Perimeter Compression in self-healing swarms.

Neil Eliot[1,*], David Kendall[2], and Michael Brockway[2]

[1]*Northumbria University, Faculty of Engineering and Environment, Department of Computer and Information Sciences*
[2]*Hexham University, Faculty of Computer Science*
[*]*Corresponding author: Dr Neil Eliot, neil.eliot@northumbria.ac.uk*

***Abstract*—Perimeter Compression is a technique where by a void reducing effect can be added to a basic swarming algorithm. The effect is dependant upon perimeter identification and is controlled by applying two weighting factors to the existing swarming formulae. One to the cohesion calculation and the other to the repulsion calculation.**

I. Introduction

When cohesion and repulsion field effects are used to create a swarming effect, the stable structures that develop are limited to either straight edges or partial lattices [8]. The maintenance of a well-structured swarm is crucial to effective deployment, including reconnaissance or artificial pollination, where 'blind spots' are best eliminated [7], and containment, where the swarm is used to surround an object or region [5]. Over time swarms form regular shapes [17] and perimeters form of partial lattices may contain so-called *anomalies*, such as concave 'dents' or convex 'peaks'. These anomalies contribute to the disruption of an otherwise well-structured swarm. The key, therefore, is to ensure that these *anomalies* are dynamically removed from a swarm.

Perimeter compression is a technique that creates a 'pull' effect between perimeter agents. It is dependant upon perimeter agent identification as discussed by Eliot et. al. in [8], [9], [10] and shown in Section IV.

The aim of the algorithm is to reduce the spacing between perimeter-based agents by reducing the repulsion field and increasing the cohesion effect on perimeter agents. Figure 1) shows an agent and it fields. S_b is the sensor field. O_b is the obstacle field. C_b is the cohesion field and R_b is the repulsion field. The implementation involves introducing two controlling weights; p_c (Perimeter Compression Cohesion) which increases the cohesion vector ($k_c \rightarrow p_c k_c$) and p_r (Perimeter Compression Repulsion) which reduces the size of the repulsion field ($k_r \rightarrow p_r k_r$) of the perimeter-based agents.

Assumption 1: $p_c >= 1$

Assumption 2: $p_r <= 1$

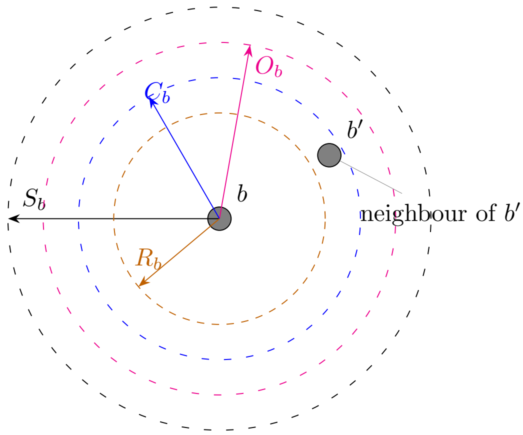


Fig. 1: Agent Fields

II. Related work

As far back as 1987 swarm theory has adopted the use of potential fields to coordinate agents [18] and this has continued since then in an attempt to improve the structure of a swarm, coordinate obstacles, and improve navigation [1], [2], [3], [4], [9], [11], [13], [20], [21]. A prototype framework for self-healing swarms was developed by Dai et al., which considered how to manage agent failure in hostile environments [6]. This was similar to work by Vassev and Hinchey, who modelled swarm movement using the ASSL (Autonomic System Specification Language) [23]. 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 is focused towards failure of an agent's internal systems, rather than on the removal of anomalies in a swarm distribution.

In the context of swarm structure maintenance, Roach et al. focussed on the effects of sensor failure, and the impact that has on agent distribution [19]. Lee and Chong identified the issue of concave edges within swarms in an attempt to create regular lattice formations [15], 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 [14]. They have also considered the effect that failed agents can have on a swarm when traversing a terrain [22].

This void reduction effect is an extension of the work presented by Eliot et al. [10], Ismail and Timmis [14], [22], and also builds on the work of Lee and Chong on concave edge identification [15], and on the work of McLurkin and Demaine on the detection of perimeter types [16]. However, perimeter type identification requires a communications infrastructure. The technique employed in this paper does not explicitly require the identification of the perimeter type as it would limit the size of the swarm[10], [15] and is therefore a reduced algorithm to identify any perimeter.

III. Basic swarming model

In the Original work by Eliot et. al. the resultant vector of an agent was calculated using Equation 1. Where k_c, k_r, k_d, k_o are weighting factors for the summed vectors associated with each interaction. i.e. v_c, v_r, v_d, v_o for cohesion, repulsion, direction and object avoidance respectively.

$$v(b) = k_c v_c(b) + k_r v_r(b) + k_d v_d(b) + k_o v_o(b) \quad (1)$$

In this paper we will only be considering the cohesion and repulsion components of the equation to create the new compression effect.

A. Cohesion

The cohesion component of an agent is calculated in a similar way to the repulsion in that it is dependent upon the proximity of neighbours. Where $n_c(b)$ is the set of neighbour agents for b (Eq. 2). The inclusion of an agent from a swarm (S) in by the agent's cohesion field (C_b).

$$n_c(b) = \{b' \in S \;:\; b' \neq b \wedge \|\vec{bb'}\| \leq C_b\} \quad (2)$$

The effect of an agent being within this set is that it will generate a vector that should 'encourage' agents to maintain their proximity. i.e. generate a cohesive swarm. The general weighted (k_c) formula for agents to maintain their proximity is to direct their motion towards the central point of all neighbouring agents as shown in Equation 3. This formula includes the k_c quotient that allows the cohesion effect to be 'balanced' with respect to other vector influences as described in [8], [9], [10]

$$v_c(b) = \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} k_c \vec{bb'} \quad (3)$$

B. Repulsion

The repulsion component of an agent's movement is calculated from interaction with its neighbours $n_r(b)$ (Eq. 4) in a swarm ($\mathcal{S}$) that are within the agent's (b) repulsion field (R_b).

$$n_r(b) = \{b' \in \mathcal{S} : b \neq b' \wedge \|\vec{bb'}\| \leq R_b)\} \quad (4)$$

The repulsion is then calculated as the average of all the vectors created by the agent (b) to the neighbours (b') (Eg. 5) and its proximity ($\|\vec{bb'}\| - R_b$).

$$v_r(b) = \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} \left(\|\vec{bb'}\| - R_b\right) \widehat{bb'} \quad (5)$$

IV. Perimeter detection

For perimeter compression to be applied to a swarm the perimeter needs to be detected. A perimeter can be defined as a continuous 'surface' of agents that are not enclosed by other agents. These agents may form an outer (green) or inner (red) boundary (Fig. 2).

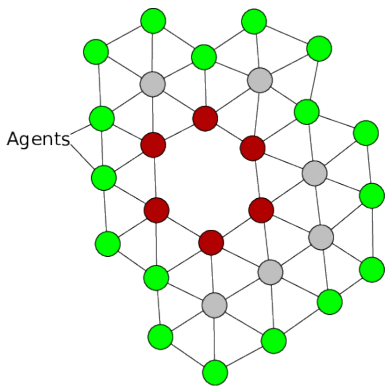


Fig. 2: Outer and inner swarm perimeters.

The detection process is achieved using a cyclic analysis of the agents that surround an agent (Fig. 3). Ghrist et al. discusses a similar technique using sweep angles [12] as does McLurkin et al [16].

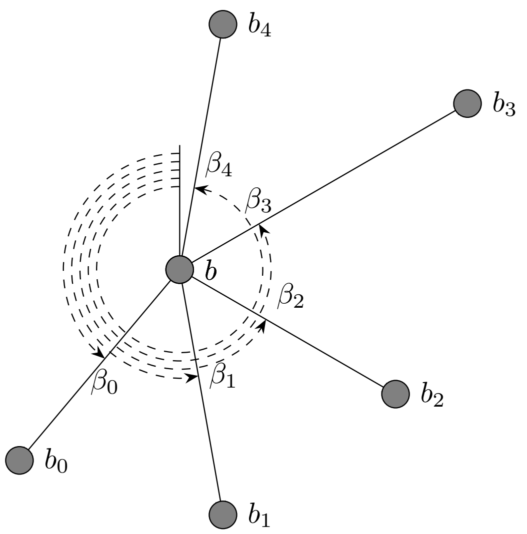


Fig. 3: Agent neighbours

The initial detection of the agents is based on the distance that each agent in the swarm is away from the current agent as described in Section III-A and shown in Equation 2. The perimeter detection set is based on the azimuth angle (β) of each neighbour agent as shown in Fig. 3.

The neighbour set $n_c(b)$ is then sorted in ascending order such that $\beta_0 < \beta_1 < \ldots < \beta_n$. Each consecutive pair of agents in the sequence now defines an *edge* which has a length that can be calculated from their relative positions.

An agent is therefore on the perimeter of the swarm if it is not enclosed by the polygon defined in the sorted set $n_c(b)$.

The polygon generated by $n_c(b)$ is considered to be 'open' if two successive agents on the perimeter are separated by more than the size of the cohesion field (C_b).

One further condition must be checked to detect a perimeter agent. When the polygon defined by $n_c(b)$ is closed it is possible that two or more neighbour agents are compressed to the point that they are with C_b but are 'behind' agent b. This condition can be detected based on any neighbour pair angle being greater than π. Figure 4 shows this condition, assuming all the neighbour agent pairs are within range C_b.

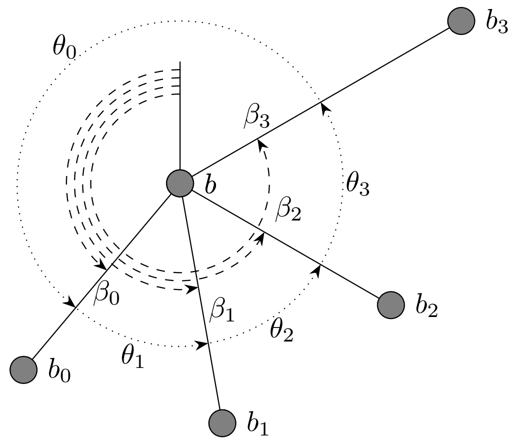


Fig. 4: Agent neighbour angles

V. Perimeter compression model

The new algorithm requires each individual agent to modify the movement vector generated based upon the perimeter status of the agent and each neighbour. The equation has been simplified (Eq. 6) and the cohesion and repulsion weighting factors (k_c, k_r) have been transferred into the calculations. The additional weighting factors (p_r, p_c) are applied to specific agents within the cohesion and repulsion vector calculations.

As in the basic model (Section III) the formula is simplified to only account for cohesion and repulsion (Eq. 6) although obstacle avoidance and direction can be added.

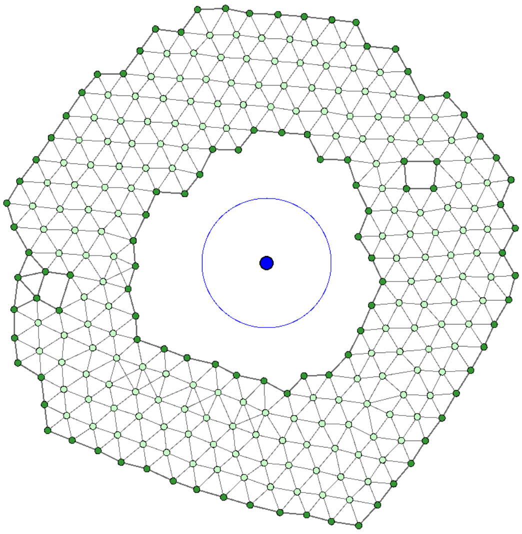
$$v(b) = v_c(b) + v_r(b) \tag{6}$$

The effect of introducing the additional weighting factors (p_c and p_r) can be seen in § VI which demonstrates the additional internal disturbance of the swarm caused by the compression algorithm and the removal of internal swarm voids.

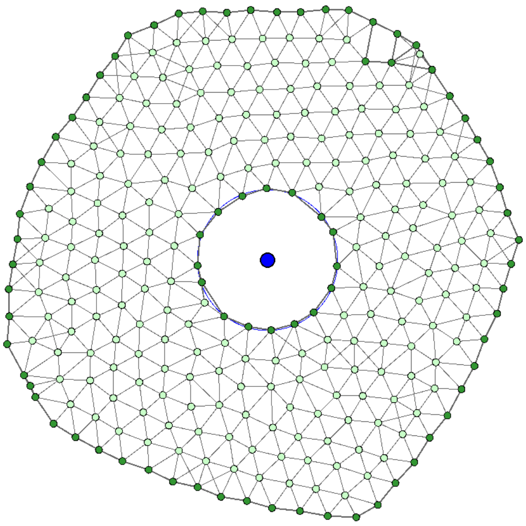
Figure 5 shows how the compression effect can remove a void from a swarm by surround an obstacle in a similar manner to the method described by Eliot et al. in [10].

A. Modified cohesion model

The cohesion component of the compression effect (p_c) is applied when an agent (b) and its neighbour (b') are both perimeter-based. If the agents are not both perimeter-based then the agents cohesion vector is scaled by k_c (Eq. 8). The effect of the additional cohesion-compression weighting is to increases the size of the effective, generated cohesion-vector (**ef**fective *c*ohesion) $\mathsf{efc}(b, b')$



(a) Void removal start



(b) Void removal finish

Fig. 5: Void removal through perimeter compression

(Eq. 7) which causes the agent to have a tendency to move towards that neighbour.

$$v_c(b) = \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} \mathsf{ekc}(b, b')\, \vec{bb'} \tag{7}$$

$$\mathsf{ekc}(b, b') = \text{if } \mathsf{per}(b) \text{ and } \mathsf{per}(b') \text{ then } \mathrm{p}_c k_c \text{ else } k_c \tag{8}$$

B. *Modified repulsion model*

The repulsion compression component of the perimeter is applied by adjusted the **e**ffective **r**ange **f**ield ($\mathsf{erf}(b, b')$) if the agents are both perimeter-based. Equation **??** shows the new formula to calculate the adjusted repulsion. As the agent's repulsion field is always within the cohesion field (Eq. 2), the repulsion neighbours can also be defined as a subset of the cohesion neighbours $n_c(b)$ (Eq. 10).

$$n_r(b) = \{b' \in \mathcal{S} : b \neq b' \wedge \|\vec{bb'}\| \leq \mathsf{erf}(b, b')\} \tag{9}$$

$$n_r(b) = \{b' \in n_c(b) \;:\; \|\vec{bb'}\| <= \mathsf{erf}(b, b')\} \tag{10}$$

An agent is identified as a perimeter agent using the technique described in § IV and shown by Eliot et.al. in [10] which uses a cyclic-check of neighbour agent angles to identify 'gaps' in the neighbours.

If the condition of both agents being a perimeter is met ($\mathsf{per}(b)$ and $\mathsf{per}(b')$) the repulsion field distance is multiplied by the compression factor (p_r) and the new field effect is used to generate a resultant-repulsion-vector (Eq. 12).

The effect of Equation 11 will be to reduce the repulsion of inter-perimeter-based agents allowing them to be closer together before a reduced repulsion-vector is applied.

> **Important:** The repulsion-vector that is generated is based upon $p_r R_b$, the reduced repulsion field, and not the full R_b field. This is to scale the resultant-repulsion-vector as well as reducing the repulsion field.

$$v_r(b) = \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} k_r \left(\|\vec{bb'}\| - \mathsf{erf}(b, b') \right) \widehat{bb'} \tag{11}$$

$$\mathsf{erf}(b, b') = \text{if } \mathsf{per}(b) \text{ and } \mathsf{per}(b') \text{ then } p_r R_b \text{ else } R_b \tag{12}$$

VI. Experimental results

A. *Baseline*

For all the experiments the parameters used to create the basic swarming effect are shown in Table I. Where C_b is the cohesion field, k_c is the cohesion weighting, R_b is the repulsion field, k_r is the repulsion weighting. The swarm consists of 200 agents which are distributed with a void at the centre. These initial parameters create a hexagonal-based distribution of agents that stabilise as shown in Figure 6. This basic swarm is used as the initial state for all the experiments.

Swarming Variable	Value
C_b	3.00
k_c	0.15
R_b	2.00
k_r	50.00
k_g	25.00

TABLE I: Swarming effect parameters

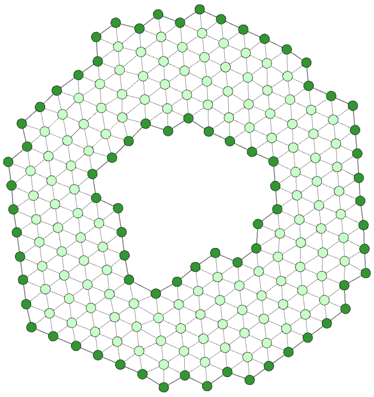


Fig. 6: Baseline swarm in stabilised configuration.

When the simulation is ran with no compression the resultant magnitudes generated are shown in Table 7. These states are used as the baseline for the experiments to measure the effects of the compression algorithm and compare the new algorithm to the existing void reduction algorithm discussed by Eliot et al. in [9].

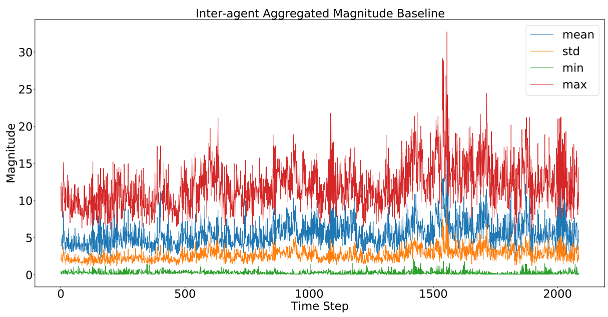


Fig. 7: Baseline swarm in stabilised configuration.

B. Compression Effects

The compression effect parameters are shown in table II

Comp.	1	2	3	4	5	6
p_r	0.10	0.15	0.20	0.25	0.30	0.35
p_c	20.00	30.00	40.00	50.00	60.00	70.00

TABLE II: Compression effect parameters

Pr/Pc	10	20	30	40	50
0.1	0.1/10	0.1/20	0.1/30	0.1/40	0.1/50
0.2	0.2/10	0.2/20	0.2/30	0.2/40	0.2/50
0.3	0.3/10	0.3/20	0.3/30	0.3/40	0.3/50
0.4	0.4/10	0.4/20	0.4/30	0.4/40	0.4/50
0.5	0.5/10	0.5/20	0.5/30	0.5/40	0.5/50
0.6	0.6/10	0.6/20	0.6/30	0.6/40	0.6/50
0.7	0.7/10	0.7/20	0.7/30	0.7/40	0.7/50
0.8	0.8/10	0.8/20	0.8/30	0.8/40	0.8/50
0.9	0.9/10	0.9/20	0.9/30	0.9/40	0.9/50

TABLE III: Experiment parameters 1

Pr/Pc	60	70	80	90	100
0.1	0.1/60	0.1/70	0.1/80	0.1/90	0.1/100
0.2	0.2/60	0.2/70	0.2/80	0.2/90	0.2/100
0.3	0.3/60	0.3/70	0.3/80	0.3/90	0.3/100
0.4	0.4/60	0.4/70	0.4/80	0.4/90	0.4/100
0.5	0.5/60	0.5/70	0.5/80	0.5/90	0.5/100
0.6	0.6/60	0.6/70	0.6/80	0.6/90	0.6/100
0.7	0.7/60	0.7/70	0.7/80	0.7/90	0.7/100
0.8	0.8/60	0.8/70	0.8/80	0.8/90	0.8/100
0.9	0.9/60	0.9/70	0.9/80	0.9/90	0.9/100

TABLE IV: Experiment parameters 2

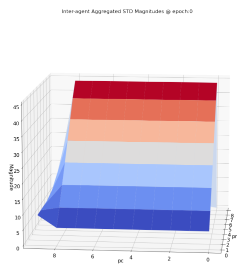


Fig. 8: Experiment 1

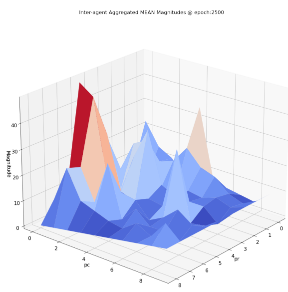


Fig. 9: Experiment 2

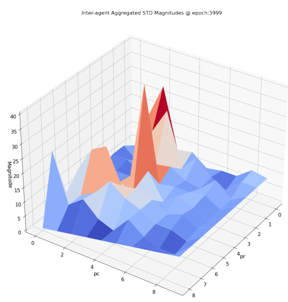


Fig. 10: Experiment 3

The first area of comparison is the effect of the algorithms on the number of perimeter agents. The baseline swarm's agents oscillates but remain in a relatively stable state with a constant number of perimeter agents and the internal anomaly persists (Fig. 6). The maximum and minimum number of perimeter agents is shown in table V.

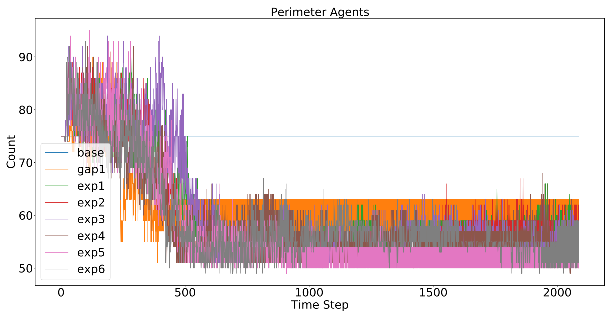


Fig. 11: Perimeter Count of baseline, gap reduction and perimeter compression.

Comp.	Base	Void	1	2	3	4	5	6
Max	75	90	90	90	94	92	95	93
Min	75	51	51	51	51	49	49	49
Mean	75	62	59	58	60	59	57	59
Std	0	6	9	10	10	8	10	8

TABLE V: Perimeter agents

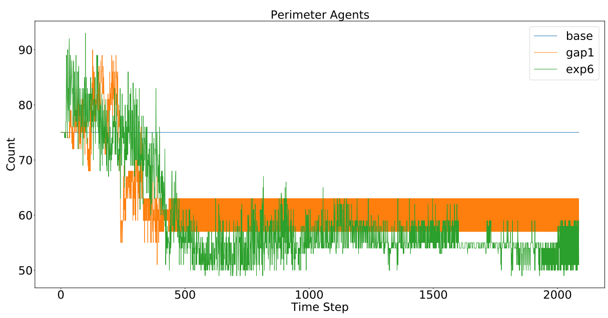


Fig. 12: Perimeter Count of baseline, gap reduction and Experiment 6.

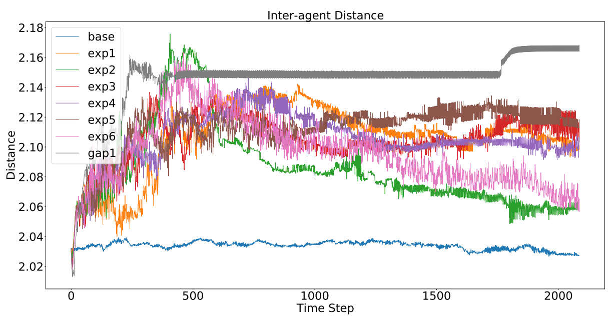


Fig. 13: Distance metric of baseline, gap reduction and perimeter compression.

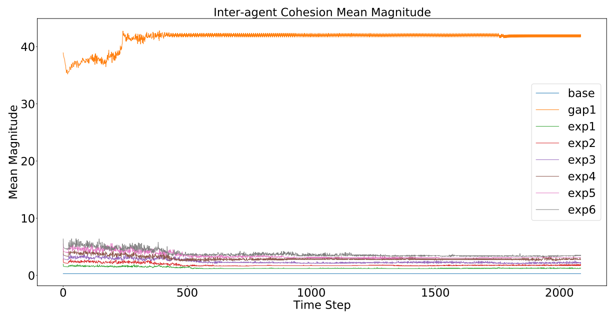


Fig. 14: Inter-agent Cohesion Mean.

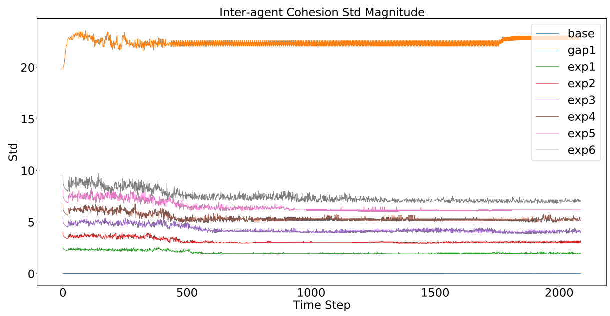


Fig. 15: Inter-agent Cohesion Std.

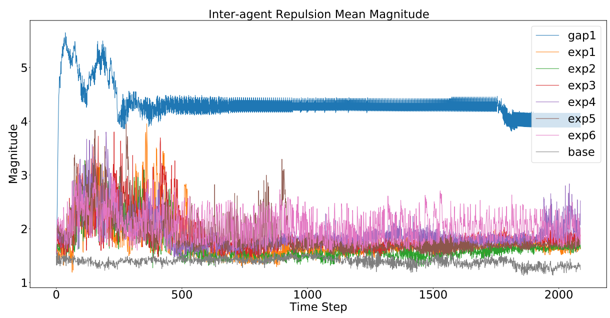


Fig. 16: Inter-agent Repulsion Mean.

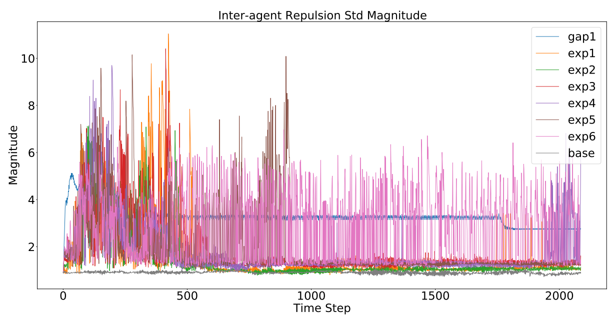


Fig. 17: Inter-agent Repulsion Std.

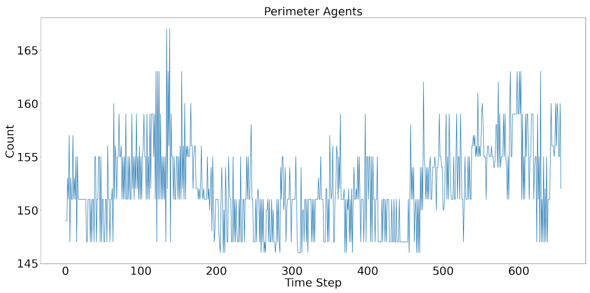


Fig. 18: Baseline swarm in stabilised configuration.

C. Gap compression

D. Perimeter compression

Compression 1

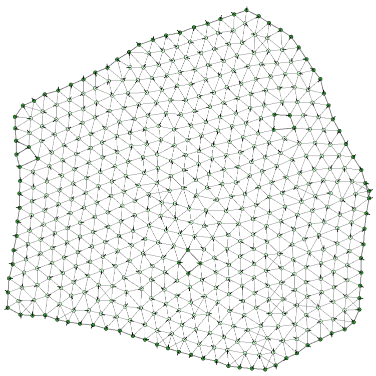


Fig. 19: Baseline swarm in with compression set 1 resultant configuration.

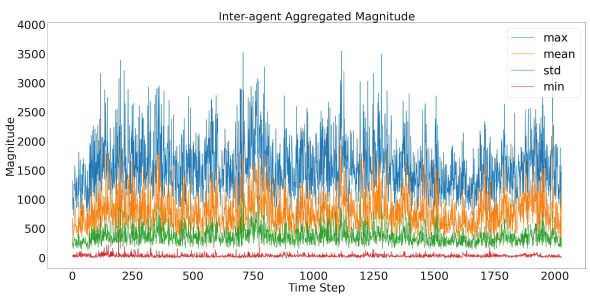


Fig. 20: Baseline swarm in with compression set 1.

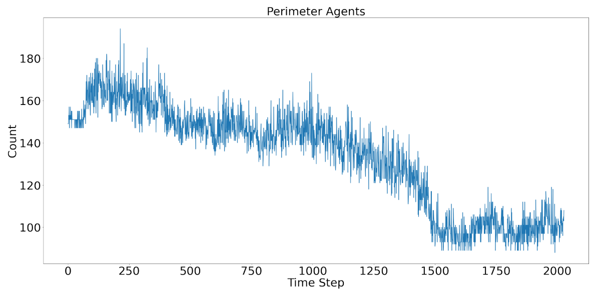


Fig. 21: Baseline swarm in with compression set 1.

E. Comparison

VII. Conclusions

From the initial simulations it is possible to show that the technique is able to successfully remove voids and surround an obstacle as shown in the video https://youtu.be/3eY1vvq0JWo.

VIII. Future Work

References

[1] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno. 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*, pages 85–94. IEEE, 2006.

[2] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno. Swarm formation control with potential fields formed by bivariate normal functions. In *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, pages 1–7. IEEE, 2006.

[3] L. Barnes, M. Fields, and K. Valavanis. Unmanned ground vehicle swarm formation control using potential fields. In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–8. IEEE, 2007.

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

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

[6] Y. Dai, M. Hinchey, M. Madhusoodan, J. Rash, and X. Zou. 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*, pages 3–10, Sept 2006.

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

[8] N. Eliot. *Methods for the Efficient Deployment and Coordination of Swarm Robotic Systems*. University of Northumbria at Newcastle (United Kingdom), 2017.

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

[10] N. Eliot, D. Kendall, A. Moon, M. Brockway, and M. Amos. Void reduction in self-healing swarms. In *Artificial Life Conference Proceedings*, pages 87–94. MIT Press, 2019.

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

[12] R. Ghrist, D. Lipsky, S. Poduri, and G. Sukhatme. Surrounding nodes in coordinate-free networks. In *Algorithmic Foundation of Robotics VII*, pages 409–424. Springer, 2008.

[13] S. P. Hou and C. C. Cheah. Multiplicative potential energy function for swarm control. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4363–4368, Oct 2009.

[14] A. R. Ismail and J. Timmis. Towards self-healing swarm robotic systems inspired by granuloma formation. In *Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on*, pages 313–314. IEEE, 2010.

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

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

[17] T. Räz. On the application of the honeycomb conjecture to the bee's honeycomb. *Philosophia Mathematica*, page nkt022, 2013.

[18] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics*, volume 21, pages 25–34. ACM, 1987.

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

[20] F. E. Schneider and D. Wildermuth. A potential field based approach to multi robot formation navigation. In *Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on*, volume 1, pages 680–685 vol.1, Oct 2003.

[21] J. Son, H. Ahn, and J. Cha. Lennard-jones potential field-based swarm systems for aggregation and obstacle avoidance. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 1068–1072, 2017.

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

[23] E. Vassev and M. Hinchey. 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*, pages 77–86, April 2009.