SURVEY PAPER

Biologically-inspired behaviour based robotics for making invisible pollution visible: a survey

John Oyekan\textsuperscript{a} and Huosheng Hu\textsuperscript{b}

\textsuperscript{a}Centre for Research into Distributed Technologies, University of Bedfordshire, Luton LU1 3JU, UK; \textsuperscript{b} School of Computer Science \& Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK

(Received 26 July 2013; revised 18 November 2013; accepted 20 November 2013)

Behaviour-based robotics is a paradigm that was proposed by Rodney Brooks in 1986. This paradigm proposes that robotic solutions should be developed by combining various reactive behaviours in an architecture. This survey overviews various biologically inspired source-seeking and multi-agent algorithms that could be combined in a behavior-based architecture towards providing visual concentration or profile information of invisible hazardous pollutants in dangerous environments. The need for this becomes even more necessary when hazardous substances that cannot be observed by human eyes are present in the environment. An example of such a scenario was the nuclear disaster caused by a Tsunami in Japan in 2011. Causalities from this disaster event could have been greatly reduced if the concentration profile of the resulting invisible pollution and radiation was made visible by using a swarm of flying microrobots.

Keywords: robotics for safety; security and dependability; swarm intelligence / collective intelligence; robot cooperation and coordination; biologically-inspired robots and systems; environmental intelligence

1. Introduction

Traditionally, the functional or sense-plan-move-based approach has been used to control robots to achieve some autonomous behaviours. In this approach, the models of the world in which the robots reside are constructed and programmed into the robot’s software. However, since the world is complex with dynamic objects, the amount of models needed to describe the world increases and so does the memory needed to store these models. It is thus a big challenge for engineers to programme the robot’s response to every single event that could happen in the robot’s world. This led to the ‘symbolic grounding problem’, i.e. the difficulty in attaching the full attributes, features and consequence of interacting with a sensed object symbol in the environment.\cite{1} Another problem was that a breakdown of one of the modules in the structure immediately caused the robot to stop functioning.

Behaviour-based approach proposed by Brooks in 1986 aims to get rid of the need of symbols to represent objects in the environment and resulted in an agent that gracefully deteriorates if one or more of its modules fail. It is a parallel structure in which each layer was a reactive control layer working in parallel with others and having direct connection between sensors and robotic actuators. In this approach, each control layer was a behaviour capable of responding to the occurrence of an event in the environment. By choosing the right set of behaviours, solutions are provided to the problems that the robotic agent might encounter whilst operating in the environment. One of the highest selling point of Brooks’ technique is that, because of the close coupling between sensors and actuators, the robotic agent could deal with and respond immediately to an unpredictable, highly dynamic and hazardous world.

This was proven previously in 1984 by Bratenberg in his thoughts experiments in which vehicles equipped with sensors and connected directly to motors were used to develop agents that exhibited seemingly complex behaviours such as fear, hate, love and so on. His research was further expanded by Hogg et al. in experiments conducted in the MIT computational labs.\cite{2} Another advantage of using a behaviour-based approach includes the ability to add necessary behaviours in a bottom-up approach in order to perform the required task. Such behavioural architecture enables developers to add future behaviours and capabilities in order to expand the operating environment of the agent. This capability could be swimming as in the case of a robotic fish or flying as in the case of an Unmanned Aerial Vehicle.

There are various ways of choosing behaviours when performing behaviour-based robotics. One of the ways is by using an ethological approach in which models of animal behaviour are chosen from the biological field and then implemented on robots. This approach was followed by Saito et al. in developing their robotic snake.\cite{3} Hu et al. also studied the motion of fishes and used the knowledge to
develop robotic fishes that are capable of swimming in water. An advantage of using fish as an inspiration for robotic development is that the robot would have the advantage of energy efficiency and reduction in noise when compared to propeller driven AUVs.[4–6]

This survey is unlikely previous ones in literature. Previous survey such as in [7–10] focus on either chemical sensing or odour source localisation using single or multiple robots. However, this survey investigates how a swarm of robots can be used to form a visual distribution of invisible pollution present in a hazardous environment using the behaviour-based robotics paradigm. The goal is that the distribution of robots would be such that their numbers in various locations are directly proportional to the pollution concentration at the locations. Such information would enable people to keep away from contaminated areas including enabling rescuers to prioritise rescue depending on severity of contamination. In order to achieve this in the behaviour-based paradigm, a source-seeking algorithm is required to explore and find the sources of pollution in the environment, and at the same time, multi-agent control is necessary in order to make the robots act as a group towards achieving the previously stated goal. By looking at a swarm of robots in formation, people could see clearly where is safer and then escape from danger. The use of a swarm of robots in disaster scenarios (called Chemical, Biological, Radiological, Nuclear and Explosive (CBRNE) incident response [11]) is an area of research that is increasingly being investigated as in [12]. In [12], an investigation into how a swarm of robots could be used to aid fire fighters in searching a large area was carried out as part of the Guardians project.

The rest of the paper is organised as follows. Section 2 presents a comprehensive literature review of potential biologically inspired source-seeking behaviours or algorithms that can be used in the stated task while Section 3 discusses biologically-inspired flocking and swarm algorithms. Providing a visual representation of an invisible pollutant could be viewed as a coverage task. As a result, for the purpose of completeness, we discuss present state of the art coverage algorithms in Section 4. Finally, a brief conclusion is given in Section 5.

2. Source-seeking algorithms

2.1. Spiral-Surge algorithm

Hayes et al. used biological inspiration from the moth in developing their spiral surge algorithm.[13,14] Their approach was developed with the aim of making it possible to detect and follow plumes to their sources in very turbulent environments.

In turbulent environments, plumes have the tendency to break into smaller packets. This makes it very difficult to use gradient-based methods such as bacterial chemotaxis behaviour to transverse the plume and find its source. To solve this problem, they divided the task into three subtasks: plume finding, plume traversal and plume source declaration. In finding the plume, they start with a spiral that has a large diameter depending on the prior information. If the plume is known to be in the immediate area, then a smaller spiral gap would be needed. However, if the plume is not in the immediate area then a larger spiral gap is used.

For the plume traversal sub task, whenever a plume patch is encountered, the agent moves upwind for a set distance called StepSize (Because it can be assumed that the source would be upwind from the plume patch). If another plume patch is detected while moving upwind, the distance moved is reset and the agent moves upwind again for another StepSize value. After the StepSize has been moved, if it does not encounter another plume hit for a set time called the CastTime, it reverts back to the plume finding behaviour but with a smaller Spiral gap value. In their experiments, they concentrated on plume traversal. Their algorithm uses a single chemical sensor that returns a binary information. If the plume is detected, a one (1) is returned or zero (0) otherwise. In their experiments, they found out that using more than one sensor was not really practical.

2.2. Moth inspired casting algorithm

The Moth Inspired Casting algorithm is an algorithm inspired by the way male moths find females for mating. They do this by facing the direction the wind is blowing the female pheromone from and then perform a zigzag flying maneuver. If the plume is detected, the moth stays in the plume and keeps zigzagging in it. If the moth detects that it is out of the plume, it turns through an angle and flies back in the opposite direction into the plume. It keeps doing this until it detects the female. This algorithm requires a method of detecting wind direction and is made up of the three subtasks of plume finding, plume traversal and plume source declaration.

The moth inspired technique was followed by Farrell et al. using a behaviour-based approach to separate the three subtasks of plume finding, plume traversal and plume source declaration into behaviours.[10] Their robot is programmed to carry out a wide side to side search for a plume.

Once detected, the robotic agent reduces its cross sectional search width to the width of the plume. Whenever the robot goes out of the plume, it turns to go back into the plume. This behaviour is continued against the flow direction of the medium until the source is found. In [15], Pang and Farrell used the plume tracing approach just described in combination with a Bayesian inference source-likelihood mapping approach to estimate the location of the source of the pollutant. Using this approach could potentially help reduce the time spent searching for the source of the plume. In [16], Li et al. decomposed the task of finding the source of the pollution into four subtasks instead
of three. These subtasks are: plume finding, plume maintaining, plume reacquisition and source found declaration.

Li et al. [17] went on to test their approach on an Autonomous Underwater Vehicle in a near shore ocean condition of 250–300 m along-shore and 100 m cross-shore. Their approach was the first known experiment to be conducted in a real-world environment with the Autonomous Underwater Vehicle, tracking a plume successfully over 100 m. They used a behaviour-based adaptive mission planner based upon a subsumption behaviour architecture in their work. Their approach was accurate in declaring the source to a range of tens of meters due to issues in coordinate resolving. Lochmater et al. compared the Surge-Spiral algorithm in Section 2.1 and the casting algorithm presented in this section and came to a conclusion that the surge-spiral algorithm performed faster and was more robust in locating a source in a low-speed, laminar flow environment.[18]

2.3. Crab inspired algorithm
Zimmer-faust et al. discussed how blue crabs forage for food in a flow environment of an estuarine tidal creek. [19,20] In this environment, the flow is not as turbulent as an open air environment that flying insects forage in. Flying insects have to rely heavily on casting especially when the food ‘signal’ is lost, whereas blue crabs use a sort of binary mechanism to find their food sources in the estuarine environment. The crab moves laterally so that it remains in the plume generated by the food source. However, if a crab has a set of its legs outside the plume and one part of its body still in the plume, it immediately changes direction to the part of its body with the highest concentration of the food source. This type of foraging behaviour is called Tropotaxis.

Grasso and Ayers et al. [21,22] used the Tropotaxis principle of the blue crab by using two sensors on their Robotic lobster as shown in Figure 1. They were able to localise the source of a plume using this approach. However, the structure of their vehicle might result in a consumption of a lot of energy during maneuvering and is only limited to the river bed. Nevertheless, by replacing their vehicle with one similar to a fish, as in [15], the power required to maneuver might be reduced.

2.4. Braitenberg vehicle algorithm
Lilienthal and Duckett used Braitenberg Vehicle structures to find the source of a simulated environmental variable.[23] Ethanol was used in their experiments. They were able to prove that it was possible to use the Braitenberg Vehicle’s simple localisation strategy in locating a source. The Braitenberg Vehicle was discovered and developed by Valentino Braitenberg during his thought experiments. The Vehicles use a direct sensor to motor coupling relationship as shown in Figure 2.

The velocity of each wheel $v$ is related to the sensor reading $x$, so that an increase in sensor reading $x$ reduces the velocity of the connected wheel. As a result, whenever a vehicle with the structure of Figure 2(a) is placed in an
environment containing an environmental variable, the vehicle turns towards it because the measurement reading of the sensor closest to the environmental variable reads maximum. This reduces the velocity of the connected wheel, hence turning the vehicle towards the environmental variable. This behaviour was called ‘permanent love’ by Valentino.

However, if the sensor to motor coupling were changed as in Figure 2(b), the vehicle would always turn away from the environmental variable resulting in an exploration of other areas in the environment. If there was another source in the sensors effective range, the vehicle would move away from the present source towards the other source. This behaviour is called ‘exploring love’. By using these behaviours, Lilienthal et al. investigated how to develop a test bed for comparing the strategies performance directly and then statistically evaluated to find the strategy which had the best performance. In addition, they investigated which strategy would perform best in an uncontrolled environment having various local maxima. They also investigated a random search algorithm’s performance in their test bed.

Their results pointed to the conclusion that using the ‘permanent love’ vehicle structure reduced the average path length covered by the robot (measured from the beginning of the experiment to the source of the environmental variable) by half when compared to the random search method. As a result, the ‘permanent love’ vehicle structure’s performance was better.

The ‘exploring love’ however required a longer average path length to localise at the source. This is because of its nature of turning away from the local maxima each time it discovers it. Nevertheless, it was able to explore its environment thoroughly. With the ‘exploring love’ structure, a source can be recognised as an area that has the less density of robot path coverage. In addition, the robot did not get trapped in a local maxima during its search for the source of an environmental variable. Local maxima are common for gaseous environmental variables especially in environments that have a strong turbulence. Using the ‘exploring love’ vehicle structure made sure that a vehicle did not get too close to the source. A vehicle getting too close to the source would result, in it being contaminated and becoming a source itself.

2.5. Bacteria chemotaxis inspired algorithm

Having studied the motions of bacteria in [24], Berg and Brown were able to mathematically describe the motion of a single bacterium using Equations (1), (2) and (3) in [24].

\[
\tau = \tau_o \exp \left( \alpha \frac{dP_b}{dt} \right) \tag{1}
\]

\[
\frac{dP_b}{dt} = \frac{1}{\tau_m} \int_{-\infty}^{t} \frac{dP^b}{dr} \exp \left( \frac{(r - t)}{\tau_m} \right) dr, \tag{2}
\]

\[
\frac{dP^b}{dt} = \frac{k_d}{(k_d + C)^2} \frac{dC}{dt} \tag{3}
\]

where \(\tau\) is a time constant dependent on the bacterial system or type of bacterium, \(\tau_o\) is the mean run length in the absence of a concentration gradient, \(\alpha\) is a constant of the system based on the chemotaxis sensitivity factor of the bacteria and \(P_b\) is the fraction of the receptor bound at concentration \(C\). \(k_d\) is the dissociation constant of the bacterial chemoreceptor. \(\frac{dP_b}{dt}\) is the rate of change of \(P_b\). While \(\frac{dP^b}{dt}\) is the weighted rate of change of \(P^b\). This model has been implemented in simulations by Jackson et al. [25] with success and it is believed that it can be easily adapted into a controller for a robotic agent.

According to Dahlquist et al., during the tumble phase, the random direction \(\sigma\) chosen is governed by a probability distribution which makes the probability of turning either right or left azimuthally symmetric about the previous direction.[26] Muller et al. [27] model this in their experiments as a Gaussian distribution for both the right and left directions. Furthermore, in the Dahlquist et al. model, the velocity \(\nu\) is assumed to be constant.

Biologists have conducted simulations based upon the Berg and Brown model in studying the effects of various environmental conditions on the bacterial system. Jackson for example conducted investigations on how changes in the bacterial system parameters affect bacteria motion.[25,28] It was concluded that the optimal parameters of the bacterial system was dependent on the velocity at which the agent was traveling. So far, no researcher has investigated the possibility of implementing the Berg and Brown model on a robotic agent. Most researchers have used if – else rules as in [29,30] to direct agents towards a chemical source. However, using if – else rules makes it difficult to analyse the robotic agent’s behaviour and compare it with its biological counterpart.

Some researchers have argued that the bacterium-inspired behaviour fails in environments with high Reynold numbers, i.e. a highly turbulent environment.[14,15] Nevertheless, it has been observed that marine bacteria are able to still find their food sources even in high turbulent environments. They do this, by using a zig-zag motion like modification to their behaviour. This behaviour becomes a control function that makes it possible for the agents to stay in a plume produced by a food source.[31] This is further supported by Barbara and Mitchell [32] where they showed that the frequency of turns and tumbles depended on the concentration gradient in the environment.

Many researchers have described the chemotactic response of a population of agents in response to spatiotemporal chemical fields over a long time spanning four or more decades. Researchers describe the motion of the population as macroscopic probability density functions comprising of a diffusion term (to simulate random walk) and a drift term (to simulate chemotactic response to the chemical field) using various models ([33] and references there in). These two basic ingredients have been modelled in various ways for various purposes by researchers over the years.
Researchers with a robotic bias have used the mathematical macroscopic equations (4) and (5) derived by Keller-Segel [34] in various forms to describe the evolution of the population density of the agents in response to spatiotemporal chemical fields.[35,36]

\[ \frac{\partial b}{\partial t} = \nabla . (\mu (C) \nabla b) - \nabla . (\chi (C) b \nabla C) + g(b, C) - h(b, C) \]  

(4)

\[ \frac{\partial C}{\partial t} = D \nabla^2 C - f(b, C) \]  

(5)

where \( b = b(x, t) \) is the density of bacterial population, \( C = C(x, t) \) is the attractant concentration at location \( x \) at time \( t \) and \( \chi (C) \) is the chemotactic coefficient. The chemotactic coefficient describes the rate of bacteria entering a location, \( \mu (C) \) is the diffusion coefficient for the bacterial population with \( g(b, C) \) and \( h(b, C) \) describing cell growth and death. Equation (5) describes the evolution of the chemical field in which the bacteria is present with \( f(b, C) \) and \( D \) representing functions describing the attractant degradation and diffusion coefficient of the attractant [37], respectively. As this Equation evolves with time, the bacteria population dynamics described by Equation (4) can be studied.

In addition, studies have shown that bacterial systems have a level of sensitivity to their environment. It has been discovered that when a bacterium is changed from an environment having a high concentration to one having a low concentration, the behaviour changes to adjust to the new one. One of the ways this is done is by adjusting the gains of their receptors to compensate for changes in environment. This makes it possible for them to amplify weak signals so that they can still navigate up the gradient of an attractant when changed from a high concentration environment to a low concentration environment. Bacterial adaption, changes in gain and sensitivity in different environments containing noise and their effects on each other are still currently being investigated extensively by researchers in the biological field.[37–40] Another way of achieving optimal foraging in the new environment is through genetic mutation of individuals. This would introduce new organisms into the genetic pool that are able to forage better than the previous ones.

In the robotic field, Passino used the chemotactic gradient climbing behaviour observed in the E.Coli bacteria to achieve Swarm aggregation behaviour of a group of bacteria.[41] The chemotactic behaviour of bacteria is a sort of behaviour in which the organism climbs up an ever increasing concentration gradient in the environment in search of more favourable food conditions. In his work, Passino used additional biological concepts of reproduction, death, elimination and dispersal. In [41], the Equation (6) was used to control the bacteria so that they move towards areas of favourable environment.

\[ \theta^i (j + 1, k, l) = \theta^i (j, k, l) + C(i) \phi (j) \]  

(6)

where \( j \) is the chemotactic index step, \( k \) is the reproduction step and \( l \) is the index for elimination-dispersal. If at \( \theta^j (j + 1, k, l) \), the cost was lower than at \( \theta^j (j, k, l) \) then the bacteria keeps moving up in that direction.

Pugh et al., also used bacterial chemotaxis behaviour in finding the source of a target. However, in addition to the bacterial chemotaxis algorithm, they also used a distributed particle swarm optimization (PSO) to adapt the parameters of the bacterial chemotaxis algorithm.

\[ v_{i,j} = w.v_{i,j} + p.w.rand() (l(x^*_{i,j} - x_{i,j}) + n.w.rand() (x^*_{i,j} - x_{i,j}); x_{i,j} = x_{i,j} + v_{i,j} \]  

(7)

Their PSO algorithm was described according to Equation (7). One particle of the swarm was allocated to a robotic agent. The particle had an initially randomly generated value, in the range of \([0.0, 1.0] \). This range was scaled to the corresponding values of the free parameters. The velocity of each agent was also randomly chosen between \([-0.5, 0.5] \) with \( p.w = n.w = 2.0 \) and \( w = 0.5 \). Each agent was allowed to run the bacterial chemotaxis algorithm in parallel with the distributed PSO. The PSO particle on each agent was evaluated by taking the value of the measured intensity after a certain time. By following this approach, free parameters that enable agents to stay around the target were favoured. During the evaluation, it was assumed that every agent was synchronised.

According to Pugh et al., the advantage of using the PSO algorithm to adapt the parameters, is that it was robust against noise. Also by giving the agents the ability to learn unsupervised and adapt their controllers using the environment, it is possible to achieve much higher system performance. This makes it possible to adapt the agents behaviour to an unknown or constantly changing environment.

During the evaluation of the particles on the agents, an average of the previously known best fitness values with the new fitness value was calculated in order to get a more accurate measure of the actual fitness. Using this approach, reduced the effects of noise on the learning process.

3. Multi-agent control algorithms

The terms of flocking and swarming are sometimes used interchangeably. Flocking is a natural phenomenon observed in nature where a large body of animals move together in unison as can be seen in the starlings of Italy, school of fish, swarm of bees, herds of cows and so on.

The benefits of flocking in nature are numerous including the chances of protection against predators, energy saving benefits such as the ‘V-flight’ formation of geese during migration and increasing the chances of finding food collectively. These benefits were realised by engineers and as such decided to realise the same benefits on robotic systems. By using a team of robots, a robotic system would be present everywhere at once. In addition, failure of one agent does not necessarily mean the end of the mission as others can
A concept that could be used to perform multi-agent control is one called Particle Swarm Optimization developed by James Kennedy [43]. This involves using a swarm of agents to find the optimal value in a multi-dimensional search space. PSO relies on the fact that every individual member in the swarm communicates to its neighbour, the best position found so far and also the best global position found by the swarm so far. To do this, swarm members are sent ‘flying’ through the search space. By doing this iteratively, the optimal position is found. This idea is taken further in [44] to control a swarm of robots towards a destination point. However, using PSO has some drawbacks, in that sending the best local and global information to all swarm members could be a communication burden, due to the amount of data that has to be transferred.

3.2. Game theory

Game theory has been used to coordinate Robotic teams in many applications such as in the Robocup rescue,[45] multi-agent tasking and [46] flocking [47,48] amongst others. It has also been used to control a heterogeneous UAV team in [49]. This theory can be used to study the individual robots best response to each other through the use of the Nash equilibrium game concept. The Nash equilibrium game concept has been used to coordinate a group of team of robots using variants of the theory.[48]

However, the use of Game theory is not without its drawbacks. As the number of robots increase, the amount of computing required to compute the equilibrium to achieve robotic coordination increases.[50] Also, the level of complexity increases with the number of robots. Kok et al. [50] used a system of assigning roles to individuals in the robotic team. Under each role is a set of actions. A role is carried out depending on the computed equilibrium in the robotic team. This makes it possible to reduce the number of actions to be computed and hence equilibrium computation to just role computation. In using this approach, a group of actions to be carried out in a role by each robot can be suppressed.

3.3. Smoothened particle hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is another approach that can be used to control a swarm of agents. This relatively new approach is based on Computational Fluid Dynamics but less demanding computationally. The approach was first proposed by Lucy in 1977. It uses a particle-based approach in which a fluid is represented by particles. Each particle is affected by other particles. The effect on each particle by other particles in the system is defined according to a weighted average dirac delta function W as shown in Equation (8).

\[ \rho_i = \sum_j \rho_j \frac{m_j}{\rho_j} W(r_i - r_j, h) \]  

where, \( \rho_i \) is a particle i’s density and \( \rho_j \) that of other particles.

The kernel \( W \) is used to compute the contribution of the other particles \( j \) in the system to the particle \( i \) property. \( h \) is the effective distance of the kernel \( W \), while \( |r_i - r_j| \) is the distance between the two particles \( i \) and \( j \). When using SPH, the various properties of the fluid such as density, viscosity, mass, etc. are defined by the user in order to get a close enough realistic simulation of the fluid. For example, if a gas is to be simulated, the fluid properties must match those of a gaseous substance.

SPH has been used by Pac and Erkmen [51] in order to control a swarm of robots for optimal coverage of an area and to perform obstacle avoidance in an area. However, computation required to use this technique might make it a disadvantage during implementation on physical agents.

3.4. Flocking algorithm

The first flocking algorithm that mimicked flocking birds was developed by Craig Reynolds after studying the flocking phenomenon of starlings.[52] He then simulated the phenomenon in a computer program in 1986. In his simulation, he used three simple rules to achieve flocking. The three simple rules were: Keep close to your neighbour (cohesion); Avoid collision with your neighbours (separation) and head in the same general direction as your neighbours (alignment). This resulted in a flocking behaviour.

Researchers have taken inspiration from his work and have developed various approaches to achieve flocking behaviour. Some have used a behaviour-based approach with fuzzy logic as in [53] in which a fuzzy logic approach was used to design the three behaviours necessary for flocking. Their behaviour coordination approach is shown in Figure 3. The dynamics of the agents were defined by Equation (9) where \( u_i \) is the force, \( p_i \) is the acceleration and \( q_i \) is the position of the agent been controlled.

\[ \dot{q}_i = p_i(t) \]
\[ \dot{p}_i = u_i(t) \]  

Figure 3. Behaviour coordination structure.[53]
In the implementation of the alignment behaviour, the
aim was to match the velocity of the agents. If the velocity
of the agents matched each other, then according to the
definition of velocity (rate of change of distance in a given
direction), the agents would be aligned in the same direction
θ_i as follows:

\[ \theta_i = \arctan \left(\frac{p_{iy}}{p_{ix}}\right) \] (10)

As a result, if there is a difference in the velocity of the
agent i and its neighbours, the alignment behaviour works
to correct it. They used an output fuzzy control function
defined in Equation (11).

\[ f_v(p_{ja} - p_{ia}) = \frac{\sum s \mu_s(p_{ja} - p_{ia})l_s}{\sum s \mu_s(p_{ja} - p_{ia})} \] (11)

In Equation (11), \(\mu_s\) is the membership function cor-
responding to the \(s\)th fuzzy input, \(l_s\) corresponds to the output of
the \(s\)th fuzzy rule. Hence, the output of the alignment
behaviour is described by Equation (12).

\[ u_{ei} = \sum_{j \in N_i} \overrightarrow{f}_v(p_j - p_i) \] (12)

where \(\overrightarrow{f}_v(p_j - p_i)\) is expressed as:

\[ \overrightarrow{f}_v(p_j - p_i) = \left[ f_v(p_{jx} - p_{ix}) \right] \] (13)

For the separation behaviour, a distance \(d\) is defined. If
the distance \(r_{ij}\) is less than \(d\), then a repulsion force is gen-
erated according to the separation fuzzy control function in
Equation (14); where \(d = 1\).

\[ f_p(r_{ij}) = \begin{cases} < 0 & (0 < r_{ij} < d) \\ = 0 & (r_{ij} > d) \end{cases} \] (14)

The output from this behaviour was expressed as:

\[ u_{pi} = \sum_{j \in N_i} \overrightarrow{f}_p(q_j - q_i) \] (15)

where \(\overrightarrow{f}_p\) is defined as:

\[ \overrightarrow{f}_p(q_j - q_i) = \left[ f_p(r_{ij}) \right] \] (16)

In the implementation of cohesion behaviour, an adap-
tive navigation gain was used. This is because when agents
are tracking a common goal, they tend to move towards the
common goal and hence collide with each other. To prevent
this from happening, navigation gain was reduced when agent i had many neighbours and increased when it had
very little. This was expressed as follows:

\[ k_i = \begin{cases} \frac{(6-S_i)}{6} k_{max} & (0 \leq S_i \leq 6) \\ 0 & (S_i > 6) \end{cases} \] (17)

where \(S_i\) is the number of close neighbours, \(k_{max}\) is the
maximal navigation gain. The value of 6 in Equation (17) was the maximum number of agents addressed in their experiments.

Similarly, Gu and Hu [54] used a stability analysis
approach to design the potential function for the separation
component of the flock. The final control function consisted
of T-R rules and Gaussian membership functions. The fuzzy
logic designed separation component was investigated to
replace the well-known artificial potential force repulsive
method used in [55]. This is because when using the artificial
potential force method, the repulsive force becomes very
large when the agents are close together. This pushes
the agents violently away from each other resulting in system
instability as the system tries to use its cohesion force to
compensate for this behaviour.

However, the use of fuzzy logic enabled them to design
the separation component such that the inputs to their sys-

3.5. The use of self organisation and templates in ants

The use of natural self organisation observed in natural
organisms such as ants in robotics has been increasing in
recent years. Self organisation is a phenomenon observed
when termites, for example, build their nests as shown in
Figure 4. It is also a phenomenon observed when ants
arrange dead ants in cemetery clusters and sort out their
brood. In these cases, simple organisms obey simple rules
using only information local to them to perform tasks. The
collaboration between the individuals gives rise to complex
structures such as the Termite Hill. To an external observer,
seems that the individuals are under a centralised control
but this is not the case.

Mathematically, biologists have tried to model these
phenomenon by using agent-based models. For example,
Deneubourg et al. tried to model the cemetery formation
of ants in [60] while in [61], Franks and Sendova-Franks
were able to model the ants brood sorting. In all these
investigations, they hypothesise that clusters are formed
when similar materials are placed at the same location. The
greater the quantity of similar materials at a location, the
more the probability of ants depositing items at that location. This creates a snowball effect at that location.

However, initially, many small clusters are formed but as time goes on, due to the random walk of the ants, fewer large central clusters are formed. The formation of central clusters and their locations however depend on the heterogeneity of the environment. Experiments conducted in [60] showed that if the environment is not large enough, the dead ants are placed on the peripherals of the environment. Experiments also show that the formation of clusters depend on the heterogeneities of the environment.

The formation of clusters in ants was studied and a mathematical model developed. In this model, if the number of perceived items in the neighbourhood of an agent is smaller than the pick up threshold, then the agent is likely to pick up the item. If otherwise, then the agent is unlikely to pick up any item simulating a situation where there is a pile up of items in the environment.[60]

However, the behaviour described and modelled above does not explain fully the natural behaviour of ants. In [62], Johnson et al., tried to explain the sorting pattern of the honey bee nest. It is known that honey bees arrange their brood in the centre of the hive, followed by pollen and then honey combs. This sorting pattern was modelled by Johnson with appreciable results in simulations.

In his work, he used templates. Templates are a sort of pre-pattern that are present in the environment and which natural organisms use during self-organisation processes to restrict their behaviour or build structures. Johnson used natural templates of gravity and the queen pheromone in these experiments. The gravity template was used to drive nectar carrying bees to the top of the nest whilst the queen template was used to generate the compact arrangement of the brood. Nursing bees tend to remove pollen differentially to the brood location. As a result, this leads to the formation of the bands seen in the natural honey bees hive. The use of environmental templates was also tested in [63], where it was shown that the wind currents affect the corpse clustering of ants discussed above. This study shows that templates can be used to control the behaviour of ants.

It has also been proven that the use of templates are responsible for building the royal chamber in Macroteer-mes subhyalinus specie. The template in this case is the pheromonal template of the queen. Self organisation on its own would result in the formation of clusters; however, there would be lack of control over the process as cluster formation is random and spontaneous. However, through the use of templates, self organisation can be controlled and restricted to a particular location where building, sorting or clustering is needed. Consequently, if the template could be controlled, moved and modified, it is possible to control the behaviour of the ants or robots using self organisation in their process.

This phenomenon was partially captured by [64] where they used a robotic cockroach to control the behaviour of cockroaches. Cockroaches normally prefer dark areas and where others are hiding. The more cockroaches are hiding in a dark area, the more cockroaches are attracted to that location. By using robots sprayed with cockroach pheromone, it was possible to control the behaviour of the group so that they aggregate where the robotic cockroaches are hiding even in a less darker area. Also, by assuming that the dark areas are templates, it is possible to control where the cockroaches are hiding by moving the dark area. It is the combination of template and self organisation that would be used in this thesis to provide visual imaging of an invisible spatiotemporal quantity. By relying on the simplicity of these processes, reactivity and fluidity in swarm motion as the distribution of the spatiotemporal quantity changes can be achieved. This approach results in the use of natural swarm intelligence for invisible hazardous spatiotemporal quantity monitoring in the environment. The potential usefulness of the principle of self organisation in controlling multiple robots was also realised and implemented by Sekiyama et al. [65].

3.6. Energy optimisation in swarm foraging

In [66, 67], Lui et al. developed a finite state robotic controller that would enable a swarm of robots forage for food in the environment in the most energy efficient way. The finite state robotic controller is shown in Figure 5. In their experiment, food was randomly distributed in the environment and ‘grows’ over time.
Once a robot finds a food item, it is brought back to the nest. By using two time thresholds; resting time threshold $T_h$, and searching time threshold $T_h$, it is possible to adapt the number of robots foraging for food. Searching time, $T_s$, is the amount of time spent searching for food while resting time, $T_r$, is the time spent resting at the nest. When $T_r > T_h$, the homing behaviour in Figure 5 is activated, while when $T_r < T_h$, the agent leaves home and performs a random walk in search of food. By using three cues, it is possible to modify the threshold values so that the optimal number of robots are left foraging in the arena. The three cues with their explanation are:

- **Internal cues**: If I successfully return a food item to the nest, reduce resting time as there may be more food, if otherwise, increase resting time as there may be less food in the arena.
- **Environment cues**: As I collide with agents in the arena, increase rest time and reduce search time as there might be too much robots for a little number of food items.
- **Social cues**: If my team mates successfully return food to the nest, increase my search time as there might be more food in the arena otherwise reduce my search time.

In addition to the behaviours in Figure 5, an avoidance behaviour is also used to avoid collision with walls, and other robots.

Through the use of a probabilistic finite state machine, Lui et al. [69] were able to derive the macroscopic Equations governing a swarm of foraging robots using the rules above. This made it possible to analyse the system as a whole. It also made it possible to obtain the optimal parameters in terms of $T_h$, $T_h$, speed of robots and so on needed on each individual robot. From Equations, it was also possible to derive the optimal number of robots that should be carrying out resting and searching tasks so that a certain level of energy is achieved in the swarm.

### 3.7. Slime mold aggregation algorithm

Whenever there is a food scarcity, the individual amoebas of the Dictostelium discoideum each release a chemical called cyclic adenosine 3', 5'-monophosphate (cAMP) into the environment. This chemical release results in other amoebas releasing even more chemicals into the environment, thereby resulting in high cAMP concentrations. If the extracellular cAMP concentration exceeds a particular threshold at a location, the amoeba releases its reserve of intracellular cAMP. This causes in a big pulse of cAMP to be released into the environment. After this, the amoeba enters into a refractory state where its receptors are no longer affected by cAMP. This big pulse release causes other amoeba to release their internal reserves too resulting in a travelling wave of cAMP originating from the first amoeba.

Schmickl et al. applied this slime mold aggregation of the Dictostelium discoideum amoeba in controlling a swarm of robots for performing a cleaning task in an environment. [70] The aim was for the robots to return from a work site to a dump site without having a prior knowledge of the dump site or the work site. The knowledge of these areas are propagated through the swarm when it is found by one robot using emitted light pulses. Whenever a dirt site is found, the finding agent emits a color of light, red for example, for a period of time before going into refractory state. In the refractory state, it is no longer sensitive to light pulses. These light pulses are picked up by another agent which emits its own red light. Similarly, when a dump site is found the robots carry out the same behaviour but with a different light color. This results in waves travelling in opposite directions from both sites. The loaded robots navigate against the blue coloured light in the direction of the dump site. The empty robots navigate against the red light towards the dirt site.

Due to the likelihood of robots aggregating at either the dump site or the dirt site, there was a possibility of the robots blocking each other's signal. This prevents the signal from reaching all parts of the arena. To solve this problem, the researchers dedicated a fraction of the swarm of robots...
to random walker mode. These robots exhibited a random walk all over the arena and sometimes acted as communication bridges between the two aggregated groups of robots. Additionally, these random walkers had weighted values from 0.0 to 1.0 that determined the degree of randomness of their walk. A low value near 0.0 was pure random walk while a value near 1.0 was directed navigation.

Robots with directed navigation did not go against the flow of robots heading towards the dump site or the dirt site. Furthermore, preferential collision avoidance of loaded robots and random walkers was implemented using two extra boolean light signals. Using virtual potential fields, they were able to use parameters to adjust the strength of repulsion and the distance moved away between the loaded/random walker robots and the empty robots.[70]

The researchers then went on to improve the performance of the algorithm by using an evolution strategy to optimise the density of agents in the environment among other factors. This was needed because if the density of the agents were too high, they would block the light signals from other members of the swarm. If the density was too low, the light signals might not get to everyone in the swarm as the distance between each agent would be too large. By studying the work done by Schmickl et al., this algorithm depends on the size of the environment. A larger environment would need a larger number of robots so that the density of the robots would be high enough to achieve communications between each individual robot.

By making the robots go at a slower speed and having a higher density of robots, it was possible to achieve a more dense trail towards the dump. This behaviour resembled the biological counterpart of the slime mould. It was shown in [70] that it was possible to use their algorithm to find the shortest route between two points.

4. Achieving multi-agent coverage – present state of the art techniques

In this section, possible present day multi-agent techniques that could be used to explore the environment, find a pollutant and then subsequently provide coverage are discussed.

4.1. The use of Voronoi diagrams (also known as Voronoi tessellation)

The Voronoi partitioning method is one of the commonly used methods for providing coverage in robotics. In order to use this technique, the area of interest must be partitioned into Voronoi partitions. In order to partition a metric space \( \mathbb{R}^d \) into Voronoi partitions, a number \( n \) of points in the space are chosen either deliberately or randomly. Assume that these points belong to the set \( P = \{ p_1, p_2, p_3, \ldots, p_n \} \). Voronoi cells, \( V_i \), are then generated from these points so that the relationship in Equation (19) is true.[71]

\[
V_i = \{ x \in \mathbb{R}^d | dist(x, p_i) \leq dist(x, h) \forall h \in S(P) \} \tag{19}
\]

In other words, various positions, \( x \), in \( V_i \) should be closer to point \( p_i \) than to other points \( h \) in other Voronoi cells. \( dist \) is an Euclidian distance function. In order to generate a Voronoi partition, a line is drawn between two points and a bisector line is drawn to divide this line into two equal halves. If this procedure is performed for each of the points in the space, the resulting bisector lines join up to produce a Voronoi diagram as shown in Figure 6.

This results into the partitioning of the \( \mathbb{R}^d \) space and a Voronoi cell enclosing each of the points in set \( S_P \). Each of these point is called a Voronoi generator. In the use of this approach, the Voronoi points are replaced by robotic agents. In the presence of a density function \( \rho(q) \) in \( \mathbb{R}^d \), the centroid of each Voronoi cell can be calculated using the Equation (20).[72]

\[
M_V = \int_V \rho(q) dq, \quad C_V = \frac{1}{M_V} \int_V q \rho(q) dq \tag{20}
\]

\[
\mathcal{H}(P, V_i) = \sum_{i=1}^{n} \int_{V_i} f(||q - p_i||) \phi(q) dq \tag{21}
\]

By using a minimum cost function of Equation (21), Cortes et al. [73] were able to direct agents to arrange themselves according to the density function \( \rho(q) \) of \( \phi(q) \). Where \( \phi(q) \) is the sensory function in the environment or the environmental variable. An example of such function or environmental variable is pollution and \( f(||q - p_i||) \) is a function used to assess the performance of the agents (Figure 7).

One of the methods for computing the centroids of the Voronoi partitions is the Lloyd’s algorithm. For each iteration of the Lloyd’s algorithm, the Voronoi cells are first computed using the robotic agents in the environment as points. Then the centroid of each Voronoi cell is computed by integrating the cell and then using Equation (20). After calculating the centroid, the robotic agents are commanded to move to the new centroid. The entire process is repeated over and over again until no more progress is made towards the source. In other words, the algorithm works so that the minimum cost function of Equation (21) is satisfied. During each iteration, the algorithm aims to place the agents so that their positions correspond to the calculated centroid of each computed Voronoi cell. Summarising, using Voronoi partitioning to perform coverage involves two steps: first calculating the Voronoi cells using the ordinary Voronoi cell formation system and then calculating the mass centroid of each Voronoi cell.

However, to be able to use the Voronoi partition approach, the sensory function \( \phi(q) \) must be known before hand. This is because the sensory function density for each Voronoi cell must be calculated and it is not possible for the robotic agent to go around its cell collecting data without upsetting the already formed Voronoi cell. This problem could be partially solved by using a highly accurate positioning device so that the agent is able to go around its cell.
collecting data about the cell density and then return to its Voronoi calculated position. However, the boundaries of the Voronoi cell must be known and errors due to robot wheel slippage would not make this system practical. In addition, the computational cost of running the Lloyd’s algorithm every run time is costly.

To solve the first problem of finding the density for each newly calculated Voronoi cell, researchers have used a variety of machine learning algorithms. For example, Schwager et al. investigated the use of a learning function to enable the agents to learn the underlying distribution \( \phi(q) \).\(^{[74]} \) This was done by using a linear combination of a set of Gaussian functions \( \mathbf{K}(q)^T \). This set of Gaussian functions could be replaced with other functions depending on the application in order to get good results. By using a vector of unknown parameters \( \mathbf{\bar{a}}_i \), it was possible to weight the functions so that they approach an estimate of the distribution \( \phi(q) \) in the environment. This was done according to Equation (22) and shown in Figure 8. Each of the robots \( i \) in the team using this approach has the same \( \mathbf{K}(q)^T \), but not necessarily the same parameter values in \( \mathbf{\bar{a}}_i \). This value would depend on their sensory perception which in turn depends on the sensors on their platform. In addition, extreme noise might corrupt the data collected and as a result, the estimated sensory distribution in the environment.

\[
\hat{\phi}_i(q, t) = \mathbf{K}(q)^T \mathbf{\bar{a}}_i(t)
\]
23
25

Figure 8. Function approximation using a combination of various basic Gaussian functions. True function is in the darker curve (φ(q)), the dashed curves (q) are the vectors of the Gaussian basic functions and the parameter vector q denotes the weighting of each Gaussian function. The grey curve is the approximate function formed from the dashed curves.\[74\]

In order to calculate the parameter vector q, the researchers used Equations (23) to (25).\[74\]

\[\lambda_i(t+1) = \lambda_i(t) + \kappa(p_i(t))\phi(p_i(t))\]

(23)

\[\Lambda_i(t+1) = \Lambda_i(t) + \kappa(p_i(t))\kappa(p_i(t))T\]

(24)

\[\hat{\Lambda}_i(t) = \hat{\Lambda}_i(t) + \gamma[\lambda_i(t) - \Lambda_i(t)\hat{\Lambda}_i(t)] + \zeta\sum_{j\in N_i(\hat{x}_i(t) - \hat{x}_i(t))}\]

(25)

where \(\phi(p_i(t))\) is the present sensory function value measured by the robot and \(\kappa(p_i(t))\) is the value of the Gaussian function at position \(p_i(t)\). \(\zeta\) and \(\gamma\) are positive gains. The centroid for each Voronoi cell was calculated by using Equation (26).

\[\hat{\phi}_V(t) = \frac{\sum_{q\in V}q\phi_i(q, t)\Delta q}{\sum_{q\in V}q\phi_i(q, t)\Delta q}\]

(26)

By using the above approach, the researchers were able to control a group of robots to arrange themselves based on an underlying environmental function in the environment. Nevertheless, an additional disadvantage of the Voronoi partitioning algorithm is that it only works effectively for polygon derivative environments. Nevertheless, the Voronoi partition algorithm has the problem of getting into local optimal configurations. In order to improve the coverage performance, the ladybug algorithm was introduced. The ladybug algorithm was introduced as a result of its exploration properties. By using these two behaviours in parallel, they were able to maximise the sensor network coverage of an area. Each agent had the dynamics shown in Equation (27).

\[\dot{p}_i = u_i = Ke_i\]

(27)

where \(K = \begin{bmatrix} k & -f_i \\ f_i & k \end{bmatrix}\), \(e_i\) is the error between the centroid point \(C_v\) and the present point \(p_i\). \(f_i\) is the exploration gain for the ladybug controller and \(k\) is a control gain. \(f_i\) is a gain that causes the agents to be either right or left biased. This is determined randomly at the start of the experiment.\[75\]

4.2. The use of virtual spring mesh

Shucker et al. used virtual physics spring mesh to co-ordinate a group of agents so that they cover an area effectively.\[76\] By reducing the total kinetic energy generated by the spring mesh of the system, they were able to track a simulated pollutant. The Lyapunov function was used for the energy reduction. Their approach could be used for hybrid systems or any system without knowing the full details of the system. By specifying a switching function, that is unique to a system, it is possible to achieve a final stable configuration for that system without knowing the details of its final stable configuration.

Each agent’s dynamics is affected by virtual springs connected to the agent from the surrounding agents. This takes the form shown in Equation (28).

\[\tilde{x}_i = \sum_{j\in N_i} k_s(||x_i - x_j|| - l_o)\hat{u}_{ij} - k_d\tilde{x}_i\]

(28)

where \(x_i\) is the cartesian coordinates of the agent’s position, \(\tilde{x}_i\) is the agent’s acceleration, \(\tilde{x}_i\) is the agent’s velocity, \(N_i\) is the set of springs connected to the agent, \(k_s\) is the spring stiffness, \(l_o\) is the spring’s natural length, \(\hat{u}_{ij}\) is the unit vector from agent \(i\) to agent \(j\) and \(k_d\) is the damping coefficient.

Figure 9 shows an example of how the network transforms from an initial state to a final state. Springs can be created or destroyed depending on how the global energy reserve would be affected during a switch. The switching function \(\sigma(t)\) is time dependent and follows the dwell time analysis in \[77\]. It also specifies how often the system state changes. According to the dwell time analysis, if the members belonging to a certain class of linear system \(T\) change their stable states arbitrarily, the entire hybrid class of system \(T\) would also be stable provided that the switching rate is slow on average.\[77\] However, instead of putting a limit on the time of change, Shucker et al. use a global energy reserve to create the same effect. Each individual agent was able to estimate the global energy reserve by using a zero sum consensus algorithm. The researchers went on to use their approach to track targets using the following control law:

\[\tilde{x}_i = u_i = \left[ \sum_{j\in R} \nabla P_R(x_i, x_j) \right] + \left[ \sum_{k\in T} \nabla P_T(x_i, x_j) \right] - k_d\tilde{x}_i\]

(29)

where \(k_d\) is the damping coefficient, \(P_R\) is the potential function acting between agents and \(P_T\) is the potential function acting between agents and target points. The potential functions could be defined by the user but in the case of the virtual springs mesh method, it is a simple spring potential as discussed above.

By dynamically adjusting the distance between agents, it was possible to achieve constant stability even when the targets motion or points were going to cause instability in...
the system. The downside to this approach is that it relies on choosing individual target points so that the target can be tracked. This approach is not practical if it is to be used to track an environmental quantity like temperature for example. This is because it might not be feasible to define individual temperature points.

4.3. Optimotaxis
Mesquita et al. used a technique called Optimotaxis to find the global maxima of a chemical field.[78] The technique was also used to adjust the probability density of agents so that they matched the probability density of the signal. This was achieved by developing two controllers inspired by the bacterial chemotaxis behaviour. Both controllers adjusted the velocity and the tumbling rates of the agents. In their experiments, the agents did not communicate with each other and did not know their locations in the ‘world’. They were only capable of taking measurements of the signal.

The two controllers proposed were a run and tumble controller and a diffusion controller. In the experiments, a prior knowledge of a shaping function $Q(.)$ was needed by the agents. This shaping function was used in the controllers to choose the appropriate velocities to ensure that the agents converge to the signal function $F(.)$.

4.3.1. Run and Tumble Controller
The run and tumble controller has two control parameters – the velocity $v$ and the tumble rate $\lambda$. The velocity $v$ changes to a random value $v \in V$ with a probability density $T_v^{-}$ which may depend on the velocity $v_-$ before the tumble. The tumble rate $\lambda$ and the velocity $v$ both depend on $x$ and $v$ through the measurements $F(x(\tau)); 0 \leq \tau \leq t$. Where $x$ is the position of the agents.

The probability density $p(x, v, t)$ of finding a vehicle at position $x$ with velocity $v$ and at time $t$ was shown to satisfy

\[
\frac{\delta p(x, v, t)}{\delta t} + v \cdot \nabla_x p(x, v, t) = -\lambda p(x, v, t)
+ \int_V T_v'(v)\lambda(x, v')p(x, v', t)d\mu(v')
\]

(30)

Equation (30) is known as the linear transport Equation. In order to control the distribution $p(x, v, t)$ of the agents so that they form a distribution $Q(x)$ of the spatial function under investigation, the right values for the parameters of $v$ and $\lambda$ must be chosen. In order to obtain the right values, $p(x, v, t)$ is substituted by the known or estimated spatial function $Q(x)$, resulting in the Equation (31).

\[
v \cdot \nabla_x Q(x)
= -\lambda(x, v)Q(x) + Q(x)\int_V T_v'(v)\lambda(x, v')d\mu(v')
\]

(31)

By dividing both sides by $Q(x)$ in Equation (31) and rearranging the terms, the Equation (32) was obtained. By using a uniformly distributed velocity jump as in Equation (33), the control law in Equation (34) was obtained.

\[
\lambda(x, v) = \int_V T_v\lambda(x, v')d\mu(v') - v \cdot \nabla_x \ln Q(x).
\]

(32)

\[
T_v'(v) = \frac{1}{\mu(V)}
\]

(33)

\[
\lambda(x, v) = \eta(x) - v \cdot \nabla_x \ln Q(x).
\]

(34)

where $\eta(x) = \int_V T_v\lambda(x, v')d\mu(v')$ is a function chosen by the designer and depends on $x$ only through $F(x)$. This control law depends on the past measurements of $\{F(x(\tau)); 0 \leq \tau \leq t\}$ or known measurements as discussed above so that the shaping function $Q(.)$ could be constructed.

4.3.2. Diffusion Controller
The diffusion controller was developed for vehicles that turn constantly. This controller would be useful if the controllers have a high tumbling rates. The controller is given by the

\[
\frac{\delta p(x, v, t)}{\delta t} + v \cdot \nabla_x p(x, v, t) = -\lambda p(x, v, t)
+ \int_V T_v'(v)\lambda(x, v')p(x, v', t)d\mu(v')
\]

(30)
stochastic differential Equation:

\[ dX_1 = \rho \cos \theta dt \quad dX_2 = \rho \sin \theta dt \quad d\theta = \sigma(x, \theta)dw \]  

(35)

where \( w(t) \) is a continuous Wiener process, \( x_1 \) and \( x_2 \) identify the position vector in the plane, \( \rho \) is the velocity of the agent and \( \sigma(x, \theta) \) is the turning intensity which can be adjusted to get the attained behaviour.

As the improvement in measurements increases, the turning intensity increases. This results in the Fokker-Planck Equation for \( p(x, v, t) \) in Equation (36).

\[ \frac{\partial p}{\partial t}(x, v, t) + v \nabla_x p(x, v, t) = \frac{1}{2} \frac{\partial^2}{\partial \theta^2} (\sigma^2 p(x, v, t)) \]  

(36)

Substituting \( p(x, v, t) \) with \( Q(x) \), as performed on the previous controller, in order to obtain the right value for the turning intensity parameter \( \sigma \) and performing integration twice on \( \theta \), results in the following Equation:

\[ v \nabla_x Q(x) + \frac{1}{2} \sigma^2(x, \theta)Q(x) = \theta c_1 + c_2(x). \]  

(37)

Solving for \( \sigma^2(x, \theta) \) in Equation (37) results in the following control law:

\[ \sigma^2(x, \theta) = \eta(x) - 2v \nabla_x \ln Q(x). \]  

(38)

By using the run and tumble controller with a constant \( \eta(x) \), the results in Figure 10 were obtained. In the experiment, the desired stationary density was stated as \( Q(F(x)) = cF^n(x) \). Their controllers were able to detect and escape local minima and hence move the agents on to the global maxima. It was also discovered that their approach was robust against additive white Gaussian noise applied to the vehicle’s body frame. However, this approach requires a knowledge of the signal to be profiled prior to deployment and did not take collisions between agents into consideration.

4.4. Deterministic annealing

Deterministic annealing pioneered by Rose is similar to simulated annealing, in that it uses a cycle of temperature lowering to enable a system to gradually descend towards the lowest energy state which corresponds to the optimal of a cost function.[79] This is achieved by using system phase changes as the temperature passes below critical values. In this case, the aim is to minimise a cost function designated by Equation (39).

\[ D = \int_Q \phi(q) \sum_{i=1}^{n} P(p_i|q) f_i(||q - p_i||)dq \]  

(39)

where \( \phi \) is a distribution density function, \( q \) are points in \( \phi \), \( P(p_i|q) \) is the probability of a point \( q \) being associated with an agent \( p_i \) and \( f_i \) is the assessment function used to investigate the quality of the solution derived by the agents.

\[ H = -\int_Q \phi(q) \sum_{i=1}^{n} P(p_i|q) \log P(p_i|q) \]  

(40)

As the temperature reduces, the sensor positions obtained as a result of Equation (44), become unstable and a phase change is performed. In order to determine when this critical temperature is reached, perturbations \( \gamma \) were introduced with a scaling factor \( \epsilon \). These perturbations would cause an

Figure 10. Showing different stages in optimotaxis in the presence of two maxima. Black dots are the agents where as the background intensity represents the signal intensity \( F(x) = 0.4e \).

Kwok and Martinez used this technique to control a simulated group of agents in [80]. As Equation (39) cannot be minimised directly, the Shannon entropy in Equation (40) is used. The deterministic annealing algorithm could then be viewed as a way of minimising the Lagrangian Equation \( F = D - TH \). However, in order to be able to minimise the Lagrangian Equation, the probability distribution \( P(p_i|q) \) must satisfy the Gibbs distribution of Equation (41), where the normalising factor is presented in Equation (42).

\[ P(p_i|q) = \frac{\exp \left[ -\frac{\delta(||q - p_i||)}{T} \right]}{Z(q)}, \quad i = 1, \ldots, n \]  

(41)

\[ Z(q) = \sum_{i=1}^{n} \frac{f_i(||q - p_i||)}{T} \]  

(42)

\[ \hat{F} = -T \int_Q \phi(q) \log Z(q) dq \]  

(43)

where \( \sum_{i=1}^{n} P(p_i|q) = 1 \). Substituting Equations (41) and (42) into the Lagrangian Equation results in Equation (43). By differentiating Equation (43) with respect to the sensor positions as in Equation (44), it is possible to achieve gradient descent towards an optimal configuration.[80]

\[ \frac{\partial \hat{F}}{\partial p_i} = -T \sum_{k \in C_i} \int_{D_k} \phi(q) \frac{1}{Z(q)} \frac{\partial Z}{\partial p_i} dq \]  

(44)

\[ \frac{\partial Z}{\partial p_i} = \frac{2}{T} (q - p_i)^T \exp \left[ -\frac{||q - p_i||^2}{R^2} \right] \]  

(45)
agent to change its position following $x_i = p_i + \epsilon \gamma_i$. Critical temperatures occur either when $\frac{\partial \hat{F}}{\partial \epsilon} |_{\epsilon = 0} = 0$ or $\frac{\partial \hat{F}}{\partial p_i} = 0$. If this condition is true and $\frac{\partial^2 \hat{F}}{\partial \epsilon^2}$ is true then, a phase change is performed as a result of the perturbations discussed earlier. Kwok and Martinez then went on to implement this technique on simulated distributed agents with results shown in Figure 11. In Figure 11, it is seen that a single run of their algorithm does not cover all the spatially distributed function present in the environment. A maximising coverage of the environment is first needed followed by a high temperature cycle. This could be seen as an exploration of the environment. Finally, the cooling procedure described above is carried out to settle the agents into optimal configurations. However, studying the distributed algorithm closely revealed that it would rely heavily on a communication mechanism that must ensure connectivity to all other agents in the network in order to synchronise all the agents in the group. Consequently, the technique might not be robust and any error (such as noise in sensor readings) could cause the whole technique to fail.

As would be seen from the coverage techniques presented above, none of them have used a behaviour-based approach in their implementation. These techniques include the Voronoi partition, virtual springs and deterministic annealing methods. It was gathered from their implementation procedure that they are either computationally expensive, need extensive communication mechanisms, and as such might not be capable of responding to dynamic spatiotemporal profiles or even being deployed on simplistic agents.

However, using a behaviour-based approach offers the reactivity needed to respond to sudden distribution changes caused by wind or other factors during the tracking of a airborne dynamic spatiotemporal process. It also offers the flexibility to add more behaviours. These advantages are desirable especially if a coverage scheme that would be capable of operating in the real natural environment is to be developed.

5. Conclusion

This paper overviews various behaviours that could be used in a behaviour-based paradigm for the purposes of visualising invisible pollution. The need for this is especially useful during search and rescue operations when responding to natural and man-made disasters. The paper also discusses present day techniques that could be used. The behaviour-based paradigm was chosen over the computational expensive nature of traditional sense-plan-move-based approaches because of their unsuitability to visualise invisible pollution in the dynamic real world.

On the other hand, the behaviour-based paradigm is able to react to sudden changes in pollution distribution due to its reactive nature and has the flexibility to add more behaviours. These advantages are desirable especially if a coverage scheme that would be capable of operating in a
natural environment is to be developed. As the behaviour-based paradigm involves combining behaviours in order to solve a problem, various source-seeking algorithms and multi-agent systems that could be ideal candidates in developing a solution were considered.

As the choice of the individual behaviours making up a behaviour architecture affects the final solution, solutions produced by natural organisms were considered in this survey. These solutions have been tested by nature over millions of years resulting in biological systems that use computationally simple mechanisms to forage for food in their environment. It is possible to transfer the computationally simple, cheap and robust mechanisms of the biological organisms to robotics, with the added advantage of improving these mechanisms where necessary. This is called the ethological approach of choosing behaviours.

By following the ethological approach, it is possible to rely on the well-established field of biology in order to provide novel solutions to the relatively new field of robotics. The bacteria chemotaxis behaviour was focused upon, due to its simplicity in finding its food source in the environment. Furthermore, flocking behaviour for multi-agent co-ordination and collaborative foraging was also reviewed. The simplicity and efficiency of the biological algorithms would make a swarm of robots respond in real-time to changes in dynamic invisible pollutants, with the fluidity of a flock of starlings in flight for effective visualisation of pollution.

Notes on contributors

John Oyekan (M’08) received the PhD degree in robotics from the University of Essex, Colchester, UK. He is currently a Senior Research Fellow with the Centre for Research in Distributed Technologies, University of Bedfordshire, Luton, UK. He has published over 14 international conference and journal papers including coauthoring two book chapters. His current research interests include robotics, computational intelligence, multiagent systems, unmanned aerial vehicle control in disaster situations, and computational neuroscience. He has served as a reviewer for the IEEE Transactions on Robotics, IEEE Transactions on Systems, Man and Cybernetics, IEEE International Conference on Robotics and Biomimetics (ROBIO), the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), and the IEEE International Conference on Robotics and Automation (ICRA), among others. He is also part of the program committee on ROBIO.

Huosheng Hu (M’94–SM’01) is a Professor in the School of Computer Science and Electronic Engineering at the University of Essex in the United Kingdom, leading the robotics research group. His research interests include behaviour based robotics, human-robot interaction, service robots, embedded systems, data fusion, learning algorithms, mechatronics, and pervasive computing. He has published over 400 papers in journals, books and conferences in these areas, and received a number of best paper awards. Prof. Hu is a founding member of IEEE Robotics and Automation Society Technical committee on Networked Robots, a Fellow of IET and InstMC, a senior member of IEEE and ACM, and a Chartered Engineer. He has been a Program Chair or a member of Advisory/Organising Committee for many international conferences such as IEEE ICRA, IROS, ICMA, ROBIO, ICIA, ICAL, and IASTED RA, CA, and CI conferences. He currently serves as Editor-in-Chief for International Journal of Automation and Computing, Founding Editor-in-Chief for Robotics Journal and Executive Editor for International Journal of Mechatronics and Automation.

References


