Void Reduction in Self-Healing Swarms

Neil Eliot[1], David Kendall[1], Alun Moon[1], Michael Brockway[1] and Martyn Amos[1]

[1] Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, NE1 8ST, UK
neil.eliot@northumbria.ac.uk

Abstract

Swarms consist of many agents that interact according to a simple set of rules, giving rise to emergent global behaviours. In this paper, we consider swarms of mobile robots or drones. Swarms can be tolerant of faults that may occur for many reasons, such as resource exhaustion, component failure, or disruption from an external event. The loss of agents reduces the size of a swarm, and may create an irregular structure in the swarm topology. A swarm's structure can also be irregular due to initial conditions, or the existence of an obstacle. These changes in the structure or size of a swarm do not stop it from functioning, but may adversely affect its efficiency or effectiveness. In this paper, we describe a self-healing mechanism to counter the effect of agent loss or structural irregularity. This method is based on the reduction of concave regions at swarm perimeter regions. Importantly, this method requires no expensive communication infrastructure, relying only on agent proximity information. We illustrate the application of our method to the problem of surrounding an oil slick, and show that void reduction is necessary for full and close containment, before concluding with a brief discussion of its potential uses in other domains.

Introduction

The natural phenomenon of *swarming* in organisms such as insects, fish and birds has, for a long time, served as inspiration for algorithmic solutions to problems (Blum and Merkle, 2008). Swarm-based algorithms use a number of *agents* which behave according to local rules (locality often being defined in terms of spatial proximity), but which - collectively - are capable of synergistically cooperative behaviour. Problems to which such methods have been applied include path finding (Hou et al., 2009), distribution across a space (Ekanayake and Pathirana, 2010; Gazi and Passino, 2002, 2004a), or foraging as a colony (Gurfil and Kivelevitch, 2007; Hereford, 2011). In order to model inter-agent interactions, many algorithms use *field effects*, which capture *attractive* and *repulsive* forces between agents (Andreou et al., 2009; Barnes et al., 2006a,b; Bennet and McInnes, 2009; Gazi and Passino, 2002, 2004b, 2005, 2011; Mohan and Ponnambalam, 2009). Attraction is used as a *cohesive* force to bring agents close together, and repulsion is used to prevent collisions.

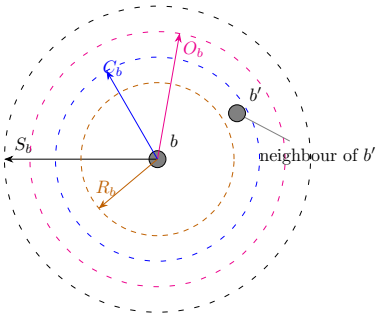

Figure 1: Agent field ranges. R_b implements repulsion, C_b implements cohesion, S_b is the agent's sensing range, and O_b is used to manage collisions with obstacles.

Forces are generally defined in terms of *ranges* around an agent, and the field effects are derived as vectors from these ranges (Figure 1). For any agent, b, all ranges must fall within the *sensing capability* of the agent, S_b, which might represent a visual or auditory range, some chemical sensing capability, or (in the context of mobile robotics) a communication range. It is usual for the cohesion field, C_b, to have a radius which is larger than the repulsion radius, R_b (so that agents are encouraged to group together, but not too closely). When another agent, b', moves into the cohesion range of b then b' becomes a *neighbour* of b; when b' moves into the *repulsion* field of b, then b is also subject to repulsion. When the repulsion magnitude exceeds the cohesion magnitude, then b has a tendency to move away from b', i.e., it is repelled. When b moves too close to an obstacle, i.e., an obstacle is within the obstacle repulsion range, O_b, the repulsion vector is applied and the agent tends to move away from the obstacle.

When cohesion and repulsion are the only field effects used to create a swarming effect, the number of stable structures that can develop is limited (Eliot, 2017). These structures effectively take the form of either straight edges or partial lattices (Figure 2). The maintenance of a well-structured swarm is crucial to their effective deployment in a number of applications, including reconnaissance or artificial polli-

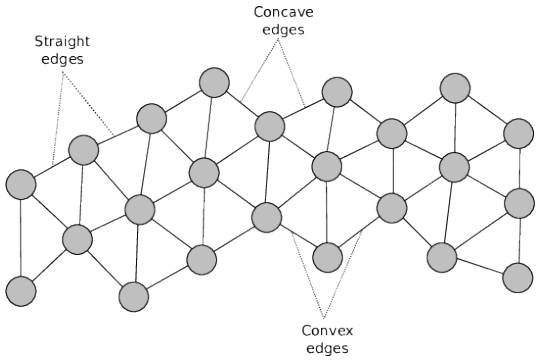

Figure 2: Stable swarm structure containing two types of anomaly.

nation, where coverage "blind spots" are eliminated (Elamvazhuthi and Berman, 2015), and containment, where the swarm is used to surround an object or region (Cao et al., 2012). Over time, the perimeters of partial lattices may contain so-called *anomalies*, such as concave "dents" or convex "peaks", and these anomalies all contribute to the disruption of an otherwise well-structured swarm. The key, therefore, is to ensure that concave *voids* are dynamically removed from a swarm.

Here, we describe our *void reduction* technique for swarm management, which is a form of self-healing that encourages a swarm to coalesce into a more geometrically stable shape. This is achieved by removing voids and concave edges. Importantly, the techniques defined in this paper function without the need for inter-agent or global *messaging* (which can carry a significant overhead), and rely only on local *proximity detection*.

The rest of the paper is organized as follows: we first briefly review related work in the area of self-healing swarms, and then describe the baseline swarming model and our novel perimeter detection and void reduction mechanisms.We describe the results of computational studies in a specific application domain (surrounding an oil slick), before we conclude with a brief discussion of our results, and give pointers to possible future work.

Related Work

A prototype framework for self-healing swarms was developed by Dai, *et al.*, which considered the problem of agent failure in hostile environments (Dai et al., 2006). This was similar to work carried out by Vassev and Hinchey, who modelled swarm deployment using the ASSL (Autonomic System Specification Language) (Vassev and Hinchey, 2009). This technique was used by NASA (US National Aeronautics and Space Administration) when developing their ANTS (Autonomous Nano Technology Swarm) for use in asteroid belt exploration. However, this work was focused more towards the failure of an agent's internal systems, rather than on the removal of anomalies in a swarm distribution.

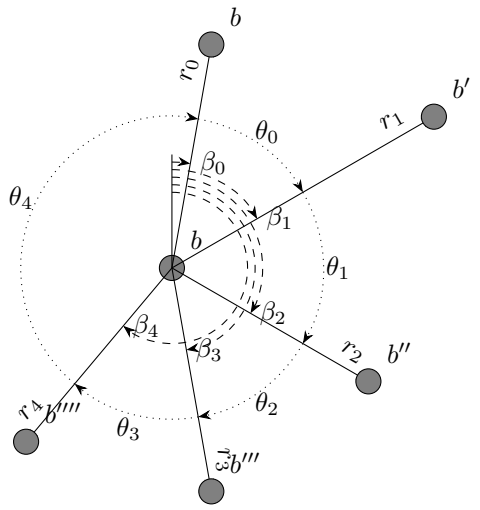

Figure 3: Swarm model: representation of interaction with neighbouring agents.

In the context of swarm structure maintenance, Roach, *et al.* focussed on the effects of sensor failure, and the impact that this 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 and Chong, 2008), and 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 and Timmis, 2010). They have also considered the effect that failed agents can have on a swarm when traversing a terrain (Timmis et al., 2016).

Our void reduction technique is an extension of the work presented in (Ismail and Timmis, 2010; Timmis et al., 2016), and also builds on the work of Lee and Chong on concave edge identification (Lee and Chong, 2008), and on the work of McLurkin and Demaine on the detection of perimeter types (McLurkin and Demaine, 2009). However, the technique employed in this paper does not explicitly require the identification of the perimeter type, as this would require a communication infrastructure.

Swarm Model

In this Section, we define the baseline swarm model. A swarm, $\mathcal{S}$, comprises a number of agents; in our application context, each agent is a mobile robot or drone, but this may remain unspecified. An agent $b \in \mathcal{S}$ has a sensor range, S_b, within which it may detect other agents in the swarm, and determine both their *range*, r, and *bearing*, β (Figure 3). At each time step, the agent generates a set of neighbours, $\mathcal{N}_b$, comprising other agents that are within a specific range

(usually defined as the range of the cohesion field, C_b), as given in Equation 1. These range and bearing pairs contain the relative position vector for each neighbour, b', with respect to the sensor reference frame of agent b. This model was defined by Eliot, *et. al.* in a paper which introduced a new magnitude-based metric for the analysis of swarms (Eliot et al., 2018).

$$\mathcal{N}_b = \{(r, \beta) \ldots\} \tag{1}$$

In order to calculate the new vector, v, for b, Equation 2 defines a weighted model that includes cohesion, repulsion, direction and obstacle avoidance ($v_c(b), v_r(b), v_d(b)$, and $v_o(b)$, respectively). The weightings k_c, k_r, k_d, k_o allow each component to be scaled in order to tailor the swarming effect.

$$v(b) = k_c v_c(b) + k_r v_r(b) + k_d v_d(b) + k_o v_o(b) \tag{2}$$

Repulsion, $v_r(b)$, defined in Equation 3, is the directional movement required to prevent agents colliding. $\mathcal{R}_b$ is defined as the set of agents that are within the repulsion range of b.

$$v_r(b) = \frac{1}{|\mathcal{R}_b|} \left(\sum_{b' \in \mathcal{R}_b} \left(1 - \frac{|b'|}{R_b} \right) b' \right) \tag{3}$$

Cohesion, $v_c(b)$, defined in Equation 4, calculates the movement required to make an agent move towards other agents in order to form a cohesive structure. $\mathcal{C}_b$ is defined as the set of agents that are within the cohesion range of b.

$$v_c(b) = \frac{-1}{|\mathcal{C}_b|} \left(\sum_{b' \in \mathcal{C}_b} b' \right) \tag{4}$$

Direction, $v_d(b)$, defined in Equation 5, generates a directional vector for an agent to move towards some destination, d.

$$v_d(b) = d \tag{5}$$

Obstacles, like agents, may be represented as a point. As an agent moves, it may enter an obstacle's *repulsion field*. If this occurs, then the agent should move away (as we assume that an obstacle is unable to take evasive action itself). Here, agents have a fixed *obstacle repulsion field*, O_b. If an obstacle enters the field, a vector of magnitude O_b is applied. If more than one obstacle is present within the field, the applied repulsion vector is the sum of the repulsion vectors (Figure 4). The resultant vector is normalised and scaled such that the magnitude is the same as the field distance, O_b, as given in Equation 6.

Equation 6 shows the repulsion vector, $v_o(b)$, for an agent. $\mathcal{O}_b$ is the set of obstacles within the range of agent

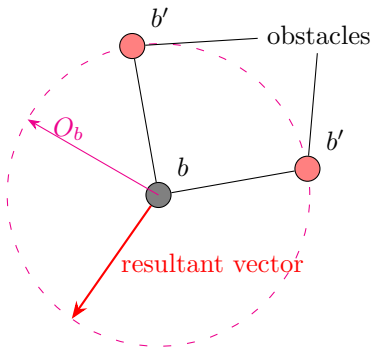

Figure 4: Repulsion from obstacles.

b. The obstacles are identified by comparing their Cartesian distance to the fixed obstacle repulsion field O_b, so $\forall o \in \mathcal{O}_b : |o| \leq O_b$. The applied repulsion is calculated by scaling the normalised sum of the normalised vectors $\hat{o}$ by O_b. Note that $\hat{}$ is the equivalent of $\hat{v} = \frac{v}{|v|}$, the normalised vector.

$$\begin{aligned} v_o(b) &= O_b \hat{q}_o \\ \text{where } q_o &= \sum_{o \in \mathcal{O}_b} \hat{o} \\ v_o(b) &= O_b \left(\sum_{o \in \mathcal{O}_b} \hat{o} \right)^{\wedge} \end{aligned} \tag{6}$$

An agent's *movement vector* is defined as the sum of all the component vectors, as shown in Equation 2 (similar to that used by Hashimoto, *et. al.* (Hashimoto et al., 2008)). In order for a vector to be used for movement, it must be normalised before the agent's speed, s_b, can be applied. The resulting movement vector, m_b, is defined in Equation 7, and is calculated using unit time, speed and the normalised *movement vector*.

$$m_b = s_b \hat{v}(b) t \tag{7}$$

Over time, applying the calculations described in this Section to all agents in turn creates the global swarming effect. This provides the *baseline* algorithm for swarm movement. We now describe how the swarm may be *dynamically reconfigured*, which is the main novel contribution of this paper. After describing our new algorithm for void reduction, we show how it may be applied to a specific problem.

Perimeter Detection

In order to dynamically restructure a disorganised swarm, we must first identify the *perimeter* agents. This is due to the fact that anomalies occur at swarm boundary locations. With reference to Figure 5, these agents may form part of an outer (green) or inner (red) edge.

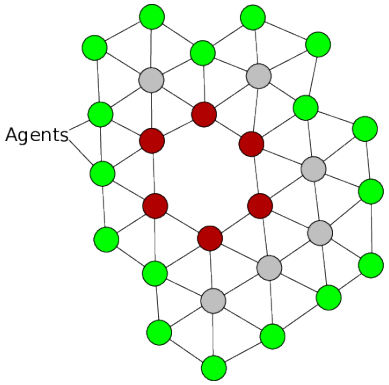

Figure 5: Outer and inner swarm perimeters.

Our detection mechanism detects both the outer edge of a swarm and any internal features (voids) that satisfy the same set of conditions (Figure 5). It is therefore possible to have both voids and "islands" of agents within the same swarm. Voids are best defined as perimeters that are both concave (Equation 15) in nature and which exist inside another perimeter. McLurkin (McLurkin and Demaine, 2009) describes two types of perimeters, convex and concave, where a convex perimeter is an edge where the average angle of the exposed faces of relevant agents is $> 180°$, and a concave perimeter is one where the average exposed angle is $< 180°$.

The set of neighbours, $\mathcal{N}_b$ (Equation 1) is sorted into the sequence $\mathcal{P}_a$, in ascending order of bearing:

$$\mathcal{P}_a = \langle (r_0, \beta_0), \ldots, (r_n, \beta_n) \rangle \tag{8}$$

such that $\beta_0 < \beta_1 < \ldots < \beta_n$

This set of agents forms the perimeter of an enclosing polygon of agent b. Each consecutive pair of agents in the sequence defines an *edge*, which has length d and an angle θ given by the difference in bearings of successive neighbours. The sequence of edges that forms this polygon is:

$$\mathcal{P}_e = \langle (d_0, \theta_0), \ldots, (d_n, \theta_n) \rangle \tag{9}$$

where

$$\theta_i = \beta_{i+1} - \beta_i \tag{10}$$

The index addition is modulo $|\mathcal{N}_b|$, making β_0 the successor bearing to β_n ($n + 1 = 0$). The angles θ must lie in the range $0 < \theta \leq 2\pi$. This restriction on the values of θ enforce the condition that

$$\sum \theta_i = 2\pi \tag{11}$$

The length of a perimeter edge is given by the cosine rule

$$d_i^2 = r_{i+1}^2 + r_i^2 - 2 r_{i+1} r_i \cos\theta_i \tag{12}$$

An agent is therefore on the perimeter of the swarm if it is not enclosed by the polygon defined in $\mathcal{P}_e$. Simple geometry shows that this is the case, given by the predicate in Equation 13.

$$\exists \theta_i \in \mathcal{P}_e : \theta_i \geq \pi \tag{13}$$

The polygon is considered to be "open" if two successive agents on the perimeter are unable to "see" one another; that is, their separation, d, is greater than the range of the attractive field. An open polygon does not enclose the agent b, so it is considered to be on the perimeter.

Formally, an agent, b, is on the perimeter of the swarm if the predicate in Equation 14 is true.

$$\exists d_i \in \mathcal{P}_e : d_i > C_b \vee \exists \theta_i \in \mathcal{P}_e : \theta_i \geq \pi \tag{14}$$

An agent is at the apex of a concave region of the perimeter if

$$\exists (\theta_i, d_i) \in \mathcal{P}_e : d_i > C_b \wedge \theta_i < \pi \tag{15}$$

The orientation is independent in so much as: if the agent b is rotated through an angle of γ then the bearings are rotated by $-\gamma$,

$$\beta_i \mapsto \beta_i - \gamma$$

The angle between successive agents is now

$$\theta_i = (\beta_{i+1} - \gamma) - (\beta_i - \gamma) = \beta_{i+1} - \beta_i - \gamma + \gamma = \beta_{i+1} - \beta_i$$

Void Reduction

In a static swarm, where there are essentially no *destination vectors*, void reduction will result in a restructuring motion that creates a more "rounded' swarm. Void reduction also creates a *surrounding* effect, as it removes voids from a swarm. This is discussed in more detail in the next Section. Although these effects improve the potential applications of swarms, negative effects may also be introduced (e.g., in some circumstances void reduction can create an artificial *destination vector*, in that the swarm will appear to have a directional movement).

In order to implement void reduction, full *perimeter detection* is required in order to identify candidate agents (Eliot, 2017). Void reduction does not require the perimeter *type* to be identified, and no communications infrastructure is required. Many existing swarm coordination algorithms require inter-agent communication (Jung and Goodrich, 2013; McLurkin and Demaine, 2009; Saldana et al., 2012; Navarro and Matía, 2009; Zhang et al., 2013), and this imposes a significant limitation on swarm size, due to the requirement for message propagation. Our method avoids the problems associated with this.

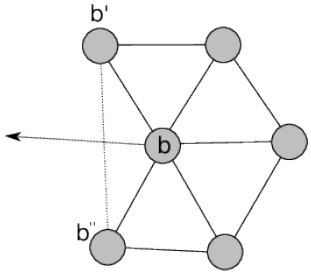

Figure 6: Agent void reduction motion: agents b' and b'' form a concave edge (depicted by the lighter line). Agent b must therefore move to remove this edge.

Void Reduction: Agent Movement

The addition of a further characteristic to the motion of a swarm means that we must augment the existing agent model (Equation 2). With void reduction, this revision is based on the identification of the agents that are connected by a concave edge, as shown in Figure 6 as $(b^{'}, b, b^{''})$.

When an agent is identified as being a component of a void characteristic (Equation 15), the normal *movement-direction vector* is replaced by a *void reduction vector*. This new vector causes the agent to move in a direction that will reduce or remove a concave edge, by moving the agent towards the identified gap. This either straightens an outer perimeter, or reduces/removes a void. The change in direction also affects the distance and magnitude variances. Figure 7 shows this effect in more detail; the top figure shows the initial positions of the agents before void reduction is applied , and the bottom part of the figure shows the effects on its relationship with its neighbours. The aggregate change is an increase in the inter-agent distances, and an increase in the resultant magnitude effects.

As part of the perimeter detection process, we may generate a set of agents, G_b, that produce a gap for a particular agent, b (that is, the first two agents identified as creating a "gap" in agent b's neighbours). Equation 16 is then used to calculate the centroid of the "gap" agents:

$$D_{pos}(b) = \frac{1}{2} \sum_{b' \in G_b} b' \tag{16}$$

The centroid $D_{pos}(b)$ is then used to calculate the *void reduction vector*:

$$D(b) = D_{pos}(b)b \tag{17}$$

$D(b)$ is the vector from the coordinates of agent b to the centroid coordinates, $D(b)$. This new vector is used as the void reduction vector in order to implement the necessary void reduction movement (Equation 17).

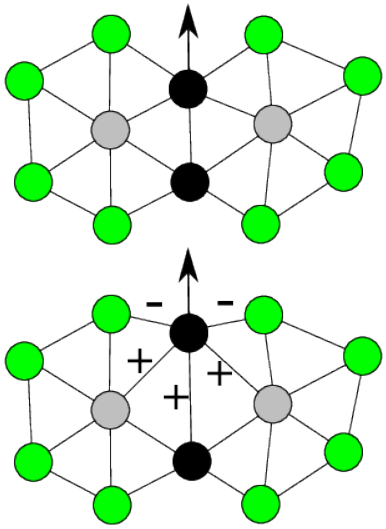

Figure 7: Initial position (top), and reduced position (bottom). +/- labels show relative changes in inter-agent magnitude.

In addition to agent proximity, the void reduction movement process must also include *obstacle avoidance* (Equation 18). As with the earlier vector-based calculations, a weighting, k_{cr}, is applied to the void reduction vector in order to allow the model to adjust the application of the effect. The resultant void reduction vector is normalised to produce a *directional vector*, as shown in Equation 18. This is then applied to the agent in order to effect movement.

$$V(b) = (k_{cr}D(b) + k_o v_o(b))\hat{} \tag{18}$$

Experimental Results: Oilslick Containment

In this Section we give the results of experiments to simulate a specific scenario; that of *oilslick containment* using a mobile robot swarm. Oil spills (from ships or drilling operations) can cause significant environmental, social and economic damage, and removing them can be hazardous and expensive. Several alternatives to traditional spill dispersal/containment procedures have been proposed, with some proposals relying on the use of robot swarms to surround a spill (Fritsch et al., 2007; Kakalis and Ventikos, 2008; Zhang et al., 2013) (details of remediation processes are outside the scope of this paper, but they may include skimming of the surface, deposition of a dispersal agent, or oil containment using a boom). However, these proposals all require the use of a communications infrastructure to facilitate message passing between agents. Our proposed method has the *significant benefit* of not requiring any such mechanism, relying only on local proximity detection.

The scenario is schematically depicted in Figure 8; we have an oil slick in some environment, and a swarm of robots

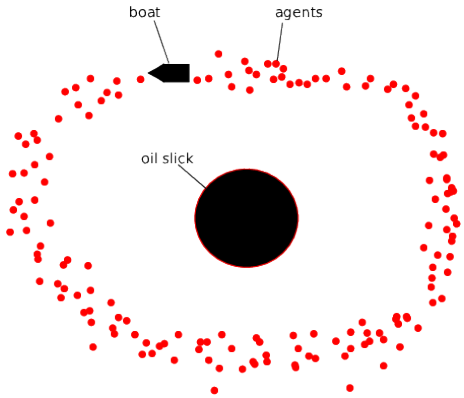

Figure 8: Oil slick containment scenario.

that are deployed by boat around the perimeter of the slick. Figure 9 shows the results of simulating the containment process using both the baseline movement algorithm (top) and the baseline method with void reduction (bottom). In our simulation, we use 200 agents, which is significantly greater than the number of agents than are generally simulated when inter-agent communication is required.

Without void reduction (i.e., simply using the field-effect-based movement algorithm) the swarm expands and then stabilises into a structure containing a void. The swarm "vibrates" slightly as cohesion and repulsion forces fluctuate to maintain the swarm's structure, but the void does not fully close, and full and "tight" containment is not achieved. The agents that do come in contact with the obstacle are repelled by the obstacle repulsion field. If, however, we activate void reduction, then the swarm expands as expected, due to the field effects, but then the void is completely removed, achieving full and close containment of the slick.

Figures 10 and 11 show the effect of the void reduction on the distribution of agents compared to the baseline method. Figure 10 shows the distance distribution of the swarm for both the baseline method (grey/black) and the void reduction method (red). The baseline swarm initially expands, then settles after approximately 6 seconds (this is also the case for the void reduced swarm). Following the initial expansion, the baseline swarm remains relatively slow-changing with respect to distance and magnitude. However, the void reduced swarm is affected more significantly; after approximately 10 seconds the swarm's internal void perimeter makes contact with the oil spillage (obstacle). This has the effect of disrupting the average distance and average *inter-agent magnitudes*. This effect diminishes slightly after approximately 18 seconds, when the swarm's *void reduction vectors* cause the swarm to surround the spillage. The slick surrounding process is followed by a few remaining changes caused by the "snapping" of agents at the spillage perimeter, and then the containment process is complete.

Figure 11 compares *inter-agent magnitudes* for the baseline and void reduction swarms. When initially deployed, the swarm is so dense that the average *inter-agent magnitude* is negative, indicating a high level of expansion. Within 2 seconds the expansion has reached a point where the average magnitude is positive, indicating the swarm is cohesive. This means that the swarm will remain as a single entity, and therefore be capable of surrounding an object without breaking apart.

When the swarm shrinks to surround the obstacle, we see an erratic change in the number of perimeter agents. Figure 12 shows the number of perimeter agents over the duration of the simulation. We see that the baseline swarm perimeter size decreases steadily and then settles (the swarm has not enclosed the spillage). The perimeter count has settled, but, as shown in Figures 10 and 11, the agents are still moving (magnitude variance and magnitude >0); however, the movement does not affect the overall structure.

When the void reduction swarm encounters the obstacle at approximately 10s there is a change due to "snapping", as the agents "fold" around the obstacle. Snapping is an oscillation of relations between four agents (Eliot, 2017). The perimeter size then continues to fall gradually as the void percolates out of the system. The perimeter size then stabilises as the slick obstacle is fully surrounded.

Conclusion

In this paper, we have shown how the structure of a simulated swarm of robots may be controlled by the identification and removal of perimeter anomalies. Importantly, the identification of anomalies is achieved locally by individual agents using only proximity detection, without any need for an inter-agent communication structure. This could offer significant benefits in terms of cost, simplicity, and fault-tolerance. The technique works with arbitrary-sized swarms; here we use 200 agents, but we have successfully simulated swarms of up to 500 agents with no appreciable performance degradation.

This work demonstrates one possible application of our void reduction technique. Future work will focus on its use with mobile swarms (e.g., for reconnaissance) which must navigate past/around a number of obstacles whilst maintaining a coherent and compact structure.

References

Andreou, P., Zeinalipour-Yazti, D., Andreou, M., Chrysanthis, P. K., and Samaras, G. (2009). Perimeter-based data replication in mobile sensor networks. In *Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 244–251. IEEE.

Barnes, L., Alvis, W., Fields, M., Valavanis, K., and Moreno, W. (2006a). Heterogeneous swarm formation control using bivariate normal functions to generate potential fields. In *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, pages 85–94. IEEE.

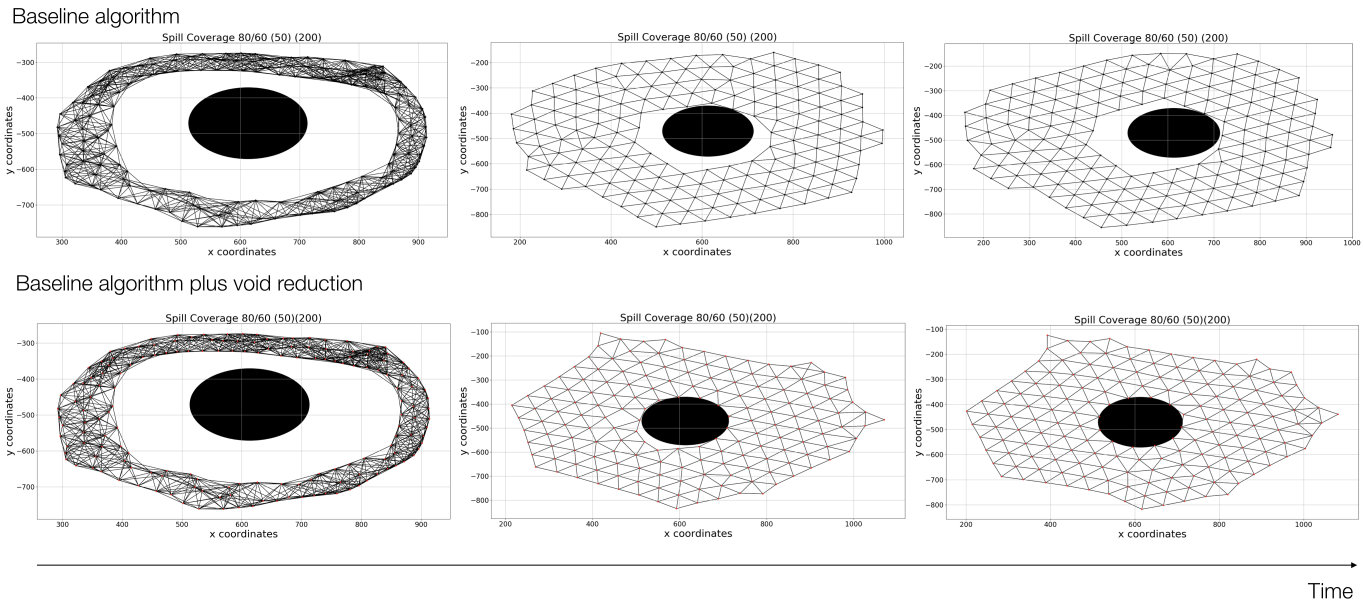

Figure 9: Simulation results for oil slick containment scenario. Top three frames show the evolution of the swarm using only the baseline movement algorithm; bottom three frames show the impact of adding our void reduction method.

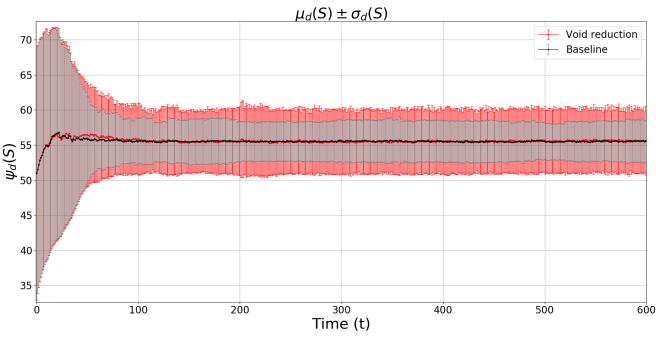

Figure 10: Oil spill containment distance (time shown in 10 millisecond slices).

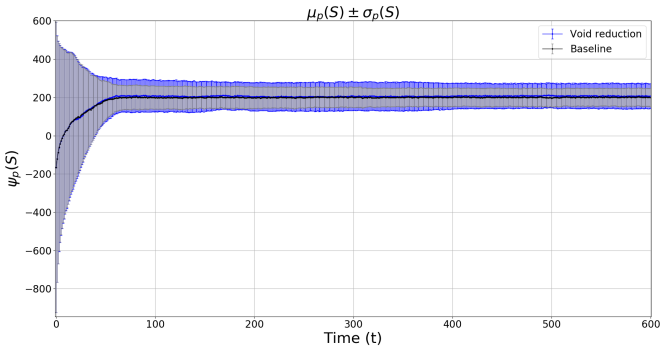

Figure 11: Oil spill containment magnitude (time shown in 10 millisecond slices).

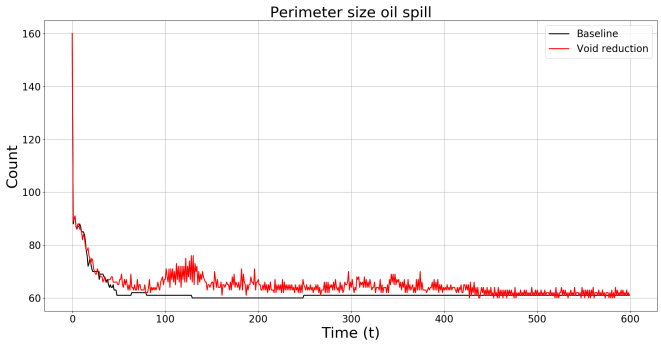

Figure 12: Swarm perimeter size comparison (time shown in 10 millisecond slices).

Barnes, L., Alvis, W., Fields, M., Valavanis, K., and Moreno, W. (2006b). Swarm formation control with potential fields formed by bivariate normal functions. In *14th Mediterranean Conference on Control and Automation*, pages 1–7. IEEE.

Bennet, D. and McInnes, C. (2009). Verifiable control of a swarm of unmanned aerial vehicles. *Journal of Aerospace Engineering*, 223(7):939–953.

Blum, C. and Merkle, D. (2008). *Swarm Intelligence: Introduction and Applications*. Springer.

Cao, Y., Ren, W., and 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. S., Hinchey, M., Madhusoodan, M., Rash, J. L., and Zou, X. (2006). A prototype model for self-healing and self-reproduction in swarm robotics system. In *2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 3–10.

Ekanayake, S. W. and Pathirana, P. N. (2010). Formations of robotic swarm: an artificial force based approach. *International Journal of Advanced Robotic Systems*, 7(3):173–190.

Elamvazhuthi, K. and Berman, S. (2015). Optimal control of stochastic coverage strategies for robotic swarms. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1822–1829. IEEE.

Eliot, N. (2017). *Methods for the Efficient Deployment and Coordination of Swarm Robotic Systems*. PhD thesis, Northumbria University, UK. Available at `http://nrl.northumbria.ac.uk/32575/`.

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

Fritsch, D., Wegener, K., and Schraft, R. D. (2007). Control of a robotic swarm for the elimination of marine oil pollutions. In *IEEE Swarm Intelligence Symposium*, pages 29–36. IEEE.

Gazi, V. and Passino, K. M. (2002). Stability analysis of swarms in an environment with an attractant/repellent profile. In *Proceedings of the American Control Conference*, volume 3, pages 1819–1824. IEEE.

Gazi, V. and Passino, K. M. (2004a). A class of attractions/repulsion functions for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579.

Gazi, V. and Passino, K. M. (2004b). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):539–557.

Gazi, V. and Passino, K. M. (2005). Stability of a one-dimensional discrete-time asynchronous swarm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(4):834–841.

Gazi, V. and Passino, K. M. (2011). *Swarm Stability and Optimization*. Springer Science & Business Media.

Gurfil, P. and Kivelevitch, E. (2007). Flock properties effect on task assignment and formation flying of cooperating unmanned aerial vehicles. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 221(3):401–416.

Hashimoto, H., Aso, S., Yokota, S., Sasaki, A., Ohya, Y., and Kobayashi, H. (2008). Stability of swarm robot based on local forces of local swarms. In *SICE Annual Conference, 2008*, pages 1254–1257. IEEE.

Hereford, J. (2011). Analysis of beeclust swarm algorithm. In *IEEE Symposium on Swarm Intelligence*, pages 1–7.

Hou, S. P., Cheah, C. C., and Slotine, J. J. E. (2009). Dynamic region following formation control for a swarm of robots. In *IEEE International Conference on Robotics and Automation*, pages 1929–1934.

Ismail, A. R. and Timmis, J. (2010). Towards self-healing swarm robotic systems inspired by granuloma formation. In *IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 313–314. IEEE.

Jung, S.-Y. and Goodrich, M. A. (2013). Multi-robot perimeter-shaping through mediator-based swarm control. In *International Conference on Advanced Robotics (ICAR)*, pages 1–6. IEEE.

Kakalis, N. M. and Ventikos, Y. (2008). Robotic swarm concept for efficient oil spill confrontation. *Journal of Hazardous Materials*, 154(1-3):880–887.

Lee, G. and Chong, N. Y. (2008). Self-configurable mobile robot swarms with hole repair capability. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1403–1408.

McLurkin, J. and Demaine, E. D. (2009). A distributed boundary detection algorithm for multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4798. IEEE.

Mohan, Y. and Ponnambalam, S. (2009). An extensive review of research in swarm robotics. In *World Congress on Nature and Biologically Inspired Computing*, pages 140–145. IEEE.

Navarro, I. and Matía, F. (2009). A proposal of a set of metrics for collective movement of robots. In *Proc. Workshop on Good Experimental Methodology in Robotics*.

Roach, J. H., Marks, R. J., and 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.

Saldana, D., Ovalle, D., and Montoya, A. (2012). Improved algorithm for perimeter tracking in robotic sensor networks. In *Conferencia Latinoamericana En Informatica (CLEI)*, pages 1–7.

Timmis, J., Ismail, A., Bjerknes, J., and 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. and Hinchey, M. (2009). ASSL specification and code generation of self-healing behavior for nasa swarm-based systems. In *IEEE Conference and Workshops on Engineering of Autonomic and Autonomous Systems*, pages 77–86.

Zhang, G., Fricke, G. K., and Garg, D. P. (2013). Spill detection and perimeter surveillance via distributed swarming agents. *IEEE/ASME Transactions on Mechatronics*, 18(1):121–129.