

# Void Reduction in Self-Healing Swarms

Neil Eliot<sup>1</sup>, David Kendall<sup>1</sup>, Alun Moon<sup>1</sup>, Michael Brockway<sup>1</sup>,  
Martyn Amos<sup>1</sup>

<sup>1</sup>Department of Computer and Information Sciences  
University of Northumbria

ALife, 2019



# Introduction

- 1 How did we do that?
- 2 Set the Ground Rules!
  - Swarm Rules
- 3 Communications and/or Sensing
  - Communications
  - Sensing
- 4 Void Reduction
  - Model
  - Local Effect
  - Global Effect
  - Simulated Results

- Swarms consist of many agents (mobile robots or drones) that interact according to a simple set of rules.
- We consider swarms of agents that:
  - ▶ are capable of detecting their neighbours (proximity detection).
  - ▶ do not require any another form of communication.
- Swarms can be made fault tolerant (resilient to agent loss).

# Why no communications?

- Communication propagation protocol demands computing overhead.
  - ▶  $n_1 \rightarrow n_3$
  - ▶  $n_1 \rightarrow n_2$
  - ▶  $n_3 \rightarrow n_2$  (decision!)
- Message propagation takes time which limits swarm size.

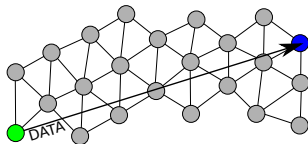
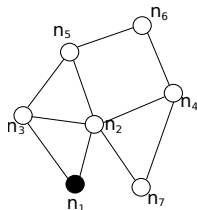
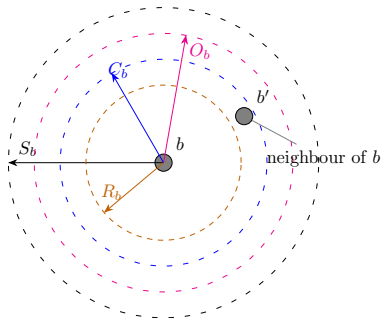


Figure: Swarm Communications

# Proximity Sensing

- No communication is necessary apart from proximity detection.
- Arbitrary sized swarms are possible.
- Agent attributes include various ranges as shown in figure.



**Figure:** Ranges:  $S_b$  = sensing,  $O_b$  = obstacles avoidance,  $R_b$  = repulsion,  $C_b$  = cohesion.

- Movement of agent  $b$  is computed as the weighted sum of 4 vectors, as shown in equation 1

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

- $v_c(b)$ : cohesion term ensures agents remain part of the swarm.
- $v_r(b)$ : repulsion term ensures agents do not collide.
- $v_d(b)$ : destination vector for goal based swarms.
- $v_o(b)$ : obstacle avoidance vector.
- Numerical weights  $k_c, k_r, k_d, k_o$  to allow tuning of relative effects.

# The Swarm

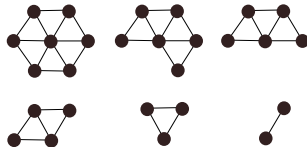


Figure: Stable Structures

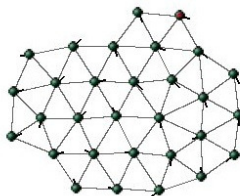


Figure: The Swarm - (Simulator)



## NOTE

Perimeter detection is used as part of the *void reduction* process. This will be discussed later.

- Perimeter detection allows for directional coordination with reduced resource usage.
  - ▶ 'Internal' agents don't need to use their GPS.
- Reduces computational overhead in agents.

# What is a Perimeter?

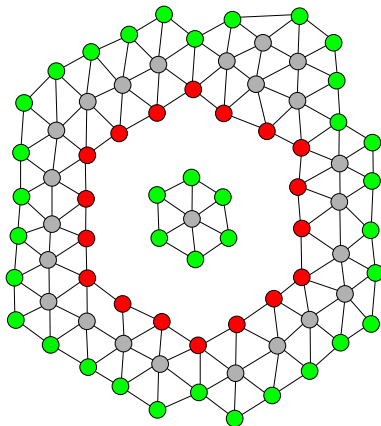


Figure: Internal (red) and external (green) perimeters

# What is a Void?

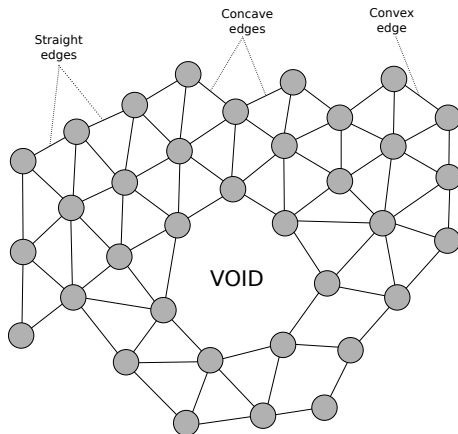


Figure: Concave components of a perimeter (voids)

# Perimeter Detection (Concave)

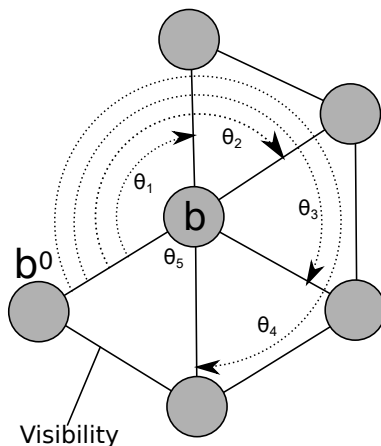


Figure: Concave gap (*Void Reduction*)

# Perimeter Detection (Convex)

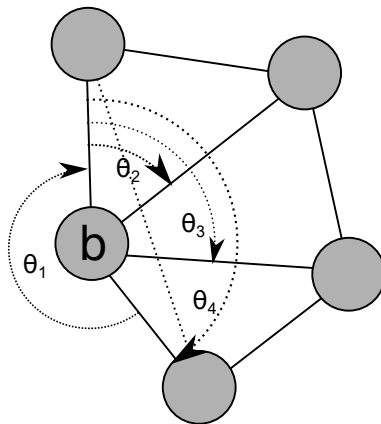


Figure: Convex gap

# Void Reduction Movement

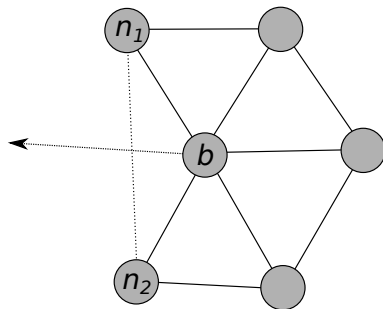


Figure: Concave detection

As part of the perimeter detection a pair  $G_b = \{n_1, n_2\}$  of agents is generated. These are the first two agents identified as creating a 'gap' in agent  $b$ 's neighbours. Equation 2 calculates the centroid of the identified 'gap'.

$$D_{pos}(b) = \frac{1}{2}(n_1 + n_2) \quad (2)$$

( $n_1, n_2$  label agents and also denote their position vectors.)

# Local agent movement

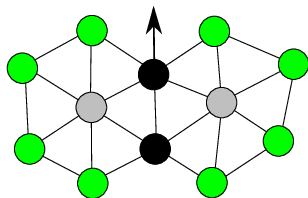


Figure: Initial positions

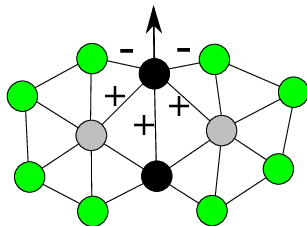


Figure: Agent movement

# Global agent movement

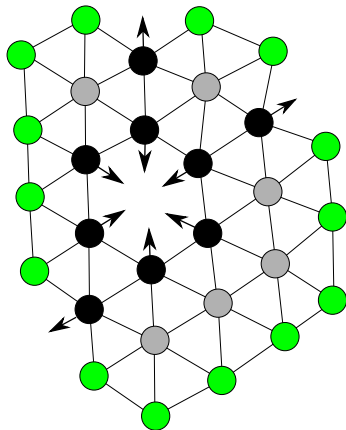


Figure: Initial positions

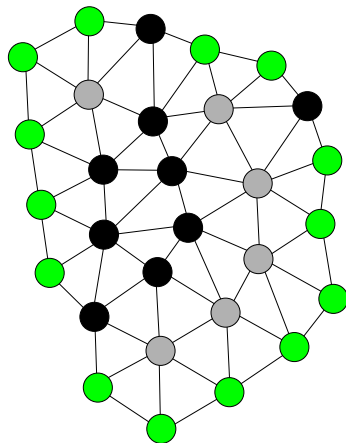


Figure: Agent movement



# Scenario 1

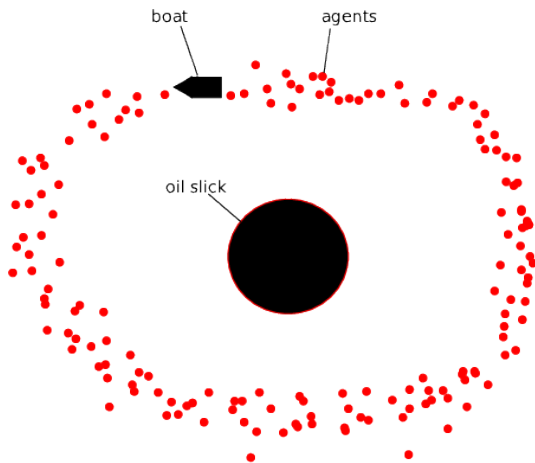


Figure: Oil Slick Encapsulation

# Scenario 1

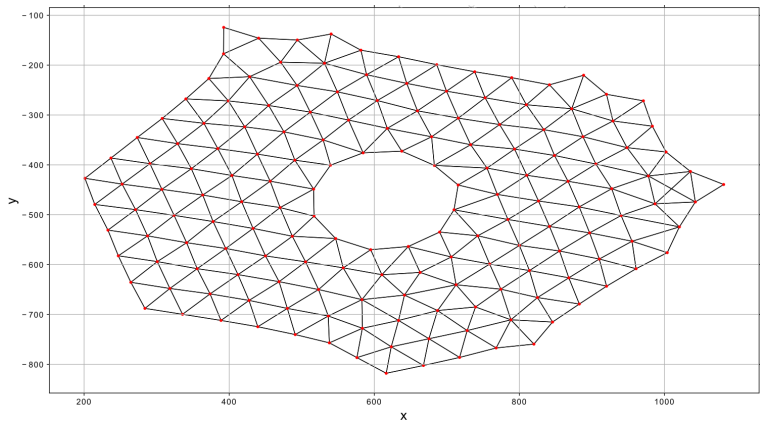


Figure: Oil Slick Encapsulation

## Scenario 2

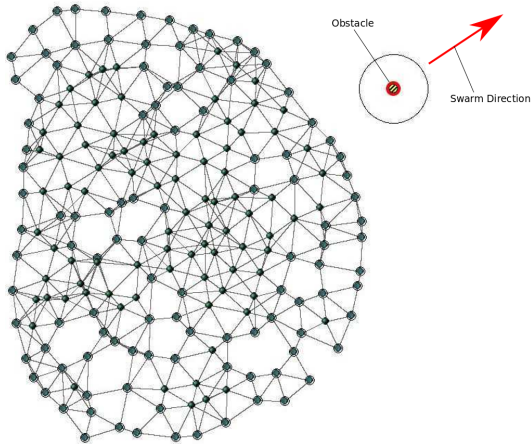


Figure: Mobile Swarm

## Scenario 2

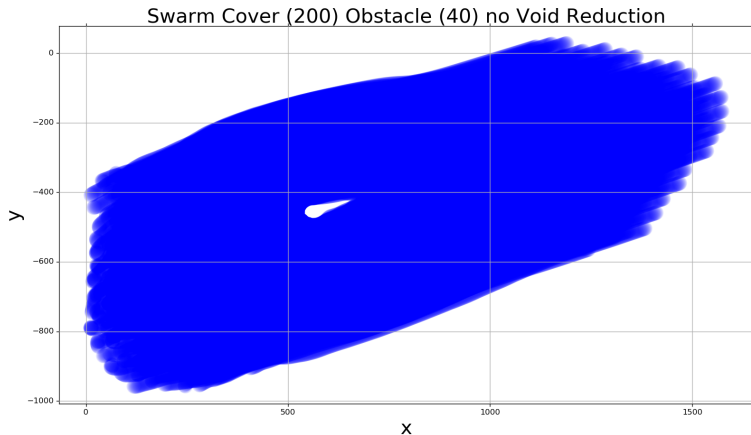


Figure: No void reduction - path plot

## Scenario 2

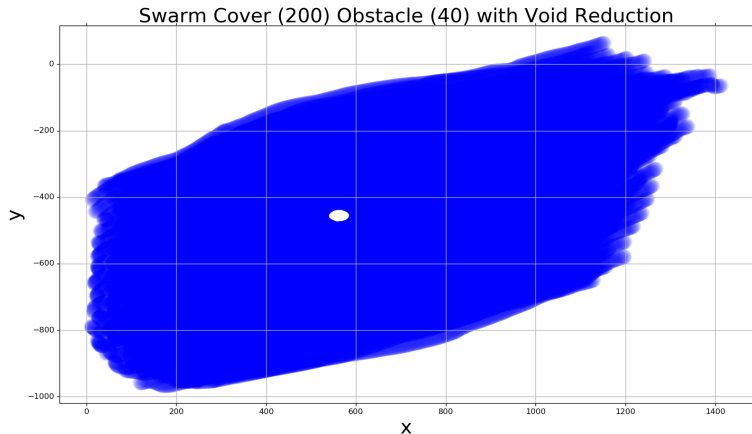


Figure: Void reduction - path plot

# Summary

- **Void reduction** has a local effect creating a global emergent behaviour that improves the shape and structure of a swarm.
- **Void reduction** removes anomalies on internal and external perimeters.
- **Void reduction** can be applied to both static swarms and directional swarms.

# Thank You

THANK YOU!  
QUESTIONS?

# Agent Movement

Here are the equations defining the movement vectors for cohesion and repulsion.

$b, b'$  denote agents and also (inside the  $\sum$ s), their position vectors.

$\mathcal{R}_b$  is the set of agents within agent  $b$ 's repulsion range;  $\mathcal{C}_b$  is the set of agents within its cohesion range.

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

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



$$v_d(b) = d \quad (5)$$

$d$  is a constant vector which points to a destination.

$$v_o(b) = O_b \left( \sum_{o \in \mathcal{O}_b} \hat{o} \right)^\wedge \quad (6)$$

$O_b$  is  $b$ 's detection range;  $\mathcal{O}_b$  is the set of obstacles within this range.

$o$  denotes an obstacle and (within the  $\sum$ ) its position vector. The caret  $^\wedge$  denotes normalisation of a vector to unit length.