Ecological Complexity, Volume 19, September 2014, Pages 68-79
DOI:10.1016/j.ecocom.2014.04.004

A model for using self-organized agents to visually map environmental profiles

John Oyekan *[1], Gu Dongbing[2] and Huosheng Hu[2]

[1]School of Applied Sciences, University of Cranfield
[2]School of Computer Science and Electrical Engineering, University of Essex

April 24, 2014

Abstract

In this work, we investigate the possibility of using inspiration from the self-organising property of organisms in nature for providing visual representation of an invisible pollutant profile. We present a novel mathematical model of the bacterium and use it to find pollutants in the environment. This model has the capability of exploring the environment to search for sparsely distributed pollutants or food sources and then subsequently exploiting them upon discovery. We also combine the bacterium model in a bacterium-flock algorithm for the purposes of preventing collisions between robots or organisms in addition to providing coverage to a pollutant. By adjusting the velocity of individuals, we show that we are able to control the coverage provided by the population as a whole. Furthermore, we compare the bacterium-flock algorithm with a novel gradient-ascent-flocking algorithm and the well established Voronoi partition algorithm. Results show that bacterium-flock algorithm and the Voronoi partition algorithm are capable of adapting the distribution of the individuals of a population based upon the underlying pollutant profile while the gradient-ascent-flocking algorithm is not. This shows that the bacterium-flock and the Voronoi partition algorithm can potentially be used to track a spatiotemporal function. On the other hand, the gradient-ascent-flock algorithm has a faster convergence time in some cases with the Voronoi partition algorithm having the slowest convergence time overall.

keywords: self-organization, natural templates, mathematical modelling, flocking, robotics

1 Introduction

In the event of an accidental leakage or deliberate release of an invisible hazardous substance as a result of the act of terrorism, human casualties could be reduced by making the invisible hazardous substance visible. In such situations, a visual reference is possibly the only clue that humans can use to keep away from contaminated areas. Such invisible hazardous substances could include Nuclear radiation as was the case in the recent 2010 Japanese Tsunami Nuclear disaster (Dauer et al., 2011) when nearly 60,000 people had to evacuate areas close to the affected Fukushima reactor (Matanle, 2011); Nerve gas as in the 1991 gulf war (Kang & Bullman, 1996) and carbon dioxide as in 1986 when the Lake Nyos in Cameroon released its underground storage of carbon dioxide. This led to the asphyxiation of both animals and unsuspecting 1700 humans in the surrounding areas (Baxter et al., 1989). A similar event occurred two years earlier when Lake Monoun released carbon dioxide resulting in the death of up to 37 people (Kling, 1987). According to Kling (1987), the month that these events occur can be predicted as both occurred in August. However, due to the invisible nature of the gas, it is presently not possible to know when a release event actually occurs. A similar event is likely to occur in Ethiopia around the Lake Kivu in the near future with potentially greater consequences (Schmid et al., 2005). The above cases indicate a humanitarian challenge that calls for a solution.

In order to meet this challenge, a swarm of robots could be used to provide a visual reference so that a previously invisible substance becomes visible. Providing this solution could lead to a reduction in human casualties. Furthermore, providing a visual representation of such biological hazards would reduce the pressure on emergency services who might be overwhelmed with dealing with the already injured. In such cases, communication of hazardous areas to the population would be instantaneous as opposed to relying on state of the art robots to collect information, relay it to a

*oyekanjohn@gmail.com

base station in order to build an electronic map, before communicating the situation to the population via the emergency crew. This cycle would result in unnecessary delays resulting in more causalities. The possibility of using this proposed visual approach to provide information to a population is becoming more feasible as devices get smaller and cost less as a result of technological advances. This is especially true with the continuing advance in nano-technology. In the near future, nano-bots could be deployed as a means to visually represent invisible spatio-temporal quantities. However, presently, implementing a flock of flying agents outside the lab is still a challenging issue to be solved. Nevertheless, progress is being made in this area as in Kushleyev et al. (2013).

In robotics, providing a solution to the problem raised above, could be viewed as a coverage or optimal resource allocation task (Cortes et al., 2004). Researchers in robotics have investigated various methods of solving this problem. This includes the use of virtual springs in which the distance between individual robots in the swarm are controlled using virtual forces. The virtual springs generate virtual forces which are used to keep the agents from each other (Shucker et al., 2006, 2008). By varying the length of the spring in accordance to the concentration level of the pollutant in the vicinity, the agents can be used to represent the distribution of the pollutant. Their approach however needs to track individual points in the pollutant and in addition, requires a heavy communication requirement. Another approach is the use of deterministic annealing which involves the use of the theory behind metal annealing to find the optimal configuration of the position of agents with respect to a pollutant's distribution (Kwok & Martínez, 2011). Metal annealing involves heating a metal to a very high temperature and then slowly cooling it. The high temperature phase makes the metal's atoms vibrate vigorously in order to escape their present state which might not be the most optimal. This phase could be termed as an explorative state in which the search for an optimal state takes place. This phase is followed by a slow cooling phase which makes the metal's atoms settle into low energy configurations that is representative of the optimal configuration. The deterministic annealing algorithm also makes use of these phases in order to make the agents explore their environment and then settle into configurations representative of the pollution distribution. However, the algorithm requires a heavy communication requirement in order to synchronize the agents.

The most commonly used approach in robotics for multi-agent coverage as evident from literature (Cortes et al., 2004; Schwager et al., 2007, 2008, 2009) is the Voronoi partition method. This method aims to minimize a cost function represented by the pollutant with respect to the agents' positions. It does this by using robot positions to divide the area containing the pollutant into Voronoi cells. The mass density of the Voronoi cells are then calculated followed by the centre of mass of the cells. The robots are then moved to the positions of the newly calculated centre of masses at the next iteration.

The problem with this approach is that it relies on unrealistic sensor constraints that could only be satisfied by using a machine learning paradigm. It requires that a robot's sensor has a radius of pollutant perception beyond the immediate position of the robot so that the mass density and then the centre of mass of the corresponding Voronoi cell can be calculated. This is not possible as the robot can be in only one place at a time. As a result, in order to obtain this information, a machine learning paradigm is often used to estimate the pollutant data beyond the immediate position of the robot. As a result of the use of the machine learning paradigm in addition to the computation of the mass density of the Voronoi cell, a heavy computational cost arises. Furthermore, it does not have an exploratory behavior for the initial search of the environment for a spatiotemporal quantity of interest and is consequently prone to local maximum traps (Cortes et al., 2004; Schwager et al., 2007, 2009).

Due to the heavy computational or communication costs required by the methods discussed above, it becomes difficult to make use of these approaches in providing visual representations of an invisible hazardous substance especially if the substance is dynamic and time is of essence. As a result, this paper investigates the possibility of using an alternative technique by taking inspiration from nature. Organisms in the natural environment have undergone evolution over million of years resulting in robust, efficient and cost effective mechanisms for carrying out their everyday activities. This paper investigates the possibility of using the self-organizing properties of natural organisms in order to provide a solution to the visual representation of invisible hazardous substances as discussed above.

The phenomena of self-organization has been observed in the brood sorting of honey bees, termite hill building, the emergence of dead ant clusters among other phenomenon. Self organization relies on multiple interactions between agents, a balance of exploitation and exploration with a feedback mechanism (Bonabeau et al., 1999).

Bonabeau et al. (1999) discussed that the emergence of patterns or structures by self organization is not always predictable. However, a way of making it predictable is through the use of templates. Templates are predefined structures or commands that agents follow during the construction of a structure. For example, it has been shown that the termites of the specie Macrotermes subhyalinus use the queen's pheromone as a template to build the royal chamber (Bonabeau et al., 1997). Templates make it possible to predict self organization and Johnson (2009), was able to use templates to obtain biological plausible results of honey bee nest construction. In addition, Jost et al. (2007) showed that ant cemetery clusters are actually dependent on wind direction templates. By controlling the wind direction or flow, it is possible to change the structure of the cemetery clusters. It was observed that when the wind was flowing

longitudinally through the nest, ants arranged their dead longitudinally into the direction of the flow. However, without wind flow, the ants made a nearly circular heap in the centre of the nest (Bonabeau et al., 1999). From this observation, it can be seen that the cemetery cluster shape formed by ants is dependent on wind flow direction in the nest.

Collectively, bacteria too are known to exhibit self organization as is evident when they form rings or other various shapes around food sources in the environment depending on the type (Shapiro, 1998; Ben-Jacob, 2003; Marrocco et al., 2010). When food is found, it has been observed that the swarm of Bacteria move towards it while maintaining their self-organization (Shklarsh et al., 2012).

Individually, bacterium use chemotaxis to navigate up chemical gradients in the environment (Brown & Berg, 1974; Segall et al., 1986). However, due to noise in the environment caused by turbulence, internal mechanisms and so on, individual estimates might be wrong. In order to increase accuracy, it has been shown that animal groups including bacterium often school in order to collectively improve their individual estimates and navigate up gradients (Simons, 2004; Shklarsh et al., 2011). However, Berdahl et al. (2013) argued that flocking could emerge as a result of interactions between individuals and not just for the purpose of improving individual estimates. In experiments, they showed how individual golden shiners were affected more by the group in which they were placed compared to the environmental gradients they individually experience. They came to a conclusion that the gradient tracking capabilities of schools were better than an individual's gradient tracking capability. Nevertheless, the above observations have led to the study of the swarming capabilities of organisms including bacteria for various purposes including micro-cargo transporting (Shklarsh et al., 2012), micro-assembly (Martel & Mohammadi, 2009), air turbine inspection (Correll & Martinoli, 2009), and so on.

In this paper, we investigate how the emergent phenomenon of bacteria self-organization can be used to achieve the visual representation of invisible hazardous substances in the environment. It should be noted that this is not the only potential application of the proposed method. It can be used for any application in which providing coverage of a spatiotemporal function is needed. This could include visualising the profiles of various environmental quantities such as temperature, light intensity, exotic natural resources in the environment and so on. In this paper, we show a parameter rich controller based upon a bacterium model developed by Berg and Brown. By controlling the parameters of the controller, it is possible to adjust its explorative and exploitative attributes. Since one agent is simply not enough to visualize a pollutant, we use a multiple of agents to achieve this. This however introduces a control complexity in coordinating the dynamics of multiple agents and preventing collisions which could be damaging to robotic agents. In order to address this challenge, a co-operative foraging controller in the form of a flocking controller is introduced. In the next section, we discuss how we combined the outputs of these two controller together in order to achieve our goal.

2 Methods

Berdahl et al. (2013) discussed how the emergent interaction between the golden shiners used in their experiments could be explained using a social cue and an environmental cue representing the gradient of an environmental quantity. The combination of similar cues were investigated in Shklarsh et al. (2011) where they used a weight to combine individual bacterium motions with the swarm level interactions. The bacterium individual motions were caused as a result of chemotaxis. The weight was determined individually by using the gradient information at the individual's location. If the sensed gradient was positive, the weight was assigned a value of 1 and 0 otherwise. As a result, when there is a positive gradient (weight = 1), the agents navigated as a group and individually when the gradient was 0 or negative. Following this approach introduces a search behaviour as a result of individual chemotactic motions.

Another implementation as discussed in Torney et al. (2009) seems to use flow velocity of the medium in moving agents to the source of a pollutant. This means that in order to be implemented in robotics, information about the medium's flow is needed. In addressing swarm level interactions, the researchers modified the radius of attraction and orientation zones in response to signal strengths. If there is a decrease in signal strength, the attraction radius was increased while if there is an increase in signal strength, the interaction radius was set to 0. All the examples above are classified as context dependent interaction where the context observed in the environment determines the interaction level between swarm members. Berdahl et al. (2013) were able to show in their experiments that the velocities of individual golden shiners were affected by light levels in the environment hence affecting the interaction between members of the school. This experiment provides evidence of the theory of context dependent swarm level interactions in biological organisms.

Nevertheless, the model presented in Torney et al. (2009) does not provide a solution to the problem of exploration in order to find the plume. Their results seems to suggest that they assumed that the agents were in the plume at the beginning of the experiment. On the other hand, the model presented in Shklarsh et al. (2011) presents a solution to individual exploration but does not take into account that natural organisms still avoid collisions with each other during

Parameters	Meaning	Bacterium behaviour range of values	Bacterium and Flocking behaviour values
k_d	Chemical Sensitivity of robotic agent	$2 \rightarrow 30$	2
τ_o	Run length in absence of pollutants	$2 \rightarrow 30$	2
α	Amplification factor	$2 \rightarrow 30$	2
β_o	Velocity in absence of pollutants	10	10
G_S	Flocking separation gain	-	1
G_R	Flocking repulsion gain	-	0.99
G_F	Flocking behaviour gain	-	0.8
G_B	Bacterium behaviour gain	-	0.8
$C(X,t)$	Pollutant value range	$1 \rightarrow 100$	$1 \rightarrow 100$
n	Number of Neighbours	-	5

Table 1: Table showing parameters, meanings and typical values in experiments

this process.

In other to address these issues, knowledge obtained from a robotic paradigm called behaviour-based robotics is used in this work. This paradigm addresses a problem by combining sub-behaviours in a user defined architecture (Arkin, 1998). Sub-behaviours are chosen based upon the sub-challenges that make up the challenge. In this case, the challenge is made up of sub-challenges involving environmental exploration, pollutant detection, and subsequent visualization using a multiple of agents. Environmental exploration and pollutant detection is solved by using a bacterium behaviour while the control of multiple agents to achieve visualization of an invisible pollutant is achieved by using a flocking behaviour. Combining these two behaviours together is achieved by a linear addition of their outputs. A dwelling behaviour is embedded into the bacterium behaviour in order to simulate how organisms slow down when in a favourable condition. This could be as a result of abundance of prey as in sharks foraging trajectories or low light intensity conditions as in golden shiners (Berdahl et al., 2013). This approach makes it possible to avoid inter-individual collisions during exploration and still collectively form a visual representation of an invisible pollutant.

In order to benchmark our approach, we compare it to the Voronoi partition method which is a well established technique of providing coverage in robotics. This technique divides an area into Voronoi cells and then uses the density of pollutants in each cell to calculate the movement of robots and hence their positions (Cortes et al., 2004). This results in the placement of more robots in areas of higher pollutant levels when compared to areas of low pollutant levels.

We also compared our approach with a relatively simple controller that combines a simple gradient ascent controller with a flocking controller. This controller could be viewed as a more conceptual or higher level view of our approach. This is because at a high level, a bacterium uses a gradient ascent approach in searching for food in the environment.

2.1 Proposed Bacterium-Flocking model

The Bacterium and Flocking Behaviours are combined together using Equations 1 and 2. G_F and G_B are gains applied to both the flocking controller velocity output $\dot{x^i_F}$ and bacteria controller velocity output $\dot{x^i_B}$ respectively and i is an individual agent in a population N of agents. A summary of the parameters used in this work is provided in Table 1.

$$\dot{x^i}_{final} = \dot{x^i_F} * G_F + \dot{x^i_B} * G_B \tag{1}$$

$$\dot{y^i}_{final} = \dot{y^i_F} * G_F + \dot{y^i_B} * G_B \tag{2}$$

Equations 1 and 2 provide the velocity updates for the individual agents making up a flock. We use gains of $G_F = 0.8$ and $G_B = 0.8$ from trial and error experiments conducted. As a result, these choice of parameter values are ad-hoc. More studies into why these values work for the purposes of this work should be conducted in future.

2.1.1 Bacterium Behaviour

In order to find the pollutants in the environment, we take inspiration from the bacterium and model our robotic agent as a biological agent utilizing a composite search strategy. This search strategy has been found to be very effective at discovering patchy distributed resources in the environment especially in the absence of detailed or lack of knowledge

of its environment (Benhamou, 2007; Reynolds, 2009). The composite search strategy involves both extensive and intensive search modes. Extensive search mode involves movements of the organism such that its motion can be classed as long, straight-line moves with small turning phases while intensive search mode involves movements of shorter straight-lines and more turning phases leading to a thorough search of a resource rich area (Hinkelman et al., 2013). This results in a search strategy in which organisms spend more time in areas in the environment that have more resources than areas having little or no resources. This strategy possibly increases organism's energy efficiency at finding food sources.

Casting this into robotics, maximizing the pollutant patches encountered and identified per battery life time is very important. As a result, the robot should spend less time in areas with little or no pollutants and more time in areas containing pollutants in order to map the pollutant distribution at that location. In this work, we assume that the robotic agent does not clean or destroy the pollutant. Both the extensive and intensive search modes of the composite search strategy have been modelled in various ways by researchers in literature. For example, Plank & James (2008) modelled a Brownian composite search strategy in which the extensive search mode was similar to a ballistic motion with a higher agent velocity and few turns while the intensive search mode had a smaller velocity with more frequent turns. It was mentioned in Plank & James (2008) that the motion resulting from this model is similar to that observed in the study of Klaassen & Nolet (2006) on swans. In Hinkelman et al. (2013), composite Levy walks with different μ parameters were used for the extensive and intensive search modes. A high μ was used for the intensive search mode while a smaller μ was used for the extensive search mode.

The switch between the extensive search mode and intensive search mode is still a topic of research among various researchers (Hinkelman et al., 2013). In Hinkelman et al. (2013) and Plank & James (2008), a giving up time (GUT) was used to switch from intensive search to extensive search after an elapsed time. A similar approach was followed by Reynolds (2009) by using a give up distance. This model has reportedly been demonstrated by studies on fruit flies (Bell, 1990). In Barto et al. (2013), the switch between an intensive search (uncorrelated random walk) and extensive search (straight lines) was performed by using the organism's "satiety" state. It has also been discussed in literature that the switch between the extensive mode and intensive search mode can also be caused by sensory perception cues in various organisms (Hinkelman et al., 2013) such as in the E-Coli bacterium and Salmonella typhimurium (Adler, 1975; Dusenbery, 1998). In these organisms, a switch from extensive to intensive search mode is caused by coming into contact with a food source. In this work, this mode switching strategy is used.

The bacterium behaviour is composed of a combination of tumble and run phases (Passino, 2002). The tumble phase is a bacterium motion in which it uses it's flagellum to reorient itself into a different direction. During this phase, a bacterium can randomly choose any direction in an angle distribution that is dependent on the previous heading of the bacterium (Jackson, 1987). This phase is used to reorient the bacterium to a favourable direction when heading in the wrong direction and introduces a random component into the bacterium's behaviour.

The run phase is a bacterium motion that is approximately a straight line after ignoring all disturbances (Passino, 2002). The bacterium uses the alternation of these two phases to drive itself towards the source of food in the environment.

The bacterium behaviour has been studied in various ways in the past and mathematical models developed (Marques et al., 2002)(Dhariwal et al., 2004). However, in this work, the Berg and Brown model (Brown & Berg, 1974) as shown in Equations 3 to 5 are used.

$$\tau = \tau_o e^{(\alpha \frac{d\overline{P_b}}{dt})} \tag{3}$$

$$\frac{d\overline{P_b}}{dt} = \tau_m^{-1} \int_{-\infty}^{t} \frac{dP_b}{dt'} e^{(\frac{(t'-t)}{\tau_m})} dt', \tag{4}$$

$$\frac{dP_b}{dt} = \frac{k_d}{(k_d + C(x,t))^2} \frac{dC}{dt} \tag{5}$$

τ is the mean run time, τ_o is the mean run time in the absence of concentration gradients. This parameter is responsible for controlling the bacterium's run length in the absence of pollutants. Increasing this parameter would increase the bacterium's exploration range but reduce it's ability to respond immediately and correct itself if going in a wrong direction.

τ_m is the time constant of the bacterium system while P_b is the fraction of the number of chemical particles bound to the bacterium's chemical receptors. $\frac{dP_b}{dt}$ is the rate of change of P_b and $\frac{d\overline{P_b}}{dt}$ is the weighted rate of change of P_b.

k_d is the dissociation constant of the the bacterial chemoreceptor and controls the chemical sensitivity of the bacteria. Lower levels mean more sensitivity to food and vice versa. In bacterium, the value of this constant is adjusted

to a low level when food is scare in order that it might find food easily. However, a low value of k_d means that the agent would be trapped in local maximums when food is abundant. As a result, this value is adjusted to a higher level when food is abundant. This can be modelled as an adaptive term that increases or reduces based upon the rate of encountering food items in the environment.

$\frac{d\overline{P_b}}{dt}$ resembles an output from an exponential moving average low pass filter and is used by the bacterium to filter out noise arising from the environment or its sensors. In this work, the exponential low pass filter uses data from the past 4 seconds similarly to what a bacterium does with its chemoreceptor readings (Passino, 2002) (Oyekan et al., 2012). In this work, $\frac{dC}{dt}$ is given by $[C(x,t) - C(x - \Delta x, t - \Delta t)]$ where Δx is obtained as a result of the agent's motion in Δt period. α is an amplification constant of the bacterial system.

The above set of Equations were developed by Berg and Brown as a model of an individual bacterium in Brown & Berg (1974) after collecting data about an individual bacterium cell's motion.

The model above was adapted into a controller for an individual robotic agent using the control law 6.

$$motion = \begin{cases} tumble() & if \quad \gamma > \tau \\ run() & else \quad \gamma < \tau \end{cases} \tag{6}$$

Where γ is a counter that increments every second but gets reset after it is greater than τ. When $\gamma > \tau$, the tumble motion occurs and a new θ is chosen causing the robotic agent to reorient itself and move in a different direction while when $\gamma < \tau$, the robotic agent continues moving in the previously defined θ angle value. For the tumble phase, the robotic agent chooses from a range of angles in the uniform distribution of the set of $\theta \in \{0...360\}$ where θ is in degrees. As seen in Equations 3 to 5, τ is related to measured gradient. When the measured gradient is large as a result of progress towards the source, τ would be large and γ would take a longer time to be greater than τ. As depicted in Equation 6, this leads to a longer run phase and hence progress towards the source.

In a 2-dimensional environment, Equations 7 and 8 show how these two motions are implemented. Where $\dot{x_B^i}$ is the x velocity component and $\dot{y_B^i}$ is the y velocity component of V_B of agent i. V_B is defined in 9. These two Equations are computed every iteration or time step.

$$\dot{x_B^i} = V_B^i cos\theta \tag{7}$$

$$\dot{y_B^i} = V_B^i sin\theta \tag{8}$$

We also introduce Equation 9 in order to represent the behaviour of an organism when it finds a food item. Before finding a food item, the velocity of the organism or the distance covered would be large as it explores the environment in search of food. During this stage, we assume that it will have a velocity which would be called β_o. However, once it has found a rich food source or concentration of food sources, its velocity V_B would reduce as the organism either stops or lingers at the location to forage. This behavior was integrated into the bacteria controller and is given by Equation 9.

$$V_B^i(x,t) = \begin{cases} \frac{\beta_o}{(C(x,t))} & if \quad pollutant\ found \\ \beta_o & if \quad pollutant\ not\ found \end{cases} \tag{9}$$

As a result, the values of V_B in Equations 7 and 8 is obtained according to Equation 9. Where β_o is a constant that is the velocity of the agent when exploring for food (i.e in the absence of concentration C readings) and can be used to tune the agent's speed of foraging. In this work, β_o is user defined and should be a value that would be capable of making the agent get to the source of the pollutant without stalling during its progress towards the source. This value would also depend on the agent's velocity capacity. Furthermore, the value of β_o would determine by how much the agent's position would fluctuate when it finds the food source. If β_o is large compared to the underlying pollutant concentration $C(x,t)$ then the agent's positions would fluctuate more and vice versa.

In real animals, the β_o value could be dependent on the amount of food observed by an organism at that location. If food is scarce, then the value would be large so that it does not dwell at that location for long before moving on to explore for richer food sources.

2.1.2 Flocking Behaviour

As the reader will probably appreciate, the bacterium behaviour with the right parameters would cause a high chemotactic force resulting in individual agents migrating towards the source of the pollutant or meeting at local maximums in the environment. This could lead to collisions between agents. Furthermore, they would not show a very useful visualisation of the pollutant profile as all agents would be at the centre of the source.

In order to solve this problem, a flocking behaviour was used. The first flocking algorithm was developed by C.W Reynolds in 1986 (Reynolds et al., 2005). The aim of this algorithm was to mimic the behaviour of a flock of birds by making sure that each individual in the flock avoided collisions with their neighbours (called Separation) but were also not too far apart from each other (called Aggregation). This behaviour enables agents to keep a relative distance from each other whilst ensuring cooperative foraging towards a goal in the environment. Following from the initial model developed by Reynolds et al. (2005), there have been many other models such as in Wang & Gu (2007) and Liu & Passino (2004). However, the morse potential flocking model (Smith & Martin, 2009) represented by Equation 10 was chosen because of the simplicity in tuning it's separation G_S and aggregation G_A parameter gains when compared to that used in (Wang & Gu, 2007) for example.

$$v_F^i(t+\delta t) = [G_S^i * exp(-r_c(t)/20) - G_A^i * exp(-r_c(t)/20)] \tag{10}$$

Gains of 1 for the separation term G_S and 0.99 for the aggregation term G_A were used. This is because if there is too large of a positive difference between G_S and G_A, then the agents would be further apart from each other. This would result in small pollutant profiles not being well covered as agents would be too far apart. r_c is the closest distance in a set $\{r_1.....r_n\}$ between an individual and its neighbours n at time t. Equations 11 and 12 show how the closest distance r_c is converted into the corresponding x_F and y_F velocity components for an individual i. x_c and y_c are the closest x and y co-ordinates in the neighbourhood set of $\{x_1.....x_n\}$, $\{y_1.....y_n\}$ of agent i. In this work, $n = 5$ was chosen.

$$\dot{x_F^i} := \dot{x_F^i} + \frac{v_F^i * x_c}{r_c} \tag{11}$$

$$\dot{y_F^i} := \dot{y_F^i} + \frac{v_F^i * y_c}{r_c} \tag{12}$$

In summary, Equation 10 ensures that agents will either move directly towards each other (Aggregate) or repel each other (Separate) depending on the distance between the agents. In the light of self-organization, the flocking behavior would enable multiple interactions amongst agents, explicit communication, and also supply the negative feedback when there are many agents at a position. Velocity feedback is introduced by Equations 11 and 12 according to the distance between individual flock members.

2.2 The Voronoi Partition method

The use of Voronoi partitioning in robotics was pioneered by Cortes et al. In Cortes et al. (2004), they showed how a group of robotic agents P, could be controlled to achieve optimal coverage of a simulated spatial distribution $C(X)$ in an area Q. This was achieved by dividing the area Q into Voronoi cells V_i using the individual robotic agents positions p_i, as in Equation 13. In Equation 13, Voronoi cells are obtained by iteratively comparing every point q in the environment Q with the agent positions p_i. The points q closest to robot position p_i are grouped as belonging to a Voronoi cell V_i. The calculation of V_i is what makes the use of the Voronoi partition method costly computationally.

Following this, the mass density M_v of each Voronoi cell is calculated using the spatial quantity ρ_c in each cell. ρ_c is found by summing up pollutant particles found at the points q that belong to V_i of robot position p_i. The mass density Equation is shown in Equation 14. From the mass density value, the position of the centre of mass C_V of the Voronoi cell is calculated using Equation 15. This Equation is similar to finding the average of positions captured in V_i except that the average is weighted according to the pollutant values found at the positions q in V_i. The robot is then moved to the position C_V using Equation 16 assuming the dynamics of Equation 17 where the term $(p_i - C_{V_i})$ is the error between the present robot position and the centre of mass of the Voronoi cell V_i with k_{prop} being a proportional gain. The greater this gain, the faster the robot would move to reduce the error term $(p_i - C_{V_i})$.

Due to the problem of computing V_i according to Equation 13, a perception radius is often defined for each robot. This is the radius within which they can sense the spatial quantity in their vicinity. This is needed to calculate the mass of the Voronoi cell within which the robot is located. As most spatiotemporal sensors can only perform point measurements, users of the Voronoi partition often use machine learning to estimate the profile of the spatial function

and then use the estimated function to compute the mass density of each Voronoi cell. In this work, we used a radius value of 50 pixels.

$$V_i = \{q \in \ Q \ \|q - p_i\| \leq \|q - p_j\|, \forall \neq i\} \tag{13}$$

$$M_V = \int_V \rho_c(q)dq \tag{14}$$

$$C_V = \frac{1}{M_V} \int_V q\rho_c(q)dq \tag{15}$$

$$u_i = -k_{prop}(p_i - C_{V_i}) \tag{16}$$

$$\dot{p_i} = u_i \tag{17}$$

The Voronoi cell partition approach can be viewed as trying to minimise the cost function Equation 18, where $f(.)$ could be any function used to simulate the cost of the robotic agent's sensor being far away from the position q. For more information on the Voronoi partition technique the reader is referred to Cortes et al. (2004).

$$M_V = \sum_{i=1}^{n} \int_{V_i} f\|q - p_i\|\phi(q)dq \tag{18}$$

2.3 Gradient Ascent and Flocking model

The gradient ascent algorithm is a simple algorithm that relies on using the gradient of a function to find its optimum. It is given by Equation 19 and 20 where ζ is the resolution of an agent's movement and was set to one in this work. Corresponding gradients of $\nabla C(x,t)$ and $\nabla C(y,t)$ were obtained by using the relationship $\frac{C(x-1.0,t)-C(x+1.0,t)}{2}$ and $\frac{C(y-1.0,t)-C(y+1.0,t)}{2}$ respectively. Obtaining the gradient value of a spatial function is essential for a gradient ascent algorithm to work. In order to compare it with the bacterium-flocking algorithm, we combine it with the flocking model outputs $\dot{x_F^i}$ and $\dot{y_F^i}$ as in Equations 21 and 22.

$$x_{t+\Delta t} = x_t + \zeta \nabla C(x,t) \tag{19}$$

$$y_{t+\Delta t} = y_t + \zeta \nabla C(y,t) \tag{20}$$

$$\dot{x^i_{final}} = \dot{x^i_F} + (W * (\Delta \eta_x)) \tag{21}$$

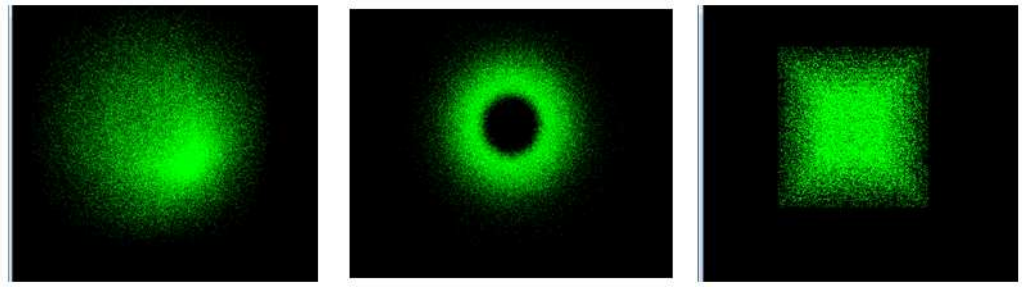
$$\dot{y^i_{final}} = \dot{y^i_F} + (W * (\Delta \eta_y)) \tag{22}$$

Where W is a gain. η_x is the difference between the present position x of the agent and $x_{t+\Delta t}$ according to Equation 19 and η_y is the difference between the present position y of the agent and $y_{t+\Delta t}$ according to Equation 20. W was set to one. According to Equations 19 and 20, the gradient of the function $C(\mathbf{X}, t)$ drives the agents.

2.4 Constructing the simulation environment

The simulation environment was constructed using JAVA programming language and each agent was assumed to have a point mass model. The position of the agent was determined by the output of the combination of both the bacteria and flocking behaviour using Equation 1 and 2. The environment had a size of 4000 pixels by 4000 pixels.

The environment was simulated using three overlapping virtual grids - One grid was used for representing the robots motion, another grid was used for communication purposes during flocking so that it was possible to know when an agent was within communication range of another agent. The third grid was used for pollutants. The pollutant grid was a boolean array of 4000 cells by 4000 cells with each cell representing a pixel position.



(a) Skewed Gaussian: Generated using a circle with an increasing radius value.

(b) Doughnut: Generated by subtracting a smaller Guassian from a larger Guassian.

(c) Square: Generated by using a Square function.

Figure 1: Showing various spatial profiles.

2.4.1 Smooth Pollutant Profiles

Smooth functions (that is functions that do not have any noise and hence called "smooth" functions in order to differentiate them from the functions generated by using a "pixel approach" as discussed in section 2.4.2) were used in order to compare our proposed approach, the standard Voronoi and the gradient ascent-flocking model against each other. These smooth functions were constructed using Gaussian functions of the type $A*\exp(\frac{|\mu-X|^2}{2\sigma^2})$. Where A is the amplitude, μ is the mean position of the function and σ is the standard deviation of the function. A smooth Gaussian function p having parameters of $\mu_x = 200$, $\mu_y = 200$, $\sigma_x = 90$, $\sigma_y = 90$, $A = 10$ was used as the simulated pollutant with comparison results presented in section 3.1 and 3.2.

2.4.2 Complex Pollutant Profiles

In generating complex or more realistic pollutant profiles, we used pixels to simulate pollutant particles. Whenever a pollutant particle was to be simulated, the position in which that particle was located was set to true. A robot was able to measure the pollutant concentration $C(\mathbf{X}, t)$ at a position $\mathbf{X} = \{x, y\}$ by counting the number of pollutant particles in a 10 by 10 pixel area.

Three static pollutant profiles were used to test the feasibility of visually forming an invisible hazardous substance by combining the bacterium behaviour and the flocking behaviour. The three pollutant profiles that were used are a skewed Gaussian pollutant profile as shown in Figure 1(a) (This was used to simulate a point source slowly diffusing into the environment under the influence of a small advection force), a doughnut pollutant profile as shown in Figure 1(b) (to simulate a scenario where the continuous pollutant profile has been broken up due to a cylindrical obstruction) and a square pollutant profile as shown in Figure 1(c) (to investigate what happens if the pollutant profile is of a unique shape). Using these different pollutant profiles made it possible to make sure that the algorithm was versatile no matter what profile the pollutant takes.

In order to simulate the doughnut pollutant profile, a Gaussian function having parameters of $\sigma_x = 70$, $\sigma_y = 70$, $\mu_x = 400$ and $\mu_y = 400$ was used to randomly place $20,000$ particles in the environment (That is setting the corresponding particle positions on the pollutant grid to boolean "true"). Then a smaller Gaussian function having the parameters of $\sigma_x = 25$, $\sigma_y = 25$, $\mu_x = 400$ and $\mu_y = 400$ was used to randomly remove the particles (That is setting the corresponding particle positions on the pollutant grid to boolean "false"). This created the doughnut pollutant profile seen in Figure 1(b).

In order to simulate the skewed Gaussian pollutant profile, a simulated skewed Gaussian function at the position $(x, y) = (400, 400)$ was used. This was generated by randomly placing particles within an expanding circle having a centre obtained by subtracting values of (D_x, D_y) according to Equations 23 and 24 from the initial position (x, y) over a number N iterations.

$$x := x - D_x \tag{23}$$

$$y := y - D_y \tag{24}$$

Following Equations 23 and 24 results in a diagonal trajectory for (x, y). The combination of the expanding circle and the diagonal trajectory results in a skewed pollutant profile. The radius R of the expanding circle is obtained using Equation 25.

$$R := R + J \tag{25}$$

Where D_x, D_y and J are user defined values that can be used to define the skewed pollutant profile. For the square pollutant profile, particles were randomly placed in a square section having dimensions 200 by 200 pixels to simulate the source. The square section's dimension was then progressively made larger over 10 iterations. The approach of randomly placing particles in the environment results in noise as seen in the results and was used in section 3.3.

3 Simulation results

Investigation into the effects of combining the bacterium behaviour with a flocking behaviour was carried out. This is necessary as without the flocking behaviour, the robots would collide with each other resulting in damage and stress in the case of biological organisms. In this set of simulations, the velocity Equation 1 and 2 were used. Simulations conducted in this section are divided into three sets. The first set of simulations compare the bacterium-flock algorithm, Voronoi partition algorithm and the gradient-ascent-flock algorithm together. In the gradient-ascent-flock algorithm, the bacterium component is simply replaced by a gradient ascent component. The second set of simulations investigate how the bacterium's model parameters affect the flocking model parameters while the third set of simulations deploy the bacterium-flock algorithm in more challenging noisy functions.

3.1 Comparisons of bacterium-flock algorithm, Voronoi partition algorithm and gradient-ascent-flock algorithm

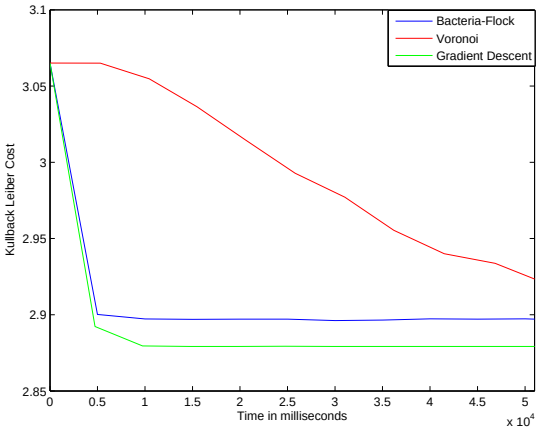
In this work, two performance metrics were tested: (i) How the different approaches would make the agents close to the pollutant distribution (ii) Rate of convergence to the pollutant distribution. In order to carry out these comparisons, the Kullback Leiber function was used as a cost function. This function is often used to compare a given function's continuous representation with its discrete or approximate representation. It is often used in robotics to see how closely two distributions are to each other (Kullback et al., 1987). The lower the value returned by the function, the closer the two distributions. This function is represented by Equation 26 where R_X and R_Y is the number of grids the simulated environment was divided into in the X and Y axis. The number of grids used in this set of experiments was 4000 by 4000. q is the concentration reading observed by the agents and p is the pollutant profile.

$$KL_{measure} = \sum_{x=1}^{R_X} \sum_{y=1}^{R_Y} \left[q_{(x,y)} log \frac{q_(x,y)}{p_{(x,y)}} \right] \tag{26}$$

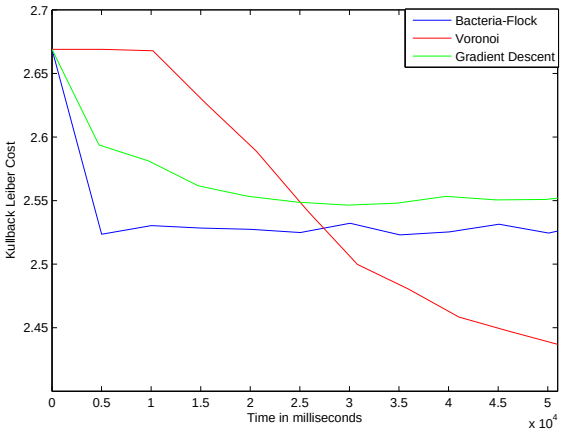
As mentioned earlier, a smooth Gaussian function p having parameters of $\mu_x = 200$, $\mu_y = 200$, $\sigma_x = 90$, $\sigma_y = 90$, $A = 10$ was used as the simulated pollutant.

For the **Flocking and Bacteria** combination, G_F and G_B were set to 0.8 each. Parameter values of k_d = 2, τ_o = 2 and α = 2 were used in order to present a fair comparison to the other methods. Using these values would result in low exploration of the environment containing a noiseless spatial function as discussed in section 2.1.1. 50 robotic agents that were randomly distributed at position $(x, y) = (600, 600)$ were used. The robotic agents had a radio communication radius of 50 pixels so that once an agent was within its neighbour's range, they could effectively "see" each other by communicating their positions to each other. This communicated position could then be used by individual agents to avoid each other. β_o was set to 1 in this simulation. For the **Voronoi partition method**, a radius value of 50 was used with the gain k of the agents set to 1 while for the **Gradient Ascent Algorithm**, ζ = 1. 50 robotic agents that were randomly distributed at position $(x, y) = (600, 600)$ were also used here.

When comparing the algorithms, simulations were ran for 50 seconds each. Time is used here because we were investigating how long each algorithm would use to reach convergence as each algorithm has its own unique property. Each experiment was repeated 20 times and an average calculated. In the simulations, the agents were bounded to a region of 0 to 4000 in both the x and y axis. This means that the agents were not allowed to go into regions less than 0 and greater than 4000. Figure 2(a) shows that the gradient-ascent-flocking algorithm has the lowest Kullback Leiber cost when compared with the Voronoi partition algorithm and the bacterium-flocking algorithm. This means that in the case of Figure 2(a), the gradient-ascent-flocking algorithm performs better. If however the pollutant parameters were changed slightly from $\mu_x = 200$, $\mu_y = 200$, $\sigma_x = 90$, $\sigma_y = 90$, $A = 10$ to $\mu_x = 200$, $\mu_y = 200$, $\sigma_x = 45$, $\sigma_y = 45$, $A = 10$, the bacterium-flock algorithm performs better than gradient-ascent-flocking algorithm in terms of speed of convergence and coverage as seen in the costs shown in Figure 2(b). Additionally, the Voronoi partition method has



(a) Using $\mu_x = 200$, $\mu_y = 200$, $\sigma_x = 90$, $\sigma_y = 90$, $A = 10$.



(b) Using $\mu_x = 200$, $\mu_y = 200$, $\sigma_x = 45$, $\sigma_y = 45$, $A = 10$.

Figure 2: Showing the average Kullback Leiber values of the various approaches. Lower costs mean better coverage.

a better coverage at the end of 50 seconds but slower convergence when compared to the other two approaches. This shows that the performance of the algorithms in comparison to each other depends on the pollutant profile $C(\mathbf{X}, t)$.

However, it should be noted as the gradient-ascent-flocking relies heavily on the gradient of the pollutant, in the absence of this knowledge the mechanism would not be able to navigate up the pollutant. Furthermore, in real life, pollutants or distributions are often not smooth but broken up into sparse pollutant segments. In this situation, the bacterium-flocking algorithm would still be able to explore its environment using Equation 9 and find the pollutant whereas, the remaining two mechanisms would not.

3.2 Effects of bacterium model parameters on bacterium-flocking algorithm

In this section, simulations were allowed to run for just over 3000 iterations (2 minutes). It was discovered that when using the combination of the bacterium and flocking behaviours, the bacterium's behaviour parameters did not have much effect on the level of coverage provided by the flock of robotic agents. $k_d = 2$ and $k_d = 20$ for $\alpha = 2$, $\tau_o = 2$ and $\beta_o = 16$ pixels per run were tested to see the effects on the coverage level provided by the flock of agents. In Figure 3(a), it can seen that the Kullback Leiber costs for using $k_d = 2$ and $k_d = 20$ were not very different.

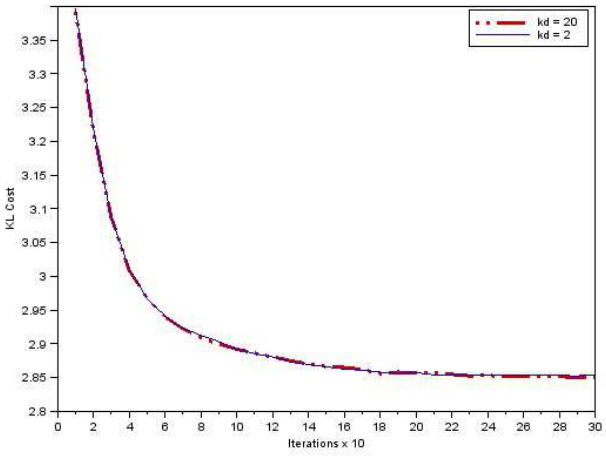
Varying values of α were also tested for $\alpha = 2$ and $\alpha = 20$ using $\tau_o = 2$, $k_d = 2$ and $\beta_o = 16$ pixels per run. Figure 3(b) shows that the results were also very similar. However, the results were different when different values of β_o were tested for $\alpha = 2$, $\tau_o = 2$ and $k_d = 2$. Figures 4(a) and 4(b) show that increasing β_o caused the coverage level provided by the agents to shrink. Also, using a higher β_o resulted in the robotic agents exploring more of the environment compared to using a lower β_o. It can also be observed in both Figure 4(a) and 4(b) that the robotic agents' motion were of the ballistic type when not in the vicinity of the food source.

By comparing both values of β_o using the Kullback Leiber cost function, it can be seen that there is a difference in coverage when using $\beta_o = 16$ and $\beta_o = 64$. The reason why the bacterium model parameters of k_d and α did not have any effects on the bacterium-flocking model is because they are not directly related to the flocking velocity of the agent as depicted in Equation 10 whereas the parameter β_o is.

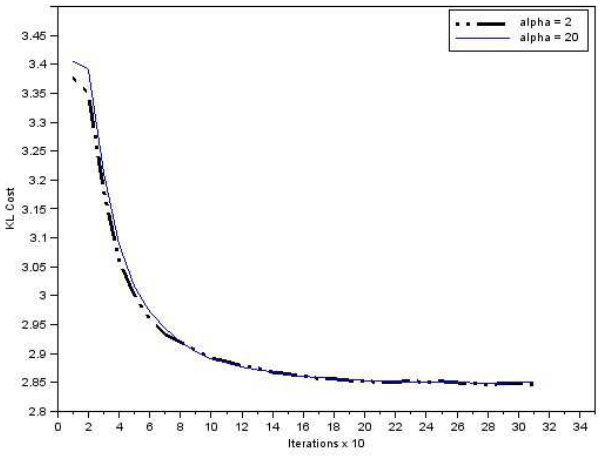
3.3 Testing on the complex pollutant profiles

After the experiments in the last section above, the algorithms were tested in more challenging environments containing noisy profiles. Various profiles were used in order to ascertain that the algorithms were capable of converging to various profiles regardless of their shape. The agents were placed at the position $(x, y) = (50, 50)$. Due to the inherent nature of the Voronoi and gradient-ascent flocking algorithms, both did not work. This is because both rely on gradients or contact with the pollutant which was not present at their start location. As a result, the agents using these algorithms were stuck at their start positions.

Following these observations, experiments were conducted for the parameters of k_d = 2, $\tau = 2$, α = 2, β_o = 32 for the bacterium-flocking algorithm. The parameter values for the bacterium behaviour were chosen because as discussed

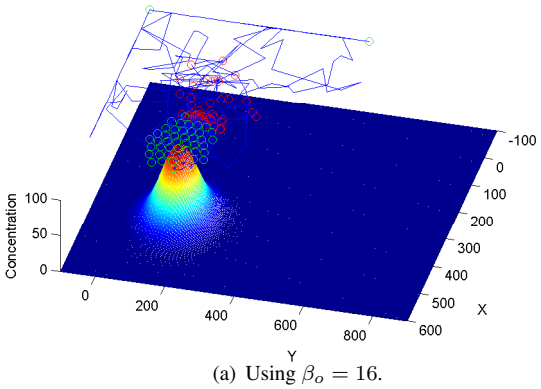


(a) For $\alpha = 2$, $\tau_o = 2$ and $\beta_o = 16$ pixels per run.

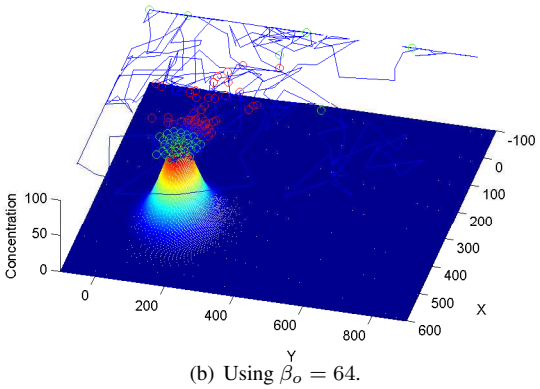


(b) For $k_d = 2$, $\tau_o = 2$ and $\beta_o = 16$ pixels per run.

Figure 3: Showing how k_d in Fig. (a) and α in Fig. (b) affects the Kullback Leiber cost function respectively. The x-axis is the number of iterations with 3000 corresponding to just over 2 minutes of simulation time.



(a) Using $\beta_o = 16$.



(b) Using $\beta_o = 64$.

Figure 4: Showing how β_o affects flock coverage when using $k_d = 2$, $\tau_o = 2$ and $\alpha = 2$. The "Red" dots are the starting point of the agents while "green" dots are the ending positions. The blue lines are the trajectories of the agents.

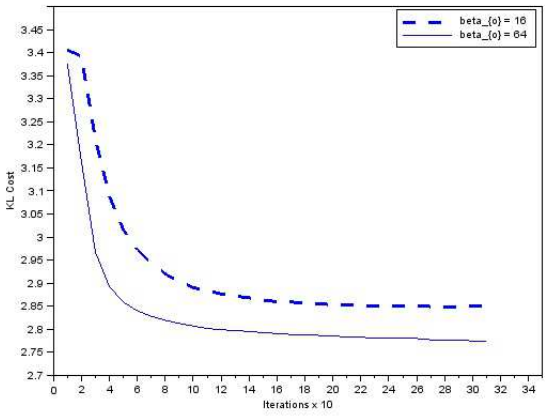


Figure 5: Showing how β_o affects the Kullback Leiber cost function for $k_d = 2$, $\tau_o = 2$ and $\alpha = 2$. The x-axis is the number of iterations with 3000 corresponding to just over 2 minutes of simulation time.

in the previous section, these values did not have effect on the coverage provided by the swarm. It was only the β_o parameter that had an effect. It is seen that the approach is able to form the shape of the presented pollutant profile even when noise is present. The noise in the Figures 6(b), 6(d) and 6(f) occur as a result of the randomly placed particles in the environment.

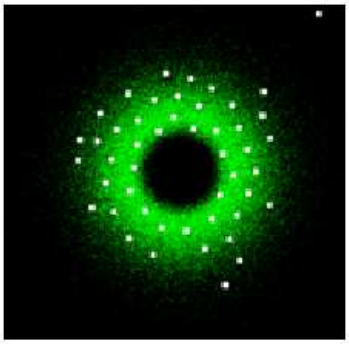
As discussed in the previous section, a lower velocity would result in a greater coverage area by the agents. This however comes at the expense of not prominently showing the simulated pollutant source. With lower velocities, the agents move slower as a result of Equation 9 and hence provide coverage to more polluted areas. With high velocity values, they would move faster and hence have the opportunity to escape less polluted areas towards more densely polluted areas.

4 Discussion

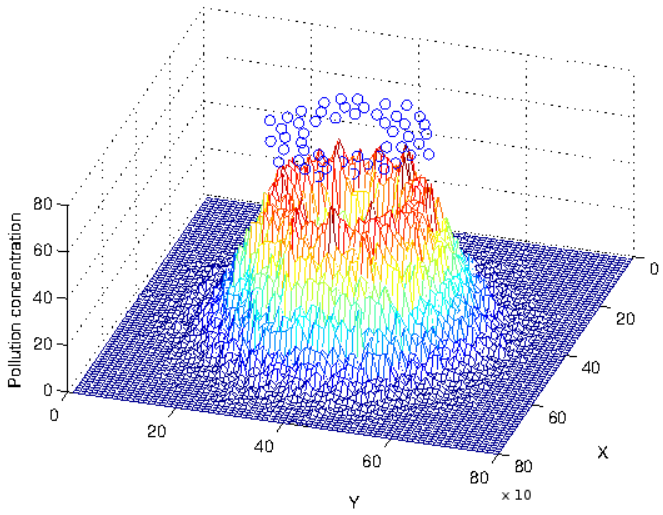
We have presented a novel algorithm for finding the source of pollutants or functions of interest in the environment. The bacterium model was used to create a composite search strategy for a robotic agent. This model uses a uniform angle distribution and a combination of run and tumble phases in finding pollutants in the environment.

The Berg-Brown bacterium model (Brown & Berg, 1974) can be tuned to achieve faster convergence at the source or greater exploration. The use of run and tumble phases to simulate foraging bacterium motion has been discussed, modeled and implemented in simulations by many researchers (Marques et al., 2002; Dhariwal et al., 2004; Wang et al., 2011). However, this is the first time a mathematical model has been converted directly into a robotic controller for the purposes of visualizing invisible hazardous pollutants. The behaviour of the composite search strategy controller that we developed from the Berg and Brown model is similar to what a forager does. It explores the environment and when it encounters a food patch, it lingers there until it is either satisfied or the entire food consumed. Using the "satiety" state of an organism to switch between a exploratory or extensive search mode and an exploitative or intensive search mode is supported by work in Barto et al. (2013). At a population level, individuals of our robotic swarm got closer to each other in more concentrated areas of the pollutant. This was independently observed as well in the experiments conducted by Berdahl et al. (2013) on golden shiners. In their experiments, they observed that golden shiners reduced by velocity and got closer together in darker areas than lighter areas.

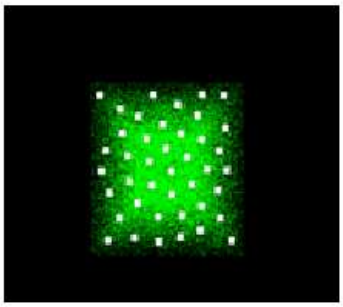
By combining the Berg-Brown bacterium model with a morse flocking model in 3.1, it was shown that the combination could be used to cover and subsequently show the distributions of pollutants in the environment. This combination was compared to the well established Voronoi partition method and a developed gradient-ascent-flocking algorithm. The gradient-ascent-flocking algorithm was developed by replacing the Berg-Brown bacterium model with a simple gradient ascent algorithm. Results of comparisons show that the performance of each algorithm varies depending on the pollutant profile. If the spread of the pollutant is small, then the Voronoi and bacterium-flocking algorithm performs coverage better than the gradient-ascent-flocking algorithm. This could be because of their ability to adapt based upon the structure of the pollutant in the environment.So when the pollutant spread was large, both algorithms spread



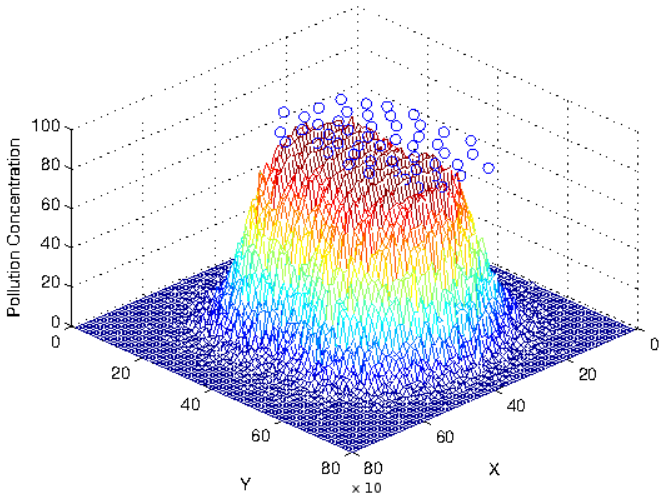
(a)



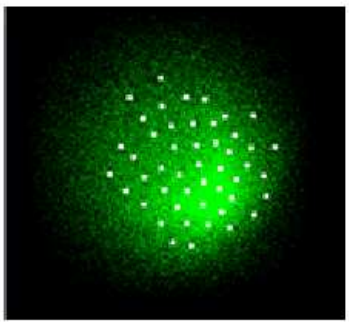
(b)



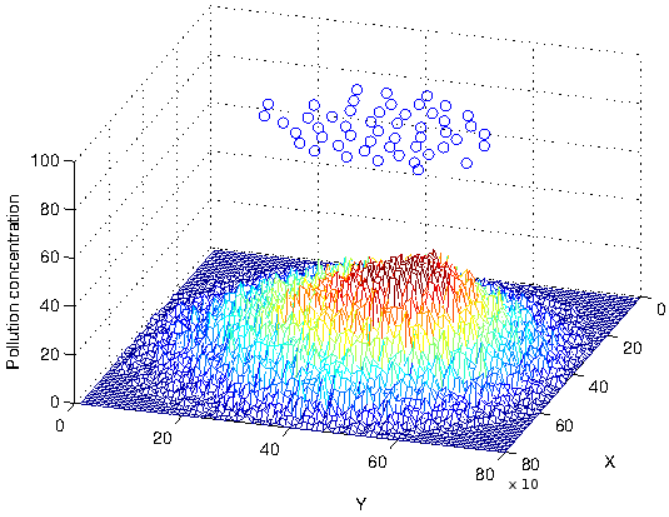
(c)



(d)



(e)



(f)

Figure 6: Showing how robots are distributed in various simulated pollutant profiles:Doughnut, Square and Skewed Gaussian functions respectively using $\beta_o = 32$. The left panels show the agents in the pollutants, while the right panels show a plot of how noisy the pollutants are. The right panels plots were obtained using Matlab's Surf function.

out their agents in order to achieve optimal coverage and vice versa unlike the gradient-ascent-flocking algorithm. Furthermore, the Voronoi has the longest time to convergence every time.

Even though the gradient-ascent-flocking and the Voronoi partition algorithms out perform the Berg-Brown bacterium model in some cases, it should be noted that the assumptions that both of them rely on (i.e knowledge of the pollutant profile in terms of its gradient or being in contact with it) would need some modification in order to use them on robots. For both the gradient-ascent-flocking algorithm and Voronoi partition algorithm, machine learning might be necessary in order to estimate the profile of the pollutant. This would further slow down the Voronoi partition algorithm and cause longer convergence times for the gradient-ascent-flocking algorithm. On the other hand, this is not required by the Berg-Brown bacterium model. This model has the ability to explore and exploit its environment in addition to dealing with a patchy environment using its composite search strategy. Additionally, it should be noted that the Voronoi partition algorithm and the gradient ascent function do not have an exploration function. This is the reason why Schwager et al. (2008) combined the Voronoi partition algorithm with a ladybug algorithm. Furthermore, the gradient-ascent-flocking algorithm can be used more effectively if the pollutant distribution is known and continuous. If however, the pollutant distribution is not known and patchy, then the bacterium-flocking algorithm is best fitted for this due to its explorative component. In addition, the presence of the individual Berg-Brown model parameters makes it possible to study the effects of individual heterogeneity on the population as a whole in future.

In 3.2, we study the effects of the bacterium model parameters on the bacterium-flocking model. It was discovered that the velocity parameter β_o had the most effect on the population as a whole. This is because this parameter was closely linked to the flocking model unlike other parameters of k_d, α and τ_o. Stafford et al. (2011) mentioned that individual parameters do not have an overall effect on the population. This could be because the parameters studied might not have a direct relationship with the aggregation link between the individuals. For example, bacterium are known to form aggregates through the use of quorum sensing during food scarcity. A bacterium secretes a chemical and this is detected chemotactically by other bacteria. They tend to navigate up the chemical field to form aggregates. Because the group mechanism has a direct link with each individual's chemotactic ability (or parameters), it might be possible to control the formation of aggregate structures through amplifying or inhibiting this mechansim. This could be investigated in future work. Another reason that the individual bacterium parameters did not have an effect on the population could be explained by the results of Berdahl et al. (2013). They observed in golden shiners that attraction to other members of the school was affected more by the number of school members than by environmental gradients. In other words, social cues had a stronger influence on the school structure than environmental cues. As a result, it is quite possible that the flocking model was overpowering the chemotactic components of each individual agent in the population. This will be studied in detail in the future as well.

In 3.3, we showed that the bacterium-flocking algorithm can be used to explore the environment to find pollutant profiles with complex profiles. We showed that the distribution of the agents is dependent on the profile of the pollutants. This shows that the bacterium-flocking algorithm can be used to visually represent various previously unknown spatial functions located in the environment. However, in real life, a pollutant profile is often not static but dynamic. This is especially true if the pollutant has a high diffusion rate. Nevertheless, provided that the speed of the agents are faster than that of the pollutant's rate of change, it is possible to use our composite search strategy model to track the spatiotemporal function.

References

Adler, J. (1975), 'Chemotaxis in Bacteria Motile Escherichia coli migrate in bands that are influenced by oxygen and nutrients', *Science* **153**, 708–715.

Arkin, R. C. (1998), *Behaviour-based Robotics*, The MIT Press, Cambridge, Massachusetts, London, England.

Barto, A., Kamil, A. & Hovestadt, T. (2013), 'Prey density, value, and spatial distribution affect the efficiency of area-concentrated search.', *Journal of theoretical biology* **316**, 61–69.

Baxter, P. J., Kapila, M. & Mfonfu, D. (1989), 'Lake Nyos disaster, Cameroon, 1986: the medical effects of large scale emission of carbon dioxide?', *BMJ (Clinical research ed.)* **298**(6685), 1437–1441.

Bell, W. J. (1990), 'Searching behavior patterns in insects', *Annual review of Entomology* **35**, 447–467.

Ben-Jacob, E. (2003), 'Bacterial self–organization: co-enhancement of complexification and adaptability in a dynamic environment', *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* **361**(1807), 1283–1312.

Benhamou, S. (2007), 'How Many Animals Really Do the Lévy Walk ?', *Ecological Society of America* **88**(8), 1962–1969.

Berdahl, A., Torney, C. J., Ioannou, C. C., Faria, J. J. & Couzin, I. D. (2013), 'Emergent sensing of complex environments by mobile animal groups.', *Science* **339**(6119), 574–576.

Bonabeau, E., Dorigo, M. & Theraulaz, G. (1999), *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press.

Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., Franks, N. R., Rafelsberger, O., Joly, J.-L. & Blanco, S. (1997), The emergence of pillars, walls, and royal chambers in termite nests, Working papers, Santa Fe Institute.

Brown, D. A. & Berg, H. C. (1974), 'Temporal stimulation of chemotaxis in Escherichia coli', *Proceedings of the National Academy of Science U.S.A* **71**, 1388–1392.

Correll, N. & Martinoli, A. (2009), 'Robust Distributed Coverage using a Swarm of Miniature Robots', *Control* **54**, 127–136.

Cortes, J., Martinez, S., Karatas, T. & Bullo, F. (2004), 'Coverage control for mobile sensing networks', *IEEE Transactions on Robotics and Automation* **20**(2), 243–255.

Dauer, L. T., Zanzonico, P., Tuttle, R. M., Quinn, D. M. & Strauss, H. W. (2011), 'The Japanese tsunami and resulting nuclear emergency at the Fukushima Daiichi power facility: technical, radiologic, and response perspectives.', *Journal of nuclear medicine : official publication, Society of Nuclear Medicine* **52**(9), 1423–1432.

Dhariwal, A., Sukhatme, G. S. & Requicha, A. A. G. (2004), 'Bacterium-inspired robots for environmental monitoring', *Proceedings of IEEE International Conference on Robotics and Automation, New Orleans, LA, April* pp. 1436–1443.

Dusenbery, D. B. (1998), 'Spatial sensing of stimulus gradients can be superior to temporal sensing for free-swimming bacteria', *Biophysical Journal* **74**, 2272–2277.

Hinkelman, T. M., Nolting, B. C., Brassil, C. E. & Tenhumberg, B. (2013), 'Composite random search strategies based on non-proximate sensory cues', *Preprint submitted to Ecological Modelling* .

Jackson, G. A. (1987), 'Simulating chemosensory responses of marine microorganisms', *American Society of Limnology and Oceanography* **32**(6), 1253–1266.

Johnson, B. R. (2009), 'Pattern formation on the combs of honeybees: increasing fitness by coupling self-organization with templates.', *Proceedings of the Royal Society-Biological sciences* **276**(1655), 255–261.

Jost, C., Verret, J., Casellas, E., Gautrais, J., Challet, M., Lluc, J., Blanco, S., Clifton, M. J. & Theraulaz, G. (2007), 'The interplay between a self-organized process and an environmental template: corpse clustering under the influence of air currents in ants.', *Journal of the Royal Society- Interface* **4**(12), 107–116.

Kang, H. K. & Bullman, T. A. (1996), 'Mortality among U.S. veterans of the Persian Gulf War.', *The New England journal of medicine* **335**(20), 1498–504.

Klaassen, R. H. G. & Nolet, B. A. (2006), 'Movement of foraging tundra swans explained by spatial pattern in cryptic food densities', *Ecology* pp. 43–64.

Kling, G. W. (1987), 'Seasonal mixing and catastrophic degassing in tropical lakes, cameroon, west africa', *Science* **237**(4818), 1022–1024.

Kullback, S., Burnham, K. P., Laubscher, N. F., Dallal, G. E., Wilkinson, L., Morrison, D. F., Loyer, M. W. & Eisenberg, B. e. a. (1987), 'Letter to the editor: The kullback leibler distance', *The American Statistician* **41**, 340–341.

Kushleyev, A., Mellinger, D., Powers, C. & Kumar, V. (2013), 'Towards a swarm of agile micro quadrotors', *Autonomous Robots* **35**(4), 287–300.

Kwok, A. & Martínez, S. (2011), 'A distributed deterministic annealing algorithm for limited-range sensor coverage', *Control Systems Technology, IEEE Transactions on* **19**(4), 792–804.

Liu, Y. & Passino, K. M. (2004), 'Stable social foraging swarms in a noisy environment', *IEEE Transactions on Automtic Control* **49**, 30–43.

Marques, L., Nunes, U. & Almeida, T. D. (2002), 'Olfaction-based mobile robot navigation ', *Thin Solid Films* **418**(1), 51–58.

Marrocco, A., Henry, H., Holland, I. B., Plapp, M., Séror, S. J. & Perthame, B. (2010), 'Models of Self-Organizing Bacterial Communities and Comparisons with Experimental Observations', *Mathematical modeling of Natural Phenomenon* **5**(1), 148–162.

Martel, S. & Mohammadi, M. (2009), 'Using a Swarm of Self-propelled Natural Microrobots in the Form of Flagellated Bacteria to Perform Complex Micro-assembly Tasks', *Proceedings of the International conference on Robotics and Automation (ICRA), 2010* pp. 500–505.

Matanle, P. (2011), 'The Great East Japan Earthquake , tsunami , and nuclear meltdown : towards the (re) construction of a safe , sustainable , and compassionate society in Japan ' s shrinking regions', *Local Environment: The International Journal of Justice and Sustainability* **16**(9), 823–847.

Oyekan, J., Dongbing, G. & Hu, H. (2012), Hazardous substance source seeking in a diffusion based noisy environment, *in* 'Proceedings of the IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5-8 August 2012.', pp. 708 – 713.

Passino, K. M. (2002), 'Biomimicry of bacterial foraging for distributed optimization and control', *IEEE Control Systems Magazine* **22**(3), 52–67.

Plank, M. J. & James, a. (2008), 'Optimal foraging: Lévy pattern or process?', *Journal of the Royal Society, Interface / the Royal Society* **5**(26), 1077–1086.

Reynolds, A. (2009), 'Adaptive Lévy walks can outperform composite Brownian walks in non-destructive random searching scenarios', *Physica A: Statistical Mechanics and its Applications* **388**(5), 561–564.

Reynolds, R., Peng, B. & Whallon, R. (2005), 'Emergent social structures in cultural algorithms', *Annual Conference of the North American Association for Computational Social and Organizational Science (NAACSOS 2005)* pp. 26–28.

Schmid, M., Halbwachs, M., Wehrli, B. & Wüest, A. (2005), 'Weak mixing in Lake Kivu: New insights indicate increasing risk of uncontrolled gas eruption', *Geochemistry, Geophysics, Geosystems* **6**(7), 1–11.

Schwager, M., Bullo, F., Skelly, D. & Rus, D. (2008), 'A ladybug exploration strategy for distributed adaptive coverage control', *in Proceedings of International Conference on Robotics an Automation, Pasadena, CA, May 2008* pp. 2346–2353.

Schwager, M., Mclurkin, J., Slotine, J.-J. E. & Rus, D. (2009), 'From Theory to Practice: Distributed Coverage Control Experiments with Groups of Robots', *Springer Tracts in Advanced Robotics* **54**(1), 127–136.

Schwager, M., Slotine, J.-J. & Rus, D. (2007), 'Decentralized, Adaptive Control for Coverage with Networked Robots', *Proceedings of the IEEE International Conference on Robotics and Automation* pp. 3289–3294.

Segall, J., Block, S. & Berg, H. (1986), 'Temporal comparisons in bacterial chemotaxis', *Proceedings of the National Academy of Science U. S. A.* **83**(23), 8987–8991.

Shapiro, J. A. (1998), 'Thinking about bacterial populations as multicellular organisms.', *Annual review of microbiology* **52**, 81–104.

Shklarsh, A., Ariel, G., Schneidman, E. & Ben-Jacob, E. (2011), 'Smart swarms of bacteria-inspired agents with performance adaptable interactions.', *PLoS computational biology* **7**(9), e1002177.

Shklarsh, A., Finkelshtein, A., Ariel, G., Kalisman, O., Ingham, C. & Ben-Jacob, E. (2012), 'Collective navigation of cargo-carrying swarms', *Interface Focus* **2**(6), 786–798.

Shucker, B., Murphey, T. & Bennett, J. K. (2006), 'An Approach to Switching Control Beyond Nearest Neighbor Rules', *Proceedings of the American Control Conference, Minneapolis, Minnesota, USA, 14-16 June, 2006* .

Shucker, B., Murphey, T. & Bennett, J. K. (2008), 'Convergence Preserving Switching for Topology Dependent Decentralized Systems', *IEEE Transactions on Robotics* **24**(6), 1–11.

Simons, A. M. (2004), 'Many wrongs: the advantage of group navigation.', *Trends in ecology & evolution* **19**(9), 453–455.

Smith, J. A. & Martin, A. M. (2009), 'Comparison of Hard-Core and Soft-Core Potentials for Modelling Flocking in Free Space', *Cornell University Library* pp. 1 – 9.

Stafford, R., Williams, G. a. & Davies, M. S. (2011), 'Robustness of self-organised systems to changes in behaviour: an example from real and simulated self-organised snail aggregations.', *PloS one* **6**(7), e22743.

Torney, C., Neufeld, Z. & Couzin, I. D. (2009), 'Context-dependent interaction leads to emergent search behavior in social aggregates.', *Proceedings of the National Academy of Sciences of the United States of America* **106**(52), 22055–22060.

Wang, C. C., Ng, K.-L., Chen, Y.-C., Sheu, P. C. & Tsai, J. J. (2011), 'Simulation of Bacterial Chemotaxis by the Random Run and Tumble Model', *Proceedings of the IEEE International Conference on Bioinformatics and Bioengineering* pp. 228–233.

Wang, Z. & Gu, D. (2007), 'Behaviour Based Fuzzy Flocking Systems', *Proceedings of the IEEE International Fuzzy Systems Conference, London, July, 2007* pp. 1098–2004.