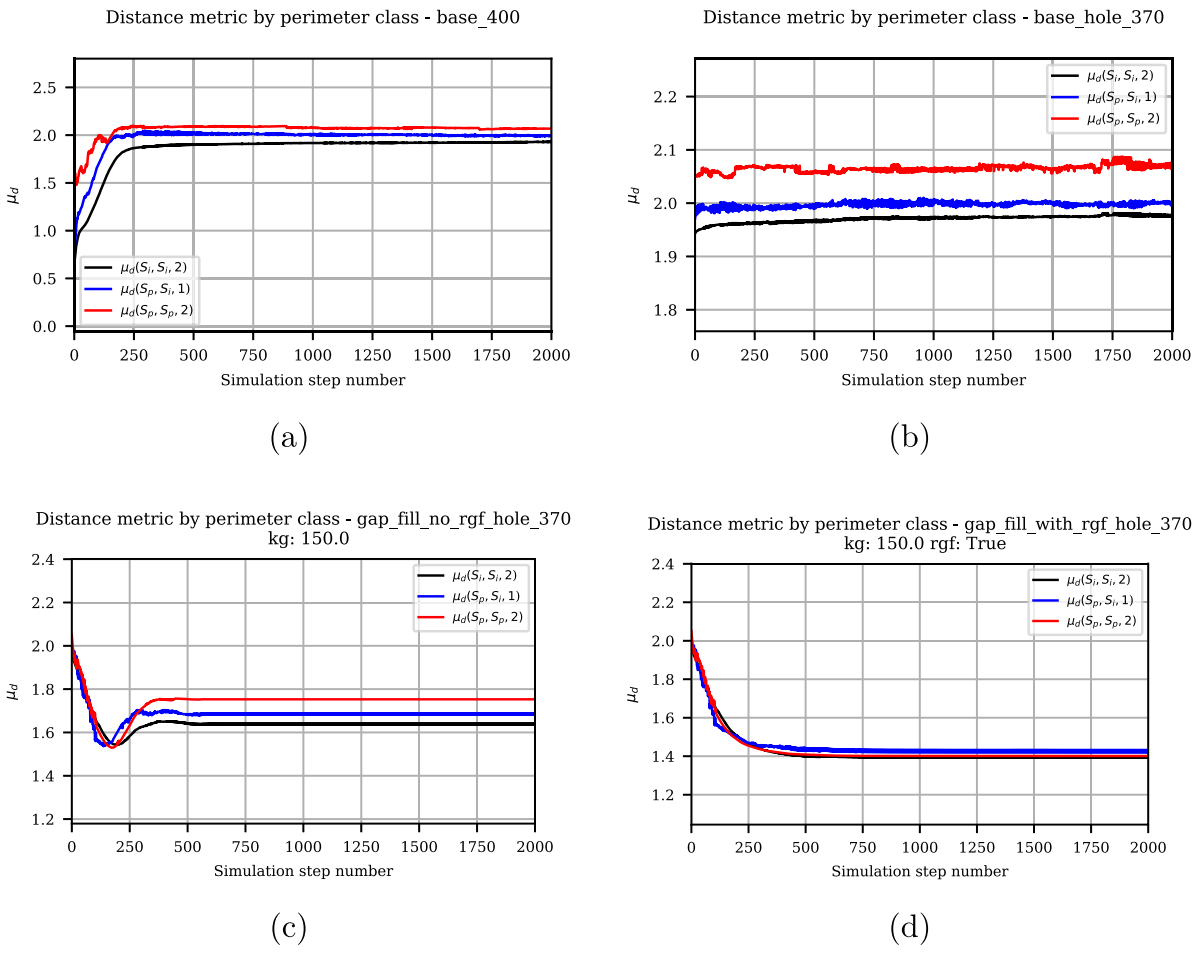


Fig. 8. Distance metrics: (a) Baseline swarm (b) Swarm with hole: base parameters (c) Swarm with hole: gap-filling without flattening of reflex angles (d) Swarm with hole: gap-filling with flattening of reflex angles.

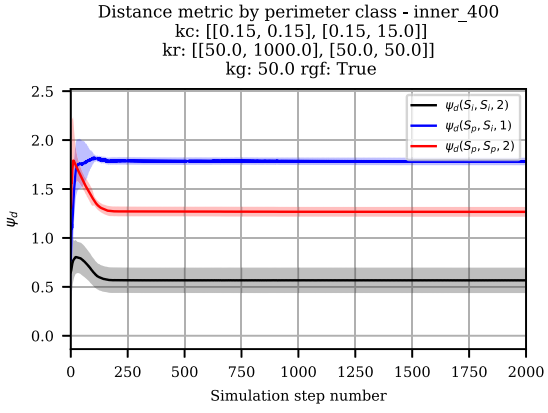


Fig. 9. Distance metric: Expanded stable perimeter.

mean distance between agent pairs in the S_{pp} class settles at about 1.27 units, giving a relative expansion factor for perimeter agents of about 2.27. The perimeter agents are established very quickly and remain stable throughout the run. This leads to very little variability in the mean distances for each perimeter class. The expansion factor can be controlled by varying the parameter values.

In contrast, Fig. 10(a) shows the run of a swarm from the same initial distribution of agents to the structure of Fig. 6(a). In this case, the perimeter is compressed relative to the internal core. The variability in the distance metric shows that the structure is volatile. In particular, the classification of agents into S_p or S_i changes from step to step, as a subset of agents oscillate between the perimeter and the internal core throughout the run. This volatility leads to a counter-intuitive distance metric graph. Fig. 6(a) shows a structure in which the perimeter agents are clearly closer together than the internal agents and there is a significant distance between the perimeter and the internal core. So we expect $\mu_d(S_p, S_p, k) \ll \mu_d(S_p, S_i, k)$ but the graph of Fig. 10(a) indicates that these values are roughly equal. This discrepancy is caused by the volatility of S_p over the course of the run. As agents move out of the perimeter, gaps appear between the remaining perimeter agents, increasing $\mu_d(S_p, S_p, k)$. At the same time, the agents that have moved out of the perimeter are now very close to a perimeter agent, decreasing $\mu_d(S_p, S_i, k)$, thus the two means are seen to converge. A glance at the outer ring of agents in Fig. 6(a) shows most agents coloured red, indicating that they are classed in S_p at step 2000, but a significant number of agents in the outer ring are coloured black, indicating that they are classed in S_i at step 2000. A dynamic view of the structure throughout the run shows that the subsets of agents, S_p and S_i, change at every step. However, it is possible to identify a core set of agents that are predominantly in S_p. In this case, there is a set of 131 out of 400 agents that are classed in S_p in more than 60% of the steps of the run. Only 4 of the agents outside this set are classed in S_p in more than 0.3% of steps and of these none is in S_p in more than 24% of steps. It is interesting to pre-compute this core set of 131 agents and treat them as the perimeter agents in calculating the distance metric. Fig. 10(b) shows the metric in this case. This graph accords much more closely with our observations about the relative distances of agent pairs in S_{ii}, S_{pi}, and S_{pp}. It can be seen here that the mean distance between

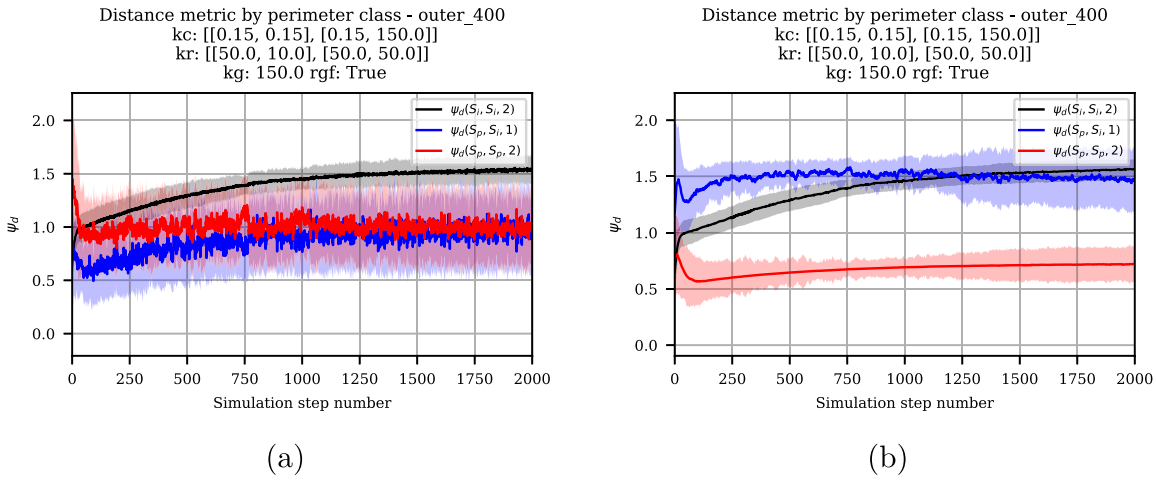


Fig. 10. Distance metrics: Compact volatile perimeter - (a) k-NN metric (b) k-NN metric with pre-computation of perimeter.

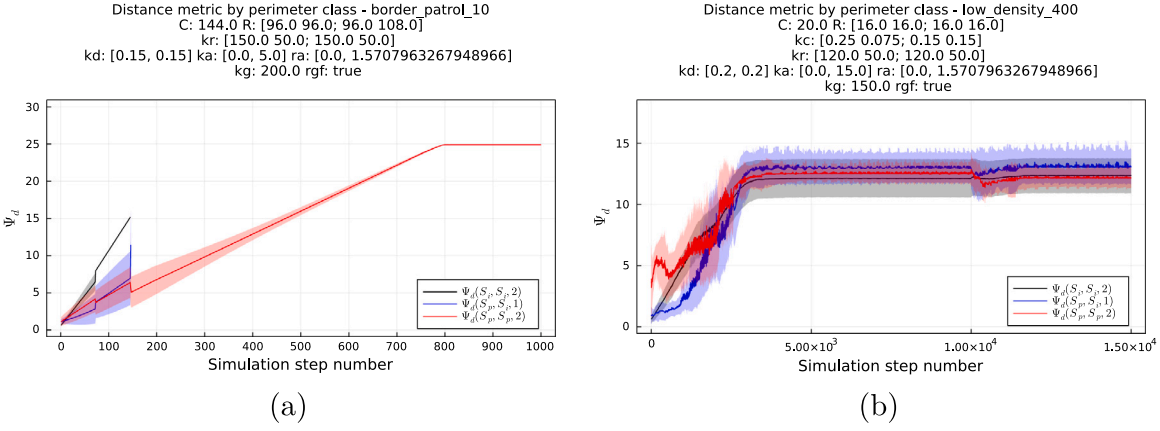


Fig. 11. Distance metrics: Rotating perimeter - (a) Goal-seeking, 'border patrol' (b) Low density with self-healing.

agent pairs in the S_{ii} class settles at about 1.56 units, whereas the mean distance between agent pairs in the S_{pp} class settles at about 0.72 units, giving a relative compression factor of 0.46. Again, the compression factor can be controlled by varying the parameter values.

Fig. 11(a) shows the distance metric for the first 1000 steps only of the border control example. It can be seen that within about 150 steps the swarm has expanded so that all agents are perimeter agents and the distance between the agents continues to expand until it reaches stability. The distance remains stable throughout the run, despite movement towards the goal and rotation.

Fig. 11(b) shows the distance metric for the evolution for 15000 steps of a low density swarm with a rotating perimeter, starting from the initial state of Fig. 4(a) (see Table 2 for parameters) and suffering agent failure at step 10,000, as illustrated in Fig. 7(b). The swarm begins to expand immediately – the minimum distance between agents never falls below the minimum distance seen in the initial state – and the disruption to the swarm caused by the agent failure at step 10,000 is seen to be only slight. It can be seen that the rotation of the perimeter agents does not cause instability in the mean distance between them.

6. Conclusions and future work

This paper proposes a new model for swarm evolution in the tradition of discrete-time, Boid-like, potential field models. The key ideas are simple but significant for the control over swarm structure that is made possible for large swarms of homogeneous agents, which can be implemented with very low computation and communication costs. The new model distinguishes four equivalence classes on the set of agent pairs, defined on the basis of the perimeter status of each agent. Array-valued parameters allow each equivalence class to have its own parameter values, leading to highly configurable swarm structures. An extension to our previous work on gap-filling adds a further level of control over swarm shape and perimeter. The use of array-valued parameters is shown to be applicable also to the case of direction and rotation vectors, allowing fine-grained control over the movement of perimeter and internal agents. Extensive experimental simulations demonstrate the effectiveness of the new model, exhibiting regular swarm structures with control over the perimeter and internal spacing of agents that could not be achieved with earlier models of this type.

Future work will include more detailed investigations to explore further the use of the new model in scenarios with obstacles and multiple goals. The tuning of model parameters remains a significant challenge: as Brambilla et al. (2013) observe, "The intuition of the human designer is still the main ingredient in the development of swarm robotics systems". So an important area for further work is the exploration of the use of machine learning techniques for automatically learning good parameter values for various objectives related to swarm structure. Finally, it is likely that the main contribution of this work will be found in real, practical applications and, therefore, the main focus of our future research will be on the use of our models for the control of swarms of physical robots and their application to real-world scenarios.

CRediT authorship contribution statement

Neil Eliot: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft. **David Kendall:** Conceptualization,

Methodology, Formal analysis, Investigation, Writing – original draft. **Michael Brockway:** Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft. **Paul Oman:** Formal analysis. **Ahmed Bouridane:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

A link to the data and code is provided on the first page of the paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eswa.2023.120183.

References

Barnes, L., Alvis, W., Fields, M., Valavanis, K., & Moreno, W. (2006a). Heterogeneous swarm formation control using bivariate normal functions to generate potential fields. In *Distributed intelligent systems: Collective intelligence and its applications, 2006. DIS 2006. IEEE workshop on* (pp. 85–94). IEEE.

Barnes, L., Alvis, W., Fields, M., Valavanis, K., & Moreno, W. (2006b). Swarm formation control with potential fields formed by bivariate normal functions. In *Control and automation, 2006. MED'06. 14th mediterranean conference on* (pp. 1–7). IEEE.

Barnes, L., Fields, M., & Valavanis, K. (2007). Unmanned ground vehicle swarm formation control using potential fields. In *Control & automation, 2007. MED'07. Mediterranean conference on* (pp. 1–8). IEEE.

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

Bruemmer, D. (2002). A robotic swarm for spill finding and perimeter formation. In *Spectrum 2002*. NV, Aug: Reno.

Cao, Y., Ren, W., & Egerstedt, M. (2012). Distributed containment control with multiple stationary or dynamic leaders in fixed and switching directed networks. *Automatica*, *48*(8), 1586–1597.

Dai, Y., Hinchey, M., Madhusoodan, M., Rash, J., & Zou, X. (2006). A prototype model for self-healing and self-reproduction in swarm robotics system. In *2006 2nd IEEE international symposium on dependable, autonomic and secure computing* (pp. 3–10).

Elamvazhuthi, K., & Berman, S. (2015). Optimal control of stochastic coverage strategies for robotic swarms. In *2015 IEEE international conference on robotics and automation* (pp. 1822–1829). IEEE.

Eliot, N. (2017). *Methods for the efficient deployment and coordination of swarm robotic systems* (Ph.D. thesis), United Kingdom: University of Northumbria at Newcastle.

Eliot, N., Kendall, D., & Brockway, M. (2018). A new metric for the analysis of swarms using potential fields. *IEEE Access*, *6*, 63258–63267.

Eliot, N., Kendall, D., Moon, A., Brockway, M., & Amos, M. (2019). Void reduction in self-healing swarms. In *Artificial life conference proceedings* (pp. 87–94). MIT Press.

Fang, H., Wei, Y., Chen, J., & Xin, B. (2017). Flocking of second-order multiagent systems with connectivity preservation based on algebraic connectivity estimation. *IEEE Transactions on Cybernetics*, *47*(4), 1067–1077.

Fedele, G., & D'Alfonso, L. (2017). A model for swarm formation with reference tracking. In *2017 IEEE 56th annual conference on decision and control* (pp. 381–386).

Fedele, G., & D'Alfonso, L. (2021). A coordinates mixing matrix-based model for swarm formation. *International Journal of Control*, *94*(3), 711–721.

Fedele, G., D'Alfonso, L., & Gazi, V. (2022). A generalized Gazi-Passino model with coordinate-coupling matrices for swarm formation with rotation behaviour. *IEEE Transactions on Control of Network Systems*, 1.

Gazi, V. (2005). Swarm aggregations using artificial potentials and sliding-mode control. *IEEE Transactions on Robotics*, *21*(6), 1208–1214.

Ghrist, R., Lipsky, D., Poduri, S., & Sukhatme, G. (2008). Surrounding nodes in coordinate-free networks. In *Algorithmic foundation of robotics VII* (pp. 409–424). Springer.

He, L., Bai, P., Liang, X., Zhang, J., & Wang, W. (2018). Feedback formation control of UAV swarm with multiple implicit leaders. *Aerospace Science and Technology*, *72*, 327–334.

Ismail, A. R., & Timmis, J. (2010). Towards self-healing swarm robotic systems inspired by granuloma formation. In *Engineering of complex computer systems (ICECCS), 2010 15th IEEE international conference on* (pp. 313–314). IEEE.

Ivić, S., Crnković, B., & Mezić, I. (2017). Ergodicity-based Cooperative Multiagent Area coverage via a potential field. *IEEE Transactions on Cybernetics*, *47*(8), 1983–1993.

Johnson, M., & Brown, D. (2016). Evolving and controlling perimeter, rendezvous, and foraging behaviours in a computation-free robot swarm. In *BICT'15: Proceedings of the 9th EAI international conference on bio-inspired information and communications technologies* (pp. 311–314). ACM.

Jung, S., & Goodrich, M. A. (2013). Multi-robot perimeter-shaping through mediator-based swarm control. In *2013 16th international conference on advanced robotics* (pp. 1–6).

Karthikeyan, S., & Ali, M. A. (2006). A general approach to swarm coordination using circle formation. In *Stigmergic optimization* (pp. 65–84). Springer.

Lee, G., & Chong, N. Y. (2008). Self-configurable mobile robot swarms with hole repair capability. In *Intelligent robots and systems, 2008. IROS 2008. IEEE/RSJ international conference on* (pp. 1403–1408).

Liang, X., Qu, X., Wang, N., Li, Y., & Zhang, R. (2019). Swarm control with collision avoidance for multiple underactuated surface vehicles. *Ocean Engineering*, *191*, Article 106516.

López-González, A., Meda Campaña, J., Hernández Martínez, E., & Paniagua Contro, P. (2020). Multi robot distance based formation using parallel genetic algorithm. *Applied Soft Computing*, *86*, Article 105929.

Marino, A., Parker, L. E., & Antonelli, G. (2013). A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling. *Journal of Intelligent and Robotic Systems*, *71*, 423–444.

McLurkin, J., & Demaine, E. D. (2009). A distributed boundary detection algorithm for multi-robot systems. In *Intelligent robots and systems, 2009. IROS 2009. IEEE/RSJ international conference on* (pp. 4791–4798). IEEE.

Pan, N., Zhang, M., Sun, Y., Chen, S., Liu, H., & Guo, X. (2021). Study on border patrol task planning of heterogeneous UAVs group based on swarm intelligence. *Science Progress*, *104*(3_suppl).

Räz, T. (2013). On the application of the honeycomb conjecture to the Bee's honeycomb. *Philosophia Mathematica*, nkt022.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics, vol. 21* (pp. 25–34). ACM.

Roach, J. H., Marks, R. J., & Thompson, B. B. (2015). Recovery from sensor failure in an evolving multiobjective swarm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(1), 170–174.

Schneider, F. E., & Wildermuth, D. (2003). A potential field based approach to multi robot formation navigation. In *Robotics, intelligent systems and signal processing, 2003. proceedings. 2003 IEEE international conference on, vol. 1* (pp. 680–685).

Son, J., Ahn, H., & Cha, J. (2017). Lennard-jones potential field-based swarm systems for aggregation and obstacle avoidance. In *2017 17th international conference on control, automation and systems* (pp. 1068–1072).

Speck, C., & Bucci, D. J. (2018). Distributed UAV swarm formation control via object-focused, multi-objective SARSA. In *2018 Annual American control conference* (pp. 6596–6601).

Timmis, J., Ismail, A., Bjerknes, J., & Winfield, A. (2016). An immune-inspired swarm aggregation algorithm for self-healing swarm robotic systems. *Biosystems*, *146*, 60–76, Information Processing in Cells and Tissues.

Vassev, E., & Hinchey, M. (2009). ASSL specification and code generation of self-healing behavior for NASA swarm-based systems. In *2009 Sixth IEEE conference and workshops on engineering of autonomic and autonomous systems* (pp. 77–86).