Perimeter Compression in self-healing swarms

Brockway M, Eliot N, Kendal D,
Hexham University
Cronkley Campus
Kiln Pit Hill
Department of Computer Science

February 8, 2021

Abstract

Perimeter Compression is a technique where by a void reducing effect can be added to a basic swarming algorithm. The affect 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.

1 Introduction

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 [1, 2, 3] and shown in § 4.

The aim of the algorithm is to reduce the spacing between perimeter-based agents by reducing the repulsion field and increasing the cohesion affect 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 $p_r <= 1$

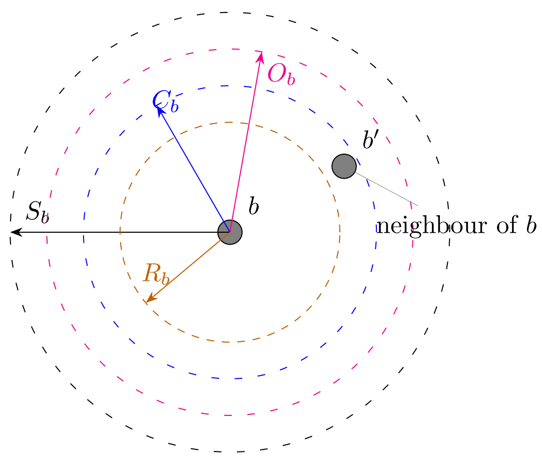


Figure 1: Agent Fields

2 Basic 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.

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

In this paper we will only be considering the cohesion and repulsion components of the equation to create the new compression effect. (Eq. 2).

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

2.1 Repulsion

The repulsion component of an agent's movement is calculated from interaction with its neighbours $n_r(b)$ (Eq. 3) in a swarm ($\mathcal{S}$) that are within the agent's (b) repulsion field (R_b).

$$n_r(b) = \{b' \in \mathcal{S} : b \neq b' \wedge \|\vec{bb'}\| \leq R_b)\} \tag{3}$$

The repulsion is then calculated as the average of all the vectors created by the agent (b) to the neighbours (b') (Eg. 4) and its proximity ($\|\vec{bb'}\| - R_b$).

$$v_r(b) = \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} \left(\|\vec{bb'}\| - R_b \right) \widehat{bb'} \tag{4}$$

2.2 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. 5). The inclusion of an agent from a swarm (S) in by the agent's cohesion field (C_b).

$$n_c(b) = \{b' \in S \ : \ b' \neq b \wedge \|bb'\| <= C_b\} \tag{5}$$

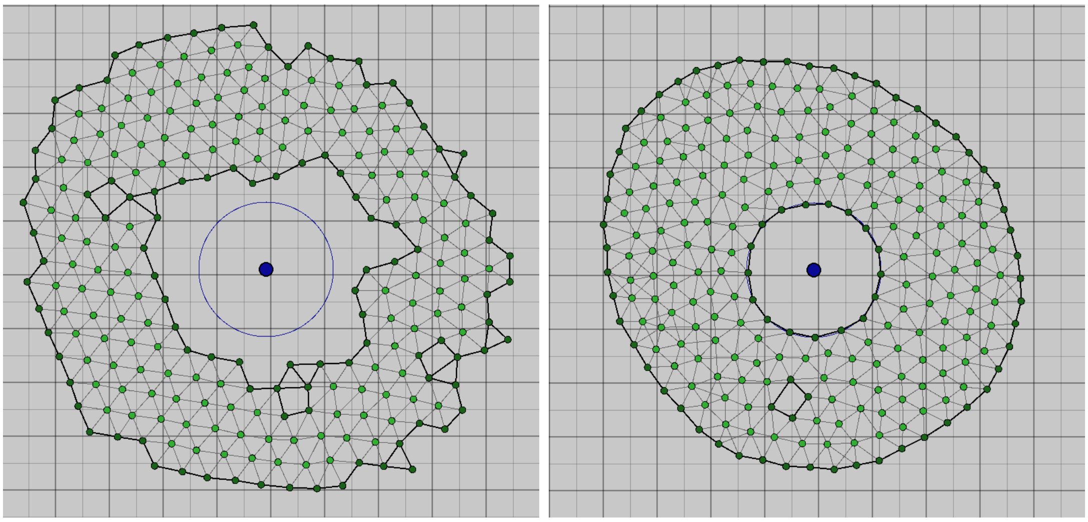
The affect of an agent being within this set is that it will generate a vector that should 'encourage' agents to maintain their proximity. i.e. generate a cohesive swarm. The general weighted (k_c) formula for agents to maintain their proximity is to direct their motion towards the central point of all neighbouring agents as shown in Equation 6. This formula includes the k_c quotient that allows the cohesion effect to be 'balanced' with respect to other vector influences as described in [1, 2, 3]

$$v_c(b) = \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} k_c \vec{bb'} \tag{6}$$

3 Compression model

The new algorithm requires each individual agent to modify the vector generated based upon the perimeter status of the agent and each neighbour. The equation has been simplified (Eq. 7) and the weighting factors have been transferred into the calculations along with the additional weighting factors that are applied to specific agents within the cohesion and repulsion vector calculations.

$$v(b) = v_c(b) + v_r(b) + v_d(b) + v_o(b) \tag{7}$$



(a) Void removal start

(b) Void removal finish

Figure 3: Void removal through perimeter compression

As in the basic model (§ 2) the formula is simplified to only account for cohesion and repulsion (Eq. 8).

$$v(b) = v_c(b) + v_r(b) \tag{8}$$

The effect of introducing these additional weighting factors (p_c and p_r) can be seen in Figure 2. The metric used to producing the graph is based upon the inter-agent magnitudes [2].

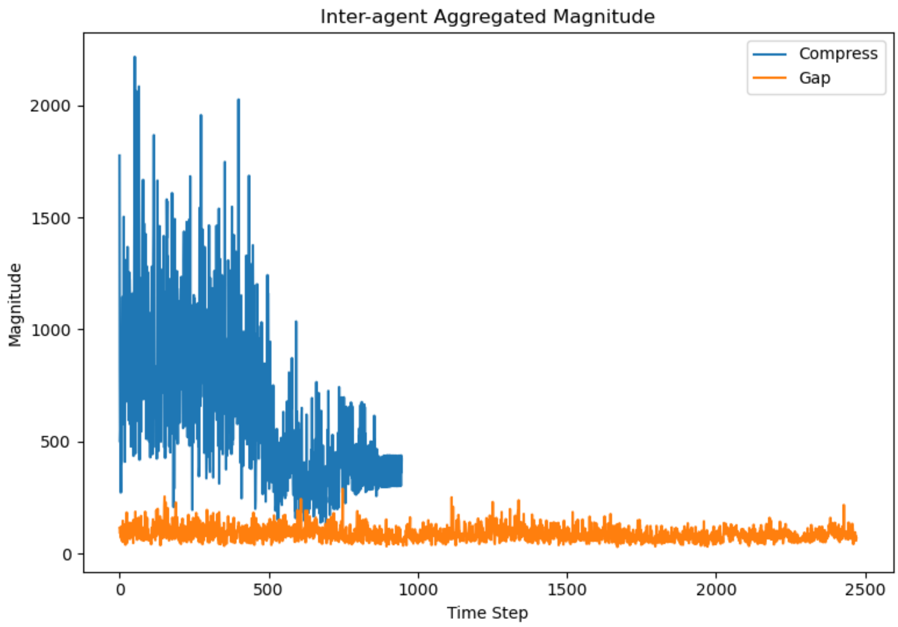


Figure 2: Comparison of magnitude change using gap reduction and perimeter compression

The graph shows that the agents inter-magnitudes are effected and the swarm stablises rapidly removing internal voids (Fig. 3). The graph shows a comparison of the new method to the existing method by Eliot et al. [3].

4 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 4. These agents may form an outer (green) or inner (red) boundary.

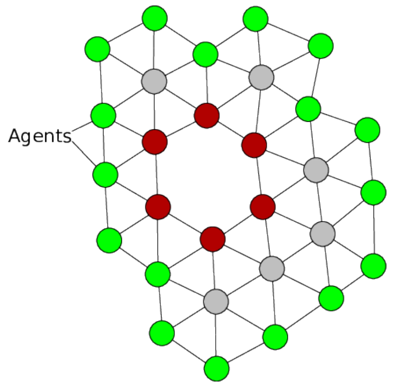


Figure 4: Outer and inner swarm perimeters.

The detection process is achieved using a cyclic analysis of the agents that surround an agent (Fig. 5). Ghrist et al. discusses a similar technique using sweep angles [4] as does McLurkin et al [5].

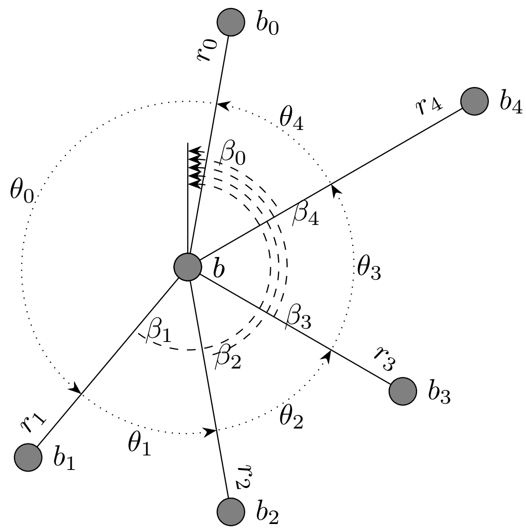


Figure 5: Agent neighbours

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 § 2.2 and shown in Equation 5. The perimeter detection set is based on the range and bearing of each neighbour agent, where r is the *range* and β is the *bearing* 9.

$$\mathcal{N}_b = \{(r, \beta) \dots\} \tag{9}$$

The set is then sorted in ascending order such that $\beta_0 < \beta_1 < \dots < \beta_n$. 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 are expressed in Equation 10.

$$\mathcal{P}_e = \langle (d_0, \theta_0), \dots, (d_n, \theta_n) \rangle \tag{10}$$

where

$$\theta_i = \beta_{i+1} - \beta_i \tag{11}$$

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

$$\sum \theta_i = 2\pi \tag{12}$$

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

$$d_i^2 = r_{i+1}^2 + r_i^2 - 2r_{i+1}r_i \cos\theta_i \tag{13}$$

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 14.

$$\exists \theta_i \in \mathcal{P}_e : \theta_i \geq \pi \tag{14}$$

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 15 is true.

$$\exists d_i \in \mathcal{P}_e : d_i > C_b \vee \exists \theta_i \in \mathcal{P}_e : \theta_i \geq \pi \tag{15}$$

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

$$\exists (\theta_i, d_i) \in \mathcal{P}_e : d_i > C_b \wedge \theta_i < \pi \tag{16}$$

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

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

4.1 Repulsion compression

The repulsion compression component of the perimeter is applied by adjusted the effective range ($\mathsf{erf}(b, b')$) if the agents are both perimeter-based. Equation 18 shows the new formula to calculate the adjusted repulsion. Equation 18 shows the calculation of the effective field. As the agent's repulsion field is always within the cohesion field (Eq. 5), the repulsion neighbours can also be defined as a subset of the cohesion neighbours $n_c(b)$ (Eq. 19).

$$n_r(b) = \{b' \in \mathcal{S} : b \neq b' \wedge \|\vec{bb'}\| \leq \mathsf{erf}(b, b')\} \tag{17}$$

$$\mathsf{erf}(b, b') = \mathsf{if}\ \mathsf{per}(b)\ \mathsf{and}\ \mathsf{per}(b')\ \mathsf{then}\ p_r R_b\ \mathsf{else}\ R_b \tag{18}$$

$$n_r(b) = \{b' \in n_c(b)\ :\ \|bb'\| <= \mathsf{erf}(b, b')\} \tag{19}$$

An agent is identified as a perimeter agent using the technique shown by Eliot et.al. in [3] which uses a cyclic-check of neighbour agent angles to identify "gaps" in the neighbours as shown in § 4.

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. 21).

The effect of Equation 20 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.

$$v_r(b) = \frac{1}{|n_r(b)|} \sum_{b' \in n_r(b)} k_r \left(\|\vec{bb'}\| - \mathsf{erf}(b, b') \right) \widehat{bb'} \tag{20}$$

$$\mathsf{erf}(b, b') = \mathsf{if\ per}(b)\ \mathsf{and\ per}(b')\ \mathsf{then}\ p_r R_b\ \mathsf{else}\ R_b \tag{21}$$

4.2 Compression cohesion

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 vector is only scaled by k_c (Eq. 23). The effect of the additional cohesion-compression weighting is to increases the size of the generated cohesion-vector $\mathsf{efc}(b, b')$ (Eq. 22).

$$v_c(b) = \frac{1}{|n_c(b)|} \sum_{b' \in n_c(b)} \mathsf{ekc}(b, b')\, \vec{bb'} \tag{22}$$

$$\mathsf{ekc}(b, b') = \mathsf{if\ per}(b)\ \mathsf{and\ per}(b')\ \mathsf{then}\ \mathrm{p}_c k_c\ \mathsf{else}\ k_c \tag{23}$$

5 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.

References

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

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

[3] 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.

[4] 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.

[5] 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.