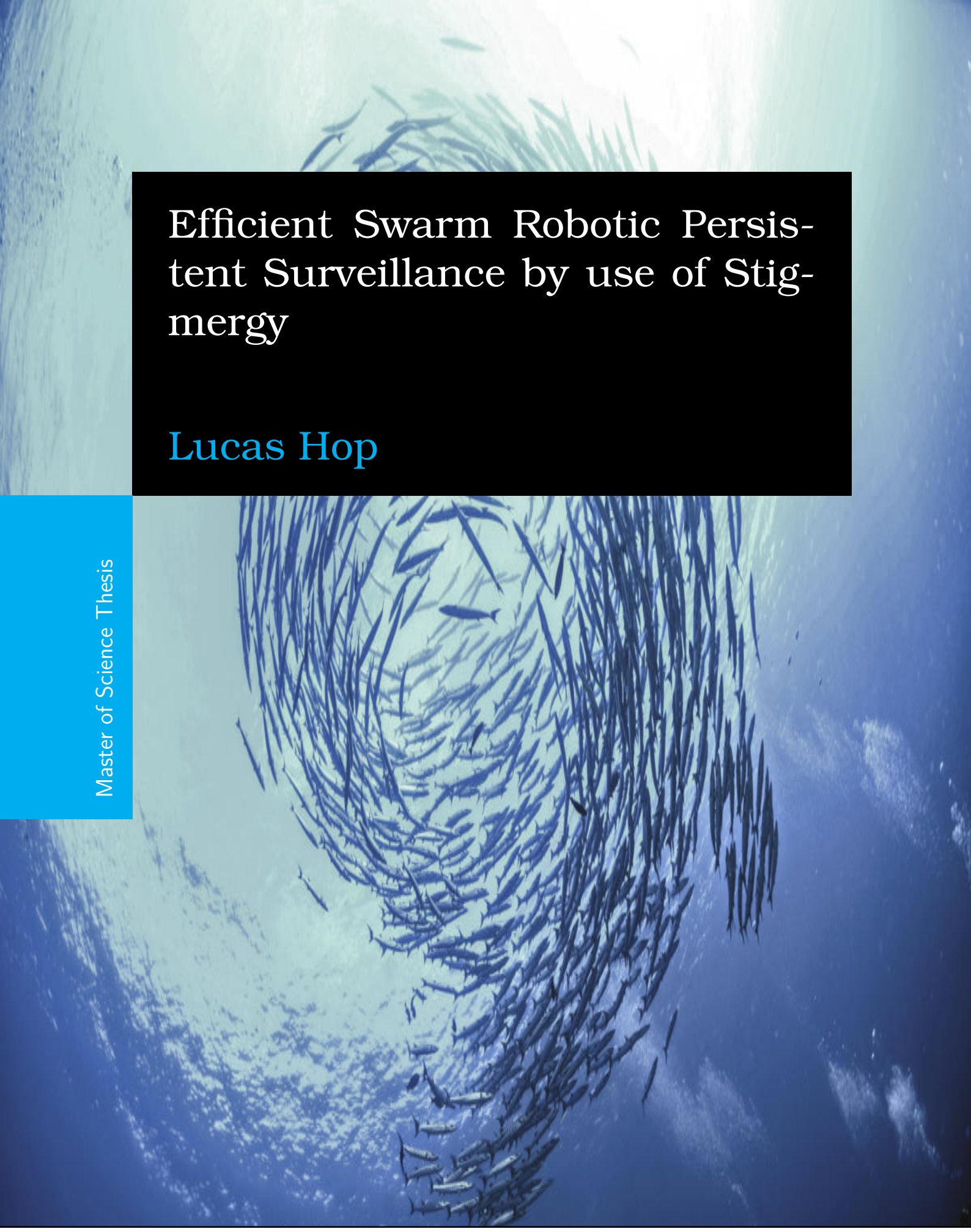


Efficient Swarm Robotic Persistent Surveillance by use of Stigmergy

Lucas Hop

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Efficient Swarm Robotic Persistent Surveillance by use of Stigmergy

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Lucas Hop

June 15, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Efficient Swarm Robotic Persistent Surveillance by use of Stigmergy

by

Lucas Hop

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: June 15, 2021

Supervisor(s):

Dr.ir. M Mazo Jr

ir. D Jarne Ornia

Reader(s):

ir. L Ferranti

Abstract

Persistent surveillance is the act of covering an environment persistently, as fast as possible. By exploiting the intelligence of the swarm, it is possible to create a swarm robotic persistent surveillance method that can deal with unknown dynamic environments without the need for complex computations or excessive storage. Using stigmergy, which is communication via the environment, robots drop pheromones to signal that a specific location is covered. Other robots sense these pheromones and avoid going there. This is, in essence, the inverted stigmergic behaviour of ants. While ants deploy pheromones to attract other ants to that location, our robots repel other robots by deploying pheromones. This thesis proposes a stigmergic swarm robotic persistent surveillance method that can deal with unknown environments and dynamic obstacles. Stigmergy is used as the sole communication mean. Additionally, an extension is included in the model that renders the model more efficient with respect to the pheromone usage. Subsequently, to demonstrate the potential for real-life application, the model is simulated in Webots, an open-source 3-D Robot simulator. Concludingly, this thesis demonstrates that stigmergy lends itself perfectly for persistent surveillance. It minimizes the computations and memory storage needed, while ensuring performance. The proposed model outperforms current literature and deploys pheromones more efficient. The model is inherently robust and flexible.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, professor Manuel Mazo Jr and daily supervisor Daniel Jarne Ornia for their supervision during my thesis writing. The frequent group meetings and many opportunities to discuss any question were of great value.

To everyone in the research group, thank you for the great open atmosphere and willingness to listen to presentations and give feedback.

Lastly, a special thanks to Daniel, who has provided me with lots of feedback and insights. Your willingness to take the extra step and the lengthy weekly discussions, which I was looking forward to, is much appreciated

Delft, University of Technology Lucas Hop
June 15, 2021

Chapter 1

Introduction

In the past decades, robotics has made an unprecedented advancement in the world. This advancement has led to the appearance of possibilities that were not conceivable before. One of these possibilities is the field of robotic coverage. In robotic coverage, robots are to cover a certain area entirely, once or persistently, preferably as fast as possible. It has a wide range of applications, such as lawn mowing, cleaning or harvesting to search and rescue missions, intrusion detection or mine clearing. The literature on robotic coverage proposes numerous approaches. The majority of these approaches rely on strong assumptions and require for instance, perfect communications and sensors [6], a perfect known environment [7] or high computational power [8]. In some cases, this is not viable. For instance, after a disaster, where communication lines are broken and the situation is unknown, coverage is needed as fast as possible for communication and visuals. Swarm robotics has the potential of circumventing these assumptions while solving the coverage task.

In 1980s Swarm intelligence was proposed. Since then, it has attracted many researchers' attention and is at the focus of many disciplines, including artificial intelligence, robotics, sociology and biology. Swarm intelligence is the emerging collective behaviour of self-organized systems of agents. In practice, this means not only many hands make light work, but many hands make light exceptional complex work. With swarm intelligence $10+10 \neq 20$, but rather 10+10=100 as the authors of [9] pointed out. This phenomenon was observed a long time ago with animals that could survive in harsh environments with the help of swarms, rather than turning to the wisdom of the individual. It became clear that poor intelligence individuals could come to complex behaviours while operating in swarms. Still, most swarm intelligence researchers are inspired by nature swarming. A swarm ranges from a few individuals to millions of individuals and has proved itself flexible and robust. The individuals act at their capacity, no entity governs and gives orders, and communicate locally either directly to an individual or via the environment. A great example is ants. Ants are poor-intelligence, small and vulnerable individuals and can perform only basic tasks. They communicate directly with sound and touch and indirectly with pheromones. In a successful quest for food, ants deploy pheromones on the environment in order to signal where the others can find the food as well. This communication method is called stigmergy and is defined as indirect communication

via the environment [10]. Stigmergy is one of the main reasons why swarming is robust and flexible. When an ant detects pheromones, it is stimulated to follow the trail and on it's turn place more pheromones. Hence, the chance of other ants following and deploying the pheromones increases. The result is the aggregation of ants at the food source. It is exactly this stigmergic behaviour of ants where the idea for this thesis originated from. What happens when the stigmergic behaviour of ants is inverted, where pheromones repel instead of attract other agents? Will this result in a stigmergic coverage method? Consequently, this master thesis proposes a swarm robotic persistent surveillance model, where stigmergy is used as the sole communication method based on the inverted stigmergic behaviour of ants. On top of that, the model includes an extension that limits use of pheromones, while keeping performance. In order to do that, the relevant background from current literature and the objectives for this thesis are given in chapter 2. In chapter 3 the problem is formulated and the model is proposed. Chapter 4 shows the results of the simulations that are done via Python, where the model is optimized and evaluated. Subsequently, a simulation with physical robots is done in chapter 5. The results are discussed in chapter 6 and these are all concluded in chapter 7. Lastly, chapter 8 gives recommendations for future work.

Chapter 2

Background

This chapter gives the relevant information that is needed for this thesis. First, swarm robotics is discussed. Next, robotic coverage is laid out and how swarm robotics can play a role in that. Then, current stigmergic robotic coverage literature is presented, and lastly the research goals are given.

2-1 Swarm robotics

2-1-1 Definition

One of the sub-divisions of swarm intelligence is swarm robotics. Swarm robotics opts to combine swarm intelligence with robotics, where multiple simple low-resolution robots complete a task while operating in a swarm. In the work of [11] a definition of swarm robotics is proposed: "*Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment.*" A robotic swarm has certain specifications. The swarm should consist of autonomous robots, a large number of robots, few homogeneous groups of robots, relatively simple robots with respect to the complex behaviour of the swarm and robots that have local sensing and communication capabilities.

2-1-2 Characteristics

Swarm intelligence has three requirements for the swarm. Decentralised agents, local communication of the individual and homogeneity. A decentralised system is a system where there is no central authority that governs or instructs the individual. It was discovered in [12] that there is no centralised coordination mechanism in nature and that the swarm is decentralised. The agent follows its own rules autonomously. "Local communication" would be the agent

that communicates either directly to another agent or indirect via stigmergy. Direct communication could be with sound, touch or smell and stigmergy could be all that that "marks" the environment, such as pheromones, paint or volatile substances. Lastly, homogeneity is a requirement for a swarm. All the agents are similar and could perform every task interchangeably. These three characteristics render the swarm robust , flexible and scalable. The power of swarm intelligence is that there is no vital subsystem that when destroyed, the whole swarm cannot operate anymore.

Robustness, flexible and scalability are the most valuable properties of a swarm and are desirable for a multi-robotic system, according to [13]. A system is robust when it remains stable or successful in its task while dealing with perturbations and uncertainty. [14] paraphrases this as "Robust systems: systems that have acceptable behaviour over a larger class of situations than was anticipated". The characteristics of the swarm render the system already robust. However, swarms rely on more. Biological systems often achieve robustness with redundancy, where individuals are dispensable. If a significant proportion of all the agents break down, the swarm is still able to complete its tasks successfully. The tasks will be delayed since there are fewer agents, but eventually, the tasks are completed. In the case of swarm robotics, this is often applied as well. A swarm could consist of thousands of agents. Nextly, flexibility. A flexible system is able to be easily modified to respond to altered circumstances, such as a dynamic environment. The authors of [11] interpret this for swarms as the ability to create modularised solutions to different tasks in order to be flexible. Again, ants demonstrate this. Individual ants take place in tasks of different nature depending on the moment, tasks like foraging, prey retrieval and chain formation. They quickly adapt when a pressing event demands attention. This is aided by the homogeneity of the agents, where the all tasks are interchangeably. Lastly, scalability requires a swarm to operate successfully under a wide range of group sizes. The decentralised and homogeneous nature of the swarm is again key here. There is no essential part that needs to be intact. The swarm is homogeneous and operates the same regardless of the group size.

More than often swarm intelligence relies on stochastic choices of the agents. The choices of an agent are a balance between a simple perception-reaction model and a probabilistic model. Through these stochastic choices, better solutions are continuously found. During foraging, ants deploy pheromones on their way back to the colony if they have found food. Now other ants are more likely to follow this path and arrive at the food food. However, there is still a chance that some ants deviate from the path and find a shorter path to the food. This path is preferred over the latter, and the pattern repeats. In that way, eventually, the shortest path is found to the food and most ants will follow that path. Stochastic choices are essential in the exploration of new territory as well. Swarm intelligence uses stochastics in a search for better solutions.

2-1-3 Benefits

Using the power of the swarm in robotics has multiple benefits that are desirable in the field of robotics. The three main benefits are already discussed in the previous section , namely

- Robustness
- Flexibility

- Scalability

These three properties are essential for robotic systems. Swarm robotics has more advantages, such as:

- **Parallel working:**
  A swarm often consist of a large population that can perform tasks autonomously. Because of the large population, many agents can cooperate to come to success faster. Again with foraging by ants. In the food search, it is hard for one ant to find food in a vast environment. However, while working in parallel with a large amount of ant's food is much faster found.

- **Economical:**
  For swarm intelligence, the individual agents are very basic relative to the behaviour of the collective. In swarm robotics, this property can be exploited, and the robots can be very economically designed, manufactured and maintained. Although there could be many robots in a swarm it is often still cheaper to mass manufacture this than building one high-quality complex robot that requires precision machining.

2-2 Robotic Coverage

2-2-1 Definition

Robotic coverage is the act of robots covering a known or unknown environment entirely, once or persistently, as fast as possible. Real-life applications for robotic coverage are in abundance. It could be as mundane as painting or cleaning and on the other hand, for something specific like crop inspection or forest monitoring. Robotic coverage aims to assist with these tasks. The authors of [15] define multi-robot coverage as the multi-robot coverage path planning (CPP) problem. Multi-robot CPP is "*the determination of multiple paths that goes through a set of viewpoints through which the team of robots must visit, each with its assigned path, in order to completely scan, explore or survey the structure or environment of interest.*" A similar but slightly different task that falls under robotic coverage problems is robotic persistent surveillance. According to [16], persistent surveillance is different from CPP since it involves perpetual coverage of a changing area, minimising the time between revisits. Therefore it is not just the repetition of a CPP problem considering that the revisit time has to improve overall and performing a CPP problem twice will not necessarily result in a better result the second time.

2-2-2 Problem formulation

There are two main subproblems under the umbrella of robotic coverage, namely CPP and persistent surveillance.

Coverage path planning

CPP is a more studied problem and has a clear problem formulation. The authors of [17] did define, already in 1988, criteria that are also applicable for multi-robot CPP nowadays:

1. Robot(s) must cover the area completely, moving through all the points.
2. Robot(s) must fill the area without overlapping paths
3. Continuous and sequential operation without any repetition of paths is required of the robot(s).
4. All obstacles must be avoided in the area by the robot(s)
5. The robot should use "simple" motion trajectories, such as straight lines or circles, for the simplicity of control
6. An "optimal" path is desired under the available conditions.

In complex environments, it is not always possible to satisfy all these criteria, and priority consideration is required.
The CPP problem is the most encapsulating and researched problem. However, there are some specific variants of coverage that are worth noting, such as:

- **The covering salesman problem:**
  A variant of the travelling salesman problem where an agent has to visit a neighbourhood of a city instead of just a city.
- **The lawnmower problem:**
  Finding a path that covers the whole region, to cut the grass.
- **The piano mover's problem:**
  Find a collision-free path from configuration A to configuration B
- **The watchman route problem:**
  Find the shortest route to the starting point so that each point in the specific region is visible at least from one point on the route.
- **The art gallery problem:**
  Find the minimum amount of agents/sensors that could be situated in a polygonal environment so that each point in the gallery is visible to at least one agent/sensor.

All these different approaches could be divided into two general subtasks in CCP: Viewpoints generation and coverage path generation. Viewpoints generation is the method of how to position and orient the sensors from where the data will be collected. This affects the overall coverage of the area. The coverage path generation is the method of creating the most efficient and complete path from which the whole area could be covered.

There are several performance metrics for CPP given in literature. The authors of [15] have compared many single- and multi-robot CPP methods. The three primary metrics that could

be distilled from literature are execution time, energy consumption and coverage completeness. Secondary metrics, like the robustness of the method and covered path length, are also widely used.

Persistent Surveillance

Persistent surveillance goes under a lot of different names, such as Persistent monitoring, area surveillance and can be seen as a generalisation of the patrolling problem. With patrolling meaning the surveillance of borders of specific areas [18]. The authors of [19] propose a problem definition for persistent surveillance. It is the minimisation of the maximum age of every cell (or point) monitored over a "long" period of time. Age is the amount of time elapsed since a point or cell lastly has been observed. Evidently, age is used as the metric for performance.

Requirements of a persistent surveillance method:

- **Robustness:**
  The method needs to be robust since it operates over a longer period of time. If one agent fails, the environment must be kept persistently surveyed, however slower.

- **Flexible:**
  The method needs to cope with stochastic dynamic environments. Persistent surveillance is about monitoring an environment perpetually. Many environments are dynamic.

- **Parallel working:** The age needs to be minimised, which is a constant process of covering multiple points fast. This is done with cooperative and parallel working.

2-2-3 Persistent Surveillance vs CPP

Persistent surveillance and CPP have slightly different objectives. CPP opts to cover the area as fast and efficient as possible and persistent surveillance opts to visit each point as frequent as possible to keep every point of the whole area "up to date". Generally speaking, the majority of methods for single- and multi-robot CPP consist of finding the most efficient path, often with complex computations. With persistent surveillance the emphasis is more on revisiting the point that has last been visited. CPP stops where persistent surveillance begins. Subsequently, the majority of CPP methods are not suitable for random dynamic environments. This has to do with the fact that many CPP methods use computations that are useless in such an environment, and methods that are able to deal with these environments are time and energy-inefficient. Also, CPP operates in a relatively smaller time window compared to persistent surveillance. The goal for CPP is to cover each point once where often a robot needs to perform a specific task at a specific point of interest, and it is done. Persistent coverage aims to monitor a specific area over a larger span of time. Therefore, it needs to be capable of adjusting to changing environments with ease and not completely restart from the beginning.

Conclusion

Table 2-1: Characteristics of swarm robotic CPP and swarm robotic persistent surveillance

Characteristics	SR CPP	SR persistent surveillance
Robust	Yes	Yes
Scalable	Yes	Yes
Dynamic environments	Yes	Yes
Inter-robot communication	limited	limited
Model	Probabilistic	Probabilistic
Performance	Poor	Good

The result of a swarm robotic coverage method is the dispersion of robots to a homogeneous distribution through the environment. This dispersion is time and energy-inefficient, however, when ultimately distributed through the environment, robots roughly stay in place and a constant optimal coverage is achieved. A comparison between the two methods are visible in table 2-1. The methods have the same characteristics, although the influence on the performance is different. As discussed in the previous section with persistent surveillance, every point or cell needs to be visited as frequently as possible. The emphasis is not on the time of the first visit of a point, but rather on what follows. This eliminates the time inefficiency of the first coverage that is unavoidable with a swarm robotic coverage method.

To conclude, a swarm robotic coverage method has specific innate characteristics. These characteristics have a positive effect on a swarm robotic persistent surveillance method, yet a negative effect on a swarm robotic CPP method. Therefore, a swarm robotic coverage method is more suited for persistent surveillance.

2-3 Robotic stigmergic coverage literature

In this section, the idea of stigmergy applied in robotic coverage is researched. To see if stigmergy could be the answer for a persistent surveillance method to have it communication-free without the need for ideal sensors and actuators. Literature is consulted to see whats tigmergic approaches are currently proposed and what the limitations and advantages ofthese methods are.

2-3-1 Stigmergy

Stigmergy was first discovered in 1959 by the biologist Pierre-Paul Grassé, where he observed termites displaying this behaviour. Since then it was discovered that many organisms exhibit this behaviour, such as ants, bees and even some bacteria. Ants use stigmergy by depositing pheromones, as discussed in chapter 1. When an ant smells pheromones, it is stimulated to follow this trail and as a consequence, reach the food source. The advantage of using stigmergy is the efficiency at which an agent can communicate with numerous other agents without direct communication. In this way, there is no need for intelligent agents with memory

or even awareness of each other. The result of colonies using stigmergy is a complex behaviour relative to the simple individual without any planning or central control. This renders the structure robust and scalable. These characteristics are desirable for robotics as well, and literature is trying to copy this behaviour [20].

2-3-2 Literature

In literature, stigmergy is occasionally used for robotic coverage. There are three general approaches with regards to applying stigmergy in swarm robotics:

1. Physical pheromones, e.g. odour, heat or amount of dust after cleaning.[2][21][22] [23] [3][24][25]
2. Digital pheromones, often with Radio-Frequency Identification (RFID) tags. These are used to store information and could be deployed in the environment beforehand or while covering.[26] [27]
3. Robots itself become stationary and act as a mark/information on the environment. [28]

Physical pheromones:
In 1999, the authors of [2] where the first who proposed three stigmergic robotic coverage methods named Edge-Ant-Walk. It presents systematic methods for local, cue-based operation of a group of robots that solves the coverage problem for unknown discrete environments. Stigmergy is used here to decide the next move and minimize revisits. The environment is composed of small rectangular tiles, and the intensity of the deposited pheromones decrease in time. In essence, robots sense which adjacent tiles are visited last by measuring the intensity of the pheromones/odour and move to the tile last visited. The method is less prone to noise since it depends on the relative difference of the intensity rather than the absolute intensity. This is a deterministic motion policy. Lastly, the algorithm can deal with changes to the environment. Since the pheromones on these places are absent, the robots will move towards those tiles. Two years later, the authors of [21] proposed a similar method named Vertex-Ant-Walk. The principle is the same. Only, agents place the pheromones on the vertex instead of the edges. The authors of [22] built on this idea and extended it to unknown continuous environments, where they proved that this method works for CPP as well as for persistent surveillance. This method is interesting since it requires homogeneous robots that are memoryless and only communicate via stigmergy. However, they do assume perfect sensors and actuators and assume that the robots have different clock phases when using multiple robots.

The most relevant and closest model that is proposed in literature is from the authors of [1]. They propose an extensive persistent surveillance method that is based on the same idea of this thesis. Inverting the effect of pheromones so that pheromones repel other robots instead of attract. The robots do not need to communicate other than via stigmergy, and limited sensors and actuators are considered. Although swarm robotics was not a well-established field at that time, the authors mention that this method is suited for teams of basic robots without memory. Thus, signalling the potential of combining stigmergy and swarm robotics

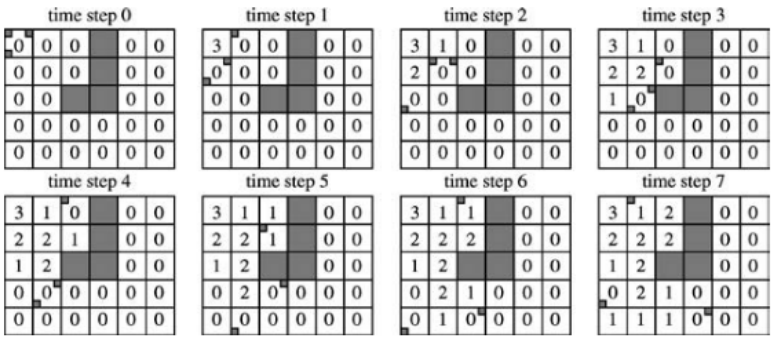


Figure 2-1: Coverage behavior of ant robots that use node counting. [1]

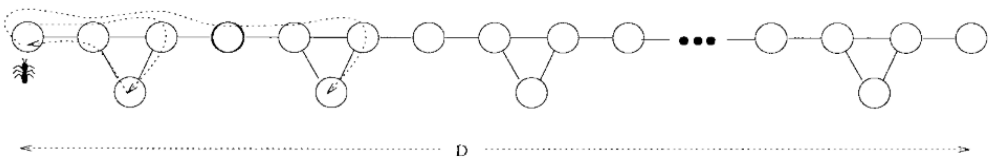


Figure 2-2: Hard case for a deterministic decision policy. There are n vertices, and about $1.25n$ edges. The diameter is about $0.8n$, and the time needed to traverse it may be as long as $\mathcal{O}(n^2)$. The dotted arrows show the worst case where each triangle of vertices is a "trap" that causes the ant to go back to its starting point. [2]

with persistent surveillance. The general idea is based on the method of node counting. Robots associate a value with each vertex that counts how often they have visited the vertex already, as depicted in figure 2-1. The values can be seen as markings that a robot leaves, identical with stigmergy. Consequently, the robot moves to the neighbouring vertex with the lowest value. Subsequently, they apply this to physical ant-robots. A major problem that they encounter is the saturation of the environment with markings. The environment gets so saturated with markings over time that it is hard to determine the smallest value of the neighbouring vertex. Therefore, after a certain saturation limit, the performance gets worse.

Similarly, this paper [23] is specifically focused on de-mining an area and propose multiple CPP methods where robots need to cooperate if one finds a mine. A discretized unknown environment is considered. The ant-based exploration process of the robots use scent in order to decide which direction to go, equivalently to the previous methods. The distinct part is that the robots decisions are based on a probabilistic model depending on the intensity of the measured scent. The robot is more likely to choose the direction where the scent is the least. The method is not deterministic. This is desirable in order for robots not to get stuck and converge faster. For instance, figure 2-2 displays a phenomenon where a robot keeps circling back. This behaviour is mitigated by adding heuristics.

A different approach is from the authors of [3] and [24], for which they later on provided a mathematical proof in [25]. This persistent surveillance method is of very low computational complexity without the need for idealized sensors and actuators. The only communication is stigmergic. It can deal with unknown environments and additionally with an extension that they provide it has the potential of dealing with dynamic obstacles. The general idea

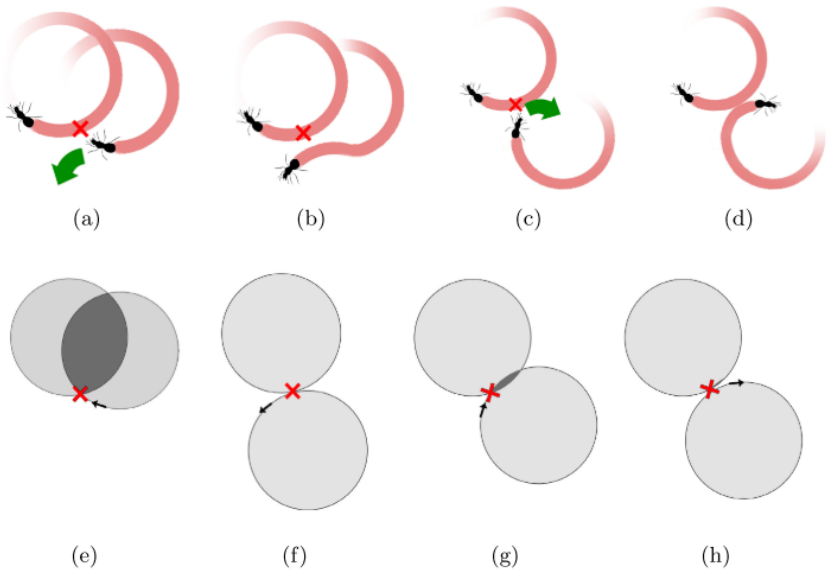


Figure 2-3: StiCo coordination principle: (a)-(b) before and after pheromone detection by internal sensor. (c)-(d) before and after pheromone detection by an external sensor. (e)-(f) covered area before and after pheromone detection by internal sensor. (g)-(h) covered area before and after pheromone detection by an external sensor. [3]

is that homogeneous robots move in a circular motion complementary to a Dubins vehicle, hence enclosing a circular area. These robots constantly deposit pheromones signalling the respective borders of the area they enclose, while changing their motion if they sense any other pheromones as depicted in figure 2-3. In this way, proved with simulations and a mathematical proof the robots disperse and converge to an equilibrium where every robot covers its own area. Consequently, the maximum coverage is achieved. This is directly the limitation of this method since it assumes successful coverage when there are no intersecting areas of individual robots. Subsequently, the maximum possible fraction of a square which can be covered by a set of disjoint identical circles is 78.5%. This is the maximum amount that can be covered with this approach. The idea is interesting since it only relies on stochastics and not on deterministic decisions of the robots. This allows for persistent surveillance of non-convex unknown complex environments.

Digital pheromones:
Some research focuses on stigmergy with RFID tags. RFID tags store information that can be read from a small distance. The authors of [26] propose a stigmergic coverage method where RFID tags are placed in the floor in a hexagonal grid. Robots can store information on RFID tags. While having a critique on installing external sensors, they place external RFID's themselves. This has high installation costs and is not suited for dynamic and completely unknown environments. The authors of [27] take an opposite approach where robots deploy RFID tags in order to build a network of reachable locations. Subsequently, a global path planning method is used when all tags are placed. At first, the robots plan a path based on the local view that they have. The fundamental problem in local exploration is minimizing the overlapping paths of multiple robots. This method could result in visiting multiple local

minima for a more extended period. The other global exploration method that is suggested requires communication other than stigmergy, which is not desirable. There is another problem with RFID tags used in this way. The robots place RFID at every pre-defined meters, which results in a grid of RFID tags. Information of multiple robots can be stored on these tags. However, because these tags are placed on the surface, it is very static, which is a problem for dynamically changing environments and could make the information on the tags obsolete.

Stigmergy via robots:
The authors of [28] present an inventive approach where robots itself become stationary and point to the direction of the objective. This method does not go in further detail about the coverage itself, but it is mentioned here for completion.

2-4 Event-triggered control

In the case of swarm robotics it is not unthinkable to have thousands of robots operating in a single swarm. In such a large multi robot system it is, generally, important to limit communication to avoid network congestion. With swarm robotics this is avoided by relying on local communication. However, there are two related problems with regards to the use of stigmergy in swarm robotics. Firstly, in current literature using stigmergy in swarm robotics is to continually deploy pheromones. This will sometimes lead to congestion of the environment with markings or pheromones such that it is difficult to measure [1]. Secondly, as discussed in section 2-5. Robots have a limited pheromone storage. It is therefore important to limit the pheromone usage while ensuring performance to make swarm robotics a viable solution in real life. In the field of control this is achieved through event-triggered control (ETC). ETC is used to limit the network congestion while keeping desired behaviour of the system. Moreover, similarities could be drawn between a stigmergic persistent surveillance method and multi-agent event-triggered consensus method, which is a sub-field of ETC. The goal for a stigmergic persistent surveillance method could be seen as the equalization of the density of pheromones everywhere in the environment. Which results in homogeneously distributed robots through the environment. This section consults current literature on the questions: What is Event-Triggered Control and how is it applied to multi-robot systems?

2-4-1 Introduction to event-triggered control

The general approach for implementing feedback control laws on digital platforms is periodic control. Where input and output is periodically updated through the network. With the increasing popularity of wireless control systems this method reaches its limit. Energy, computation and communication constraints need to be considered. Event-based control offers a great solution in satisfying these constraints, however it introduces new theoretical and practical challenges. Event-based control models are harder to analyse and to design. The discussion between periodic and event-based control is going on for years. Already in 1999 the authors of [29] and [30] showed advantages of using event-based control,which motivated the development of innovative stabilizing feedback control laws based on event-based implementations. In figure 2-4 the five parts of an event-based control loop are depicted. In ETC an event generator is added that monitors a certain event triggering condition. When this

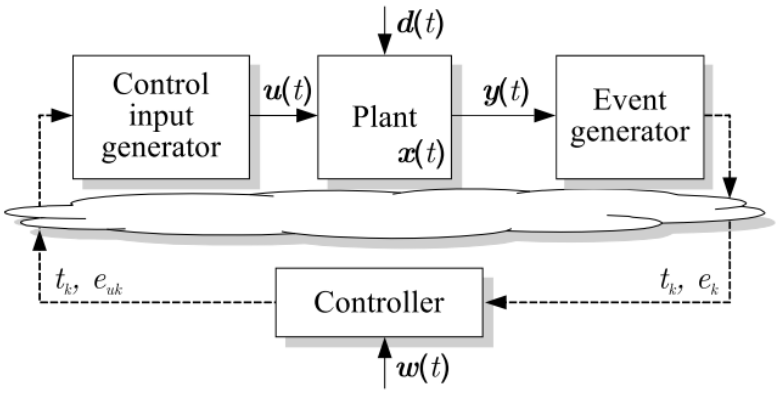


Figure 2-4: Event-based control Loop. [4]

condition is violated it sends information through the communication network in order to get an updated control input. An example with a linear plant is given from [31]. This is the most general case, however in most cases changes must be made to make it situation specific [32].

$$\frac{d}{dt}x_p = A_p x_p + B_p u, \quad x_p \in \mathbb{R}^{n_p}, u \in \mathbb{R}^{n_u} \tag{2-1}$$

With linear feedback control law.

$$u = Kx_p \tag{2-2}$$

Which stabilizes the closed loop system.

$$\frac{d}{dt}x_p = A_p x_p + B_p K x_p \tag{2-3}$$

This is a classical linear closed-loop system where the input is continuously updated and the system is asymptotically stable. However, when implementing the control law (2-2) on a digital platform it is due to obvious reasons not possible to continuously update the input. As previously mentioned, one way to solve this is by periodically computing the input and keeping the input constant in between the updates. An alternative way is to use unsatisfactory performance, instead of time, to be the deciding factor on when to update. In this way the input gets updated when the system is about to violate the performance metric. Often a lyapunov function is used , denoted as $V(x_p) = x_p P x_p^T$, for the closed loop system (2-3). Where $P > 0$ and P is symmetric. The Lypapunov function needs to satisfy :

$$\frac{d}{dt}V\left(x_p(t)\right) = \frac{\partial V}{\partial x_p}\left(A_p + B_p K\right)x_p = -x_p^T Q x_p \tag{2-4}$$

Where $Q > 0$. This ensures that $V(x_p)$ decreases along the solutions of the closed loop system (2-3). The rate of decrease is determined by Q. When a slower rate of decrease is tolerated a solution of an event-triggered implementation is required to satisfy

$$\frac{d}{dt}V\left(x_p(t)\right) \leq -\sigma x_p^T Q x_p \tag{2-5}$$

for an arbitrary σ. $\sigma < 1$ would be a slower rate of decrease for V. This suggests that in order to keep the inequality (2-5) satisfied, the input (2-2) needs to be updated when it is close to being violated. The assumption is that in between updates the input is held constant. That is called sample-and-hold and is depicted as:

$$u(t) = u\left(t_k\right) \quad \forall t \in [t_k, t_{k+1}[, k \in \mathbb{N} \tag{2-6}$$

and $\{t_k\}_{k\in\mathbb{N}}$ is the sequence that represents the time instances where the input is recomputed and updated, called triggering times. In order to write down the dynamics of the corresponding closed loop system where the input is updated only when it is necessary an error is introduced.

$$e(t) = x_p\left(t_k\right) - x_p(t) \quad \forall t \in [t_k, t_{k+1}[, k \in \mathbb{N} \tag{2-7}$$

Which results in an alternative version of $\frac{d}{dt}V\left(x_p(t)\right)$,namely :

$$\begin{aligned} \frac{d}{dt}V\left(x_p(t)\right) &= \frac{\partial V}{\partial x_p}\left(A_p + B_pK\right)x_p(t) + \frac{\partial V}{\partial x_p}B_pKe(t) \\ &= -x_p^T(t)Qx_p(t) + 2x_p^T(t)PB_pKe(t) \end{aligned} \tag{2-8}$$

Now an expression for $\frac{d}{dt}V\left(x_p(t)\right)$ is derived. These are the dynamics of the lyapunov function with an event-triggered update policy for the input. When equation (2-8) is substituted in the inequality of equation (2-5) the triggering condition appears.

$$\begin{bmatrix} x_p^T(t) & e^T(t) \end{bmatrix} \begin{bmatrix} (\sigma-1)Q & PB_pK \\ K^TB_p^TP & 0 \end{bmatrix} \begin{bmatrix} x_p(t) \\ e(t) \end{bmatrix} \leq 0 \tag{2-9}$$

The triggering time is when this equals zero. With other ETC methods this could be different, which will result in another quadratic triggering condition. The goal of the triggering condition is always to ensure a certain decay of the lyapunov function. The authors of [33] provide a proof that a lower bound for inter-execution times always exist for a linear system with linear state-feedback controller. For other systems a lowerbound exists under the right assumptions. This is necessary to guarantee that the system does not display zeno behaviour. Zeno behaviour is infinitely many triggers in a finite amount of time.

The basic idea behind ETC is to limit the communication or use of resources, while minimizing the consequent loss of performance. The intuition behind it is clear. Do not fix what is not broken. ETC will always have the trade-off between performance and amount of triggers. By choosing a triggering condition that ensures certain performance a trigger will only happen at times when it is necessary to steer the system in the right direction.

2-4-2 Multi-agent event-triggered control

Generally, event-triggered control is used for single-agent systems. However, in literature there is a growing interest in event-triggered control for multi-agent systems (MAS). With MAS it is even more critical to reduce the communication, since with multiple agents the network is easily congested. In the field of MAS there are several general tasks, like flocking, rendezvous [34], formation control [35], deployment and consensus. However with regards to ETC the focus in literature is on consensus [5][36][37], which is the problem of how could all agents in a system eventually agree on a quantity of interest.

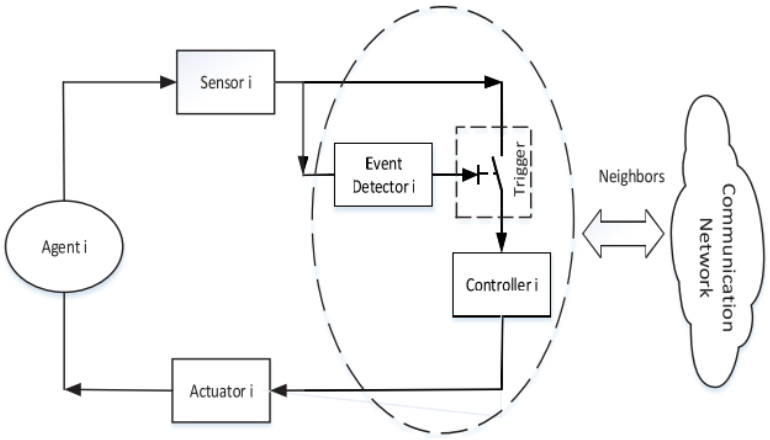


Figure 2-5: Basic event-triggered control configuration for agent i. [5]

The authors of [5] provided a general overview for MAS consensus algorithms. Consider a MAS that consists of N agents with linear dynamics:

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t), i = 1, 2, \ldots, N \tag{2-10}$$

Where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are constant matrices and $x_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}^m$ are the the state vector and input of agent i. The agents are assumed to operate on a graph G that is either connected or a spanning tree. The MAS (2-10) reaches consensus if there exists a suitable control protocol $u_i(t)$ such that every agent's state eventually reaches a common value. The most commonly used consensus control protocol is:

$$u_i(t) = K \sum_{j\in N_i} w_{ij} \left(x_j(t) - x_i(t)\right) \tag{2-11}$$

where K is a control gain matrix. For reasons that are previously mentioned it is practically not realistic to assume access to continuous signals from other agents. Consequently, time-triggered control and event-triggered control are used.

In figure 2-5, the event-triggered control-loop for an individual agent is depicted. The event detector evaluates the output, checks if the event triggering condition is violated and if violated sends an execution signal. There are multiple approaches for the event triggering condition:

- **Centralised:** All the agent's measurements are used.
- **Decentralised:** Only the agent's individual information is used.
- **Distributed:** The information from itself and its neighbours are used.

The focus in literature is namely on decentralised and distributed event triggering conditions, due to the impracticality of designing a triggering condition that uses the information of all agents. Lastly, it is imperative that there exists a lower-bound on the interevent time, otherwise Zeno behaviour occurs.

2-5 Discussion

Current methods in literature regarding stigmergic robotic coverage are undoubtedly able to successfully cover an unknown dynamic environment with limited sensing and actuating abilities and simple robots. This is mainly achieved with methods using physical pheromones. These methods proofed experimentally or mathematically to be convergent in discrete and continuous environments. Three main points could be distilled from literature. Firstly, although the authors of [1] came a long way in combining the node counting method with stigmergy and swarm robotics, it did only consider a maximum of fifteen robots and is deterministic. The reason in current literature for using stigmergy is to minimize the unnecessary revisits of points during coverage without complex computations and communication needed from the robot. When swarm robotics is taken into the equation, the focus of using stigmergy will be on the dispersion of the swarm of numerous robots through the environment without direct communication. Secondly, almost all current stigmergic coverage methods are deterministic. This is beneficial for CPP, but for persistent surveillance this is not desirable. Deterministic methods are inherently faster than probabilistic methods, however, not well suited to dynamic environments. On top of that, in swarm intelligence, stochastic choices are used to come to better solutions and avoid local optimums. Therefore, it is desirable for a swarm robotic coverage method to use a probabilistic approach. Thirdly, the use of pheromones is assumed to be infinite. This is not a viable assumption in real life, since there is a limited amount of pheromones that can be stored in a robot. Additionally, it is easy to congest the environment or network if all agents deposit pheromones constantly. For these reasons, the coverage method must be as durable as possible by minimizing the pheromones deposits. This could be achieved by applying a switching rule that is similar to a decentralised event triggering condition for the deployment of pheromones. The switching rule decides for each agent if it is necessary to deploy pheromones or not. Hence, optimizing pheromone usage.

2-6 Research goals

The goal of this thesis is to propose a stigmergy based swarm robotic model that solves the persistent surveillance problem. We intent the model to be robust, flexible and minimizing the performance metric age. On top of that, we are interested in making the model deploy pheromones as efficient as possible by including a switching rule that is inspired from event-triggered control.

In order to evaluate the proposed model, it will be compared with the persistent surveillance model of [1]. This work is the closest and most extensive of current robotic stigmergic coverage literature. Lastly, a demonstration is presented with a simulation of physical robots performing persistent surveillance with the proposed model.

Chapter 3

Problem Formulation and Model Definition

In this chapter, the persistent surveillance problem is formally defined. Subsequently, the model that solves this problem is proposed. The model consists of two parts. The decision making policy that is responsible for the direction that an agents goes next and the weight dynamics that models the dynamics of the pheromone-densities in the system. The switching rule is part of the weight dynamics and will be proposed there. At last, mathematically derived upperbounds are presented for the pheromone densities that show that the pheromones in the system will not explode.

3-1 Preliminaries

The first things that need to be considered are the spatial and temporal characteristics of the model. We consider the time to be discrete $t \in \mathbb{N}_+$ and the agents operating on a connected weighted graph. This simplifies the problem drastically while remaining relevant.

We model the environment to cover as a connected weighted graph:

$$\mathcal{G} := (\mathcal{V}, \mathcal{E}, w(t)).$$

Where $\mathcal{V}$ is a vertex set, $\mathcal{E}$ is an edge set, $|\mathcal{V}|$ is the total amount of vertices and $w(t) \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|}$ is a set of weights where each element corresponds to a specific vertex. An edge that connects i to j is referred to as $(ij) \equiv e \in \mathcal{E}$. A graph is connected if for every vertex pair $i, j \in \mathcal{V}$ there exists a set of edges $(iv_1), (v_1v_2), \ldots, (v_{kj}) \subseteq \mathcal{E}$ that connects i, j. Agents walk from vertex to vertex in one timestep. The set of agents is defined as $\mathcal{A} := \{1, 2, \ldots, n\}$, and the position of agent a is stored in the vector $x \in \mathbb{R}^{|\mathcal{A}|}$ in $x_a(t)$. The set of neighbouring vertices is defined as $\mathcal{N}_i := (j \in \mathcal{V} | (iv), (vj) \in \mathcal{E})$ for $i \in \mathcal{V}$. Let $q \in \mathbb{N}_0^{|\mathcal{V}|}$ be the number of agents present at every vertex.

The following matrices and metrics defined correspond to a certain graph and are a way of representing the graph in a structural manner.

The weight matrix:
The weight matrix $W(t) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ indicates the transition weights.

$$W_{ij}(t) := w_j(t)\ (ij) \in \mathcal{E}. \tag{3-1}$$

The adjacency matrix: The adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ depicts the adjacent vertices of each vertex. The formula yields,

$$A_{ij} := \begin{cases} 1, & \text{if } (ij) \in \mathcal{E}. \\ 0, & \text{else.} \end{cases} \tag{3-2}$$

The two layer adjacency matrix:
The two layer adjacency matrix $A^* \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ depicts the vertices that an agent can sense. In this case the agents can sense two vertices away. The formula yields,

$$A^*_{ij} := \begin{cases} 1, & \text{if } \exists k \in \mathcal{V} : \{(ik),(kj)\} \in \mathcal{E}. \\ 0, & \text{else.} \end{cases} \tag{3-3}$$

The two layer weight matrix:
The two layer weight matrix $W^* \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ indicates all the transition weights that an agent can sense. The formula yields,

$$W^*(t) = A^* diag(w(t)). \tag{3-4}$$

The out degree:
The out degree $g \in \mathbb{R}^{|\mathcal{V}|}$ is a vector with the number of neighbouring vertices in the sense radius of each vertex. Where,

$$g_i := \sum_{k\in\mathcal{V}} A^*_{ik} \tag{3-5}$$

With weighted out degree $d^{out} \in \mathbb{R}^{|\mathcal{V}|}$:

$$d^{\text{out}}_i(t) := \sum_{k\in\mathcal{V}} W^*_{ik}(t) \tag{3-6}$$

3-2 Problem formulation

As discussed in section 2-2-2, persistent surveillance is the minimisation of the maximum age of every cell (or point) monitored over an extended period of time. In the case of a graph, every vertex has an age. Let $age_i(t) \in \mathbb{N}$ be the age of every vertex i, and it evolves for any $i \in \mathcal{V}$ as:

$$age_i(t+1) := \begin{cases} age_i(t)+1, & \text{if } q_i(t+1) = 0. \\ 0, & \text{else.} \end{cases} \tag{3-7}$$

The goal is to minimize $||age(t)||_\infty$. This would result in every vertex being visited as frequent as possible.

3-3 Model

In the previous chapter, the main requirements for a stigmergy based swarm robotic persistent surveillance model are discussed. These are mainly:

1. The model is decentralized and agents can only use local information and local communication.
2. The model is probabilistic by definition.
3. The number of measured pheromones should affect the decisions/probabilities, such that a robot has a higher probability of leaving areas with a higher pheromone density.

The first requirement is necessary to satisfy the requirements of a swarm. This renders the model robust, flexible and scalable. The second requirement is set because probabilistic models are better suited to persistent surveillance. The model will be able to deal with dynamic obstacles in this way and will not get stuck in local minimums. The last requirement is the stigmergic behaviour that this paper is interested in, to ensure that the agents disperse through the environment.

3-3-1 Decision making model

The first important part of the persistent surveillance model is the decision-making model. According to the decision-making model agents decide which vertex to go next. We consider the robots to operate on a graph. Therefore, the probability decision model can be represented as a probability matrix $P(t) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where:

$$\Pr\{x_a(t+1) = j \mid x_a(t) = i\} = P_{ji}(t), \quad a \in \mathcal{A}. \tag{3-8}$$

The matrix is column-wise, summing to 1. Element P_{ji} is defined as the transition probability: the probability for a robot to traverse from vertex i to vertex j. Using a stochastic matrix is a convenient way to express the stochastic "choices" of a robot at a specific location. In this case, $P(t)$ is dynamically changing with every time step. The dynamics of the probability matrix are only affected by the vertices' weights and are responsible for the desired behaviour of robots distancing from pheromone-dense areas. There are multiple approaches for the updating formula of the probability matrix. With stigmergy problems, often high pheromone densities attract other agents. However, this results in the aggregation of robots in pheromone dense area's. To invert this behaviour, the updating policy yields,

$$\begin{aligned} P^1_{ji}(t) &= \frac{\Big(\gamma W_{ij}(t) + (1-\gamma)B_{ij}(t)\Big)^{-1}}{\sum_{k\in\mathcal{V}}\Big(\gamma W_{ik}(t) + (1-\gamma)B_{ik}(t)\Big)^{-1}}, \quad \forall j : (ij) \in \mathcal{E}, \\ &where, \\ B_{ij} &= \frac{(\sum_{k\in\mathcal{V}} W_{jk}) - W_{ji}}{\sum_{k\in\mathcal{V}} A_{jk} - 1}. \end{aligned} \tag{3-9}$$

Here, $\gamma \in (0,1)$ is a scaling parameter that scales the influence of the closer vertices vs the farther vertices. The agents can sense up to two vertices away. Thus, B_{ij} represents the average weight of the neighbouring vertices of vertex j, excluding vertex i. Depending on γ, the further away vertices have a bigger or smaller impact on the probability of moving one step closer or further away from them. In the end, the total transition probabilities are the inverted normalised weights around a vertex. One downside is that it is impossible to tune the model's aggressiveness, i.e. prioritising the adjacent vertex with the lowest weight. This is possible by the next approach. Let $P^2(w(t))$ be a probability matrix where

$$P^2_{ji}(w(t)) = \begin{cases} 1 & \text{if } W_{ij} = \inf_k \{W_{ik}(t)\}. \\ 0 & \text{else.} \end{cases} \tag{3-10}$$

This matrix would result in a non-probabilistic decision model that only will go to the best solution at hand. Combining P^1 and P^2 results in

$$P(t,\varepsilon) := \varepsilon P^1(w(t)) + (1-\varepsilon)P^2(w(t)). \tag{3-11}$$

This formula interpolates between the probabilities of the normalised weights around a vertex and the deterministic model. This results in a probabilistic decision model where ε is the probability of following the probability matrix. This is similar to the ε-greedy method used in reinforcement learning.

3-3-2 Weight dynamics

The weight dynamics is the second and last part of the complete coverage model. Agents deploy pheromones on the graph by adding weight on the weighted graph. These weights represent the density of pheromones on their specific vertices, where a higher value is a higher density. The weight dynamics consist of an evaporation term and a deployment term. Let $\rho \in (0,1)$ be the evaporation factor, $\Theta(W^*(t),t) : \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{N}_0 \to \mathbb{R}^{|\mathcal{V}|}_{\geq 0}$ be the deployment function and $\delta_a(W^*(t),t) : \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|} \times \mathbb{N}_0 \to (0,1)$ be the switching rule, a binary parameter that determines if deployment of weights is needed. This is calculated by each individual agent. Subsequently, the weight dynamics are:

$$w_i(t+1) = (1-\rho)w_i(t) + \rho\delta_a\big(W^*(t),t\big)q_i(t)\Theta_i\big(W^*(t),t\big). \tag{3-12}$$

Deployment function Θ

The deployment function has a significant impact on the performance of the coverage model. It is equivalent to the number of pheromones that each ant places. In this section, three deployment functions are considered.

The most obvious, basic and often used deployment function is incrementing the weight by a constant of one unit, regardless of the evaporation factor.

$$\Theta^1_i(t) = \frac{1}{\rho q_i(t)}. \tag{3-13}$$

Another way is to use available information to deploy more or less weights on the specific vertex depending on the surrounding weights. For instance, it can be useful to draw other agents near you and deploy less weight on the vertex where there are lots of vertices with low weight in the vicinity. This has the potential of adapting faster to a new, not covered environment. The following deployment function uses the average of the neighbouring vertices. Apart from incrementing the weight by one unit, it adjusts this unit slightly depending on the average weight of the surrounding vertices:

$$\Theta_i^2(W^*(t),t) = \frac{(1-\lambda)+\lambda\frac{d_i^{out}}{g_i}}{q_i(t)}, \tag{3-14}$$

Where $\lambda \in (0,1)$ is a scaling parameter. A slightly different approach is to take the minimal value of the neighbouring vertices. The intention is to deploy less weight when an agent senses any long not covered vertices, with a low weight. This results in:

$$\Theta_i^3(W^*(t),t) = \frac{1+\lambda \min_k W_{ik}^*(t)}{q_i(t)}. \tag{3-15}$$

These factors aid when dynamic obstacles move and create an empty area. Consequently, agents in the vicinity will reinforce surrounding vertices less when they sense the empty area to attract other agents to it. This results in three possible deployment functions.
The incremental deployment function:

$$\Theta_i^1(t) = \frac{1}{\rho q_i(t)}. \tag{3-16}$$

The average deployment function:

$$\Theta_i^2(W^*(t),t) = \frac{(1-\lambda)+\lambda\frac{d_i^{out}}{g_i}}{q_i}. \tag{3-17}$$

The minimum deployment function:

$$\Theta_i^3(W^*(t),t) = \frac{1+\lambda \min_k W_{ik}^*(t)}{q_i}. \tag{3-18}$$

3-3-3 ETC-inspired switching rule δ

We now want to address the switching rule with the question of when to deploy and when not to deploy pheromones. The objective is to limit pheromone usage while keeping desired performance. For pheromone efficiency, it is more productive and relevant to monitor and minimise the deployment instances per time-step. In this way, the final result has importance for other research fields and could be extended to, for instance, reinforcement learning.

There are some challenges to the switching rule. The agents will be relatively simple, autonomous and do not communicate via a wireless connection with eachother. Therefore, it is needed to have a decentralised individual switching policy, where individual agents estimate or measure specific properties based on locally available information. Subsequently, the switching policy should not require much memory since the agents are relatively simple.

This is something that can be stretched when necessary but has to be kept in mind. The only available local information that an individual agent has are the weights and the number of agents in the neighbour set $\mathcal{N}_i$, where $i \in \mathcal{V}$ is the vertex where the considered agent is located.

Before we propose the actual switching rule, there is one addition. Normally every agent deploys pheromones every time-step, despite the number of other agents present at that vertex. Instead of all agents deploying pheromones when they are on the same vertex, the addition is that only one agent deploys the total amount for that vertex. This reduces the deployment instances already significantly. It can easily be implemented by giving every agent a rank. Subsequently, the agent with the highest rank on the specific vertex is obliged to deploy pheromones. Since this reduction of deployment instances does not affect the weight dynamics, it is not considered a switching rule.

Finally, we consider the switching rule. The switching rule must omit the deployment instances that have the most negligible influence on the total performance. For instance, if a vertex with already a high weight gets covered, it is unnecessary to deploy even more weight on this vertex since it already has a higher weight and thus a lower probability of being covered soon after. To achieve this, several switching rules have been tried that used only the amount of nearby agents or some kind of weight distribution metric. Unfortunately, these rules did not outperform a random deploying switching rule while using all the available information. This is because these switching rules decoupled the weight of a vertex from the age of that vertex. It became apparent that the age and the weight of a vertex should be linked together. If a vertex has just been covered, it should have a relatively high weight than the surrounding vertices. With that in mind, the following switching rule will be proposed. With this rule, the weights of the vertex where the current agent is located are only deployed weights on if it has less weight than the average weight of all surrounding vertices. In other words, there is a smaller probability of going to the just covered vertex since the policy ensures a higher weight relative to the neighbouring weights. Hence, the relation between the weight and age of a vertex is reserved. Let $K \in \mathbb{R}$ be a scaling parameter. The final switching rule yields:

$$\delta_a(t) = \begin{cases} 1, & \text{if } w_i(t) < K \frac{\sum_{j \in \mathcal{N}_{x_a}} w_j}{|\mathcal{N}_{x_a}|} \\ 0, & \text{if other.} \end{cases} . \tag{3-19}$$

Parameter K affects the performance and deployment instances. A lower K results in a worse coverage performance, but less deployment instances. A higher K has the opposite effect.

3-4 Weight Bounds

It is evident for $\Theta_i^1(t)$ that the weights will not explode. It only adds a constant, and in the worst case, every agent places a constant amount of pheromones. This is, however, not so evident for the other deployment functions. These derive their pheromone deployment amount from their surroundings and thus are vulnerable to a positive feedback loop. An upper bound can be derived for $\Theta_i^2(W^*(t), t)$ and $\Theta_i^3(W^*(t), t)$ to ensure that the weights remain bounded.

The weight dynamics for $\Theta_i^2(W^*(t),t)$ are:

$$w_i(t+1) = \begin{cases} (1-\rho)w_i(t), & \text{if } q_i = 0. \\ (1-\rho)w_i(t) + \rho(1-\lambda) + \rho\lambda\frac{d_i^{out}}{g_i}, & \text{if } q_i > 0. \end{cases} \tag{3-20}$$

Let $\Omega(t) = ||w_i(t)||_1 \in \mathbb{R}$ be the sum of weights over all vertices. To find an upper-bound for $\Omega(t)$ the worst case scenario where every vertex has at least one agent is considered. Also let $V_n = \{i \in \mathcal{V} : q_i > 0\}$.

$$\begin{aligned} \Omega(t+1) &= (1-\rho)\Omega(t) + \sum_{i\in V_n} \rho(1-\lambda) + \rho\lambda\frac{d_i^{out}}{g_i}, \\ &\leq (1-\rho)\Omega_i(t) + \sum_{i\in\mathcal{V}} \rho(1-\lambda) + \rho\lambda\frac{\sum_{j\in\mathcal{N}_i} W_{ij}^*}{g_i}, \end{aligned} \tag{3-21}$$

$g_i \in (6,18)$ in a triangular lattice and $\frac{\sum_{j\in\mathcal{N}_i} W_{ij}^*}{g_i} \leq \frac{g_{max}\sum_i w_i}{g_{min}}$, which result in:

$$\begin{aligned} \Omega(t+1) &\leq (1-\rho)\Omega(t) + \rho(1-\lambda)|\mathcal{V}| + \frac{18}{6}\rho\lambda\Omega(t), \\ &\leq \big(1+\rho(3\lambda-1)\big)\Omega(t) + \rho(1-\lambda)|\mathcal{V}|. \end{aligned} \tag{3-22}$$

If $0 < (1+\rho(3\lambda-1)) < 1$, $\Omega(t) \to C$ for $t \to \infty$. Then, $\Omega(t+1) = \Omega(t)$ and yield the constant for $t \to \infty$:

$$\begin{aligned} (1-3\lambda)\Omega(t+1) &\leq 1|\mathcal{V}|(1-\lambda), \\ \Omega(t+1) &\leq \frac{1|\mathcal{V}|(1-\lambda)}{(1-3\lambda)}. \end{aligned} \tag{3-23}$$

For $\lambda \in (0, \frac{1}{3})$, $\rho \in (0,1)$ the weights will remain bounded.

Similarly, an upper-bound for the weights with deployment policy $\Theta_i^3(W^*(t),t)$ can be determined. The weight dynamics with $\Theta_i^3(W^*(t),t)$ are:

$$w_i(t+1) = \begin{cases} (1-\rho)w_i(t), & \text{if } q_i = 0. \\ (1-\rho)w_i(t) + \rho + \rho\lambda\min_k(W_{ik}^*), & \text{if } q_i > 0. \end{cases} \tag{3-24}$$

Then,

$$||w(t+1)||_\infty = (1-\rho)||w(t)||_\infty + \rho + \rho\lambda||\min_k(W_{ik}^*)||_\infty. \tag{3-25}$$

As upper-bound use $||\min_k(W_{ik}^*)||_\infty \leq ||w(t)||_\infty$, which results in:

$$\begin{aligned} ||w(t+1)||_\infty &\leq (1-\rho)||w(t)||_\infty + \rho + \rho\lambda||w(t)||_\infty, \\ &\leq \big(1+\rho(\lambda-1)\big)||w(t)||_\infty + \rho. \end{aligned} \tag{3-26}$$

If $0 < (1 + \rho(\lambda - 1) < 1$ when $t \to \infty$, then $||w(t+1)|| \to C$. As $t \to \infty$, substituting $||w(t+1)||_\infty = ||w(t)||_\infty$ in equation 3-26 yields,

$$\begin{aligned}(1-\lambda)||w(t)||_\infty &\leq 1, \\ ||w(t)||_\infty &\leq \frac{1}{(1-\lambda)}.\end{aligned} \tag{3-27}$$

The weights will remain bounded for $\lambda \in (0,1)$ and $\rho \in (0,1)$.

Weight Convergence

The previous section derived bounds for the parameters with regards to the different policies. For clarity sake, the results were:

- For the average deployment function: $\lambda \in (0, \frac{1}{3})$ and $\rho \in (0,1)$.
  This gives the upper-bound for the sum of weights when $t \to \infty$ of
  $$\Omega(t) \leq \frac{10|\mathcal{V}|(1-\lambda)}{(1-3\lambda)}.$$
- For the minimum deployment function: $\lambda \in (0,1)$ and $\rho \in (0,1)$.
  This gives the upperbound for the maximum weight when $t \to \infty$ of
  $$||w(t)||_\infty \leq \frac{10}{(1-\lambda)}.$$

These upper-bounds show that the weights do not explode when the requirements are met. Subsequently, these upper-bounds also give an indication for the equilibrium to which these policies converge. The weight dynamics can be closer examined to compute the equilibrium. For instance with $\Theta_{i3}(W^*(t), t)$, let $D \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ be a diagonal matrix with only $1's$ on the diagonal and $\mu(t) \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ be a diagonal matrix with $\{\min_k(W^*_{1k}), \ldots, \min_k(W^*_{||\mathcal{V}||k})\}$ on the diagonal. The weight dynamics in matrix formulation result in:

$$w(t+1) = (1-\rho)w(t) + \rho\Big(D + \lambda\mu\big(w(t)\big)\Big)q(t) \tag{3-28}$$

In order to compute the exact equilibrium of the weights, a stochastic process needs to be analysed. This is due to $q(t)$ that depends on the probability matrix $P(t)$, which in its turn depends on the weights of the vertices. Because of this coupling, it is challenging to determine the equilibrium. Since this is not the scope of the thesis, the weights will not be further analysed.

Chapter 4

Simulation

In this section, the proposed coverage model is simulated and examined. First, the performance metrics are explained in detail. Then, the parameters are optimized, and the complete coverage model with the three proposed deployment functions are compared without switching rule active. Next, the switching rule $\delta_a(t)$ is implemented on the best performing weight deployment function and simulated. Lastly, the results are compared with the coverage model from the work of [1]. There are two tests for the simulation. Both operate on a 101x51 triangular lattice, depicted in figure 4-1. The most extreme test covers an area with a wall and one passage in the middle for $T = 5000$ timesteps. This is depicted in figure 4-2a. The second test is to cover the area while three square objects move randomly through the environment for $T = 3000$ timesteps, shown in figure 4-2b. The timesteps vary with each test since it takes longer for the coverage methods to reach an equilibrium with the first test. The first test is designed such that the difference in the performance of different policies is increased. The second test represents the final objective where dynamic objects move randomly.

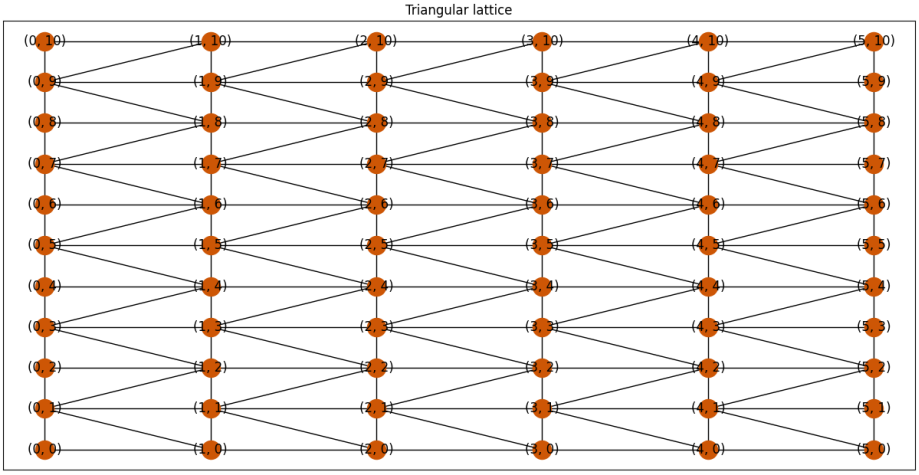
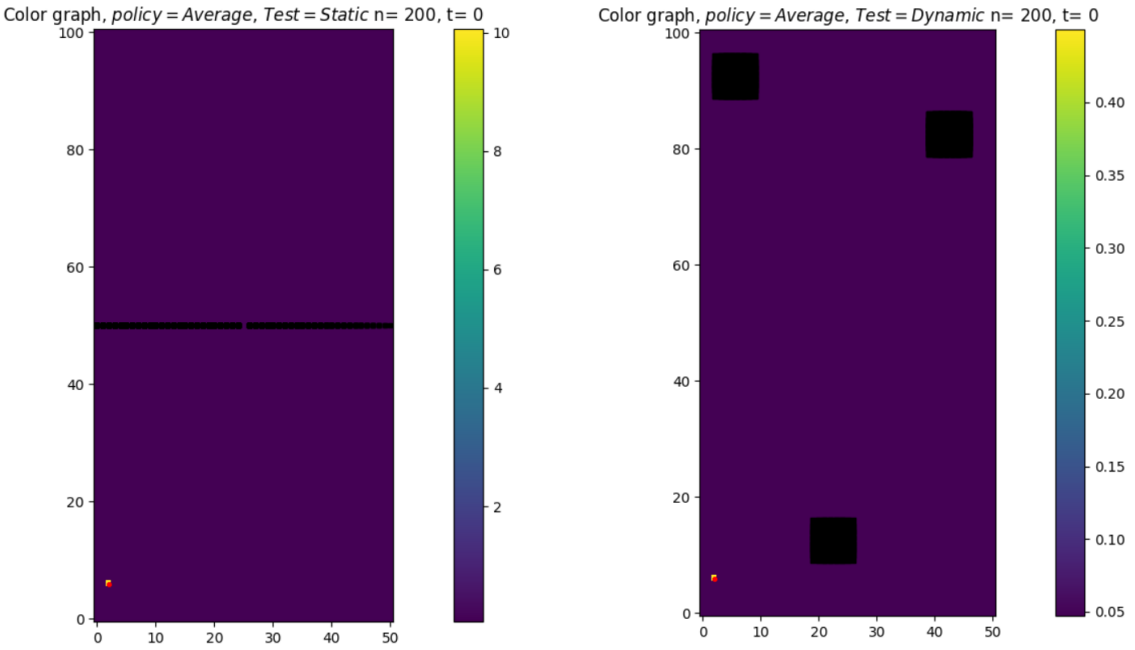


Figure 4-1: Triangular lattice



(a) Snapshot of the weight graph of the test with a wall and a passage in the middle.

(b) Snapshot of the weight graph of the test with dynamic objects.

Figure 4-2: Initial position of the tests

4-1 Performance metrics

The objective of persistent surveillance is to minimize $||age(t)||_\infty$. To evaluate the simulations, $||age(t)||_\infty$ for every t is shown. An example is depicted in figure 4-3, where the age of every vertex is visible.

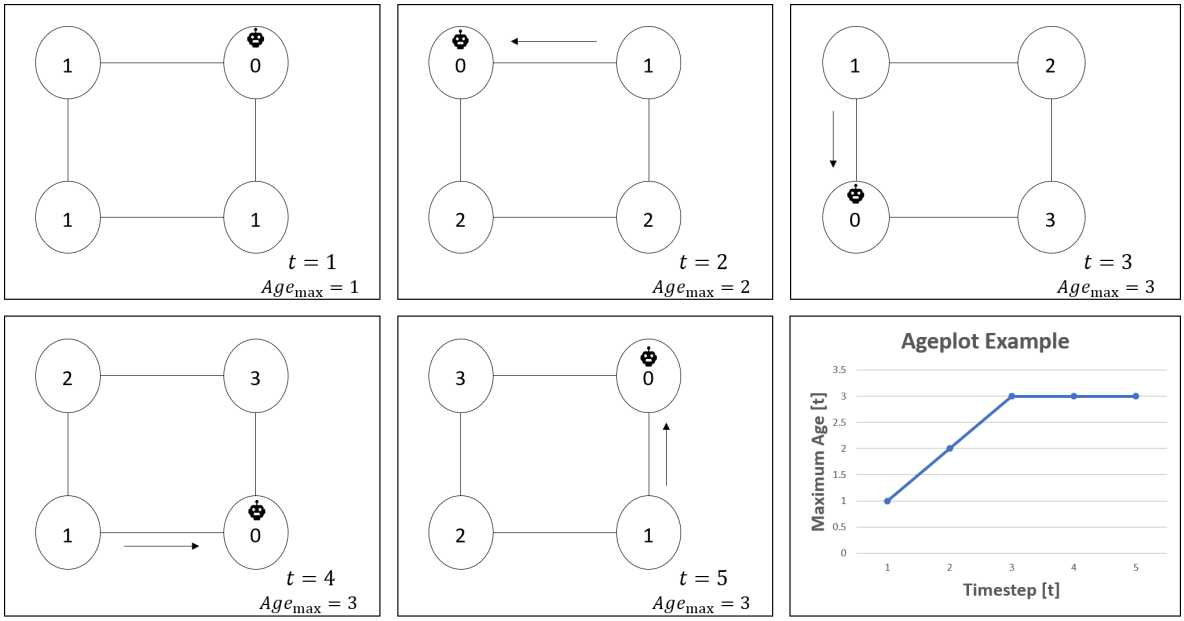


Figure 4-3: Example ageplot.

The next performance metric that monitors the pheromone usage is the number of communication instances per timestep. One communication instance is when an agent deploys pheromones, regardless of the number of pheromones deployed. This performance metric is for evaluating and comparing the pheromone usage of the specific coverage method. Usually, every agent deploys pheromones at every timestep. However, with the switching rule this is minimized. In the end, this thesis opts to minimize the deployment instances while ensuring the age to stay low.

4-2 Parameter Optimization

The first step is to optimize the parameters in the model. There are four relevant parameters. These are:

- γ, Scaling parameter for emphasis on the weight of the closer vertices.
- ρ, Evaporation factor.
- λ, The scaling parameter.
- ε, The deterministic vs probabilistic parameter.

These parameters are optimized with the dynamic objects test. In the end, the performance will also be compared with the static object test. With both tests, there are 100 agents, and the start node of all agents is at the vertex (3,6). With a pre-defined starting vertex, the performance of different methods can be compared more reliable.

4-2-1 Scaling parameter γ

The scaling parameter is visible in equation (3-9) and scales the effect of the weight of the closest vertices vs the farther vertices. With $\gamma = 1$, only the weights of the neighbouring vertices are taken into account for the decision model. With $\gamma = 0$, only the weight of the neighbours of the neighbours are taken into account. In figure 4-4 is shown that it has no beneficial effect to take the weight of other vertices than the direct neighbouring vertices into account. Therefore, $\gamma = 1$ is chosen for the remaining simulations, and equation (3-9) transforms in a regular normalization of the inverted weights.

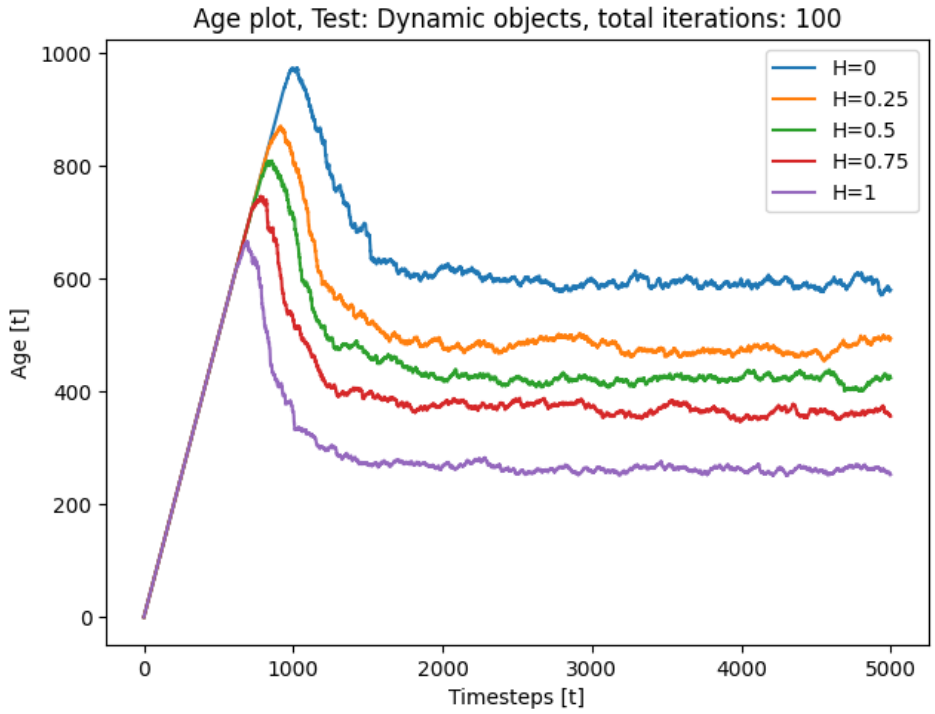


Figure 4-4: Incremental policy with different values for scaling parameter H

4-2-2 Evaporation factor ρ

The evaporation factor is responsible for the dissipation of the weights. It could be interpreted as the forgetfulness of the system. Therefore, it is convenient to view the halving time of the weights corresponding to a certain evaporation factor to evaluate the system. The formula

for the halving time of the weights is:

$$t_h = \frac{log(0.5)}{log(1-\rho)} \tag{4-1}$$

Where $t_h \in \mathbb{R}_{>}0$ is the halving time. The ideal halving time of the weights depends on the ratio of $R = \frac{\#vertices}{\#agents}$, since this is equal to the optimal amount of time steps per complete coverage. In a 101x51 grid with 100 agents, there are ≈ 52 nodes per agent to cover in a perfect scenario. The halving time has two requirements. Firstly, the weights should not decay too fast so that newly deployed weights affect only a few time-steps. This will happen when the halving time $\ll R$. Secondly, it should not be too big so that long ago deployed weights still affect the current situation. This will happen for halving time $\approx R$. In order to find the optimal evaporation factor, simulations are done with the incremental deployment function. With this deployment function, the evaporation factor only influences the performance and can easily be optimized. Figure 4-5 shows the result. The evaporation factors $0.06, 0.08$ and 0.1 have a similar performance. Table 4-1 depicts the corresponding halving times. A higher evaporation factor results in a faster first coverage. However, for persistent surveillance, a higher evaporation factor is not directly desirable, as explained previously. For this reason, an evaporation factor of $\rho = 0.06$ is chosen. It has a similar performance with a higher ρ. However, the halving time is kept in mind. With this evaporation factor after $52(\approx R)$ time-steps, deployed weights are declined with a factor of $\approx \frac{1}{23}$.

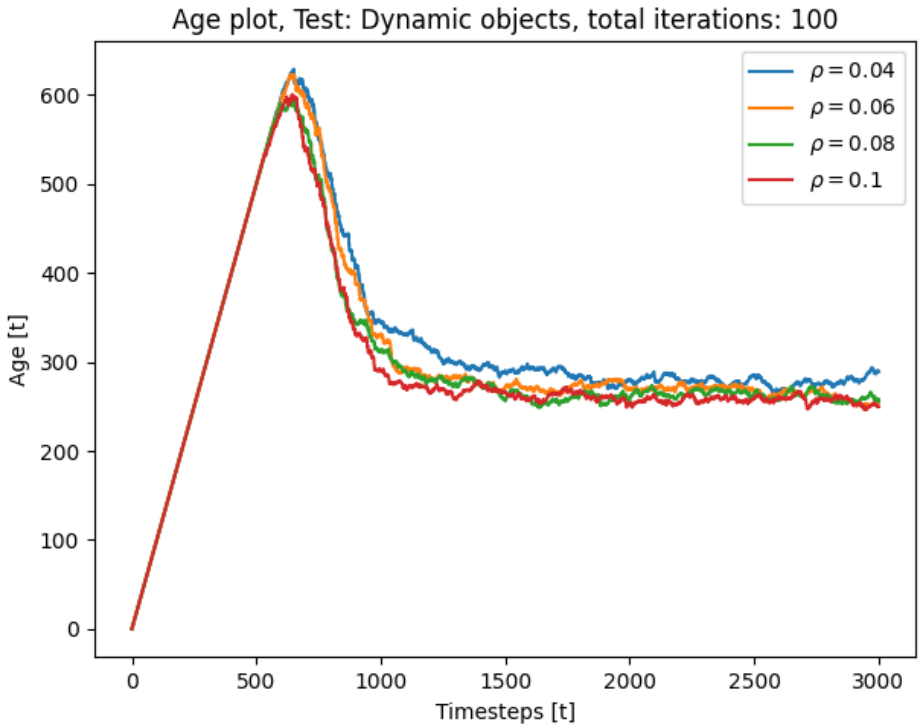


Figure 4-5: Incremental policy with different evaporation factors

ρ[-]	Halving time [t]
0.04	17.0
0.06	11.2
0.08	8.3
0.1	6.6

Table 4-1: Halving times with corresponding evaporation factors

4-2-3 Deployment aggressiveness parameter λ

The parameter λ influences how aggressive a vertex is being deployed weight on. Section 3-4 proposed limits for λ. The average deployment function: $\lambda \in (0, \frac{1}{3})$ and the minimum deployment function : $\lambda \in (0, 1)$.

Average deployment function:
Figure 4-6 shows the performance with respect to different λ. It is apparent that λ does not have a significant influence on the performance. Although the influence is insignificant the highest possible λ is chosen, since in theory this has the strongest effect.

Figure 4-6: Average deployment function with different λ

Minimum deployment function:
Similar to the average policy, figure 4-7 shows that λ does not have a big influence on the performance. Again, the highest $\lambda = 1$ is chosen.

Figure 4-7: Minimum deployment function with different λ

4-2-4 Deterministic vs probabilistic parameter ε

The parameter ε is the interpolation parameter for the probabilistic and deterministic decision model. With $\varepsilon = 0$, the deterministic model is fully applied. With $\varepsilon = 1$, the probabilistic model. By changing this parameter, the proclivity to choose the vertex with the lowest weight is adjusted. Figure 4-8 displays the effect of different ε. It is visible that $\varepsilon = 0.4$ has a slightly better performance than others for persistent surveillance. This is not significant, however. The first cover (CPP) is the fastest with $\varepsilon = 0.4$ as well. Below a certain threshold $\varepsilon \approx 0.2$, the performance decreases rapidly.

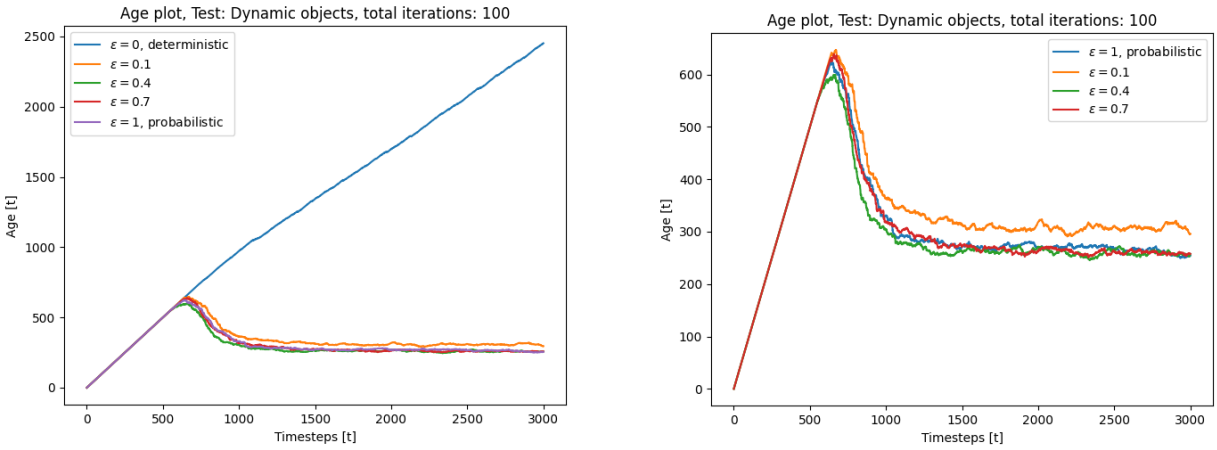


Figure 4-8: Results with different ε

4-3 Results

The optimized policies are compared with dynamic objects in figure 4-9, where the random walk is also included as a reference. The first thing that stands out is that the all models work significantly better than the random walk. The second thing that is visible is the similar performance of the other deployment functions. The average deployment function is slightly worse. This shows that the basic incremental policy is not outperformed by the policies that use local information. Subsequently, when compared in the static object test, one should expect a more significant difference in performance. However, depicted in figure 4-10, it is quite the contrary. The results are similar as well.

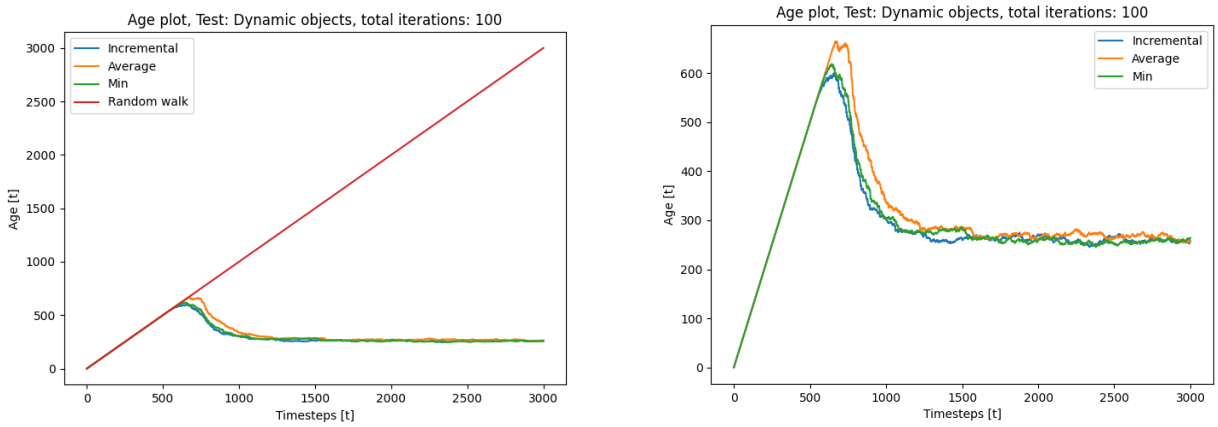


Figure 4-9: Optimal deployment functions compared in the Dynamic test

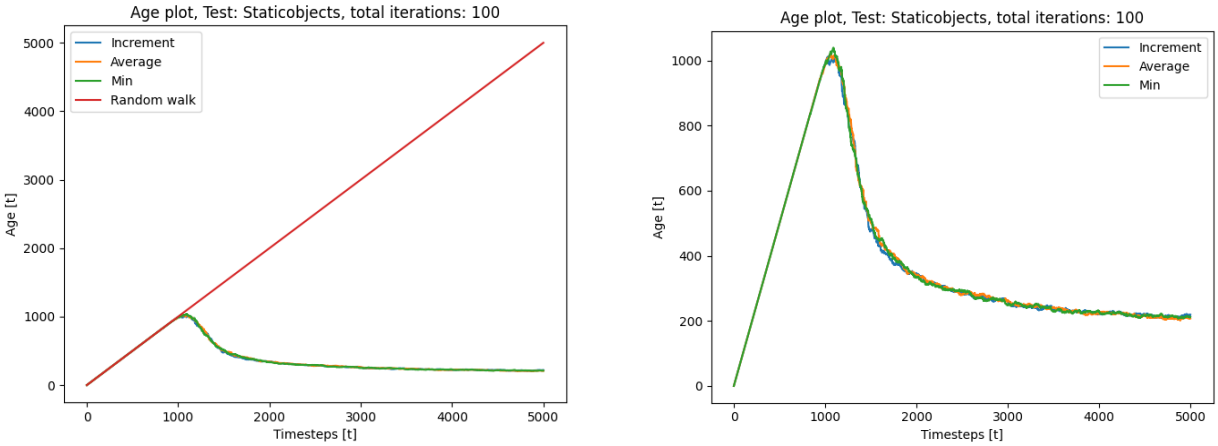


Figure 4-10: Optimal deployment functions compared in the static test

4-4 ETC-inspired Switching rule

The switching rule is about the question of when to deploy pheromones and when not to deploy pheromones discussed in section 3-3-3. From this section onward, the incremental weight deployment function from equation 3-13 is used. Since all deployment functions perform similarly, it is best to use the most simple one. This section shows the results from implementing the switching rule that is proposed in equation 3-19. First, the effect of the sensitivity parameter K is shown. At last, the performance of the switching rule is evaluated. This is achieved by comparing the proposed switching rule with the standard version without switching rule and the coverage algorithm from the work of [1], from here on called the "Svennebring" method.

4-4-1 Sensitivity parameter K

The parameter K has an influence on the threshold for deploying weights or not. The lower K is, the stricter the switching rule, resulting in fewer deployment instances. In figure 4-11 the effect of different K parameters is shown. Moreover, the incremental deployment function is added without a switching rule. For all cases, the normalization of deployment instances over the vertices, described in section 3-3-3, is applied. This is done to show that the proposed switching rule is also more efficient even when there is accounted for the normalization of deployment instances over the vertices. In table 4-2 are the quantitative measures given. Where age_{max} is the average time for the first cover, which is the goal of CPP, age_{∞} is the steady-state age, t_{∞} is the time that the steady-state age is achieved (settling time), and $instances_{\infty}$ is the quantity for the steady-state of the deployment instances. In this table, age_{max} is included for completeness. With regards to CPP, the model without the switching rule applied is slightly the fastest, followed by Svennebring and the switching rule with $K = 1.3$. For Svennebring, the peak is at $age = 630\ t$. However, from $age = 375\ t$, there are already a lot of first covers done. Concerning persistent surveillance, the age_{∞} is comparable for every value of $K > 0.4$. Only the settling times are longer when K decreases.

For $K < 0.4$, the performance starts to decline rapidly. The svennebring method does no have a steady state. Regarding the deployment instances, the coverage methods that use the switching rule are more economical with their pheromone deployments.

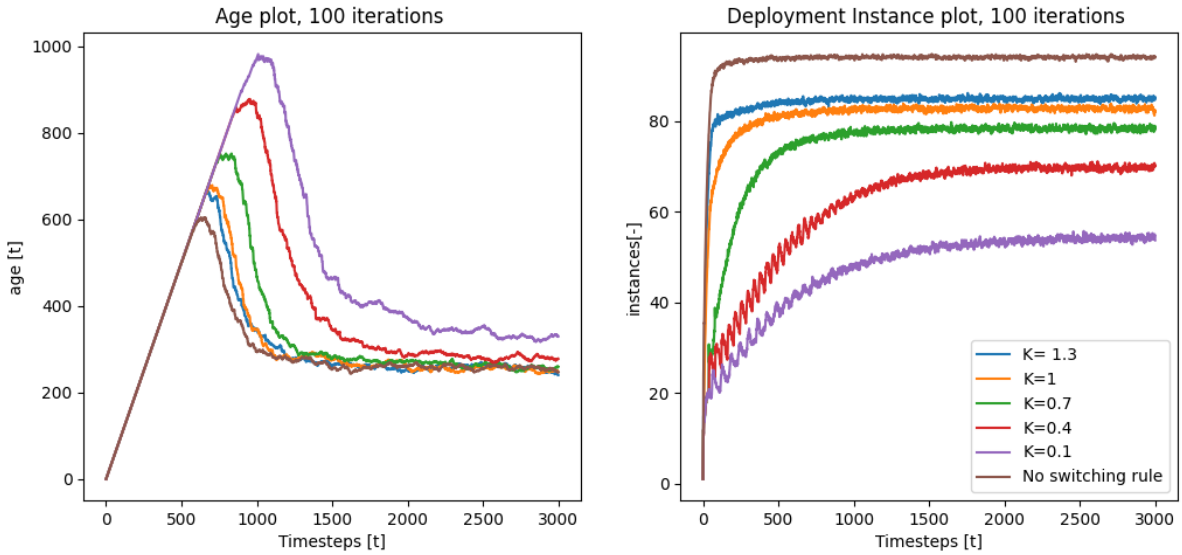


Figure 4-11: The effect of different K values

	Age_{max} $[t]$	Age_{∞} $[t]$	t_{∞} $[t]$	$Instances_{\infty}[\frac{1}{t}]$
No switching rule	625	265	1500	93
K=1.3	660	265	1600	83
K=1	700	265	1550	80
K=0.7	790	270	1750	76
K=0.4	1000	275	2500	68
K=0.1	1100	330	3500	50
Svennebring	630	-	-	100
Random	910	345	1800	47

Table 4-2: Performance metrics for different K values and other methods.

4-4-2 Combined results

Figure 4-12 compares the three final methods. Namely, the incremental deployment function without switching rule, with switching rule with $K = 1$ and the coverage algorithm from the work of [1], from now on called the Svennebring method. The first coverage of the Svennebring method is faster from 385 t; however, the age in the environment only increases from that point onwards. Subsequently, although the incremental deployment function is slower than Svennebrings' method concerning the first coverage, the persistent surveillance is significantly better.

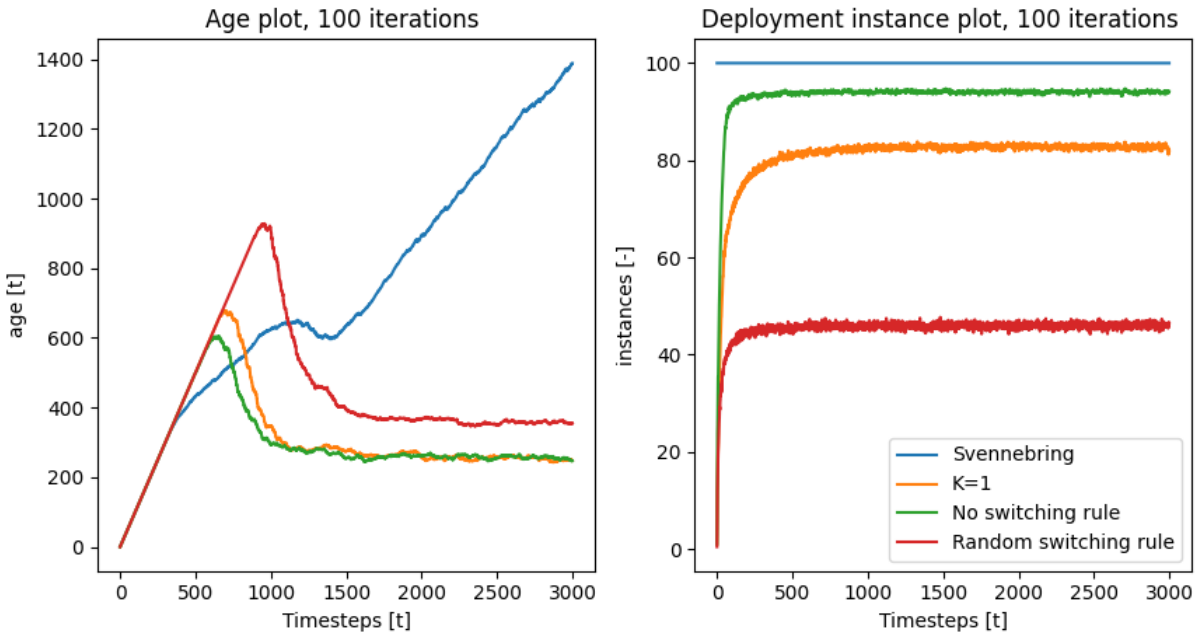


Figure 4-12: Performance plot

4-5 Reflection

The results show that the proposed model performs better than current literature. However, it requires more reflection to evaluate if the proposed model is objectively a viable solution for persistent surveillance. For instance, if the proposed model is better than literature, but the maximum age does not go under $20t$ for a 3x3 square graph with 20 agents, it is objectively not a good solution for persistent surveillance. The objective relevance can be reasoned by use of the previous mentioned constant R. R is the amount of vertices per agent in the environment. In other words, if every agent would get an equally sized subset of the environment to persistently survey it would be R vertices big. Consequently, it will take R timesteps to completely cover the environment in the optimal case. For the simulations, $R \approx 52$. The results for the proposed model show that $age_{\infty} = 265$. Hence, the proposed model is 5 times slower than the optimal case. The proposed model is designed to be robust, adaptive and probabilistic. This model will operate and perform well in almost any scenario and only simple robots are required, where stigmergy is the sole communication mean. Considering that all these characteristics are at the cost of speed, the performance is reasonably well.

Chapter 5

Demonstration

In this chapter, a simulation with physical robots is presented. The simulation acts as a demonstration and shows that the persistent surveillance model works if implemented on physical robots under several assumptions. First, information about the simulation and assumptions are discussed. Secondly, the results from the demonstration are presented.

5-1 Introduction

In this master thesis thus far, a discrete model for robotic persistent surveillance is proposed and simulated. That model is also simulated in Webots, an open-source 3-D Robot simulator, to demonstrate that the model is also working on physical robots. In Webots, persistent surveillance is done in a pre-made environment with multiple Elisa-3 GCtronic robots. Figure 5-1 displays the environment which the robots need to cover. The simulations take place over 7200 seconds with 16 Elisa-3 robots. After 2400 seconds, the box in the middle disappears, and the environment changes. The robots adapt effortless to the altered environment, and after a transition phase, a steady-state is achieved. The complete demonstration is visible on https://youtu.be/yC5Kk20LX6Q.

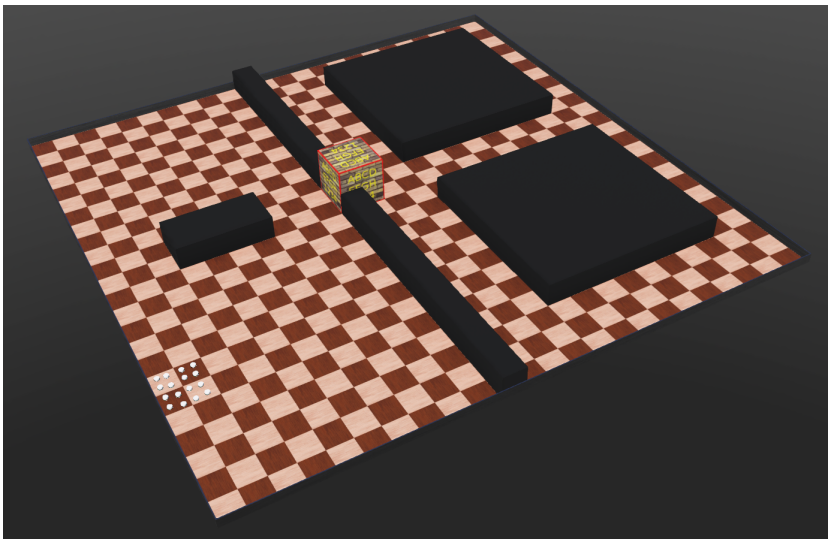


Figure 5-1: Webots simulation. Multiple Elisa-3 robots need to cover an environment

5-2 Translation to continuous space

In order to translate the discrete model and simulations to a continuous simulation, some assumptions and alterations had to be made. In table 5-1, the algorithm is given in pseudo-code for the individual robots. The robots need only simple rules.

Pseudocode individual robot:
1: Measure pheromones from neighbouring tiles (by requesting them).
2: Decide next direction based on the pheromones measured.
3: Place pheromones (by emitting them).
4: Compute rotation and translation.
5: Move.
6: While moving: Collision avoidance.
7: Go to 1.

Table 5-1: Algorithm for the individual robot

There were three points of interest during the translation to the Webots simulation. Namely, collision avoidance, pheromone placement and coordinate system. In a physical world, robots cannot be at the same location, and they need obstacle avoidance to not crash into each other. The downside is that the robots cannot go in their planned direction and need to alter their direction. Additionally, Webots cannot simulate pheromones directly, and robots cannot deploy them. Hence, it is necessary to simulate the pheromone density via a proxy. It is assumed that robots can deploy and sense pheromones. An external Python algorithm that can communicate with Webots simulates the pheromone densities. This is done similarly to the discrete simulations from the previous chapter. The environment is divided into a grid, where every tile has its weight. Therefore, the pheromone placement is still on a discrete coordinate system. The individual robots sense the pheromone density in the tiles around them

by receiving that information from the external python program. Based on that information, the robots decide which tile to go next. The underlying assumption here is that the robots have also agreed upon a discrete coordinate system on which the pheromones are deployed. In real life, a method for agreeing on a coordinate system must be applied. For instance, with digital pheromones, robots can become beacons that store pheromones for that location. These methods are, however, beyond the scope of the thesis. The Webots simulation is solely intended as a demonstration.

5-3 Results

For the Webots simulation, every nine seconds, a snapshot of the age and weights are taken. Nine seconds is what it takes on average for a robot to travel from one tile to the other and corresponds to 1 timestep of the system. Figure 5-1 shows the area which the robots need to cover. Subsequently, figure 5-2 shows the maximum age present in the system at a specific time, averaged over 25 iterations. During the first 2400 seconds, the robots only need to cover half of the complete environment. After roughly 1000 seconds, an equilibrium is achieved. From second 2400 there is a new transition phase, which reaches an equilibrium at 5000 seconds. In figure 5-3 this is also visible. At timestep $t = 2415s$, it is visible that all robots are evenly spread throughout the left half, followed by a transition phase where robots slowly enter the right half and spread through the environment. Lastly, in figure 5-4 a heat map of the weights of all the tiles are displayed at the end of the demonstration. Here it is visible that the weight distribution is roughly even through the environment.

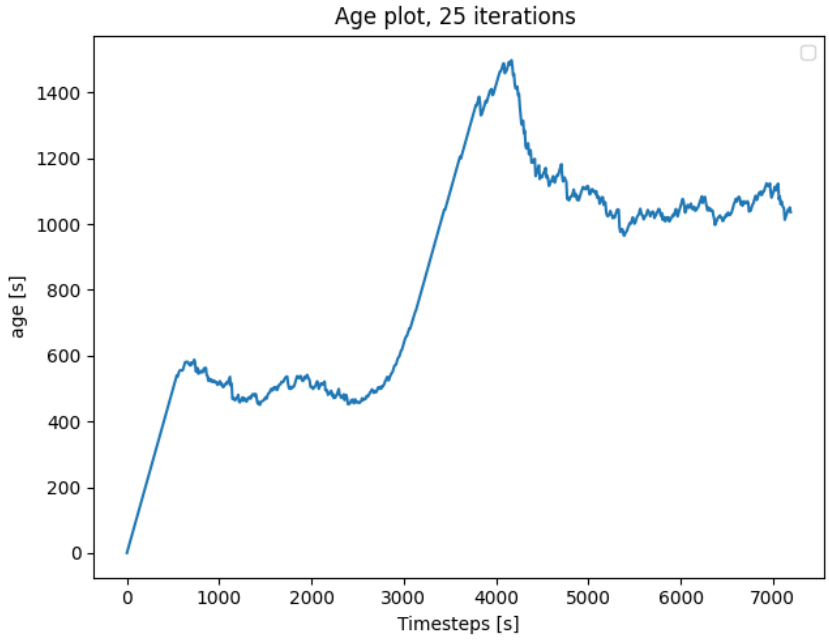
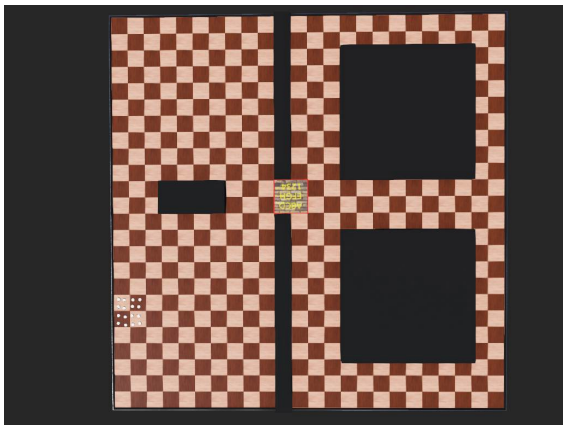
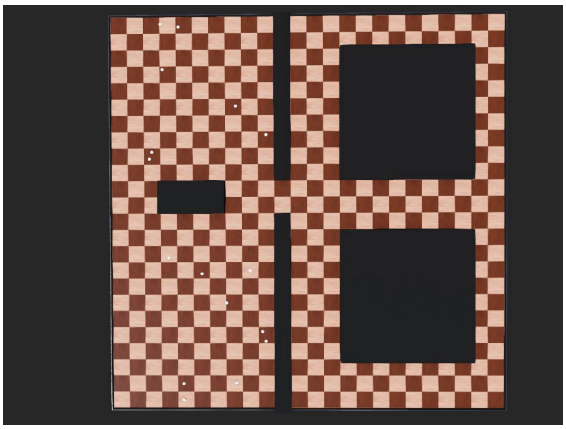


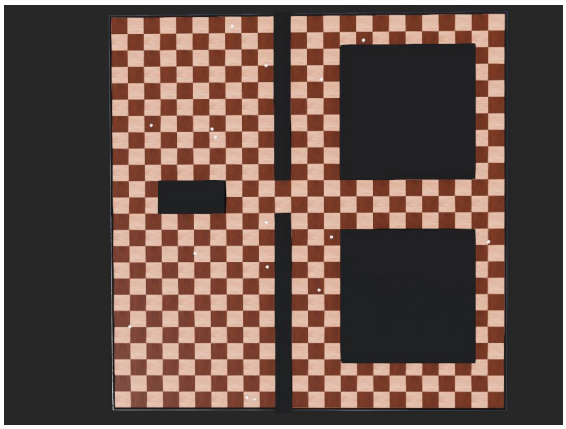
Figure 5-2: Ageplot of the demonstration



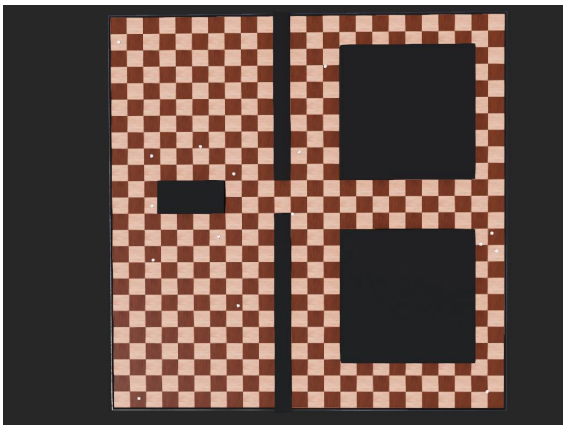
(a) t=0s, Start of the simulation



(b) t=2415s, box just disappeared



(c) t=3690s, Transition phase



(d) t= 7200s, end of simulation

Figure 5-3: Snapshots of the demonstration at different time-steps. https://youtu.be/yC5Kk20LX6Q

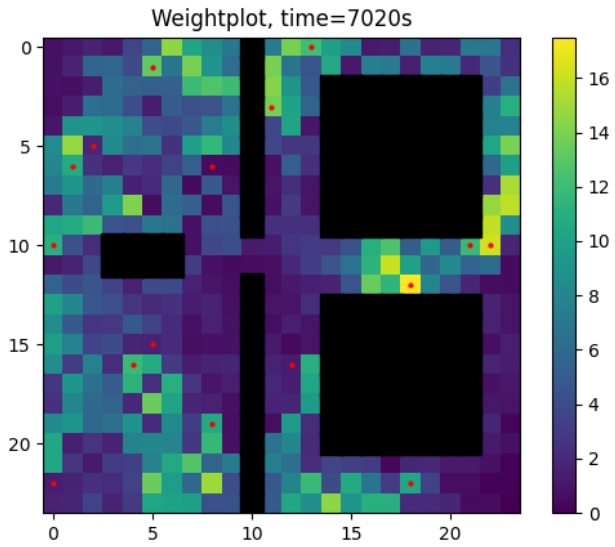


Figure 5-4: Heatmap of the pheromone-densities of the demonstration at $t = 7020s$.

Chapter 6

Discussion

The results from the simulations show that the proposed model of this thesis outperforms current persistent surveillance models. On top of that, including the switching rule in the model saves pheromone deployments significantly. The results show again that probabilistic models are not desirable for CPP. They are inherently slow and inefficient. Persistent surveillance, however, is ideal for probabilistic models. Subsequently, stigmergy seems exceptionally well suited to coverage and persistent surveillance in particular. With the use of pheromones, there is a direct relation between pheromone density and the time a location has been covered. This requires no memory or heavy computations for the robots while having accurate knowledge of which location best to cover next.

6-1 Parameters

Concerning γ, it was not expected that extra information does not help the overall performance. On the contrary, it makes the performance worse. This could be because the outer vertices often have lower weights with respect to the closer vertices and, consequently, levels out the difference of the weights of the closer vertices. Hence the probabilities will have a smaller difference, which results in more agents going to the vertex with the higher weight. Additionally, regarding evaporation factor ρ, one would expect a relatively lower ρ is beneficial for the performance since more information can be exploited from the system. A high ρ emphasizes recent covers more and erases the past quickly. However, it turned out that a higher ρ increases performance. This is probably because it reduces the, what we call, "overlap effect". The overlap effect happens in discrete space when multiple agents are on the same vertex. It is hard for them to split since pheromones have no separating effect on them. By increasing ρ, the presence of pheromones indicates that another agent is in the vicinity and results in agents staying away from each other, hence resulting in more efficient coverage. Especially if agents overlap with a relatively low ε, it is even harder for them to split. They will often travel the same path since the probability of going to the lowest weight is much increased due to ε. Their pheromone deployment will not affect the dispersion of agents that

are already at the same vertex. This will probably be less the case in continuous space and time, where the deployment of an agent's pheromones directly leads to other agents sensing that instead of a timestep later. On top of that, it is not possible to be at the same vertex in continuous space. Regarding the probability matrix, including the deterministic decision policy and ε has a negligible effect on the overall performance. For $\varepsilon > 0.2$, the performance is similar. Although there is not much of a big difference, including the ε-greedy method has its benefits. The ε-greedy method is much researched and acquiring mathematical proofs for future work is therefore less complicated.

Lastly, the "average" and "minimum" deployment functions did not increase performance while taking more information into account. Parameter λ had no effect. The idea behind these deployment functions was that the agents would adjust the number of pheromones accordingly to their surroundings. If there were a lot of uncovered low weight vertices, then the agents would deploy less. If it sensed that this area was already covered a lot, it would deploy more. One would expect that if more relevant information is present in the system, the agents come to better decisions. This hypothesis, however, turned out to be false. The performance did not change with respect to the incremental deployment function. We could not explain why it had no visible effect.

6-2 Switching rule

With sensitivity parameter, K one can adjust the trade-off between age_{max} and settling time t_∞ and the deployment instances. Fewer deployment instances lead to a slower method. However, the steady-state maximum age in the system age_∞ does not change significantly until a certain threshold $K < 0.4$. In figure 4-11 is clear that with applying the switching rule the weight usage is more efficient and persistent coverage is achieved with the same performance and less deployment instances. On the other hand, with regards to age_{max} there is a direct link between the amount of deployment instances and age_{max}. Since the agents start at the same vertex, they must spread as fast as possible. The more weights are placed, the more the agents are repelled from the initial place and each other. Hence, it is not desirable to limit weight deployment if fast first coverage is the objective. It is for this reason that this thesis focuses on persistent surveillance. The model that this thesis proposes is probabilistic, which means that it is by nature slower than deterministic models, especially with regards to first coverage. This is directly visible in figure 4-12. Here the fully deterministic model from Svennebring is faster for the first coverage. However, it quickly reaches its limit. The model is not well suited to persistent surveillance as the method deploys all the time, and there is no evaporation. As a result, the environment gets saturated, and performance goes quickly down. With a probabilistic model, this is not the case.

6-3 Demonstration

The demonstration is not one-to-one applicable in real life. The discretization of the pheromone-density field is a limiting factor in that. To let this work in real life, a discretization method should be applied. The next question would be how real-life robots are going to perform

stigmergy and place and sense pheromones. Because of the discretization of the pheromone-density field, the overlapping effect is also present in the demonstration. The pheromones do not actively repel the agents from each other, which limits the performance. This will not happen in continuous space. Subsequently, for continuous space, the robots need collision avoidance. At first sight, it seems like this will negatively impact the performance since robots cannot go where they have opted to go. However, collision avoidance does aid against the overlap effect and the dispersion of agents. If they encounter other agents, they turn away and try to get away from each other. This reduces the overlapping effect and has a positive effect on the performance.

Chapter 7

Conclusion

This thesis proposes an efficient swarm robotic persistent surveillance model that uses stigmergy as the sole communication method. The idea came from nature swarming, where stigmergy is used as a mean of communication. Robots deploy pheromones that repel instead of attract other agents. The authors of [1] propose a similar model, which is used for comparison in this thesis. Lastly, a demonstration is presented with the proposed model where a simulation of physical robots is persistently surveying a location.

It is demonstrated that a probabilistic stigmergic swarm robotic model is well suited to persistent surveillance. Stigmergy lends itself perfectly for persistent surveillance as it simply acquires a direct relation between pheromone density and the time a location has been visited. Therefore, there are no complex computations and big memory storage needed for the robots whilst having a good performance. The model of this thesis outperforms current literature and can deal with unknown environments with dynamic obstacles for a more extended period of time.

Additionally, this thesis proposes a switching rule that optimizes the use of pheromones. In that way, it performs persistent surveillance almost equally good as without switching rule, but it deploys pheromones more efficiently. This is desired to avoid congestion of the environment or network.

Lastly, the demonstration shows the potential for the model in a real-life application. A few critical assumptions had to be made, which prevents the one-to-one application in real life. Future work needs to focus on this.

Chapter 8

Future Work

This thesis, thus far, focused mainly on the proposal of a stigmergic robotic coverage method that uses pheromones efficiently. During the one year of writing this thesis, a lot of interesting other relevant questions arose. However, many of those questions turned out to be beyond the scope of this thesis and too time-consuming. In this chapter, these questions are posed for future work.

8-1 Continuous space and time

The proposed model has been mainly tested for discrete-time and space. It is, therefore, interesting for future work to translate the model to continuous time and space, ready for real-world application. The demonstration in Webots showed that there is potential, but already exposed some challenges that have not been solved for continuous-time and space. There are two points of interest. The first is to tackle the challenges that are present in the translation to continuous time and space. These include:

- How do agents agree upon a shared coordinate system and keep that fixed without a centralised entity?
- How will stigmergy be performed?
- Which physical robots to choose that can facilitate stigmergy?
- How to deal with measurement errors and faulty sensors in real-life?

The second challenge regards parameter tuning. The "overlap effect" discussed in chapter 6 will probably be less detrimental to the performance of the model. Therefore, the hypothesis is that in continuous space and time, optimising the parameters again can lead to better performances. For example, it could be the case that a lower ρ and γ is beneficial. This is because the robot has more information to base its decision on and lead to better overall decisions. There is no need to emphasise on the avoidance of the overlap effect by increasing ρ and γ.

8-2 Real-life experiments

The next step in the translation to continuous space and time is to perform real-life experiments. To apply the method on physical robots and do experiments on how well the performance is in real applications.

8-3 Different deployment function

For the proposed model, the best performing pheromone deployment function turned out to be a regular incremental deployment function, where agents, regardless of the situation around them, deploy a constant amount of pheromones. It is interesting to see if deployment functions using available information to decide the deployment amount are performing better in continuous space. For example, the "average" and "min" deployment function could be used.

8-4 Dynamic K

The switching rule uses constant K as a scaling parameter for the threshold to deploy or not. It is interesting to see what the performance will be with a dynamically changing K. In that case, the transition phases can be shortened, and dispersion of the agents can be sped up. Additionally, it is possible to decrease the deployment instances even further if a steady-state is achieved.

8-5 Mathematical proof of convergence

The mathematical proofs did not go beyond boundedness results. It would be interesting to see if further mathematical proofs can be derived. By including the ε-greedy method, it is likely to attain convergence proofs for the total weights.

Bibliography

[1] J. Svennebring and S. Koenig, "Building terrain-covering ant robots: A feasibility study," *Autonomous Robots*, vol. 16, no. 3, pp. 313–332, 2004.

[2] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, 1999.

[3] B. Ranjbar-Sahraei, G. Weiss, and A. Nakisaei, *A multi-robot coverage approach based on stigmergic communication*, vol. 7598 LNAI. Springer, Berlin, Heidelberg, 10 2012.

[4] D. Lehmann and J. Lunze, "Event-based control: A state-feedback approach," *2009 European Control Conference, ECC 2009*, pp. 1716–1721, 2014.

[5] L. Ding, Q.-L. Han, X. Ge, X.-M. Zhang, and S. Member, "An Overview of Recent Advances in Event-Triggered Consensus of Multiagent Systems," *IEEE Transactions on Cybernetics*, pp. 1–14, 2017.

[6] J. Cortés, S. Martínez, T. Karataş, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, 4 2004.

[7] A. Janchiv, D. Batsaikhan, G. H. Kim, and S. G. Lee, "Complete coverage path planning for multi-robots based on," *International Conference on Control, Automation and Systems*, pp. 824–827, 2011.

[8] J. M. Palacios-Gasos, Z. Talebpour, E. Montijano, C. Sagues, and A. Martinoli, "Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1321–1327, 2017.

[9] K. Sugawara and M. Sano, "Cooperative acceleration of task performance: Foraging behavior of interacting multi-robots system," *Physica D: Nonlinear Phenomena*, vol. 100, no. 3-4, pp. 343–354, 1997.

[10] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 851–871, 2000.

[11] E. Sahin, "Swarm Robotics: From Sources of inspiration to Domains of Application," *Springer,Berlin , Heidelberg*, 2005.

[12] S. Camazine, N. Franks, and J. Sneyd, *Self-Organization in Biological Systems*. 2001.

[13] E. Şahin, S. Girgin, L. Bayindir, and A. E. Turgut, "Swarm Robotics," *Swarm Intelligence*, pp. 87–100, 2008.

[14] G. J. Sussman, *Building Robust Systems*. PhD thesis, 2008.

[15] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, vol. 1, no. 8, 2019.

[16] C. G. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 947–961, 2013.

[17] Z. L. Cao, Y. Huang, and E. L. Hall, "Region filling operations with random obstacle avoidance for mobile robots," *Journal of Robotic Systems*, vol. 5, no. 2, pp. 87–102, 1988.

[18] J. Czyzowicz, L. Gasieniec, A. Kosowski, and E. Kranakis, "Boundary patrolling by mobile agents with distinct maximal speeds," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6942 LNCS, pp. 701–712, 2011.

[19] N. Nigam and I. Kroo, "Control and design of multiple Unmanned Air Vehicles for a persistent surveillance task," *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, MAO*, pp. 1–19, 2008.

[20] R. Fujisawa, H. Imamura, T. Hashimoto, and F. Matsuno, "Communication using pheromone field for multiple robots," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 1391–1396, 2008.

[21] V. Yanovski, I. A. Wagner, and A. M. Bruckstein, "Vertex-Ant-Walk - A robust method for efficient exploration of faulty graphs," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 99–112, 2001.

[22] E. Osherovich, V. Yanovki, I. A. Wagner, and A. M. Bruckstein, "Robust and efficient covering of unknown continuous domains with simple, ant-like A(ge)nts," *International Journal of Robotics Research*, vol. 27, no. 7, pp. 815–831, 2008.

[23] F. De Rango, N. Palmieri, X. S. Yang, and S. Marano, "Bio-inspired exploring and recruiting tasks in a team of distributed robots over mined regions," *Simulation Series*, vol. 47, no. 9, pp. 65–72, 2015.

[24] B. Ranjbar-Sahraei, G. Weiss, and A. Nakisaee, "Stigmergic Coverage Algorithm for Multi-Robot Systems (Demonstration)," tech. rep., 2012.

[25] B. Ranjbar-Sahraei, G. Weiss, and K. Tuyls, "A Macroscopic Model for Multi-Robot Stigmergic Coverage," tech. rep., Maastricht University, 2013.

[26] R. Johansson and A. Saffiotti, "Navigating by stigmergy: A realization on an rfid floor for minimalistic robots," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 245–252, 2009.

[27] V. A. Ziparo, A. Kleiner, B. Nebel, and D. Nardi, "RFID-based exploration for large robot teams," *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 4606–4613, 2007.

[28] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone robotics," *Autonomous Robots*, vol. 11, no. 3, pp. 319–324, 2001.

[29] K. J. Astrom and B. Bernhardsson, "Comparison of periodic and event-based sampling for first-order stochastic systems," in *Ifac world congress*, pp. 301–306, 1999.

[30] K.-e. Arzen, "A SIMPLE EVENT-BASED PID CONTROLLER," no. 1989, 1999.

[31] W. Heemels, K. Johansson, and P. Tabuada, "An Introduction to Event-triggered and Self-triggered Control," 2012.

[32] L. Grüne and F. Müller, "An algorithm for event-based optimal feedback control," *Proceedings of the IEEE Conference on Decision and Control*, no. 3, pp. 5311–5316, 2009.

[33] P. Tabuada and S. Member, "Event-Triggered Real-Time Scheduling of Stabilizing Control Tasks," vol. 52, no. 9, pp. 1680–1685, 2007.

[34] Y. Fan, G. Feng, Y. Wang, and C. Song, "Distributed event-triggered control of multi-agent systems with combinational measurements," *Automatica*, vol. 49, no. 2, pp. 671–675, 2013.

[35] X. Ge and Q. L. Han, "Distributed Formation Control of Networked Multi-Agent Systems Using a Dynamic Event-Triggered Communication Mechanism," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 10, pp. 8118–8127, 2017.

[36] E. Garcia, Y. Cao, and D. W. Casbeer, "Decentralized event-triggered consensus with general linear dynamics," *Automatica*, vol. 50, no. 10, pp. 2633–2640, 2014.

[37] G. Guo, L. Ding, and Q. L. Han, "A distributed event-triggered transmission strategy for sampled-data consensus of multi-agent systems," *Automatica*, vol. 50, no. 5, pp. 1489–1496, 2014.