CosPy: A simulator for robotic swarm research

Student Name: A.E.W. Reynoldson
Supervisor Name: F. Arvin
Submitted as part of the degree of MEng Computer Science to the
Board of Examiners in the Department of Computer Sciences, Durham University

Abstract – Pheromone-based communication is one of the most efficient communication methods studied in nature. We propose a novel swarm robot simulator, 'CosPy', for use in swarm robotics development and research. It aims to provide a platform for the rapid development of experiments and algorithms for mobile robots within a 2D environment. We explore the requirements of a swarm robotics platform and how we can apply pheromone-based communication to real-world applications and environments.

Index Terms— Artificial Pheromones, Collective Behaviour, Pheromone Communication, Swarm Robotics, Simulator

1 INTRODUCTION

THE field of swarm robotics (SR) has exponentially grown in the past decade. The use of bio-inspired swarm algorithms derived from natural systems, such as Ant colonies, have been studied at length and proved to be an effective tool for many categories of problems [1]. In this work we develop a new platform to explore how these swarms can communicate via pheromones, being artificially simulated in a 2D environment. We explore the associated metrics and test them on our system.

1.1 Collective Behaviour

Collective behaviour can be used for metaheuristics [2], utilising the evolved strategies of swarm systems to efficiently find near-optimal solutions to a wide variety of tasks. It is based on the premise that evolution has refined strategies and collective behaviour for groups of animals even when the individual has limited comprehension and capability [1]. Swarm robotics exploits this social behaviour and emergence to develop asynchronous and distributed methods for useful and human tasks. Self-organization is a key aspect of this behaviour [3] and it relies on a positive feedback loop. This loop requires interaction between the individuals and means that a particular action may be more likely if the neighbouring robots are performing the same action [3]. For example, if a fish's neighbourhood begins to change direction in response to a predator, then that fish will be much more likely to follow that direction also.

There are several main components of collective behaviour in SR which come together to form capable swarms. We want to be able to implement these into our robot swarms so that collective behaviour can be best replicated. The following provides a summary of the collective behaviour taxonomy proposed by Brambilla *et al* [4]. It details the main behaviours found in robotic swarms, these break down how SRs can organise, move, and make decisions [2]. These behaviours have been applied to SR, for instance Garnier *et al* [8] in which 'Alice' micro-bots applied cockroach aggregation and decision-making for selecting a shelter and estimating a shelter's size. Below details the three main behaviour groups.

Spatially Organising Behaviours describe how swarm robots can be organized and distributed in the environment.

Aggregation – Groups robots closer together so that they can cooperate and form a swarm.
Pattern Formation – Robots maintain distance to produce a certain pattern or shape which allows the swarm to cover an area most efficiently or perform a task.
Chain Formation – Robots position themselves to connect two points and form a chain, which can be used for navigation.
Self-Assembly – Robots physically connect to each other; morphogenesis is self-assembly into a particular pattern since assembled robots may perform a task better than individuals.
Object Clustering – Robots move and group objects. Assembly is when these objects are connected (used for construction).

Navigation Behaviours coordinate the movement of robot swarms and allow them to perform cooperative tasks.
Collective Exploration – Robots cooperate to explore and move through an environment. Area coverage allows swarms to evenly cover an area. Swarm-guided navigation is used to guide the swarm and optimise its navigation/routing usually by utilising pheromones (discussed in section 1.2).
Coordinated Motion – Robots move in formation (flocking) to quickly navigate an environment without collisions. Collision avoidance, velocity matching and flock centring are employed to ensure that a swarm stays together whilst not colliding or letting robots escape the aggregate.
Collective Transport – Groups of robots must work together to transport an object. Typically, due to the object being too large for a single robot to move, the robots must agree on which direction they will move in whilst moving the object.

Collective Decision-Making involves how the robots form decisions and influence each other in these decisions. The desired outcome for the SR system may be that the swarm converges or that the swarm splits apart and maximizes the performance of a given system.
Consensus Achievement **–** Allows swarms to reach a consensus from several alternative options e.g., which object to move towards. A consensus can be reached via direct communication between robots (using sensors) or indirect communication, here the properties of a given swarm are used to form a decision.
Task Allocation – Robots split up and distribute themselves so that different tasks can be undertaken. This can be used for maximising the performance of SR, with each task having a

variable number of robots assigned to it depending on the current environment.

There are several other behaviours which can form collective behaviour, such as group size regulation [13], note that not all these behaviours are required for an SR system.

1.2 Pheromone-Based Communication

This work focuses mainly on the pheromones involved in swarm intelligence and how we can easily implement them artificially with light from a projector for use in SR systems. Pheromones in nature are chemical signals [1] used for communication between individuals and collective exploration. An ant may lay a pheromone trail when it finds food, and this will attract other ants towards the food. The ants that follow this trail will also release pheromone creating a positive feedback loop. This forms an efficient communication system which allows a population to make intelligent decisions about what to do, despite the individuals having limited comprehension. Negative feedback is also used to stabilize the system, otherwise a pheromone trail would reinforce exponentially and prevent a swarm from functioning. This negative feedback is introduced through pheromone evaporation, exhaustion of the food source and the swarm size being limited. Swarms are not limited to just one pheromone type, some ant colonies have multiple specialized signals for danger, food, and mating [1]. Experimentation with using many pheromone types in one robotic swarm is limited, as this introduces far more complexity into the system. Also, with any pheromone communication system evaporation, diffusion and environmental effects should be considered and implemented.

1.3 Swarm Robotics

SR is a relatively new field which aims to apply collective behaviour and artificial pheromones to mobile robots in order to complete useful tasks. Distributed systems have been around for decades, and their benefit is massive parallel processing power due to having no centralized control system. Also, this accounts for redundancy, many agents can fail, and the swarm can still function efficiently if enough agents remain. Swarm robotics aims to replicate collective behaviour for practical uses. A robot swarm contains autonomous robots that can freely move and interact with objects in an environment [2]. There have been many mobile robots developed for use in SR such as Colias [21] and Kilobot [18]. These robots have limited comprehension but do contain the ability to be programmed (sometimes remotely e.g. with IR) by researchers and deployed. They utilise small motors for movement and contain sensors, capable of sensing varying levels of their real or virtual environment. Each robot should only have local information and the swarm should be able to complete tasks by cooperating. Swarms can comprise of few or many robots meaning it is very scalable, these mobile robots are either homogeneous or heterogeneous (identical or of different type). The largest robot swarm which has been created was by Rubenstein *et al* [17] where they used 1024 Kilobot robots [18] capable of self-assembly into any 2D shape. This large swarm self-assembly system did not utilise pheromones and instead used edge-following and gradient formation to iteratively form the shapes. Designing physical swarm systems is a difficult task, there is no standardized method for developing SR systems and the related works provided all use different programming languages and hardware. One potential reason for this may be how open-ended the field is, there is no clear goal of what SR systems should be or do and it is not clear what uses they may have. This work proposes a novel system which attempts to push research towards standardization by providing an open-source platform for SR testing using a projection system [5]. The main components required for a physical SR system [2] are:
1. Processing hardware is the hardware provided by the robots or an overhead control system. This can vary depending on processing requirements and the power of each robot.
2. Communication Methods describe how the swarm communicates inter-robotically and with remote workstations.
3. Sensors allow mobile robots to detect their surroundings, IR and visible light sensors are the most common.
4. Localization System is the system responsible for giving position awareness to the swarm robots e.g., compass or GPS.
5. Tasks differ between systems, ranging from basic navigational tasks to self-assembly and collective transport.
The methods researchers have gone about implementing and combining these components to develop autonomous, useful robot swarms is explored in section 2. Simulators in SR attempt to implement these components virtually, so that simulated swarms behave the same as physical implementations.

1.4 Project Aims

In this work we aim to develop a simulation platform for SR research and education. This project will be a success if swarm scenarios tested using the platform, correctly mimic and visually demonstrate the actions of a real-world swarm of robots. We aim to create a detailed but intuitive graphical user interface allowing for complex scenarios to be developed and fine-tuned. The goal is to give researchers an open-source platform to experiment with and fine-tune SR algorithms and scenarios, before applying them to physical robots and eventually real-world tasks. In this work, the developed simulation system, 'CosPy,' is presented and then tested on several SR scenarios. Specifically, basic pheromone following [7], BEECLUST [31] and ΦClust [28]. The efficacy and reliability of the system is discussed and compared with previous works. Future development and research using the system is also discussed, providing a basis for SR researchers to build upon. Studying mobile robots, artificial pheromones, and existing simulators, provides a starting point for this work; giving examples of artificial environments and SR setups that we can simulate. Existing literature on these topics is discussed in the next section.

2 Related Work

This section gives a brief review of the field of swarm robotics (SR), discussing both the hardware and software. SR is a comprehensive field of study with a wide range of applications and areas of development. The related work featured touch on

the physical experimental setups for SR systems and then delves into virtual implementations and tasks.

2.1 Mobile Robots

Mobile robots are able to interact with each other and the environment. Before developing a simulator, we need to be aware of the capabilities of the available mobile robots and what kind of abstractions we can use. We want cheap, robust, and simple autonomous robots which can be used in other domains such as biology or education. Several different approaches and designs have been used for the development of these autonomous robot swarm agents. This work focuses on simulating ground-based robots, but recent works have begun exploring the uses of flying drone swarms [2] and how pheromones can be represented, such as using GPS. The main components of these mobile robots are actuators, microcontrollers, batteries, and sensors [20].

Kilobots [18] are lightweight and cheap robots. They only take 5 minutes to assemble, this combined with the low cost means that they are perfect for large swarm experiments [17]. SDASH is the algorithm used to support Kilobots in self-assembly and collective behaviour, so each robot has enough memory and power to run this algorithm. Forward motion and rotation are achieved via their coin-shaped vibrating motors as opposed to a traditional two-wheeled system. Robots can inter-communicate in all directions via a photodiode and an LED infrared transmitter placed atop. An Atmega328 microprocessor is used for managing the control and sensing, and the Kilobots are programmed in C. This can be done en masse by sending the code remotely to the Kilobots via IR signals.

Alternatively, Turtlebot3 [19] is a a large and more intelligent robot, which is programmed using the Robotic Operating System, ROS. It has a 360-degree planar lidar, which allows for simultaneous localization, mapping, and autonomous navigation. They use two wheels for movement, and they contain a custom raspberry pi for use as the robot controller. These were developed for use in teaching, so they are intuitive, and many tasks or behaviours can be rapidly implemented and visualized. However, due to the high costs and big size, large-scale collectives are not feasible with the turtlebot3 robots with most work being done with 1-10 at a time. Therefore, using these large mobile bots for pheromone research would not be particularly appropriate, as we want large numbers of 'dumb' robots reinforcing paths and creating a positive feedback loop as opposed to few 'intelligent robots' which struggle to forage cooperatively in an as efficient way.

Robot Alice [20] is a microrobot focused on modality. The base robot has limited ability just housing a controller, battery, wheels, and a motor. The base module has a serial input/output bus so that additional components (extensions) can be attached to the top such as radio antennas, distance sensors or IR communication photodiodes. The robots have low level local control, and the swarm can be controlled globally using IR and an external computer locally or over the internet. This cheap, low powered robot can run for around 10 hours, and they have been used in many studies. In Caprari *et al* [20], they demonstrate the use of the swarm being controlled over the internet to perform collective transport and Garnier *et al* [5] extends this adding greater localization of the robots and a virtual pheromone system.

Many other mobile robots have been developed with different capabilities. Colias robots [21] are circular, relatively cheap bots that were originally tested with honeybee aggregation. They also use IR sensors for communication but additionally have light sensors to measure ambient light, this is especially useful when working with artificial pheromones represented by visible light. The E-puck robot [22], designed for education of robotics, it has a wide range of abilities and is a more generalised and easier to program SR platform. The vast number of SR robot platforms being used in research demonstrates the interest in and lack of standard SR systems.

TABLE 1
Comparison of available Mobile Robots

Robot	Cost	Weight	Speed	Software
Colias	£25	0.028kg	1.26km/h	C, Basic
E-puck	£580	0.20kg	0.47km/h	C, C++
Kilobot	£13	0.016kg	0.04km/h	C
Alice	N/A	0.011kg	0.14km/h	N/A
Turtlebot3	£540	1kg	0.80km/h	ROS

We require more than just a group of robots to create a useful swarm (at least currently). The aforementioned components of an SR system need to be met, so we additionally need to add support for robot localization, task giving and pheromone communication. Generally, an external workstation and additional hardware is used to track, manage, and control the swarm robots. The usage and application of these mobile robots for SR and pheromone communication will now be discussed by exploring artificial pheromone systems.

2.2 Artificial Pheromones

Pheromone communication is a useful tool in swarm robotics, however representing the pheromones in these systems is not a trivial task [2]. A plethora of research has been done to investigate methods of creating artificial pheromones for use in swarms. An effective pheromone system means that robots only need local sensing, limited memory, and that no centralized control is required, so finding good methods of implementing them is important. There have been a variety of novel pheromone models proposed and developed for the control of SR with varying levels of success.

In nature we examine the use of chemical pheromones [1] for insect swarm communication, however, the usage of this chemical communication method has limitations for SR. Fujisawa *et al* [10] used ethanol as a substitution for pheromone, where each specialized robot (ARGOS-00) had the ability to release alcohol from a tank and detect it via alcohol sensors. This method allows for realistic evaporation and diffusion; however, the autonomous swarms performance in their simulations had a lot of variance. The main issue is that environmental factors such as temperature and humidity make a stark difference, and these are difficult to control especially outside of laboratory conditions. Additionally, it only allows for one pheromone type in the system and releasing chemicals into the environment can be harmful to both humans and animals. Chemical-based pheromone communication allows for a fully autonomous swarm however, in that it requires no centralized control or localization system

(Fujisawa *et al* [11]), but it's not reliable enough for most SR applications so other methods are being explored.

Other than artificial chemical pheromones recent studies have used RFID tags [14], heat sensors, UV glow paint [12], audio and light [5][6][7]. These methods provide platforms to more easily and often cheaply implement artificial pheromone trails compared with using chemical trails. RFID technology can emulate pheromones [14], using radio frequency or magnetic field variations for communication. Here a grid of wireless battery-free RFID tags is placed, and each robot can read/write pheromone values to them without direct contact. This technique requires the robots to have more functionality than other systems, it is also not as intuitive compared with using light or chemical trails. In Mayet *et al* [12] ultraviolet light is emitted by the e-puck robots onto a phosphorescent paint. This left luminous green trails on the arena floor which 'evaporate' over time due to the glow fading out after several minutes. They used this system to mimic ant trails and present a fully autonomous pheromone system. Applying this method to real-world applications would not be feasible due to it needing the environment to be covered with paint and for it to be dark. But it can provide an efficient way to represent trails compared with using chemicals, as each agent doesn't have to release any substances.

This work focuses on simulating systems that use light to represent pheromones, Garnier *et al* [5] experimented with using projected light to represent pheromone trails. They used 5 micro-robots 'Alice' [20] equipped with watch motors connected to wheels for movement and four IR sensors and transmitters that are used for communication and object detection. Each robot also contains an extra module containing two photodiodes for tracking of the light trails and an LED so that the robots can be detected by a camera. Each robot only lays pheromones when the LED is on and thus detected by the camera. The setup includes three main components: the first is a maze built from cardboard with a source and a nest. Then above they placed a digital video camera to track the robots via the red LEDs and a video projector which allows light to be projected across the entire maze. An example construction of a light pheromone projection system is in Figure 1. The virtual pheromones are represented by spots of light (large enough for several robots to move through) projected onto the maze by the projector. The robots themselves followed the basic trail-laying and trail-following behaviour found in ants. Being a physical experiment, this work is prone to a lot of error compared with the simulations they performed in the Webots [24] physics engine. Hence, they could perform evaporation and refreshing of the projector only every 5 seconds as more precision was not necessary. For 5 robots, one path out of the two available consistently had a higher probability of being taken because of pheromone build-up. The time taken to navigate the maze also decreased exponentially until a minimum time was usually found. This paper was essentially a proof of concept of this 'Light pheromone system' and it provides a cheap and simple way of doing these experiments in a laboratory. The experiment conducted is basic yet the development of this system, built upon in later research, is a good starting ground and a similar setup will be utilised in the current work. We can build upon this pheromone system and provide it with more complex environments and rulesets. For instance, they suggest implementing algorithms to allow for adaptability and recruitment within the robot population. This would include robots which can adapt to different tasks autonomously and a system where an optimal number of robots for a given task can be found via recruitment [13]. To provide a different method of localization than camera tracking (which needs a fixed camera and calibration) Hikari *et al* [6] experimented with embedding control information within the projected light which was then read and understood by the micro bots. They used this to provide coordinate data to each robot, it provided a method to efficiently communicate information with the swarm. Before exploring other areas of research, it's worth exploring similar work and more recent usage of light pheromone trails.

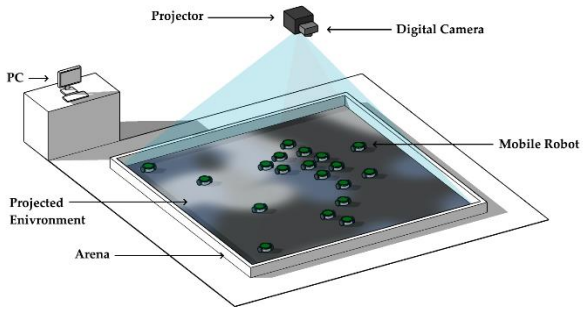


Fig. 1. Example of an artificial pheromone projection system with a digital camera for robot localization.

LCD screens have also been utilised to represent pheromones via light trails, Arvin *et al* [7] used an LCD display and a cheap USB camera aloft to provide a reliable system (COSΦ) for swarm robotic research. They used the Colias-Φ [21] robot and programmed them to mimic ant behaviour, using the 2 light sensors as if they were antennas. The amount of pheromone was represented by the light intensity shown on the LCD, this screen formed the grid in which the robots moved. Localization is achieved in this system by placing unique-black and-white patterns on the robots which can be recognised by the camera. Firstly, this allowed for the orientation of each robot to be established and secondly an encoded ID let the robots be distinguished. This is a cheap, quick, and effective method of localization which detects the positions of swarm robots and releases pheromones accordingly. In addition, they found that increasing the pheromone evaporation half-life increases the stability of the pheromone trail. Interaction between the robots was not defined so the population size did not have a big impact on performance unless due to robot collisions. Na *et al* [9] built on the COSΦ system by implementing environmental effects, such as diffusion and advection, which act on the pheromones. This employed spatiotemporal development of pheromone intensity based on the Navier-Stokes equation to model fluid flow.

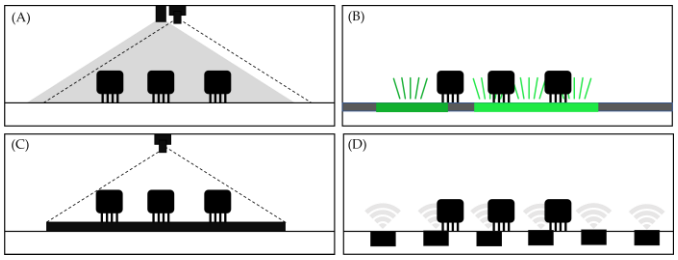


Fig. 2. Simple diagrams of the artificial pheromone models (A) Light projection, (B) UV glow paint, (C) LCD Screen COSΦ, (D) RFID Tags.

Kilogrid (Valentini *et al* [15]) is a purely decentralised virtualization environment for the Kilobot robot. It provides a physical platform that can be easily controlled and tracked by researchers to efficiently test swarm behaviours. Virtualization is used in this system, which is the process of coordinating robots based on a virtual environment as opposed to information from the physical space. The Kilogrid system is built up of three main components. (1) The Kilogrid Module: this is a PCB containing 4 cells which can communicate with the Kilobots using infrared (IR) and provide user feedback about the system using visible light. (2) The Dispatcher: the interface connecting the modules and the workstation. (3) The KilogridGUI: an application developed to control a swarm of robots. Researchers can load virtual environments, change the programming of the Kilobots/ modules (applying different parameters or behaviours) and also record and visualize in real-time data collected from the experiment. In this work a similar application is proposed for use in projection-based swarm systems, KilogridGUI demonstrates the need for these systems and the limitations involved. The Kilogrid system is expensive, due to all the specialized hardware required, and it's also limited by the fact it's a discrete system (due to its grid shape). More recently Kedia *et al* [16] proposed and developed the GenGrid platform. This open-source distributed platform can use a variety of mobile robots (e.g., e-puck, Colias) as opposed to being limited to Kilobot. It also provides a modular grid of nodes for bi-directional communication at a lower cost than Kilogrid. ARK [23] is a similar proposed SR platform which aims to bring standardization to the field and allow for easier, more quantifiable swarm research. Notably, the ARK system uses a set of cameras for tracking the Kilobots and provides virtualized environments for them yet moves them in the physical environment. Yet ARK requires lots of centralized control and it struggles to log large amounts of data, with the data having to be downloaded after the experiment. This is where Kilogrid and GenGrid have an advantage, they allow for mass data collection and testing of SR systems. These proposed systems massively enhance the capabilities of mobile robots and allow for efficient testing of SR systems and collective behaviour. However, these discrete platforms do not translate intuitively to real-world scenarios which are continuous and usually unstructured.

2.3 Simulators

Simulating swarm robotic systems is much cheaper, quicker, and simpler than setting up a physical testing environment [2]. There are existing simulators used for SR ranging from specialized software for specific academic research to more generalized systems with robots and complex environments ready to use [2]. Webots [24] is an intuitive, open-source simulator which allows users to experiment with and develop mobile robots from their collection of virtual actuators and sensors. It provides support for physics, collisions, and collision avoidance systems. It also allows for simulated code written in C, C++ or Java to then be transferred to real robots.

Gazebo [25] is a three-dimensional extension of The Player Project. It's an open-source software and different types of robots can be loaded into the model. It is a newer and more updated simulator compared to Webots meaning that it has additional complexity and processing power.

Enki [26] is a fast, two-dimensional only robot simulator. It can simulate SR systems exponentially faster than doing it physically. Since this work only focuses on 2D swarms, Enki is a good comparison simulation system to use in this work, the physics are limited but this makes very little difference to a flat swarm of mobile robots. Kilogrid, ARK and Gengrid are essentially hybrid simulators, in the fact that the environment and robot control is simulated virtually (an example of virtualization) and the robots are moved in the physical space accordingly.

PyCX [33] is a python-based simulation code repository for complex systems education. It provides a basic Tkinter GUI enabling users to run a wide range of complex tasks including an ant pheromone scenario. PyCX does not simulate robots and instead creates visualizations for multi agent algorithms. The software is aimed at students and researchers, focussing on simplicity, readability, and generalization. This is the closest system to our proposed solution, yet it has some limitations. It provides several scenarios, but in order to run specific SR tasks the user has to directly program these themselves instead of creating them in the GUI. The interface (shown in Figure 3) is basic compared with the other simulators, reflecting the fact that the project is outdated and no longer supported (last updated in 2019).

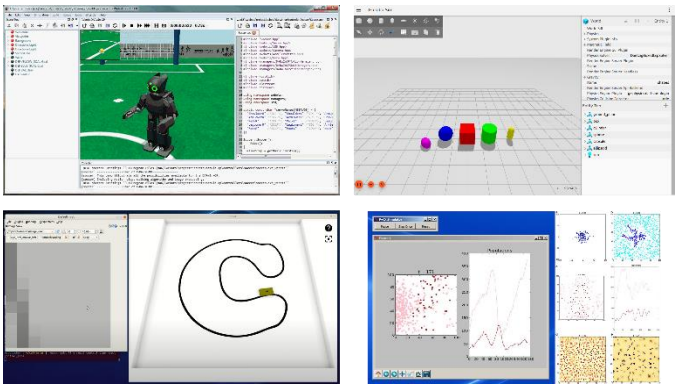


Fig. 3. UIs of the simulators (top left) Webots, (top right) Gazebo, (bottom left) Enki, (bottom right) PyCX.

The discussed simulators use varying levels of abstraction to simulate a swarm's performance. Webots and Gazebo simulate real-world physics of a system, whereas Enki and PyCX focus more on the SR algorithms and performance. Simulating the physics of mobile robots is not technically necessary to assess SR scenarios, so abstractions like these be used in our system. We are aiming to develop a simulator to rapidly develop SR tests and observe SR systems over time, simplifying the environment into a 2-dimensional space using abstraction and focussing on one type of mobile robot will help us to do this.

2.4 Swarm Robotics Metrics and Tasks

To evaluate the feasibility and correctness of our SR software we need to establish what tests and associated metrics would be useful. Nedjah *et al* [2] is a comprehensive study into the field of swarm robotics, included is a section on collective behaviour tasks. The first is spatial organisation (aggregation, dispersion, coverage, pattern formation) and the second is collective motion (exploration, foraging, collective navigation and collective transport). Self-aggregation is regarded as the simplest collective behaviour present in insect societies [29].

The BEECLUST algorithm [27] models' aggregation and is based off day-old honeybees. The robots here do not communicate directly or with pheromone, instead they stop on robot-robot collisions, proportional to an environmental value such as temperature (cue-based aggregation). This algorithm results in robots spending more time in the elevated temperature areas, resulting in more collisions in that area and an aggregation of the swarm. BEECLUST is versatile and simple to implement as there is no pheromone communication and localization is not required as the robots do not need to know their location within the environment [32].

Algorithm 1: BEECLUST Aggregation

Step 1:
Initialize robots randomly throughout the arena
Repeat:
Each robot moves in a random direction
Step 2:
For each robot:
If collision with the arena wall, **then**:
Turn robot around randomly
If collision with another robot, **then**:
Stop the robot moving
Step 3:
For each stopped robot:
Measure the temperature value at current position
Calculate a wait time based on this value and wait
Step 4:
After wait time elapsed:
Robot turns and goes back to step 1

Algorithm 1 is based off the four basic steps provided by Hereford *et al* [31]. BEECLUST has been used in follow up studies, Bodie *et al* [30] focussed on the robustness of swarms when acted upon by another. They concluded that BEECLUST can work in a homogeneous robot swarm where other robot groups are present with their own control strategies. This means that low battery robots could aggregate around a charger, whilst fully charged robots may perform a different task such as foraging, and the aggregation will still work. So, despite BEECLUST aggregation being simple, it can be one function used by robots within a larger system to complete more useful real-world tasks. In more recent work, Arvin et al [28] proposed the use of artificial pheromones to improve the performance of the BEECLUST algorithm. The artificial pheromone system used for this is COSΦ [7] (LCD setup discussed in 2.2), hence the extended BEECLUST algorithm was named ΦClust. They observed improved performance for aggregation with pheromone following, in all configurations of the system. Once they validated the simulations (using PyCX [33]) aligned with the physical observations, they used simulated swarms as it allowed them to find optimal values faster. Each pixel within the arena is modelled as three matrices $I_{\{r,g,b\}}$, the values are calculated from pheromones as:

$$I_c(x,y) = \sum_{i=1}^{n} s_i^c \ \Phi_i(x,y) \quad (1)$$

where $\Phi_i(x,y)$ is the pheromone strength at *(x,y)* and s_i^c represents the influence on the colour of the pixel. The pheromone at each time step is modelled using pheromone injection rate, ι, evaporation rate, e_Φ, and diffusion rate, d_Φ; updated by:

$$\Phi_i(x,y) = e_{i\Phi}\Phi_i(x,y) + d_i\Phi_i(x,y) + \iota_i(x,y) \quad (2)$$

The amount of time that a robot must wait for is calculated relative to the current 'temperature' represented by light intensity:

$$w(t) = w_{max} + \frac{S^2(t)}{S^2(t)+25} \quad (3)$$

where, w_{max} is the maximum waiting time and S is the cue intensity.

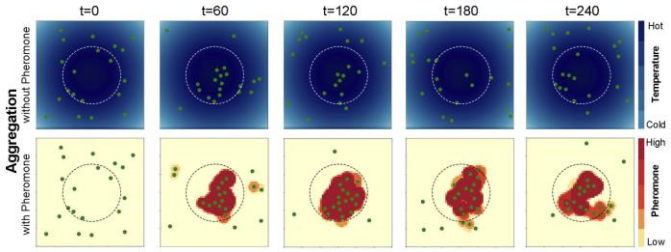


Fig. 4. Comparison of the BEECLUST (above) and ΦClust (below) algorithms, from [28].

To evaluate the performance of these aggregation algorithms, [28] measured aggregation size at regular time steps. The aggregation size is a count of how many robots are in the aggregation area (Figure 4). When comparing the aggregation size using varying population sizes, it may have been better to use the proportion of robots in pheromone instead, as a larger population will intuitively have a larger aggregation size by default. Additionally, ΦClust is harder to implement physically and it uses more system resources due to its complexity compared with BEECLUST.

The performance of a swarm in COSΦ was evaluated in Arvin et al [7] using basic pheromone following. They tracked how well follow robots can follow a pheromone trail, laid down by a single leader robot. Two metrics were used for each test: number of robots following the pheromone trail and the average distance of the swarm robots from the leader.

In this work we focus on developing a simulation system to model these light-based pheromone systems. The tasks and metrics discussed provide ways to evaluate the efficacy and correctness of our simulation system. The next section details the methodology of the system and setup for these tasks.

3 SOLUTION

The proposed solution for this project, is to create a piece of swarm robotic simulation software using Python 3.12. Creating

software with an intuitive graphical user interface allows for a wide variety of experimentation by researchers, without the need for technical knowledge. We built the system in Python as it is one of the most popular modern programming languages, with the goal of being used and updated in future research. Creating a new tool (as opposed to building onto an existing system) opens the door for more swarm robotic research, as no python system like this currently exists. We outline how the rendering system and robot logic work, the functionality of the software and discuss which tasks we tested. Looking at Figure 5, we can see the interface of the proposed software 'CosPy':

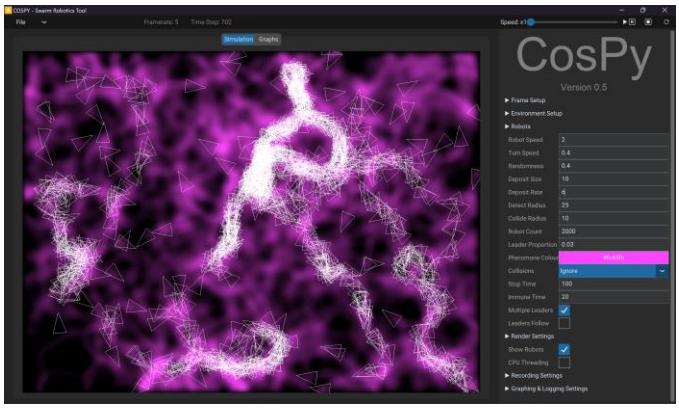


Fig. 5. Example of the CosPy software running a swarm test.

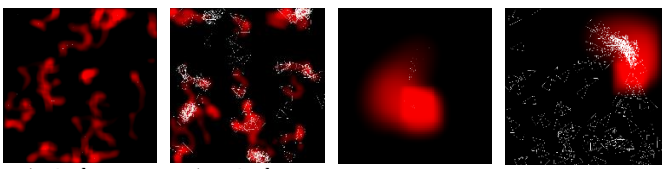
3.1 Pheromone Rendering System

Simple Direct Media Layer (SDL2) is used for the rendering, specifically the PySDL2 interface is used. The SDL2 surface is inserted into a CustomTkinter [34] window so we can have the GUI and swarm simulations in one window. The rendering system has a main drawing surface, of which the pheromone and temperature layers are copied onto each frame. These two layers are 2-dimensional RGBA (red, green, blue, alpha) pixel arrays, more layers can be added if needed. Frame size *(w,h)* and scale *s*, can be specified by the user, decreasing the scale means that we can create very small and quick renders, whilst still having them fill the window.

$I_{\{r,g,b,a\}}$ represents each pixel of the main surface, we can say that $\tau(x,y)$ is the temperature value at *(x,y)* and $\Phi(x,y)$ is the amount of pheromone deposit:

$$I_{\{r,g,b,a\}}(x,y) = \tau(x,y) + \Phi(x,y) \tag{4}$$

To represent the robots there are two available options. Firstly, drawing a triangle for each robot, this allows location and angle to be easily represented. The other method is simply drawing a coloured pixel at each location. This is useful for when we have thousands of robots to draw, but it gives no indication of the individual robot angles.



s=1, pixels | *s=1*, traingles | *s=0.1*, pixels | *s=0.1*, triangles

Fig. 6. Comparison of scales and robot rendering mode, where s = scale.

The artificial pheromone system allows multiple robots to release pheromones into the environment of varying colours. Note that for this work we only use one colour of pheromone at a time, so we only change the alpha value of each pixel of the already filled pheromone surface (*alpha=0* at *t=0*).

The amount of pheromone at each time step t, is determined by four factors:

1. Pheromone Injection, ι, the amount of pheromone released by each robot.
2. Evaporation, e_Φ, rate at which the pheromone decreases linearly.
3. Diffusion, $d_{\Phi,}$ rate at which the pheromone spreads out into the environment.
4. Wind, $w_{x\Phi}$, $W_{y\Phi}$, (or advection [9]) rate at which the pheromone is moved linearly along the axis.

Depending on the configuration of the system, we can have multiple robots releasing pheromone simultaneously. We have two methods of releasing pheromone around the robots, the first is injecting pheromone at a given radius, s_Φ. The second is to fill the subarray $s_\Phi \times s_\Phi$ around the pheromone, this tends to be quicker and there is no detected difference when diffusion is to organically spread out the pheromone.

Pheromone Injection Method 1:

$$\iota(x,y) = \begin{cases} k_\Phi \; if \; \sqrt{(x-x_r)^2 + (y-y_r)^2} \leq \frac{s_\Phi}{2}, \\ 0 \; otherwise, \end{cases} \tag{5}$$

Pheromone Injection Method 2:

$$\iota(x,y) = \begin{cases} k_\Phi \; if \; (x-x_r) \leq \frac{s_\Phi}{2} and \; (y-y_r) \leq \frac{s_\Phi}{2}, \\ 0 \; otherwise, \end{cases} \tag{6}$$

For the rest of this work, we will use method 2 for pheromone injection unless stated otherwise. Using this we can get the total amount of pheromone injection for, N, robots, with a proportion of leader robots (releasing pheromone), μ:

$$\Phi(x,y) = \sum_{i=0}^{N \times \mu} \iota_i(x,y) \tag{7}$$

Evaporation is applied first, this simply reduces the pheromone at each pixel by a fixed value every, e_Φ, time steps:

$$\Phi(x,y) = \frac{1}{e\Phi} \times \Phi(x,y) \tag{8}$$

Diffusion applies a 2D diffusion kernel convolution across the entire pixel array at rate d_Φ. This convolution spreads the pheromone outward from the source, the kernel K is a normalized kernel of size *3*.

$$\Phi(x,y) = \frac{1}{d\Phi} \sum_{w} \sum_{h} \Phi[x,y] \times K[x-w][y-h] \tag{9}$$

Wind dispersal (or advection) moves all pheromones in a given *(x,y)* direction at a rate of (Wx_Φ, Wy_Φ):

$$\Phi(x,y) = \Phi(x + Wx\Phi, y + Wy\Phi) \tag{10}$$

We can combine the pheromone injection and environmental factors to get the strength of pheromone at each pixel *(x,y)*:

$$\Phi(x,y) = \frac{1}{d\Phi}\sum_{w}\sum_{h}\frac{1}{e\Phi}\Phi(x+Wx_{\Phi}, y+Wy_{\Phi}) \times K[x-w][y-h] + \sum_{i=0}^{N\times\mu} \iota_i\,(x,y) \tag{11}$$

Temperature, τ, has its own array of pixel values that the robots can access. A temperature array is created at *t=0*, this is in a normalized gradient array with the centre of the frame as the epicentre. This temperature is used for the aggregation experiments. For each pixel we set the alpha channel using the equation below, giving use values in the range [0,255].

$$\tau(x,y) = 255 \times \left(1 - \frac{\sqrt{\left(x-\frac{w}{2}\right)^2 + \left(y-\frac{h}{2}\right)^2}}{\sqrt{\frac{w^2}{2} + \frac{h^2}{2}}}\right) \tag{12}$$

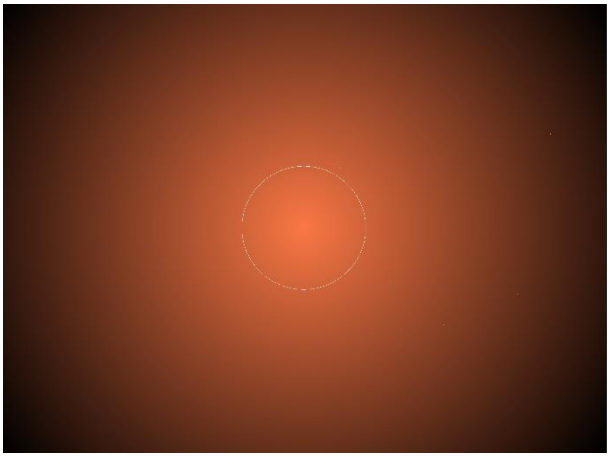


Fig. 7. Temperature array with aggregation detection area shown.

The system also has a simulation speed option, T_x, so that tests can be completed quicker whilst still providing a visual output. We only update the rendered system on every $\underline{T_x}$ time step, this results in faster processing time overall.

3.2 Robots

The simulated robots R do not have size or mass, they are represented by an *(x,y)* point in 2D space. Any individual robot can sense their environment using 2 front facing virtual sensors. These essentially access the raw pixel values from the RGBA surfaces. Comparison is then done on the left pheromone L_Φ and right pheromone R_Φ values, indicating to the robot which way it should turn. If we have a robot movement angle $\theta \in [0,2\pi]$ and a pheromone detection radius, r_Φ, then:

$$\begin{aligned} L_\Phi &= \Phi(x + cos(\theta-1) \times r_\Phi, y + sin(\theta-1) \times r_\Phi) \\ R_\Phi &= \Phi(x + cos(\theta+1) \times r_\Phi, y + sin(\theta+1) \times r_\Phi) \end{aligned} \tag{13}$$

If the array indices are out of range for the pheromone array, then we set the sensor value to 0, unless the boundary condition is 'Wrap'; in that case we loop around the array. These boundary conditions control how the robots react to hitting the virtual arena walls. When colliding they can 'Bounce', reverse their angle on the axis of collision, 'Wrap', loop back to the other side of the arena and 'Stop', don't change any values but prevent the robot from moving in that direction. The 'Wrap' option treats the arena as an infinitely repeating plane, something that we could not do with physical robots.

Each robot's angle θ, is random in the range (-ε,ε) unless prompted to turn toward detected pheromone gradients:

$$\theta(t) = \begin{cases} \theta(t-1) - \Delta\theta \text{ if } L_\Phi > R_\Phi\,, \\ \theta(t-1) + \Delta\theta \text{ else if } L_\Phi < R_\Phi\,, \\ \theta(t-1) + Uniform(-\varepsilon,\varepsilon)\ otherwise\,, \end{cases} \tag{14}$$

Using the angle, we can find that for a given robot with velocity v, at any time t, we have:

$$\begin{aligned} x(t) &= cos(\theta(t)) \times v \\ y(t) &= sin(\theta(t)) \times v \end{aligned} \tag{15}$$

Each robot may be selected as a leader at time t=0. The number of leader robots is $N \times \mu$. Each leader robot will deposit pheromone at each time step based upon the custom pheromone parameters. Non leaders will follow the pheromone paths automatically, there is also an option (*Leaders follow pheromone?*) which allows leaders to also follow pheromone gradients depending on the current test.

For experiments such as BEECLUST we require the robots to behave differently and stop upon a robot-robot collision. To calculate which robots are within collision range, r_R, we cannot do detection uniquely for each robot as there is no efficient way to sense any nearby robots. Instead, at each time step we create a NumPy array of all robot *(x,y)* coordinates in the arena. We then calculate all pairwise Euclidean distances and generate a mask of all *distances* > r_R, each unique robot included in one of these below threshold distances then runs its collision function. Schmickl *et al* [27] provides a formula for the wait time after each collision relative to the temperature sensed by the robot. We use a similar formula in our system to get the wait time at any *(x,y)* coordinate:

$$w(t) = \frac{(w + Im) \times \tau(x,y)}{255} \tag{16}$$

where w is the minimum wait time and Im is the immune time. Immune time is how long after the wait period, the robot can move but a collision will not trigger the wait function. The above equation shows the robot collision mode 'Stop linear to temperature'. We also have 'Stop', the robot stops forever, 'Ignore', robot collisions are not used and 'Stop exponentially to temperature', where $\tau(x,y)^2$ is used in place of temperature.

By default, the robots are processed in an arbitrary order synchronously. We include the option for CPU threading, this assigns each robot a thread (queues used when there are more robots than available threads) and processes them asynchronously, similar to a physical mobile robot distributed system. Running the robots asynchronously has worse performance. The overhead of running the simulations in this manner outweighs the benefits of using threading.

The robot class is built to be reprogrammed and updated for the users' needs allowing for more control over the robot behaviour and enabling a broader range of swarm experiments to be performed in the future. So, the equations discussed in this section relate only to the base behaviour of the agents.

3.3 User Interface

The main interface is developed in CustomTkinter, it consists of the option menu, rendering frame and graphs frame. The option menu supports full customization of robot swarm experiments, so that no code changes are needed between tests, making this a versatile system. The option menu consists of the sub menus Frame, Environment, Robots, Render Settings, Recording settings and Graphing/Logging Settings (these are separate to help usability). We built all the features into one window for a better user experience and quicker test development. Table 2 goes into detail on each option available with this menu:

TABLE 2
SWARM EXPERIMENT OPTIONS AVAILABLE IN THE SOFTWARE

Values	Description	Range
FRAME		
w	Render Width	400px to 2000px
h	Render Height	400px to 2000px
s	Scale	0 to 1
ENVIRONMENT		
e_Φ	Evaporation Rate	>0
d_Φ	Diffusion Rate	>0
$w_{x\Phi}$, $W_{y\Phi}$	Wind Rate x,y	>=0
	Boundary Function	{Stop, Bounce, Wrap}
	Use Temperature	{True, False}
	Temperature Colour	[0-255,0-255,0-255]
A_r	Aggregation Radius	>0
ROBOTS		
v	Robot Velocity	Float
Δθ	Angle Turn Rate	0 to 2π
ε	Random Angle Rate	0 to 2π
s_Φ	Deposit Size	0 to min(w,h)
k_Φ	Deposit Rate	0 to 255
r_Φ	Detect Pheromone Radius	>0
r_R	Detect Robot Radius	>0
N	Robot Count	0 to 5000
μ	Leader Robot Proportion	0 to 1
	Pheromone Colour	[0-255,0-255,0-255]
	Collision Function	{Ignore, Stop, Stop linear, Stop exp}
w	Collision Wait Time	>=0
Im	Collision Immune Time	>=0
	Multiple Leaders?	{True, False}
	Leaders Follow Pheromone?	{True, False}
RENDER SETTINGS		
	Show Robots (As Triangles)	{True, False}
	Use CPU Threading	{True, False}
RECORDING SETTINGS		
	File Location	Valid Folder Path
	Test Name	Any String
GRAPHING AND LOGGING SETTINGS		
	Colour Scheme	{seaborn,dark-background}

3.4 Graphs & Logging

The system facilitates live graphs of the experiments and saves the metrics as log files. Live graphs allow selected metrics to be tracked whilst running the experiments, and the log files are used to analyse results post-test and to generate the boxplots used in the results section. The graphs and the log files are updated every 100 time steps. Note that some specific metrics will be discussed in the results section, as custom metrics are used for some tests.

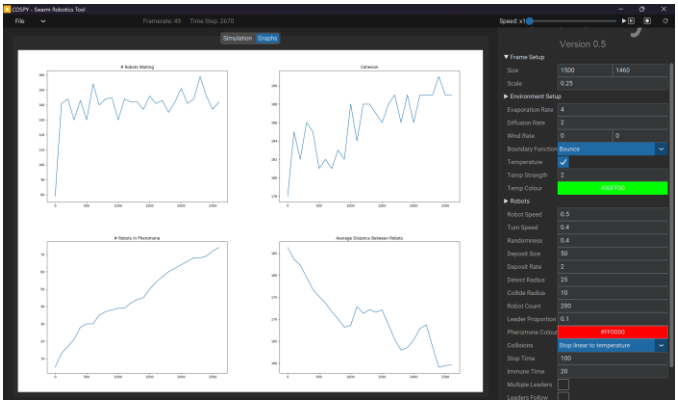


Fig. 8. Example of the live graphing system within CosPy.

The default metrics shown on the graphs are:

Aggregation – The number of robots of maximum distance, A_r, from the centre of the arena.

Number of robots waiting – How many robots have collided and a currently waiting stationary.

Average distance between robots – The average Euclidean distance between all robot pairs. This should decrease as the robots aggregate due to pheromone or temperature.

Number of robots in pheromone – How many robots are currently releasing pheromone or following a pheromone gradient.

Framerate and the current time step are also tracked, only time step is saved to the log file as framerate is only relevant to the rendering system. We can record the swarm experiments to a video file (Figure 9), enabling experiment playback if needed. There is a single record button on the menu bar so that we can control which frames of an experiment are recorded, these can be saved with the live graphs embedded within the video file.

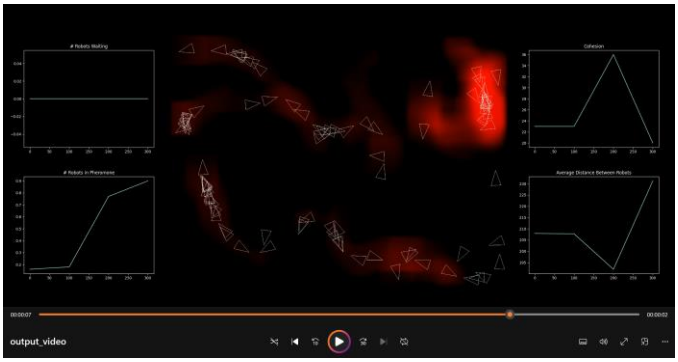


Fig. 9. Screenshot from an experiments recording file.

3.7 Experimental Setup

We evaluate the performance of CosPy by setting up and running several swarm experiments. Four experimental setups have been selected to test the correctness and versatility of this system. Each test is repeated 5 times and results are shown on boxplot.

1. Pheromone Following
2. BEECLUST
3. ΦClust

1) Pheromone Following

In this experiment pheromone is being released by 1 or more leader robots with a varying number of follower robots. The metrics we track are the number of robots following pheromone and the average distance between robots. In [7] they track the average distance to the leader, but as these tests will have multiple leaders, we will track global average distance instead. Over time separate swarms should merge and show a low average distance as the same pheromone colour is being used for these tests, meaning that each robot is part of the same global swarm. Figure 10 shows the finite state machine for each of the follow robots in this system:

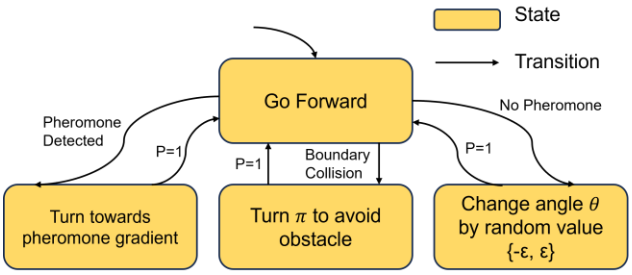


Fig. 10. Finite state machine of followers in basic pheromone following.

In this scenario we investigate the effects of evaporation rate, e_{Φ}, leader robot proportion, μ, and the boundary function on the number of robots following pheromone and the average distance of robots. Each experiment will be repeated 5 times and boxplots will be used to plot the results of each independent experiment. Table 3 shows the parameters used in these tests, with the sets representing the test values used.

TABLE 3
EXPERIMENTAL PARAMETERS FOR PHEROMONE FOLLOWING

Values	Description	Range / Value(s)
w	Render Width	400px
h	Render Height	400px
s	Scale	1
e_{Φ}	Evaporation Rate	{1, 2, 5, 20, 0 (None)}
d_{Φ}	Diffusion Rate	10
$w_{x\Phi}$, $W_{y\Phi}$	Wind Rate x,y	0,0
v	Robot Velocity	1
Δθ	Angle Turn Rate	0.4
ε	Random Angle Rate	0.4
s_{Φ}	Deposit Size	50
k_{Φ}	Deposit Rate	2
r_{Φ}	Detect Pheromone Radius	25
N	Robot Count	100
μ	Leader Robot Proportion	{1/N,2/N,0.05,0.1,0.5}
	Boundary Function	{Stop, Bounce, Wrap}

2) BEECLUST

The next two experiments use honeybee aggregation to essentially perform consensus achievement towards finding the area where temperature is highest. We use the standard BEECLUST model [27] to measure aggregation of the robots towards the epicentre of the arenas temperature $\tau(x,y)$. No pheromone is used in this scenario, when robots collide, they wait for $w(t)$, this wait time is relative to the temperature as outlined in equation (16).

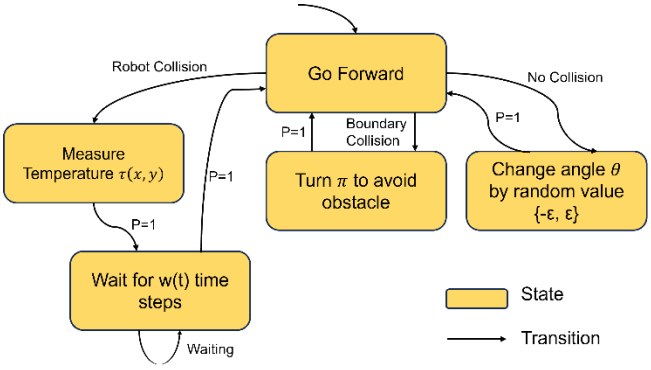


Fig. 11. Finite state machine for each robot in BEECLUST.

We investigate the performance of aggregation using this model using varying robot collision radii, $r_{R,}$ and random angle turn rate, ε. We aim to keep the parameters relatively similar to those used in the original work, as to allow for comparison of the final results.

3) ΦClust

ΦClust [28] is the extension of BEECLUST, where all robots will only release pheromone whilst waiting, relative to the detected temperature $\tau(x,y)$. Figures 11 and 12 show the different finite state machine diagrams for the robots in each clustering scenario, we can see that the only change is the addition of pheromone communication. We run the exact same experiments for both ΦClust and BEECLUST.

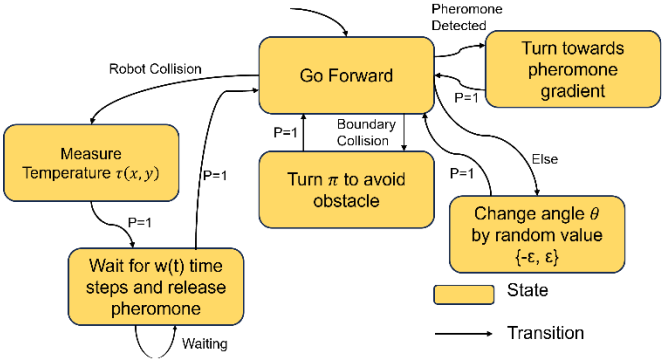


Fig. 12. Finite state machine for each robot in ΦClust.

TABLE 4
EXPERIMENTAL PARAMETERS FOR ΦClust and BEECLUST

Values	Description	Range / Value(s)
w	Render Width	400px
h	Render Height	400px
s	Scale	1
e_{Φ}	Evaporation Rate	1
d_{Φ}	Diffusion Rate	4
$w_{x\Phi}$, $W_{y\Phi}$	Wind Rate x,y	0,0
v	Robot Velocity	1
Δθ	Angle Turn Rate	0.4
ε	Random Angle Rate	{0,0.1,0.5,1,π}
s_{Φ}	Deposit Size	100
k_{Φ}	Deposit Rate	0.01
r_{Φ}	Detect Pheromone Radius	25
N	Robot Count	100
μ	Leader Robot Proportion	1
	Boundary Function	Bounce
w	Collision wait time	300
Im	Collision immune time	25
r_R	Detect Robot Radius	{1,3,10,20,50}

5 RESULTS

In this section, results for the swarm robot experiments are presented and discussed. Each unique swarm test was prepared exclusively in CosPy and repeated five times, they took between 1 and 10 minutes to run. We use time steps to track the passage of time as opposed to seconds, as the framerate in our system is not always consistent and a synchronous runtime is used; meaning if we repeated this again, we would get identical results at each time step, *t*. Log CSV files were updated every *N* time steps, these contained the swarm metrics (stated in previous section) needed to generate the boxplots with matplotlib.

5.1 Pheromone Following

1) Evaporation Rate

Here we investigate the effects of evaporation rate on the both the rate at which the leader can gain follower robots and the swarm's ability to consistently follow pheromone trails over time (stability of the system). We use $e_\Phi \in \{1, 2, 5, 20, 0 \text{ (None)}\}$ for time steps *t=0* to *t=2000* at intervals of 200 to investigate convergence rate and time steps *t=0* to *t=10000* at intervals of 1000 to investigate stability of the pheromone following. These tests were performed with a single leader robot *(μ =1/N)* using the 'Bounce' boundary condition.

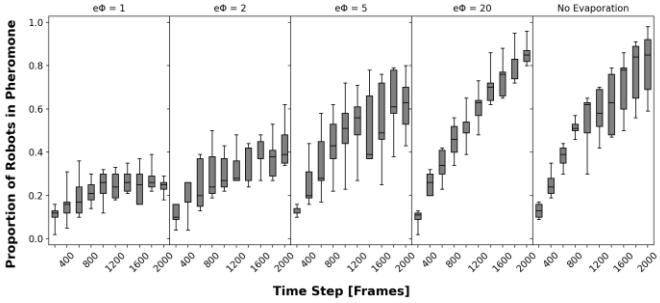


Fig. 13. Results for proportion of robots in pheromone for the pheromone following scenario, every t=200 time steps up to t=2000, using evaporation rates, $e_\Phi \in \{1, 2, 5, 20, 0 \text{ (None)}\}$.

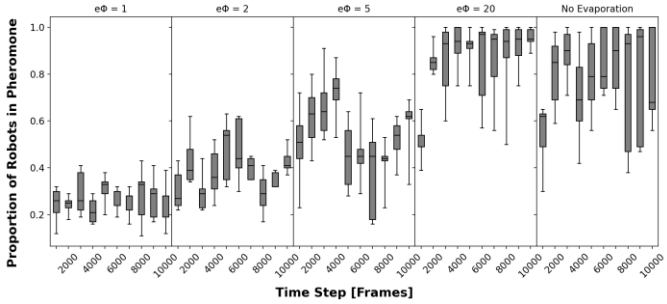


Fig. 14. Proportion of robots following pheromone over time for evaporation rates, $e_\Phi \in \{1, 2, 5, 20, 0 \text{ (None)}\}$ *up to t=10,000.*

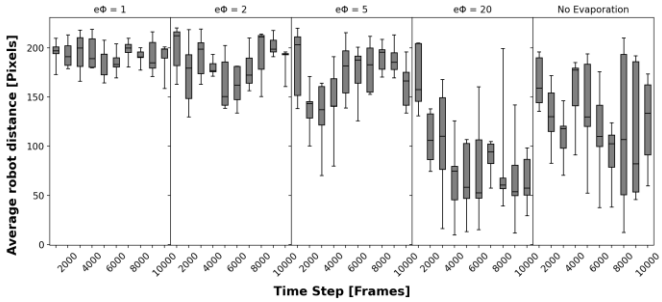


Fig. 15. Average robot distances over time for evaporation rates, $e_\Phi \in \{1, 2, 5, 20, 0 \text{ (None)}\}$ *up to t=10,000.*

Figure 13 demonstrates that as the evaporation rate decreases the proportion of robots following pheromone increases at a greater rate. This is because a decreased evaporation rate results in a higher global pheromone coverage, increasing the probability that an individual robot, *R*, will 'discover' and follow pheromone. *'No evaporation'* performed worse than e_Φ*=20*, caused by an oversaturation of pheromone in the system when there is no evaporation present. As the system becomes oversaturated, the pheromone trails contain less useful information. This results in robots following old trails which can become disconnected from the current leader. We can see an example of this happening in Figure 20 *(e_Φ=20)* where the robots get separated from the leader and become trapped, in nature this phenomenon is known as an 'ant mill' or 'death spiral'. Figure 14 shows the effect of this over time, the system with no evaporation has a much larger range of values for the proportion of robots in pheromone, we can say this system is less stable. This is supported by Figure 15, which shows that despite a large number of robots following pheromone, the robots are far apart and do not behave as a cohesive swarm. The average global robot distance was high for $e_\Phi \in \{1, 2, 5\}$, the pheromone trails dissipate so quickly that most robots do not enter the pheromone, resulting in the robots being randomly distributed. Our results show that evaporation stabilizes the system; (consistent with Arvin *et al* [7]) for each swarm there is an optimal evaporation rate, e_Φ*, which results in a maximum proportion of robots in pheromone and a minimum average robot distance.

2) Leader Proportion

The leader proportion, *μ*, is the proportion of robots that release pheromone, these randomly selected leaders may or may not follow the pheromone themselves. We investigate the differences in both cases here (*Leaders follow pheromone?* ∈ *{False, True}*) for $\mu \in \{1/N, 2/N, 0.05, 0.1, 0.5\}$, each plot shows time steps *t=0* to *t=10000* at intervals of *1000*. Figures 16 and 17 show the configuration where leader robots do not follow.

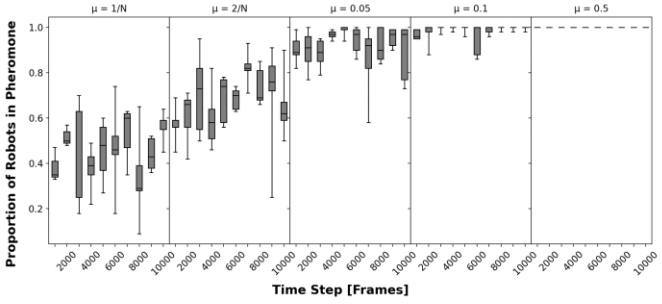


Fig. 16. Proportion of robots following pheromone over time for leader proportions, μ ∈ {1/N, 2/N, 0.05, 0.1, 0.5} where leaders do not follow.

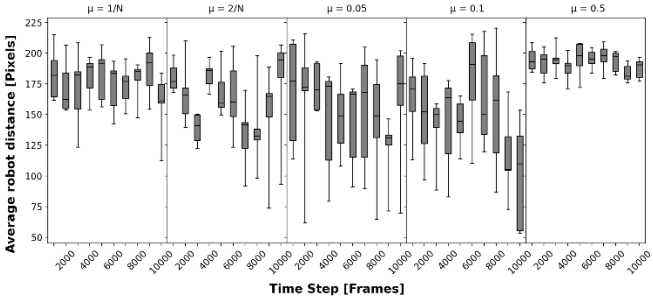


Fig. 17. Average robot distances over time for leader proportions, μ ∈ {1/N, 2/N, 0.05, 0.1, 0.5} where leaders do not follow.

When leader robots do not follow pheromone, we observe that increasing μ results in a larger proportion of robots in pheromone. For μ=0.5 all robots are in pheromone at each time step suggesting that the arena pheromone coverage is close to 100%. This is supported by Figure 17, which shows that the average robot distance in this configuration is the same as a system with no pheromone, again we see that oversaturation results in the pheromone being meaningless.

For the scenario where robots do follow pheromone (Figures 18 and 19), we get a different outcome. We see a similar pattern of robots following pheromone, yet the values for this have considerably less variance at each time step. Figure 19 shows that for high μ values we get reduced average robot distance instead. This is because when leader robots can follow, they can gather and stay in saturated pheromone trails. There is large variance in the average robot distance for μ>=0.1 for two main reasons. Firstly, multiple saturated areas of pheromone can exist in the same arena, resulting in disconnected swarms. Secondly, when the robots converge to an average distance of ~0, the pixel values all become 255 meaning that the robots sensors detect the same values for L_Φ and R_Φ. Robots within this have no pheromone gradient to follow and they end up moving out of this saturated zone.

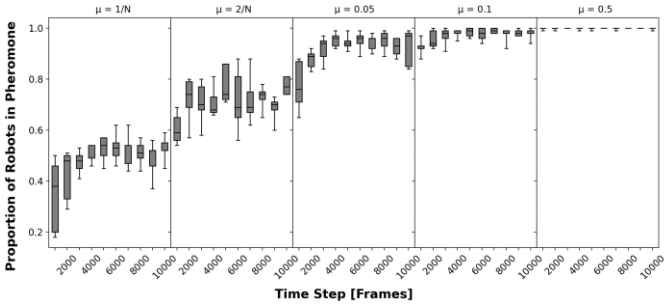


Fig. 18. Proportion of robots following pheromone over time for leader proportions, μ ∈ {1/N, 2/N, 0.05, 0.1, 0.5} where leaders follow.

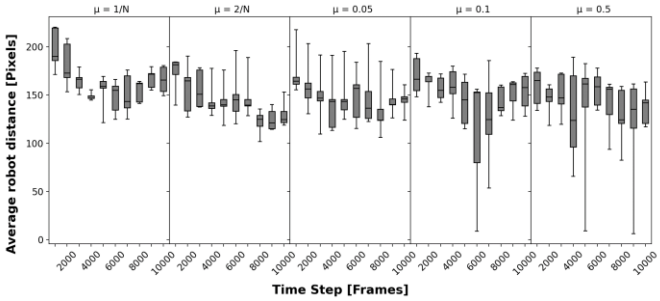


Fig. 19. Average robot distances over time for leader proportions, μ ∈ {1/N, 2/N, 0.05, 0.1, 0.5} where leaders follow.

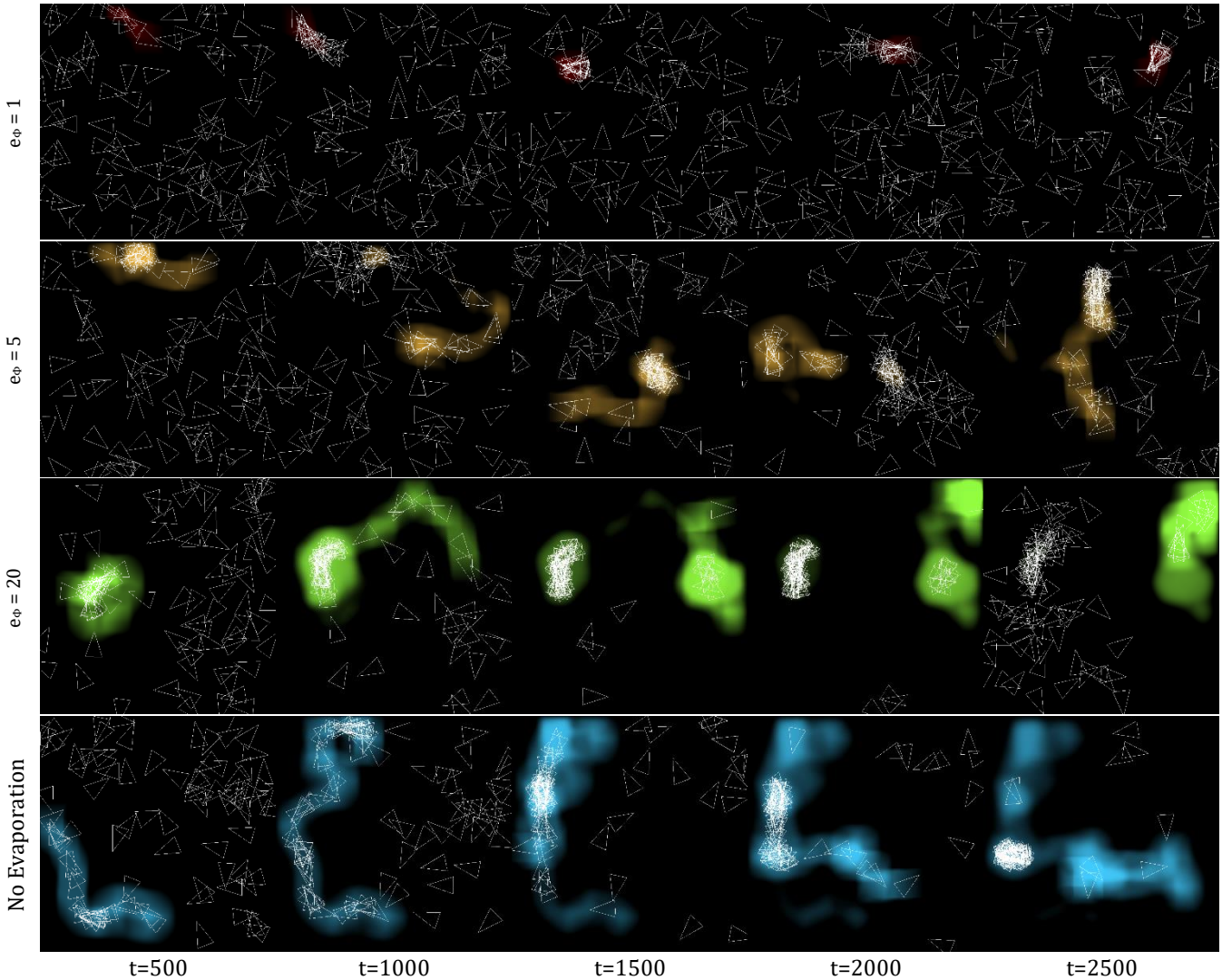


Fig. 20. Images of the robot arena at time steps {500,1000,1500,2000,2500} for evaporation rates {1,5,20, No Evaporation}. N=100 and the pheromone colour for each test is just to distinguish between them has no impact on the performance of the swarm.

3) Boundary Function

The final experiments performed for the pheromone following scenario investigate the effect of the arena boundary function on the proportion of robots in pheromone (following the trails). For each of the three boundary conditions ∈ *{Stop, Bounce, Wrap}*, we run the pheromone following scenario 10 times, to get the averages shown in Figure 21. From this figure we observe that 'wrap' outperforms the other two boundary functions in both convergence and stability of robots in pheromone. 'Stop' results in considerably less robots following pheromone trails at each time step. This outcome is expected as the probability that a robot, *R*, is in pheromone at time step *t* is proportional to the area that the robot has explored (increasing their chances of finding a pheromone trail). In our system, 'wrap' results in the highest rate of area exploration as the robots can detect and pass through the arena walls increasing the exploration rate. Whereas 'Stop' limits the robots and they spend more time stuck to boundaries.

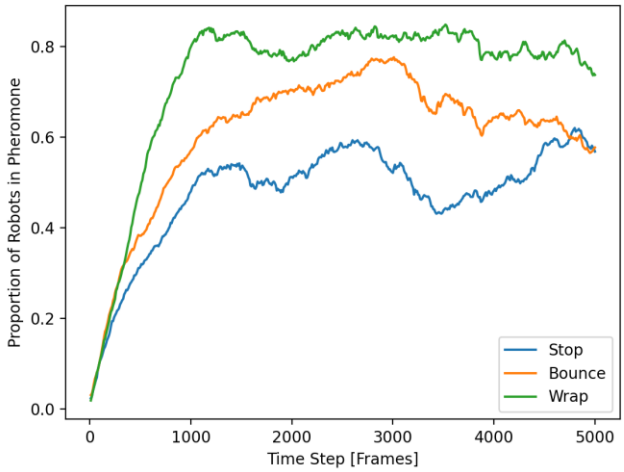


Fig. 21. Effect of the boundary function on the average proportion of robots following pheromone gradients every 100 time steps up to t=5000. For boundary functions ∈ {Stop, Bounce, Wrap}, e_Φ =10 and μ = 1/N; each test was repeated 10 times to get the mean values.

5.2 Aggregation with BEECLUST

1) Detect Robot Radius

In this experiment we see how the robot collision radius, r_R, effects the aggregation of the robots to the centre of the arena where the temperature is highest. We test varying values of radii, r_R ∈ *{1, 3, 10, 20, 50}*, this essentially changes the simulated size of the robots. However, they are represented by points, so any two robots may have a *distance* < r_R, meaning we do not have issues with the aggregation area getting too full. This abstraction is used as this work focuses on simulated swarms, if we were extending this to comparison physical robots then we would give the robots impassible boundaries. Each test runs to *t=10000*, taking around 6 minutes as the framerate is lower when using the collision detection system.

Looking at Figures 22 and 23 we observe that increasing the collision radius, increases the aggregation size and decreases the average robot distance up to a point. For the lowest collision radius, r_R=*1*, we observe the worst aggregation performance. The probability of collisions in this setup is too low for any meaningful movement of the swarm towards the temperature to occur. As r_R increases, the rate of collisions increases resulting in higher aggregation sizes and more robots waiting at any time t. We see that when the radius is increased to *50px* the performance decreases. This is because at such a large radius the rate of robot collisions is far higher, resulting in less movement and therefor a lower robot exploration rate. This lower rate means that it is less likely that a robot will move into the aggregation area and wait there for *w(t)*, but we do still see some aggregation. In these tests the maximum aggregation size observed was 52/100 robots and the minimum average robot distance observed was 140.

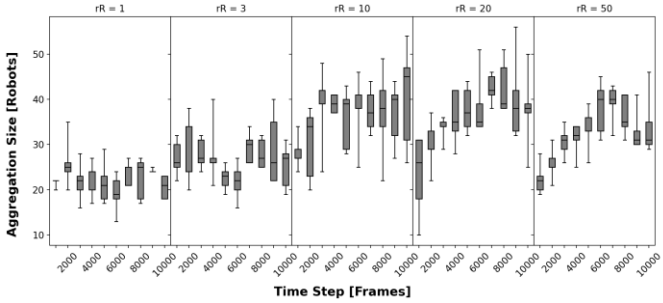


Fig. 22. Aggregation size in BEECLUST scenario, up to t=10,000 for detect robot radii r_R ∈ {1, 3, 10, 20, 50}.

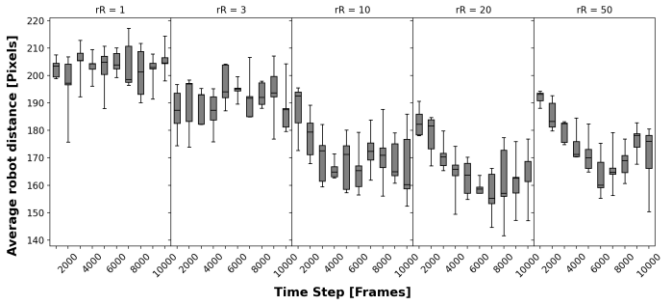


Fig. 23. Average robot distances in BEECLUST scenario, up to t=10,000 for detect robot radii r_R ∈ {1, 3, 10, 20, 50}.

5.3 Aggregation with ΦClust

1) Detect Robot Radius

Here we performed the same tests as with BEECLUST but added in the pheromone following functionality of ΦClust (refer to Figure 12). Pheromone is release by all robots where deposit size s_Φ=*100* and deposit rate k_Φ=*0.01*, meaning that at each time step, 0.01 units of pheromone is added to each pixel in the *100x100* array surrounding each robot. These values were fine-tuned to ensure we did not oversaturate or under saturate the arena with pheromone.

Using ΦClust we observe better results for the cue-based aggregation performance as also observed in [28]. From Figure 24 we observe that the system reaches higher aggregation sizes and converges faster when using pheromone following. For the values r_R ∈ *{10, 20}*, the aggregation size reaches 100% of the robots. These results show that pheromone helps massively with the robots aggregating in this experimental setup. For r_R=*10*, there is one test where the aggregation size lowered to 0 after previously being 100, yet the average distance of robots was still low (see Figure 25). This was caused by the robots being in a dense swarm, thus the pheromone in the area was too saturated. This allows robots to move away from the aggregation area which can then lead the swarm in the wrong direction. Once the swarm has become this compact, little exploration takes place, and the robots cannot

escape the pheromone. This is good if the swarm stays in the aggregation area, yet detrimental to the aggregation size if not.

From Figures 24 and 25 we again see that large collision radii such as $r_R=50$ do not perform well. The probability of collisions being too high and reducing robot exploration, still reduces performance. However, pheromone provides the robots with direction for their movement, and we observe better performance than without pheromone. From our results, we conclude that r_R makes a bigger performance impact on aggregation in ΦClust compared to BEECLUST.

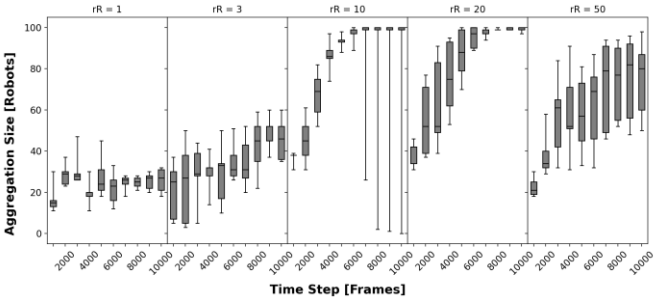


Fig. 24. Aggregation size in ΦClust scenario, up to t=10,000 for detect robot radii $r_R \in \{1, 3, 10, 20, 50\}$.

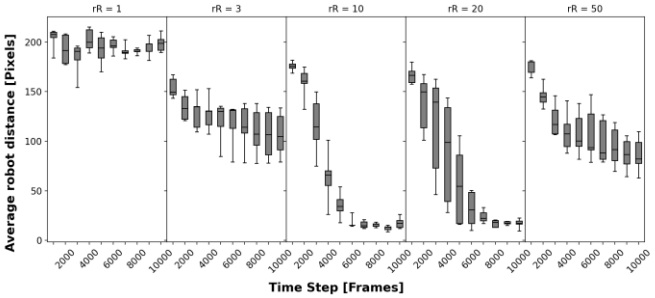


Fig. 25. Average robot distances in ΦClust scenario, up to t=10,000 for detect robot radii $r_R \in \{1, 3, 10, 20, 50\}$.

2) Random Angle Rate

We investigate the effects the random angle rate, ε, on swarm aggregation in the ΦClust scenario only. The previous test not only investigated the effects of robot detection radius, but also to compare the performance BEECLUST and ΦClust. Whereas here we just focus on random angle rate, ΦClust is used, as it performs better. We use $\varepsilon \in \{0, 0.1, 0.5, 1, \pi\}$ for time steps $t=0$ to $t=10000$ and plot at intervals of 1000 time steps. Figures 26, 27 and 28 show the aggregation size, average robot distance and total number of robots currently waiting respectively.

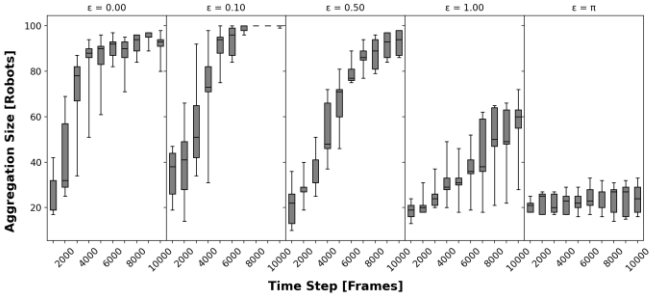


Fig. 26. Aggregation size in ΦClust scenario, up to t=10,000, for random angle rates, $\varepsilon \in \{0, 0.1, 0.5, 1, \pi\}$.

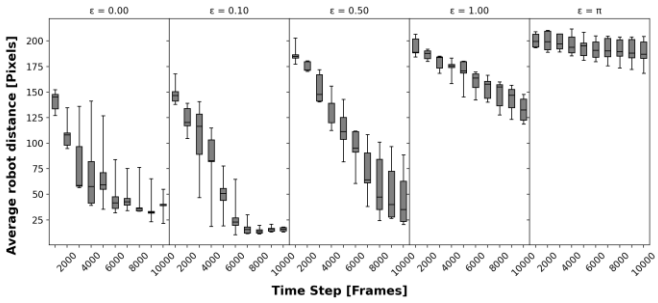


Fig. 27. Average robot distances in ΦClust scenario, up to t=10,000, for random angle rates, $\varepsilon \in \{0, 0.1, 0.5, 1, \pi\}$.

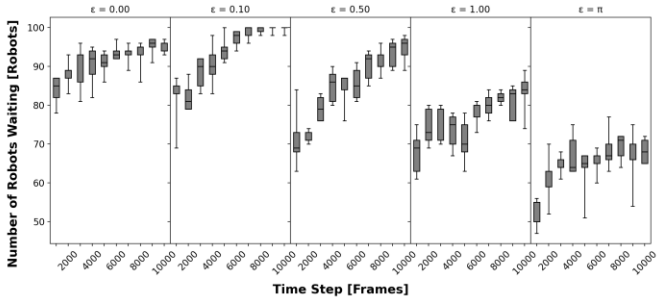


Fig. 28. Total number of robots currently waiting in ΦClust scenario, up to t=10,000, for random angle rates, $\varepsilon \in \{0, 0.1, 0.5, 1, \pi\}$.

We observe that low values of ε perform best on aggregation. $\varepsilon=0.1$ performed best, yet it was only slightly better than having no random angle movement at all. This is because for $\varepsilon=0$, a robot, R, can never get to certain (x,y) positions based on their starting conditions. This can be overserved in Figure 28, where for ε=0 we do not have a point where all the robots waiting as there are fewer collisions than $\varepsilon=0.1$. Yet as the angle increases towards $\varepsilon= \pi$, the movement of each robot decreases, thus the probability of collisions and probability of movement into the aggregation area is reduced. We can see from Figure 26 that the aggregation size changes very little over time for $\varepsilon= \pi$, because there is such little movement of the robots. So, a low random angle rate is best as it maximizes the exploration rate of the robots and ensures they can reach all areas of the arena.

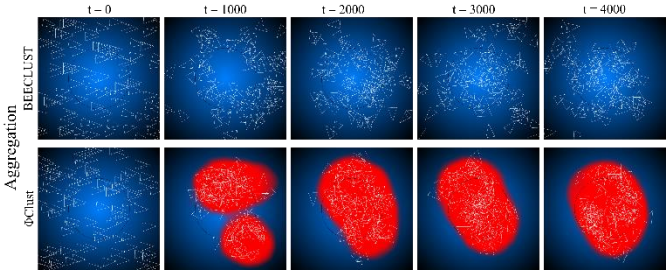


Fig. 29. Comparison of BEECLUST and ΦClust for $r_R=10$.

6 Evaluation

The proposed system has succeeded in providing a platform to develop and evaluate the performance of SR algorithms. Our results are repeatable and align with previous works [7][27][28], showing that the algorithms were correctly implemented in CosPy, this demonstrates the feasibility and reliability of the system. We observed that:

- A balanced evaporation rate is essential for a stable robot swarm.
- When there is high pheromone coverage in the system, there is less useful communication between robots, and they become randomly distributed.
- The most effective boundary condition is allowing robots to 'wrap' around the virtual arena.
- ΦClust performs better than BEECLUST at aggregation, in all experiments that we performed.
- A small random angle rate performs best for aggregation as it maximizes the exploration rate.

It was easy to change between the experimental setups without closing the software or changing the code. Being able to quickly implement swarm algorithms and test them was the main aim of this work, and for the selected tasks we have succeeded. The performance of the system is good, and we can simulate thousands of robots whilst keeping high frame rates. Our results showed that fine-tuning of swarm parameters is essential for having stable swarms and good aggregation. Fine-tuning of these parameters can be done quickly in CosPy, just by changing a value, restarting the runtime, and studying the swarm metric graphs. This shows the usefulness of CosPy in fine-tuning parameters and testing SR tasks before implementing them in physical mobile robots. Another advantage of the virtual system is the ability to use the 'Wrap' boundary condition. In our tests 'wrap' was the best performing boundary condition as it results in increased exploration of the arena. Implementing this into the physical light pheromone system is not possible, as robots cannot move to alternating sides of a space instantaneously. Buhl *et al* [36] used a ring-shaped arena, which allowed simulated locusts to loop around. Creating a physical arena in this manner would be incredibly difficult and would require specialized mobile robots, additionally it wouldn't be useful for real world tasks.

The software does have potential limitations. We observe low system performance when using multiple surface layers (e.g. temperature and pheromone) and a large number of robots. This is most profound when the image is scaled, due to the PySDL2 surface scaling system being slow. We have optimised the system, yet this could be an issue in future work, if more environmental layers need to be added. Another limitation is the over generalization and specialization of the tool. For more complex tasks (such as collective transport), they may need to add substantial amounts of code to the system. On the other hand, for basic tasks many of the features will go unused and a more lightweight solution may be possible. However, through our results we have demonstrated that the software provided a good balance between complexity and usability, providing the user with a wide range of options whilst ensuring simplicity. Other simulators have this same issue, attempting to strike this critical balance to best assist future research.

7 Conclusion

In this work we have developed an open-source swarm robotics simulator, 'CosPy'. The software performed correctly on selected tasks, achieving our goal of developing a cheap, simple, and robust platform. There is substantial future work that can build upon the proposed CosPy system. Firstly, several systems were not directly tested in this work such as the wind, multi-coloured pheromones and using multiple competing swarms. The addition of walls and movable objects into the environment is also something that we would add in the future, to enable more complex systems and simulations of real-world tasks. The system could eventually be used to manage artificial pheromones for physical robots, in a light projection environment [5][7]. Future work using this system for physical swarm tests would need to add mobile robot localization and provide an interface for programming the robots. Overall, the system is a reliable and efficient testing platform for future work. The source code is freely available online at [35].

References

[1] Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. Swarm intelligence, 1, 3-31.

[2] Nedjah, N., & Junior, L. S. (2019). Review of methodologies and tasks in swarm robotics towards standardization. Swarm and Evolutionary Computation, 50, 100565.

[3] Moussaid, M., Garnier, S., Theraulaz, G., & Helbing, D. (2009). Collective information processing and pattern formation in swarms, flocks, and crowds. Topics in Cognitive Science, 1(3), 469-497.

[4] Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence, 7, 1-41.

[5] Garnier, S., Tache, F., Combe, M., Grimal, A., & Theraulaz, G. (2007, April). Alice in pheromone land: An experimental setup for the study of ant-like robots. In 2007 IEEE swarm intelligence symposium (pp. 37-44). IEEE.

[6] Hiraki, T., Fukushima, S., & Naemura, T. (2016, December). Projection-based localization and navigation method for multiple mobile robots with pixel-level visible light communication. In 2016 IEEE/SICE International Symposium on System Integration (SII) (pp. 862-868). IEEE.

[7] Arvin, F., Krajník, T., Turgut, A. E., & Yue, S. (2015, September). COSΦ: Artificial pheromone system for robotic swarms research. In 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 407-412). IEEE.

[8] Garnier, S., Jost, C., Jeanson, R., Gautrais, J., Asadpour, M., Caprari, G., & Theraulaz, G. (2005). Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In Advances in Artificial Life: 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005. Proceedings 8 (pp. 169-178). Springer Berlin Heidelberg.

[9] Na, S., Qiu, Y., Turgut, A. E., Ulrich, J., Krajník, T., Yue, S., ... & Arvin, F. (2021). Bio-inspired artificial pheromone system for swarm robotics applications. Adaptive Behavior, 29(4), 395-415.

[10] Fujisawa, R., Imamura, H., Hashimoto, T., & Matsuno, F. (2008, September). Communication using pheromone field for multiple robots. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1391-1396). IEEE.

[11] Fujisawa, R., Dobata, S., Sugawara, K., & Matsuno, F. (2014). Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. Swarm Intelligence, 8, 227-246.

[12] Mayet, R., Roberz, J., Schmickl, T., & Crailsheim, K. (2010, June). Antbots: A feasible visual emulation of pheromone trails for swarm robots. In ANTS Conference (pp. 84-94).

[13] Talamali, M. S., Bose, T., Haire, M., Xu, X., Marshall, J. A., & Reina, A. (2020). Sophisticated collective foraging with minimalist agents: A swarm robotics test. Swarm Intelligence, 14(1), 25-56.

[14] Sakakibara, T., & Kurabayashi, D. (2007). Artificial pheromone system using RFID for navigation of autonomous robots. Journal of Bionic Engineering, 4(4), 245-253.

[15] Valentini, G., Antoun, A., Trabattoni, M., Wiandt, B., Tamura, Y., Hocquard, E., ... & Dorigo, M. (2018). Kilogrid: a novel experimental environment for the Kilobot robot. Swarm Intelligence, 12, 245-266.

[16] Kedia, P., & Rao, M. (2021, May). GenGrid: A Generalised Distributed Experimental Environmental Grid for Swarm Robotics. In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1910-1917). IEEE.

[17] Rubenstein, M., Cornejo, A., & Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. Science, 345(6198), 795-799.

[18] Rubenstein, M., Ahler, C., & Nagpal, R. (2012, May). Kilobot: A low cost scalable robot system for collective behaviors. In 2012 IEEE international conference on robotics and automation (pp. 3293-3298). IEEE.

[19] Amsters, R., Slaets, P. (2020). Turtlebot 3 as a Robotics Education Platform. In: Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., Obdržálek, D. (eds) Robotics in Education. RiE 2019. Advances in Intelligent Systems and Computing, vol 1023. Springer, Cham. https://doi.org/10.1007/978-3-030-26945-6_16

[20] G. Caprari, The Autonomous Micro Robot "Alice": a platform for scientific and commercial applications. MHA'98. Proceedings of the 1998 International Symposium on Micromechatronics and Human Science. - Creation of New Industry - (Cat. No.98TH8388)

[21] Arvin, Farshad, et al. "Colias: An autonomous micro robot for swarm robotic applications." International Journal of Advanced Robotic Systems 11.7 (2014): 113.

[22] Mondada, Francesco, et al. "The e-puck, a robot designed for education in engineering." Proceedings of the 9th conference on autonomous robot systems and competitions. Vol. 1. No. CONF. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[23] Reina, Andreagiovanni, et al. "ARK: Augmented reality for Kilobots." IEEE Robotics and Automation letters 2.3 (2017): 1755-1761.

[24] Michel, O. (2004). Cyberbotics ltd. webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems, 1(1), 5.

[25] Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) (Vol. 3, pp. 2149-2154). IEEE.

[26] Magnenat, S., Waibel, M., & Beyeler, A. (2011). Enki: The fast 2D robot simulator. URL http://home. gna. org/enki.

[27] Schmickl, T., & Hamann, H. (2011). BEECLUST: A swarm algorithm derived from honeybees. Bio-inspired computing and communication networks, 95-137.

[28] Arvin, F., Turgut, A. E., Krajník, T., Rahimi, S., Okay, I. E., Yue, S., ... & Lennox, B. (2018, October). $\Phi $ Clust: Pheromone-Based Aggregation for Robotic Swarms. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 4288-4294). IEEE.

[29] Bonabeau, E., Theraulaz, G., Deneubourg, J.L., Aron, S. and Camazine, S., 1997. Self-organization in social insects. Trends in ecology & evolution, 12(5), pp.188-193.

[30] Bodi, M., Thenius, R., Szopek, M., Schmickl, T. and Crailsheim, K., 2012. Interaction of robot swarms using the honeybee-inspired control algorithm BEECLUST. Mathematical and Computer Modelling of Dynamical Systems, 18(1), pp.87-100.

[31] Hereford, J., 2011, April. Analysis of BEECLUST swarm algorithm. In 2011 IEEE Symposium on Swarm Intelligence (pp. 1-7). IEEE.

[32] Bodi, M., Thenius, R., Szopek, M., Schmickl, T. and Crailsheim, K., 2012. Interaction of robot swarms using the honeybee-inspired control algorithm BEECLUST. Mathematical and Computer Modelling of Dynamical Systems, 18(1), pp.87-100.

[33] Sayama, H., 2013. PyCX: a Python-based simulation code repository for complex systems education. Complex Adaptive Systems Modeling, 1, pp.1-10.

[34] TomSchimansky, CustomTkinter, GitHub Repository: https://github.com/TomSchimansky/CustomTkinter

[35] Adam Reynoldson, CosPy, GitHub Repository: https://github.com/RENYREYNOLDSON/CosPy

[36] Buhl, J., Sumpter, D.J., Couzin, I.D., Hale, J.J., Despland, E., Miller, E.R. and Simpson, S.J., 2006. From disorder to order in marching locusts. Science, 312(5778), pp.1402-1406.