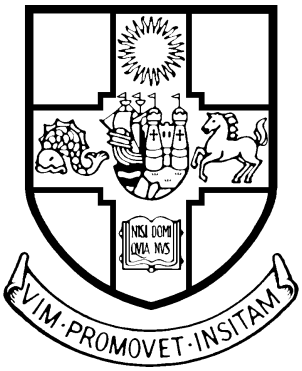
Anti-Aligning Robots

By

Adam Rhys Morris



Department of Engineering Mathematics
University of Bristol

Technical project thesis submitted to the University of Bristol in accordance with the requirements of the degree of Masters of Engineering.

April 2024

Supervisor: Dr Stuart Thompson & Professor Sabine Hauert, Engineering Mathematics

Abstract

The interest in, application of and possibilities offered by swarm robotics and active matter promote the need for research and development into our understanding of collective intelligence. Swarm dynamics are observed throughout nature due to their ability to produce complex yet robust and efficient macroscopic behaviours from microscopic simplicity. Active matter outlines the equations of motion of collectives offering unique insight into the underlying mechanics displayed by living and non-living systems. The combination of these features offer many advantages for similar robotic systems and illustrate how they may be utilised for our benefit. This report describes the investigation into the collaboration of run-and-tumble motion with the anti-Vicsek/alignment model as a collective behaviour for a swarm of Kilobots. Simulations demonstrate the effective dynamics and parameters of the model when compared to traditional alignment and random movement. Experiments examine how this behaviour can be implemented onto a real-world system using simple robotic agents. The importance of combining simulation with experimentation is outlined as a key factor in the development of collective dynamics, exemplifying how both can be utilised in the effective deployment of swarms to the desired application.

This project did not require ethical review, as determined by my supervisor Dr Stuart Thompson.

Contents

1 Introduction

1.1 Motivation

Interest in swarm robotics systems has skyrocketed in recent years, with estimates showing an expected market size growth from USD 0.8 billion in 2023 to USD 3 billion by 2028 [1]. This area of expertise involves the study, design, operation and implementation of groups of robotic agents collaborating to execute a common goal. Applications for this technology are not limited, with possibilities being explored in exploration, surveillance, search and rescue, humanitarian demining, intrusion tracking, cleaning, inspection and transportation of large objects [2]. The primary advantages of swarm robotics include enhanced fault tolerance, scalability, adaptability, cost-effective maintenance and collective intelligence. Continuing research and development is crucial for unlocking the innovative potential of swarm behaviour and enhancing the capabilities of swarm robotics systems.

Inspiration for collective behaviour arises from the evolutionary advantages displayed by swarms in nature. Peleg et al. investigated how structures constructed by swarms of honeybees withstand mechanical stress [3]. These structures are formed solely from individual bees grasping on to one another, as seen in figure 1. Simulation and experimental results show "how super-organismal structure responds to dynamic loading by actively changing its morphology to improve the collective stability of the cluster at the expense of increasing the average mechanical burden of an individual" [3]. The findings underscore the potential of mimicking such natural systems in swarm engineering to improve the resilience and adaptability of robotic collectives.

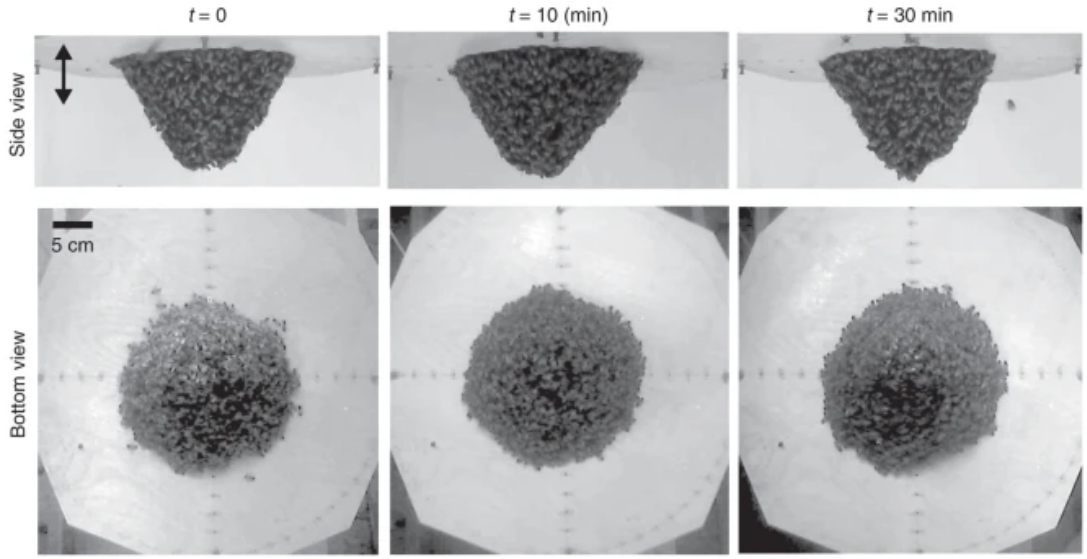


Figure 1: Figure from research investigating mechanical robustness of honeybee swarms subject to dynamic loads of varying orientation, amplitude, frequency and duration [3]. Diagram illustrates the change in the swarm structure over 30 minutes as it is subject to vertical vibrations.

Current challenges in this field involve the macroscopic design and control of collective agents. At the fundamental level, it is important to understand how the rules governing local interactions influence the emergent collective behaviour of a robotic swarm. In particular, seemingly innocuous changes in the interaction rules can lead to drastically different collective dynamics. The development and refinement of algorithms governing these interactions necessitate rigorous testing and experimentation, which are both

time-intensive and resource-heavy. Furthermore, the transition from theory to application has not been extensively explored, demanding further examination and exploration.

Similar motivation stems from the field of active matter. Active matter study concerns the statistical and mechanical modelling of sets of particles in motion [4]. Their ability to take in and dissipate energy pushes these systems far from equilibrium and linearity, making them increasingly difficult to analyse as the complexity rises. However, such models display huge potential not only in physics but in the description of mechanical forces exerted in living systems, particularly at the sub-cellular scale [4]. Active matter calls for the understanding of new models of particle motion in the hopes of uncovering their practicality and applications among a wide array of research areas. Many successful attempts have been made in recent years with the use of computational models [5]. Even so, the challenges in the field are further highlighted with the advancement of their analysis. It is noted "an overarching challenge for the whole field is the distinction between the generic and specific properties and behaviour of a particular active-matter system" [5].

1.2 Macroscopic Dynamics

A common approach to particle swarm dynamics was developed by Vicsek et al. in 1994, coining the term "the Vicsek model" [6]. This novel approach introduced the notion of a simple yet effective model that captures the behaviours of self-ordering groups of particles. Individual agents are modelled identically, moving with a specified constant speed. Crucially, particles contain one key additional behaviour; agents measure the average direction of their neighbours withing a specified local radius. This calculated mean is used to update the current direction of travel, with the addition of a noise parameter, giving rise to collective alignment. Vicsek further introduces a metric for quantifying the order of these systems, being the absolute value of the average normalised velocity. It is shown, through the gradual reduction of noise, that the model exerts a transition from a disordered state to a phase of coherent motion.

The Vicsek model operates on three fundamental components: position, heading, and average neighbour direction. The position vector of a particle is iteratively updated at each time step, Δt, as outlined in equation (1). The velocity vector, $\mathbf{v}_i(t)$, is determined by decomposing the particle's speed into the axial components based on its current heading, $\theta(t)$. This heading is adjusted according to equation (2), where $\langle\theta(t)\rangle_r$ represents the average heading of all local neighbours within a specified radius, r (including the particle itself). A key feature of this model is the incorporation of a noise parameter, $\Delta\theta$, which is drawn randomly from a uniform distribution within the range $[-\eta/2, \eta/2]$. This noise introduces variability in the heading updates, simulating real-world unpredictabilities in agents' behaviour. The average direction, $\langle\theta(t)\rangle_r$, is calculated using the arctangent of the ratio of the average sine and cosine of the local headings, as detailed in equation (3). Finally, the system's degree of collective alignment, or the order parameter, is noted v_a, being assessed through the use of equation (4).

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)\Delta t \tag{1}$$

$$\theta(t+1) = \langle\theta(t)\rangle_r + \Delta\theta \tag{2}$$

$$\langle \theta(t) \rangle_r = \arctan\left(\frac{\langle \sin(\theta(t)) \rangle_r}{\langle \cos(\theta(t)) \rangle_r}\right) \tag{3}$$

$$v_a = \frac{1}{Nv}\left|\sum_{i=1}^{N} \mathbf{v}_i\right| \tag{4}$$

The simplistic nature of this model naturally arises speculation of its validity and accuracy. However, Baglietto and Albano set out to extensively review the Vicsek model through finite-size and dynamic scaling analysis [7]. More specifically, an investigation was carried out on observed order-disorder transitions as to determine whether the results are consistent with a second-order phase transition. This concept was originally challenged by Gregoire and Chate in their paper on the onset of collective and cohesive motion [8]. It was shown that the Vicsek model does in fact display the originally perceived transition behaviours, and not the challenged first-order dynamics, hence is able to accurately capture the desired complex macroscopic dynamics.

Vicsek, along with Zafeiris, return to the topic of swarm dynamics in their 2012 paper on collective motion [9]. The report looks at flocking behaviours seen across a wide range of disciplines, discussing the experiments, mathematics and modelling of these systems. Findings of the paper illustrate the importance and relevance of this field of research and how it can be applied to a range of collective behaviours seen in nature. Examples discussed include, but are not limited to, macro-molecules, bacteria colonies, insects, fish, mammals and humans. Through the analysis of these vastly different systems, some noteworthy similarities were concluded. In general, most patterns of collective motion are universal, with simple models able to capture these behaviours. The context at which these agents exist therefore seems to play little impact on the overall movement experienced by their collectives. Additionally, the complexity of swarms is often accounted for through a singular noise term, characterised by a defined probability distribution. The capabilities and results of the standard Vicsek model are well documented and understood, making it a highly functional architecture for exhibiting collective behaviour.

1.3 Microscopic Motion

At the microscopic level, individual agents exert their own local movement dynamics. Dynamics are dependent on the capabilities of the agents, the environment in which they are moving, their desired behaviour etc. A commonly seen example is Brownian motion; random movement of particles suspended in a fluid resulting from their collision with other fast-moving molecules in said fluid. Memory-less random walks, such as Brownian motion, offer a simplistic, flexible and realistic framework to create dynamic movement. An illustration of its uses can be seen in the paper by Andrew W. Eckford [10]. Brownian motion was utilised as a theoretically viable alternative mode of communication for nano- and bio-technological systems where electromagnetic messaging may be limited.

Taking inspiration again from nature, random walks combined with low-level sensing can produce further intelligent behaviours. Run-and-tumble is an underpinning model of many types of self-propelled particles [11]. In nature it can be observed at the microbial level, particularly among wild-type bacteria [12]. It allows swarms of units to effectively explore their local environment with limited motor capabilities. Movement consists of forward propulsion, known as a "run", followed by a "tumble", indicating a stationary rotation, repeating sequentially. The process is stochastic; tumble events are sampled from a probability density function, interrupting the run to alter the particles immediate trajectory. An example

is seen in the behaviour of Escherichia coli (E. Coli), whose run duration is exponentially distributed with a mean of ∼1 second [12].

Run and tumble offers many advantages to swarm dynamics. The underlying mechanism is simplistic and low energy, making it easy to implement at the microscopic level. However, the emergent behaviours are not so predictable, with capabilities to produce complex, coordinated dynamics. Behaviour is often coupled with local environment sensing, forming further advanced collective dynamics, such as Chemotaxis in bacteria [13].

The mathematics of run-and-tumble motion in 2D can be characterised by the set of finite-difference equations and probability functions seen in equations (5) and (6).

$$\text{Particle}(x; y; \theta; T_n) = \begin{cases} x_{n+1} = x_n + v_x \cos\theta \\ y_{n+1} = y_n + v_y \sin\theta \\ \theta_{n+1} = \theta_n + T_n v_\theta \\ T_n = 0 \text{ or } 1 \end{cases} \qquad v_x, v_y, v_\theta\text{: component speed} \tag{5}$$

$$T_n(i; \lambda) = \begin{cases} Pr(T = 1) = 1 - e^{-\lambda i} \\ Pr(T = 0) = e^{-\lambda i} \end{cases} \qquad i\text{: time steps since previous tumble, } \lambda\text{: tumble rate} \tag{6}$$

1.4 Kilobots

Kilobots are small, low-cost, highly scalable robotic units able to display collective dynamic behaviours [14]. Each Kilobot is constructed with 10 main components, outlined in figure 2, measuring around 13mm in radius. The combination of these gives the Kilobots their functionalities, resulting in the ability to replicate simple, decentralised swarms. Vibration motors are the chosen method of movement, as opposed to the typically used, more expensive two-wheeled approach. A single motor is activated to induce rotation in the Kilobot, with both being used to move forwards at a rate of approximately 1cm/s. Each robot can communicate to its neighbours through the use of light signals, therefore can execute different behaviours based on their local environments. The effective range of communication is approximately 10cm in radius form the centre of the Kilobot. Its ambient light sensor can be utilised to further capture information on its local environment. A three-colour LED serves as a user-friendly indicator, providing visual feedback about the current state or task being performed by the Kilobot.

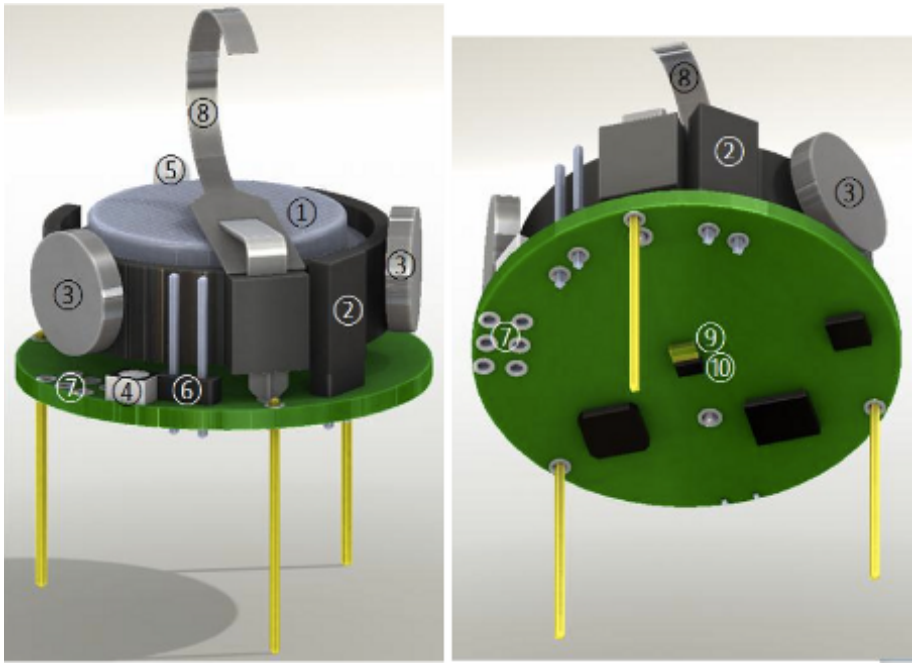


Figure 2: Schematic diagram of Kilobots to be used in swarm configuration: 1. 3.7 Volt Battery. 2. Power Jumper. 3. Vibration Motors. 4. RGB LED. 5. Ambient Light Sensor. 6. Serial Output Header. 7. Direct Programming Socket. 8. Charging Tab. 9. IR Transmitter. 10. IR Receiver [15].

An example of Kilobot collective behaviours is seen in the demonstration of morphogenesis using robot swarms [16]. Morphogenesis is undertaken naturally by cells in order to self-organise into functional shapes. Each cell is identically controlled, reacting only to their local environment, suggesting possible re-creation of these systems using Kilobots. The researchers were able to replicate this behaviour in a swarm of 300 agents without the need for self-localisation, that are additionally robust to damage. Results were obtained through the design of an algorithm where individual agents act based on the state of their simulated gene regulatory network (GRN). Kilobots orbit the swarm in search of a Turing spot, described by the appropriate reaction-diffusion equations. Agents edge follow until a Turing spot is found (using edge detection), additionally enabling the robots to detect if they are lost and recover appropriately.

Alternative approaches to controlling Kilobot swarms involves the use of overhead projectors. Kilobots include ambient light sensors and as such can receive information based on external light sources. This was utilised in an experiment where agents congregate along the edges of projected images to recreate complex shapes and patterns [17]. Results show how Kilobots are able to communicate salient features to one another, allowing them to self-organise into the desired environment setting. Although external systems are required to provide this lighting information, the swarms continue to exhibit a decentralised behaviour whilst improving the robots capabilities without the need for alterations.

Kilobots offer a highly effective platform for investigating swarm dynamics due to their simplicity, affordability, and scalability. These low-level robotic units are designed to mimic the behaviours observed in natural swarms, making them ideal for large-scale experiments that would be impractical or prohibitively expensive with more complex robots. This enables the detailed study of swarm algorithms and interaction rules in a controlled environment, facilitating advancements in understanding how local interactions lead to complex group behaviours in both natural and artificial systems.

1.5 Current Explorations

Javed et al. explore the state-of-the-art and future research challenges of swarm applications [18]. Advances in technological capabilities of Unmanned Aerial Vehicles (UAVs) has gained significant interest in recent years due to their multi-discipline applicability. The research outlines successes in the development of UAV swarms, including integration with artificial intelligence/machine learning, formation control, path planning, task assignment, co-ordination and co-operation and even energy management. An important aspect highlighted in the report is the lack of ethical review within the current literature. A summary of over 20 reports across a range of disciplines displayed no ethical implications considered alongside their findings. Swarms, in particular UAV swarms, pose a number of social and safety concerns to the public. Issues range from the invasion of privacy through photo and video capture, to the illegal deployment of weaponized systems used solely to cause physical harm. It is therefore important for new research to consider, understand and collaborate on the development of appropriate regulations for swarm technology.

Further discoveries in the sector of swarm robotics and active matter present a combination of research to achieve a self-organising robotic aggregate [19]. The development of the so called Granulobot utilises principles of solid- and liquid-like states to produce a highly adaptable locomotion technique able to traverse objects as a collective. The system is additionally robust, granting the ability to split into subdivisions that may later recombine. Levine and Goldman offer their insight into the field of "smart active matter", concerning the "integration of insights from active matter physics with principles of regulatory interactions and control" [20]. The report describes how neither active matter nor biological research alone can gain a full understanding of smart collective behaviour. The merge of these areas is noted to be essential in providing insight into living systems and the engineering of functional collectives.

1.6 Aims, Objectives and Outline

The aim of this project it to investigate the design, simulation and deployment of a swarm whose agents preferentially anti-align (in contrast to classical models of aligning particles such as the Viczek model of collective motion) through the fusion of principles from collective behaviour and active matter. The first objective is to present a simple modification of the Vicsek model to induce "anti-alignment" in swarms. The effect of which will be assessed through the use of simulations, comparing results to that of alignment. Finally, tests are to be conducted on the ability to translate the results into a real-world swarm, assessing the challenges and potential use cases of such systems.

To achieve this, the report begins by combining the mathematical models describing alignment, and hence its theorised counterpart anti-alignment, with "run-and-tumble" particle motion. Simulations are developed to explore how a simple agent performs the desired tasks along with the influential parameters of the collective system. The developed algorithms are translated to a real-world robotic swarm constructed from Kilobots. The project explores the effectiveness of theoretical models in predicting the outcome of practical systems along with the capabilities of the individual agents.

2 Anti-Alignment Swarm Formulation

2.1 Agents, Movement and Interactions

The design of the anti-alignment swarm contains 3 components; the agents, the underlying movement and the interaction behaviours. Agents are modelled on Kilobots, outlined in section 1.4. Kilobots are an example of a simple agent able to execute a limited number of commands. However, when introduced into a swarm can exhibit complex, joint behaviours. It's minimal features nonetheless grant the opportunity to embed a range of simple movement dynamics as well as localised communication. An agent is modelled as an individual particle with a specified diameter, d, speed, v, position in 2D space, (x, y), heading with respect to the plane, θ, tumble rate parameter, λ and local neighbourhood radius r, as displayed in figure 3.

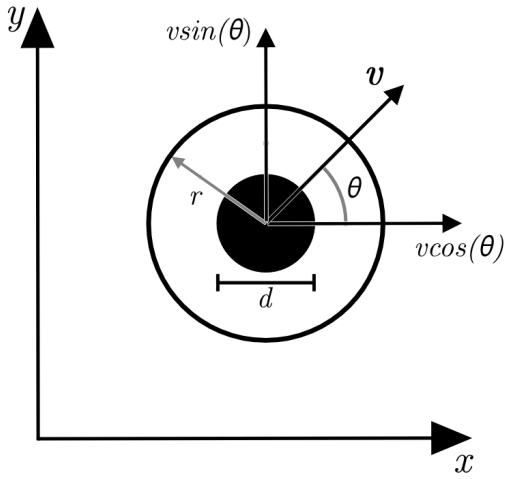


Figure 3: Diagram outlining simulated agent attributes for use in simulations. Agents are assigned a diameter, d, speed, v, position, (x, y), heading, θ, and local neighbourhood radius r.

The chosen movement dynamics for the swarm, known as run-and-tumble (section 1.2), provides a straightforward yet effective exploration strategy suitable for the capabilities of the agents. At each time-step, Δt, the agents update their position and heading according to equations (5) and (6). This basic locomotion framework not only facilitates initial exploration but also serves as a foundational model for more sophisticated behaviours. For instance, the adaptation of tumble rates in response to environmental stimuli can be integrated to enhance the responsiveness of the swarm to dynamic surroundings, thereby optimizing the collective behaviour of the swarm under varying conditions.

The interaction behaviours are formulated on the Vicsek model. As opposed to alignment, agents will locally anti-align by taking the negative of the average neighbourhood heading. The method will take advantage of the atan2(y, x) function, where the sign of both components are taken into consideration. As a result, two negative signs are included in the average neighbour direction calculation to achieve the desired behaviour, giving equation (7). Combining equation (1), (2) and (7) formulates the "Anti-Vicsek" model. This is used to update agents headings when anti-aligning on top of their run-and-tumble dynamics. The fusion of the outlined features gives the rules governing the dynamics of the anti-alignment swarm.

$$\langle\theta(t)\rangle_r = \text{atan2}(-\langle\sin(\theta(t))\rangle_r, -\langle\cos(\theta(t))\rangle_r) \tag{7}$$

2.2 Trivial Case and Agent Finite Difference Equations

Examining the anti-alignment formulation more closely reveals some trivial emergent behaviours of the system. If the neighbourhood radius is allowed to be unbounded, hence all agents retain communication with one another at all times, the system oscillates between two opposing headings. Regardless of the number of agents or distribution of initial headings, after 2 iterations of anti-alignment the agents will converge to this behaviour. Figure 4 demonstrates this case in the form of 8 static agents initialised with random headings. Iteration 1 shows the updated heading after a single step; each agent has aligned to the negative of the average direction. The next iteration, all agents rotate π radians followed by a repeat of this process in iteration 3. This behaviour will continue indefinitely. To avoid this, neighbourhood detection radii should be kept within a reasonable factor of the agents size. For this project, this radius will be kept within the bounds of the Kilobot physical communication limits, hence should not exceed approximately 10 times the radius of the agent. Furthermore, the introduction of noise will further discourage this behaviour by altering the calculated mean after each iteration. However, in place of the Vicsek noise parameter within the neighbourhood adjustment, this effect is achieved through the underlying motion, i.e. through random tumble events. The anti-alignment agent model, $\text{Agent}(x; y; v; \theta; r; \lambda)$, is therefore formulated by

$$\begin{aligned}
x(t+1) &= x(t) + v\cos(\theta(t))\Delta t \\
y(t+1) &= y(t) + v\sin(\theta(t))\Delta t \\
\theta(t+1) &= \langle\theta(t)\rangle_r + T_i\Delta\theta \\
\langle\theta(t)\rangle_r &= \text{atan2}(-\langle\sin(\theta(t))\rangle_r, -\langle\cos(\theta(t))\rangle_r) \\
\langle\sin(\theta(t))\rangle_r &= \frac{1}{n}\sum_{i=1}^{n}\sin(\theta_i(t)) \\
\langle\cos(\theta(t))\rangle_r &= \frac{1}{n}\sum_{i=1}^{n}\cos(\theta_i(t)) \\
Pr(T_i = X) &= \begin{cases} 1 - e^{-\lambda j} & \text{for } X = 1 \\ e^{-\lambda j} & \text{for } X = 0 \\ 0 & \text{otherwise} \end{cases} \\
\Delta\theta &= \text{uniform}[0, 2\pi]
\end{aligned}$$

where n is the number of neighbours within radius r and j is the number of time steps since the previous tumble.

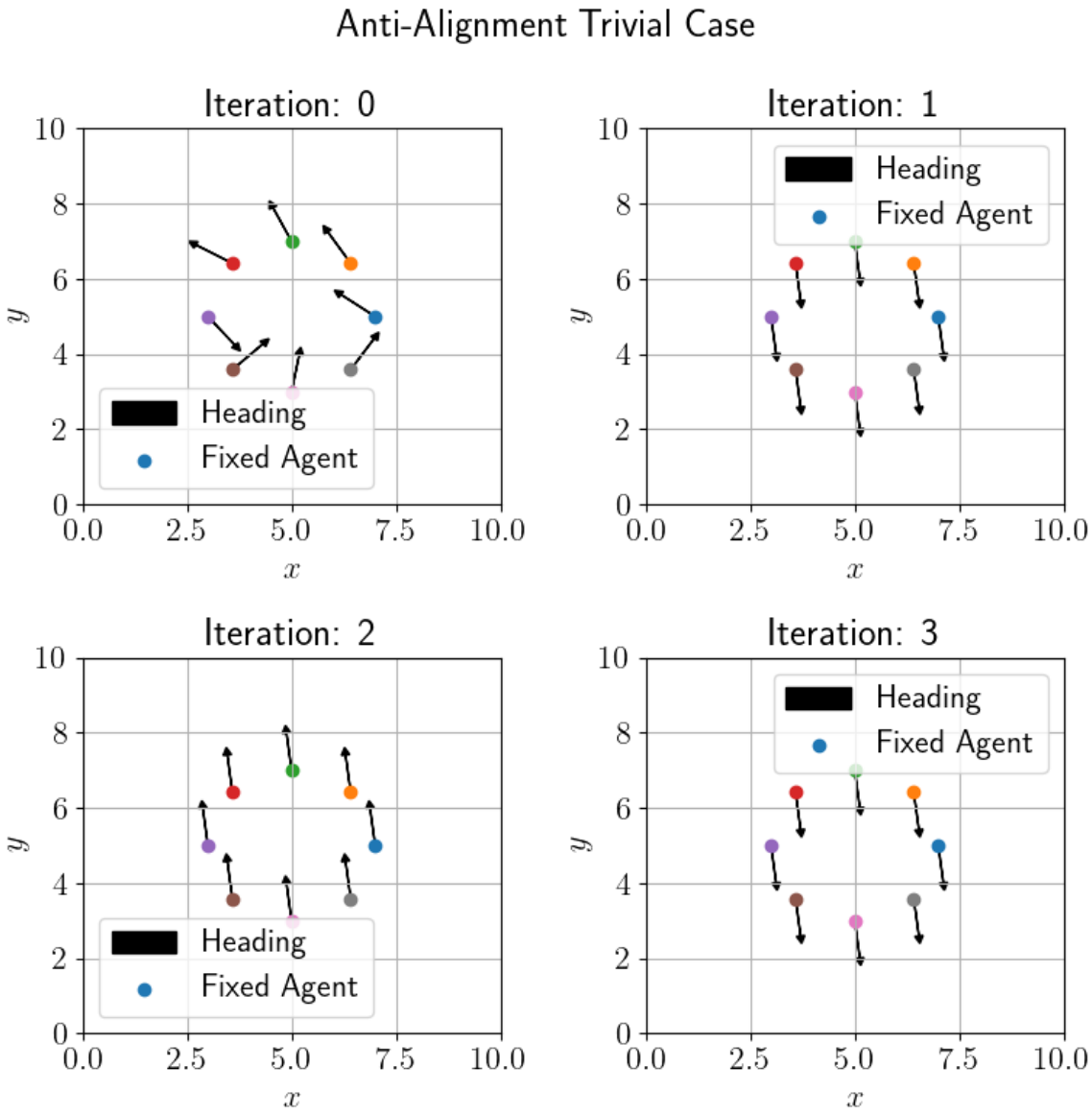


Figure 4: Figure exploring the trivial case emergent behaviour of anti-alignment. This occurs when agents are given an unbounded detection radius and no perturbation in headings. Agents will oscillate between two opposite states, as outlined in iteration 1-3.

3 Numerical Simulations

Numerical simulations offer a time-efficient method to predict and analyse the behaviours of collective dynamics. By first modelling the system mathematically, effective parameters on the emergent behaviours can be extracted quickly and efficiently before delving into experimentation. Furthermore, issues pertaining to the imposed behaviours can be more easily deciphered due to the reduced complexity. For this reason, the chosen start point for this project begins with the development of numerical simulations.

3.1 Run and Tumble

Investigations began with the simulation of a single Kilobot in a specified domain of size $W \times H$ pixels (px). To better visualise the true size of the Kilobots, a single pixel is set to equate 10mm. A scaling factor is introduced to represent the true ratio between the domain size and the Kilobots. Setting the scale factor to 1 would represent the true size and speed of the agents, as outlined in section 1.4, as compared to the domain (a scale factor of 2 would double these values, equivalent to halving the domain, see figure 25 in appendix A). This is used to constrain the simulation window to the same size while allowing simulations to be seen at different scales. Simulations are run at 30 frames per second (FPS), where agent updates occur at each frame. Kilobots update their onboard clocks approximately 31 times a second, hence 30 FPS is deemed an appropriate representation for the rate of execution for the agents. However, Kilobots are limited in their time to execute physical movement, i.e. movement is not instantaneous. As a result, a small delay, t_{tumble}, is added to the agents when exhibiting tumbling motion. Finally, agents display different colours depending on their state/action (see table 2). This will be used as a visual aid when comparing to experimental results, where Kilobots may also present this information through the use of their onboard LED.

Figure 5 shows an example path of a single particle exhibiting the implemented run and tumble behaviour.

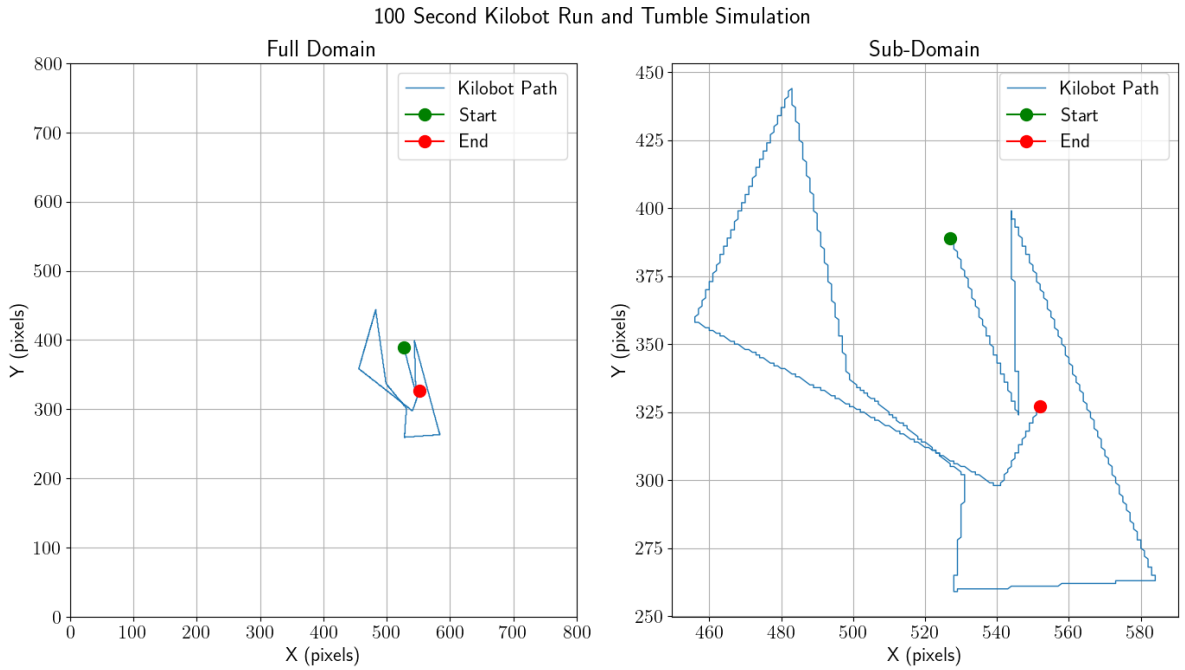


Figure 5: Initial simulation testing. An example path of a single agent run-and-tumble from a random starting location. Simulation is scaled by a factor of 8, hence the figure equates to a full domain size of approximately 1m^2. Simulation parameters: $v = \frac{1}{FPS} \times \text{Scale} = \frac{8}{30}\text{px}/s, \lambda = \frac{1}{50000}, t_{tumble} = 500\text{ms}$.

3.2 Neighbourhood Detection

The simulation is now expanded to multiple particles, each with an additional neighbourhood detection radius of length r. This is demonstrated in figure 6, where particle detection radii are displayed as an outer ring centred on the given agent. Once a neighbour is detected, the outline changes colour from black to blue until no neighbours are detected. r is set to the equivalent of 100mm in correspondence to

the Kilobot attributes outlined in section 1.4.

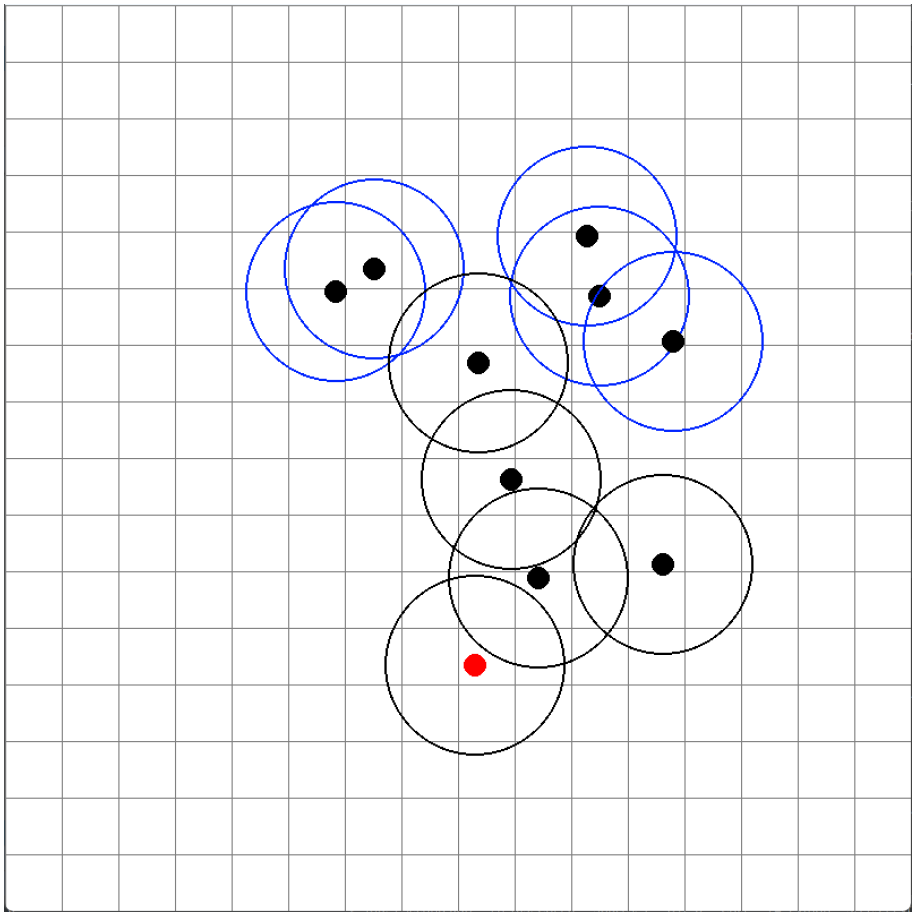


Figure 6: Expanded simulation including multiple agents and detection radii. Outer rings represent the detection radius centered around its respective agent. Rings turn blue when a neighbour is detected, otherwise are black. Agents also indicate their current state depending on their colour; black while running, red while tumbling. Domain size $= 800 \times 800$ px, scaled by a factor of 8, equating to approximately 1m^2. Simulation parameters: Number of agents $= 10, d = 2.6 \times \text{Scale} = 20.8\text{px}, v = \frac{1}{FPS} \times \text{Scale} = \frac{8}{30}\text{px}/s, r = 10 \times \text{Scale} = 80\text{px}, \lambda = \frac{1}{50000}, t_{tumble} = 500\text{ms}$.

3.3 Anti-Alignment

Using the implemented neighbourhood detection system, agents now calculate their desired alignment according to the classical Vicsek or anti-Vicsek models as specified. This is achieved through the removal/addition of both negative signs in the average neighbour heading calculation. Additionally, the centre of mass (CoM) of the system is plotted as part of the simulation. CoM is calculated according to $CoM = \frac{1}{N}\sum_{i=1}^{N}(x, y)$, where N is the number of agents. Both features are presented in figure 7, where green agents indicate (anti-)alignment adjustments. Heading adjustments require a short amount of time to complete, estimated similarly to tumbling, t_{tumble} (section 3.1). Furthermore, there is a short delay between adjustment periods, t_{adjust}. If no delay is set, agents become stuck in place constantly adjusting due to the separation between running and heading updating.

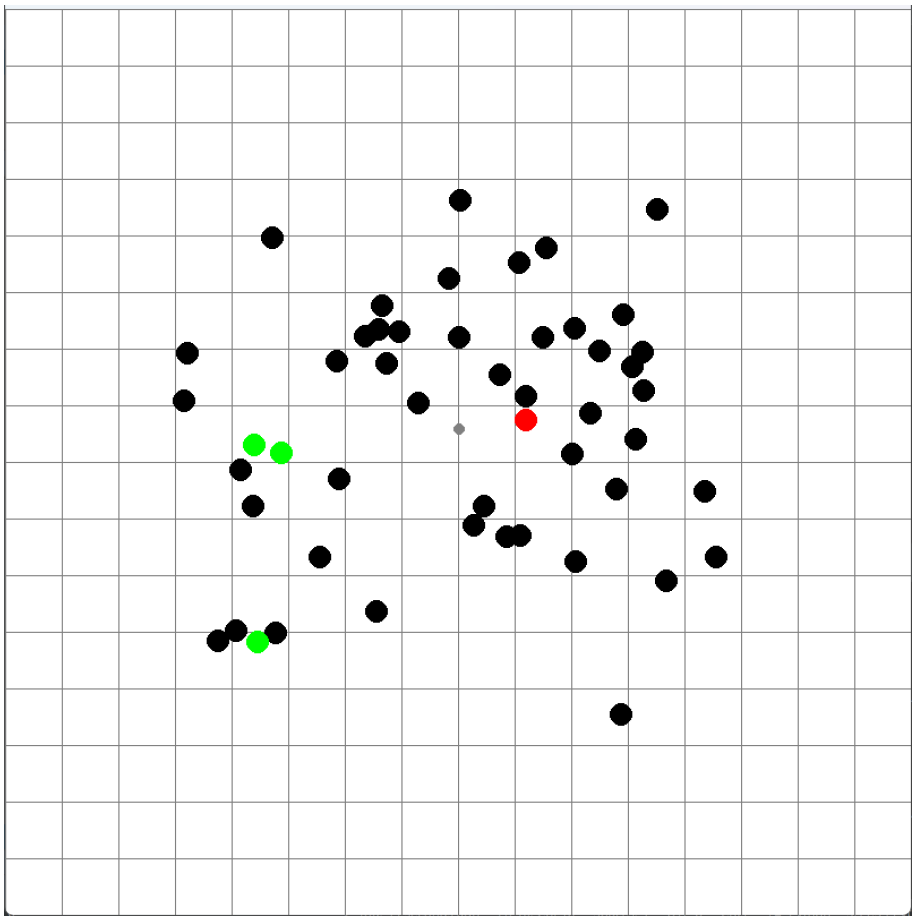


Figure 7: Final stage of simulation building. Colours represent states of respective agents; black while running, red while tumbling and green while (anti-)aligning. Smaller grey dot represents the centre of mass of the system. Domain size $= 800 \times 800$ px, scaled by a factor of 8, equating to approximately 1m^2. Simulation parameters: number of agents $= 50, d = 2.6 \times \text{Scale} = 20.8\text{px}, v = \frac{1}{FPS} \times \text{Scale} = \frac{8}{30}\text{px}/s, r = 10 \times \text{Scale} = 80\text{px}, \lambda = \frac{1}{50000}, t_{tumble} = 500\text{ms}, t_{adjust} = 1000\text{ms}$.

This simulation will form the basis of the theoretical behaviour aimed to be re-created in the Kilobot swarm. Results from experimentation will provide feedback that highlight potential improvements to be made to the model. One aspect not considered are collisions between agents. Further investigation is required to determine how Kilobot collisions should be modelled before implementing. However, due to the low speed and mass of the bots, it is assumed that the effect of these collisions can be ignored as an effect on the collective dynamics.

4 Experimental Implementation with Kilobots

The aim of this section is to establish the feasibility of implementing the desired behaviours on a real robot swarm. Translation from theory to experiments seldom maps flawlessly; simulations inadvertently fail to capture all necessary behaviours and parameters of the true system. In addition, the work aims to aid investigations to better understand the capabilities of Kilobots. Although simplistic in nature, their potential uses and extent of limitations are yet to be fully explored.

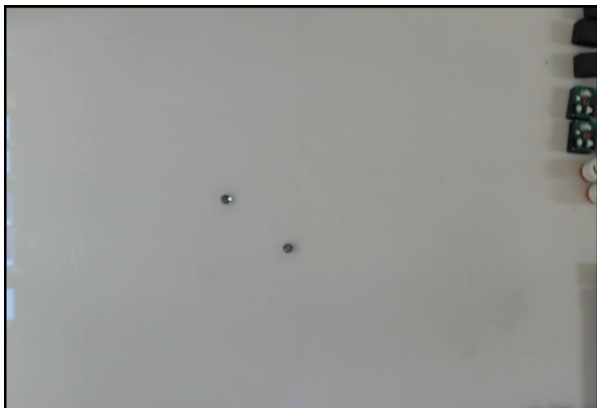
4.1 Limitations and Calibration

Before experimentation began, it was noted that Kilobots do not possess the ability to sense their orientation using onboard hardware (i.e. no accelerometers, gyroscopes or magnetometers). This presents difficulty in being incapable to determine their heading, hence cannot communicate this information to one another and therefore are unable to perform anti-alignment. To overcome this, 3 alternative methods are explored to calculate their heading using other sensors and equipment. These include the use of overhead projectors to encode information through ambient light.

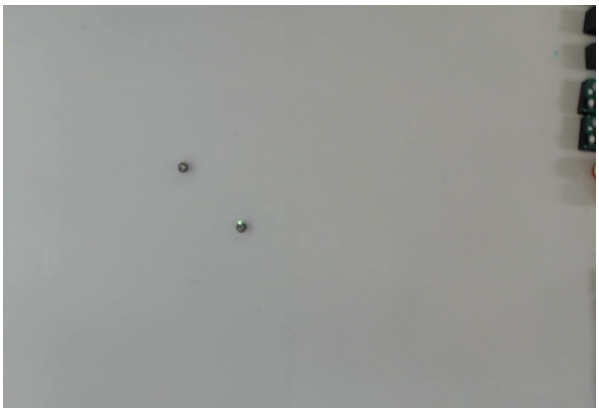
The first task was to calibrate the Kilobots. Each bot can be individually tuned to ensure its motors perform the required actions effectively. This is achieved using the Kilogui software environment, where a Kilobot is assigned a unique ID and motor powers for turning left, right and forwards [15]. Although a slow process, this is a crucial step in minimising variability across the swarm as well as determining faulty robots. It is also important to fully charge the Kilobots before testing to ensure they perform consistently with their calibration phase.

4.2 Run and Tumble

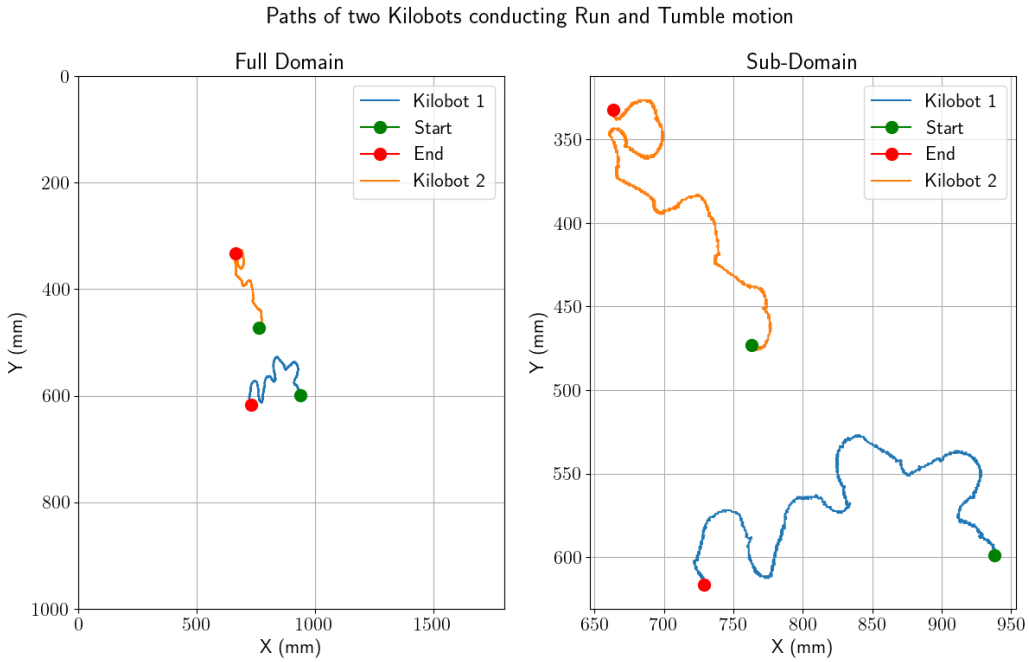
Once calibrated, tests were conducted to determine the Kilobots ability to re-create the underlying movement strategy. Figure 8 shows the captured paths from a video tracker tracking the position of two Kilobots performing run and tumble. As expected, the movement of the Kilobots contains noise and imperfections when compared to the simulations. When performing runs, Kilobots tend to drift away from their path over longer distances. However, the implementation shows clear distinctions between running and tumbling, as well differing paths generated by each Kilobot, determined by a random seed based on the unique ID. It was therefore determined that the Kilobots, with careful calibration, can perform the required motions effectively.



(a) Location of Kilobots at beginning of testing.



(b) Location of Kilobots after 30 seconds of testing.



(c) Video tracker paths plotted for both Kilobots performing run-and-tumble motion. Testing ran for approximately 60 seconds total. It is seen that Kilobots tend to drift while running, however plots show a clear distinction between performance of a run (smooth change) and a tumble (sharp change).

Figure 8: (a) Recording from overhead camera used for video tracking. Green light indicates the Kilobot is in a tumble state, while no light indicates a running state. (b) Video tracking demonstrating the Kilobots ability to perform run and tumble motions based on video captured from the overhead camera in the Kilobot arena.

4.3 Neighbourhood Detection Algorithm

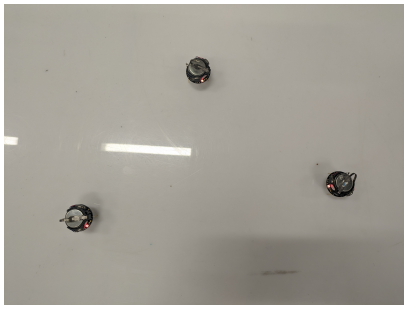
The next stage involved utilising the Kilobots infrared light transmitters/receivers for neighbourhood detection. The bots can emit a 12 byte signal roughly twice per second, with the ability to encode 9 bytes of information within this. This provides sufficient memory to store the Kilobot ID and heading values within the message. As a result, the algorithm involves checking that a message has been received every clock cycle and storing this information for each unique Kilobot message received. Kilobot ID's are used to ensure that repeat messages from one Kilobot are not processed multiple times, as well as counting the number of neighbours within the detection radius. To detect when a Kilobot has left

the radius, stored neighbour information is reset periodically, approximately once every second. Once reset, only the remaining bots will be re-added to the array. The detection radius is limited to 100mm, within which Kilobots gauge the distance of their neighbours by analysing the intensity of the signal they receive. Due to variations in the effective range over which Kilobots can successfully send signals, these radii are established to minimise such disparities and create a more precise boundary for detecting when neighbours are within or outside of this range. Finally, arrays are required to be finite in length upon compiling, therefore the number of neighbours that a Kilobot can store information about must be pre-determined. This number must also be sufficiently small that it does not exceed the onboard memory allocation, and sufficiently large so that no neighbours are ignored. The capacity was therefore set to 20 following the practical limits of how many bots can fit within a 10cm radius of another.

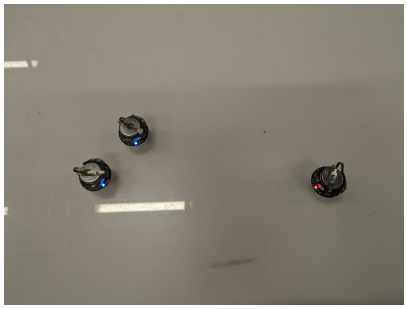
Algorithm 1 outlines a pseudocode representation of the implemented neighbourhood detection used on the Kilobots. Figure 9 shows the algorithm running on 3 Kilobots, where each Kilobot represents its state via their LED colour. It is seen that this implementation works as intended, successfully detecting when another Kilobot is present or not. Further tests were conducted to ensure Kilobots correctly count the number of neighbours, which are provided in appendix A, figure 26.

Algorithm 1 Kilobot Neighbourhood Detection Algorithm

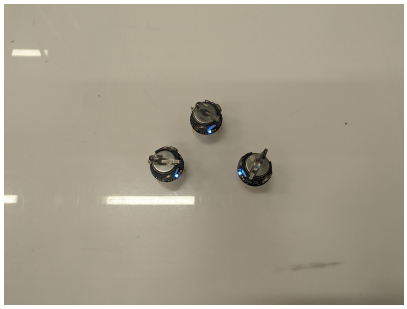
Require: $DETECT\ RADIUS = 100,\ NEIGHBOURS[MAX\ NUM] = \{0\}$
  while *running* **do**
    if *new message received* **then**
      current distance = *esitmate distance*()
      if *current distance* $<$ $DETECT\ RADIUS$ **then**
        add neigbour to $NEIGHBOURS$ *array*
      end if
    else
      reset $NEIGHBOURS$ *array*
    end if
    if *timer reached* 1 *second* **then**
      reset $NEIGHBOURS$ *array*
      reset timer
    end if
  end while



(a) No neighbours detected, indicated by the red lights.



(b) Two Kilobots within detection radius, indicated by the blue lights. One Kilobot outside of detection radius, indicated by red light.



(c) All Kilobots within detection radius of one another, indicated by the blue lights.

Figure 9: Kilobot neighbour detection algorithm demonstration. Light colours determine whether a neighbour is detected (blue) or not (red).

4.4 Heading Estimation

One feature of the Kilobots that has not yet been utilised in this projects is the use of the ambient light sensor. This sensor produces a value between 0 and 1024 corresponding to the light levels of an agents local environment. Through this, information can be sent to the robots via an overhead projector, projecting varying intensities and colours of light across the arena. This section investigates the potential of such methods to encode direction and headings without altering Kilobot hardware. An alternative method is to estimate the change in heading based on the time taken to perform a tumble motion. Such an approach has the advantage of requiring no external hardware or additional algorithms. However, in practice not all Kilobots turn at the same angular rate, even with careful calibration. Furthermore, deviations during the running phase cannot be captured. Both approaches are tested to determine the most appropriate implementation.

4.4.1 Pattern A (Triangular)

It's important that any projected images can uniquely determine left from right and up from down with respect to the plane. Patterns that are too simple, such as a sequence of alternating coloured squares, would produce the same estimated headings for multiple directions. Therefore, at least 3 colours are required per axes. Shapes containing the subsequent colours should also tile the plane, giving three options; equilateral triangles, squares and hexagons. The decision was made to use equilateral triangles as each only requires three unique adjacent colours. A tiling of triangles was therefore coloured with a 6 length sequence repeating per row. Each row is then shifted by 3 places with respect to the previous row, the result of which is demonstrated in figure 10. However, at each corner there are 6 colours located within close proximity. A white circle at this location is therefore added to ensure no more than 3 colours touch at any one point. The aim of this is to increase clarity when determining colour changes, which is vital to determining the Kilobot headings. Projected image sizes should ensure that Kilobots are fully encapsulated by any one shape, although small enough to reduce heading estimation time.

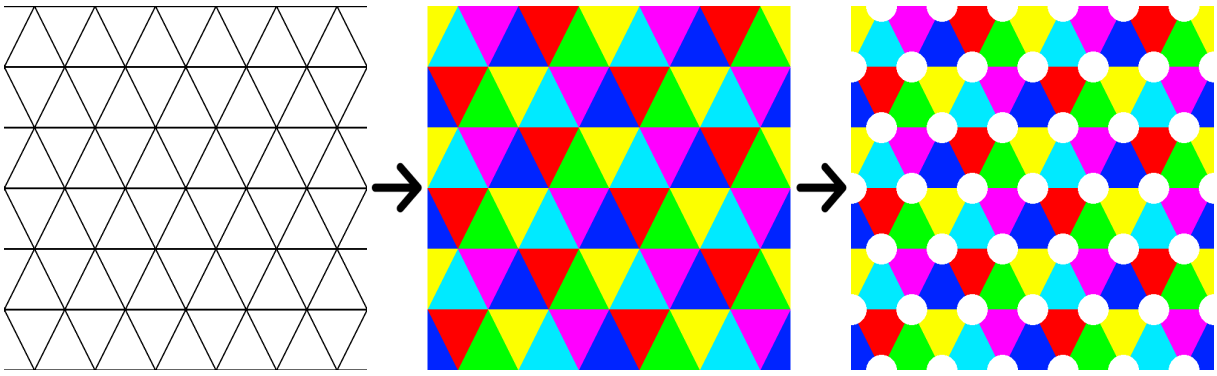


Figure 10: Pattern A: Triangular. Pattern is generated through an equilateral triangle tiling. The pattern is then coloured according to a 6 length colour sequence: [Yellow, Cyan, Magenta, Blue, Red, Green]. A layer of white circles at each triangle point is then added to eliminate a 6 colour border at these locations.

Heading estimation is calculated through the encoding of each colour and the equivalent sequence detected. Once a Kilobot has detected three colours while travelling in any direction, a heading is assigned based on the sequence combination. For example, if the agent begins on yellow, travels to blue and then to white, the Kilobot heading estimation is $\pi/2$ radians (or $90°$) counterclockwise from the x-axis. Figure 11 illustrates some example paths a Kilobot can take to estimate its heading. Each direction is approxi-

mated as $\frac{2\pi}{12}$ radians from one another, ranging from 0 to 2π. A total of $6 \times 12 = 72$ total sequences are required to be encoded (sequences are only evaluated beginning on a triangle).

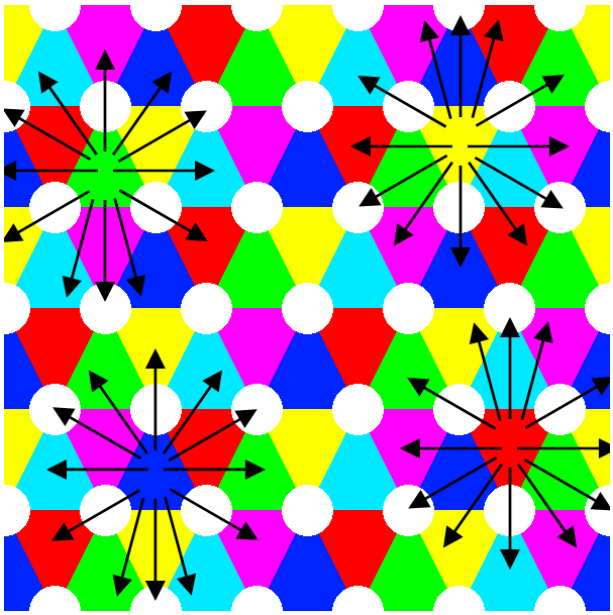


Figure 11: Example encoded paths that estimate a Kilobots heading based on the detected 3 colour sequence. There are 12 possible directions from any one starting location. Headings are discritized into bands of size $\pi/6$. Encoded sequences begin only on triangle colours.

4.4.2 Pattern B (Intensity)

A non-uniform approximation can be achieved through the utilisation of colour gradients. Figure 12 demonstrates an example pattern varying in colour from left to right, and intensity vertically. A velocity vector can be generated for a Kilobot by measuring the change in time between readings. If a change in colour is detected (i.e. form red to green) then the horizontal speed can be approximated as $\frac{l}{t_{current} - t_{detected}}$. l equates to the length of a single grid square, $t_{current}$ is the current time and $t_{detected}$ is the time at which the previous colour was detected. The negative is taken if the colour change follows the negative x direction, e.g. from green to red. Similarly, the vertical speed is approximated following the same principle, however when a change in intensity of the same colour is detected. The angle of the resulting vector is calculated using $\arctan(y/x)$, giving the heading of the Kilobot between $[-\pi, \pi]$. For consistency with numerical simulations, the value is transformed to represent the angle counterclockwise from the positive x-axis, falling in the range of 0 to 2π.

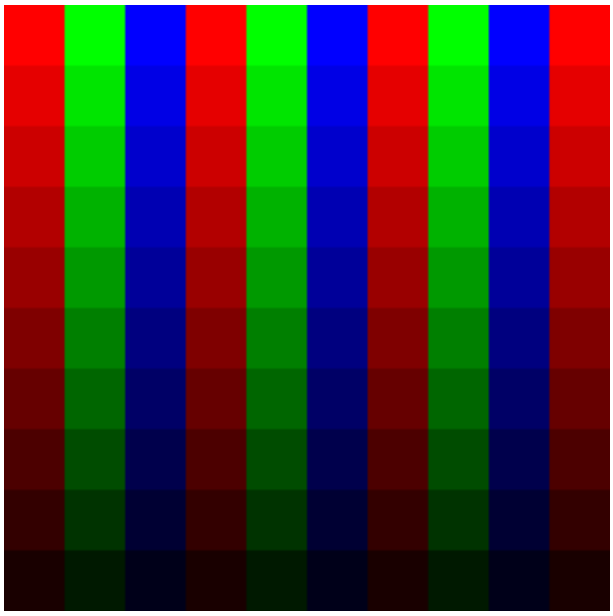


Figure 12: Pattern B: Intensity. Kilobots detect the rate of change of colour to determine an approximate velocity along the x-axis. Rate of change of intensity of the same colour is used to approximate velocity along the y-axis. Velocities are calculated depending on the time between reading changes and the length of individual squares.

4.4.3 Rotational Estimation

Environmental factors may restrict the ability to use light encoding as a method of heading estimation. Furthermore, signals may be noisy, requiring additional filtering/computation to accurately decode colours and intensities. An alternative approach is to employ the methods Kilobots already use for turning to calculate their rotational movement. Each Kilobot motor is activated for a specified time depending on the task at hand. As a result, the time taken to alter its heading, t_θ, is readily available. By measuring the average time taken for the bots to complete one full rotation, $t_{2\pi}$, an estimate for the change in orientation, $\Delta\theta$, can be deduced. This is calculated using the following formula: $\Delta\theta = 2\pi t_\theta / t_{2\pi}$. As mentioned in the beginning of section 4.4, this method is prone to errors. Furthermore, errors carry through to each subsequent heading estimation due to the inability to validate the estimation. Kilobots must also all be placed in the same direction initially, where this will indicate the zero heading. Random initialisation of Kilobot starting positions is still possible with the addition of a preliminary tumbling phase. Kilobots will begin by turning a random amount before initiating the main implemented behaviours. Rotational estimation will be carried out throughout execution providing the heading values to be communicated to neighbouring Kilobots.

4.5 Anti-Alignment

By combining the neighbourhood detection algorithm with heading estimation, anti-alignment behaviour can be achieved. The final step is to calculate the time required to turn the desired alignment adjustment. Through re-arranging the formula outlined in section 4.4.3, t_θ is approximated by the formula $t_\theta = \Delta\theta t_{2\pi}/2\pi$, where $\Delta\theta$ is calculated using the anti-Vicsek model (7) and stored neighbour headings.

Figure 13 illustrates a high level overview of the complete Kilobot operations to be applied to the final

swarm.

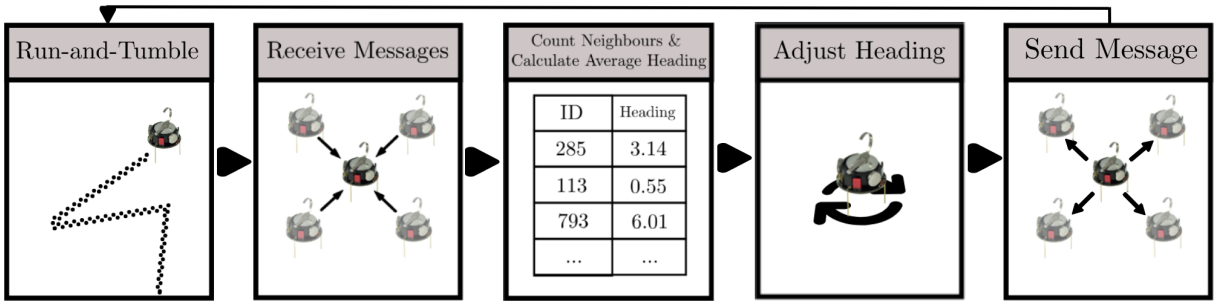


Figure 13: Overview of Kilobot operations. Kilobots begin by moving randomly by running and tumbling. Once a message (or messages) has been received, the ID and heading from the neighbouring Kilobot is stored. This data is then used to count the number of neighbours as well as the anti-alignment adjustment heading. Kilobots adjust their heading accordingly, followed by updating their output signal to their neighbours with their new heading. The process repeats sequentially indefinitely.

5 Results

5.1 Simulations

Simulations were conducted with 100 agents randomly initialised withing a sub-region of the domain, as outline in figure 14. The co-ordinates of the starting box were set as $(W/4, H/4)$ and $(3W/4, 3H/4)$. This was chosen as a sufficiently small area such that the agents are able to begin communication and minimise boundary crossing. The area must also be large enough to minimise overlapping and promote variation in neighbourhood counts. Simulations are run for 100 seconds, and states recorded every 100 milliseconds. All simulation parameters are noted alongside their respective figures.

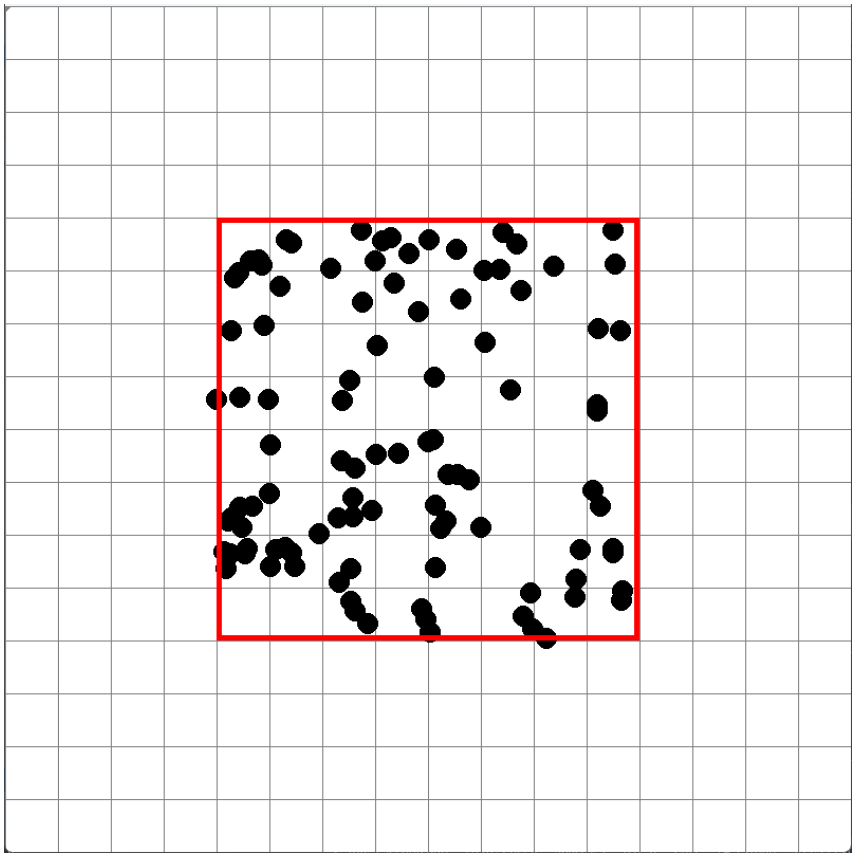


Figure 14: Simulation initialisation for data collection. 100 agents are spawned within the starting bounding box, highlighted in red. Boundary domain is periodic, therefore agents leaving on one side appear on the opposite. Domain size = 800×800 px, scaled by a factor of 8, equating to approximately 1m^2. Simulation parameters: Run Time = $100,000$ms, number of agents = $100, d = 2.6 \times$ Scale = $20.8\text{px}, v = \frac{1}{FPS} \times$ Scale = $\frac{8}{30}\text{px}/s, r = 10 \times$ Scale = $80\text{px}, \lambda = \frac{1}{50000}, t_{tumble} = 500\text{ms}, t_{adjust} = 1000\text{ms}$. Simulation is first run with alignment, then re-initialised for anti-alignment.

5.1.1 Alignment vs Anti-Alignment

Two separate simulations were run for 100 seconds each to compare the emergent behaviours of alignment against anti-alignment. Both models were run with equal domains sizes (with periodic boundary conditions), number of Kilobots, and identical agent attributes. Collected data includes the current time step, Kilobot ID, X, Y and θ co-ordinates of associated agents, the number of detected neighbours per agent and the X and Y position of the centre of mass (CoM).

Figure 15 illustrates the Kilobot counts present within a discretized space of the domain. Each grid space represents a 20×20 px section of the simulation area. Associated counts represent the number of times Kilobots are detected within the grid space at a given time step, summed for the whole run time. The resulting heatmaps show distinct differences in the collective dynamics of the two scenarios.

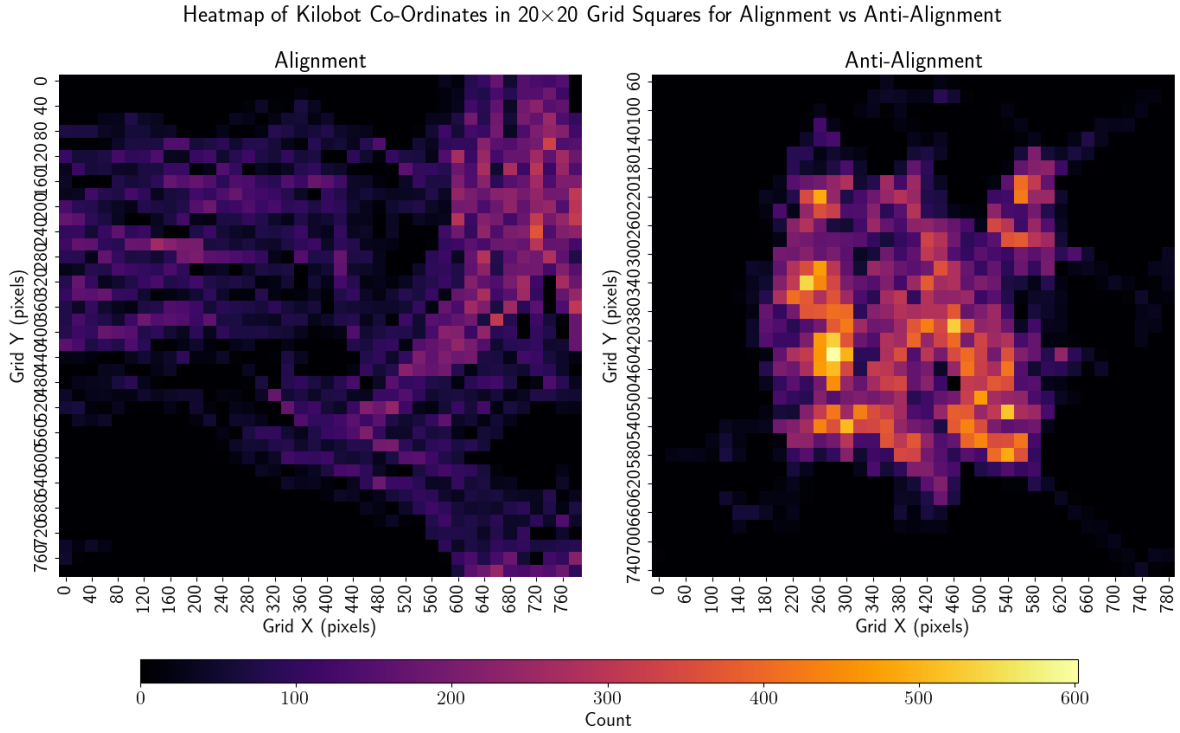


Figure 15: Agent locations for alignment and anti-alignment swarms are recorded every 100ms. Agents within a 20×20 px grid space are added to the count within that square and plotted as heatmaps. Alignment tends to move further across the domain, with lower counts per grid space, whereas anti-alignment remains centrally located around the starting location, spreading out slowly and evenly.

Alignment swarms tend to create numerous smaller clusters travelling in their respective aggregated direction. These clusters eventually collide, forming larger clusters following a new heading, repeating until a majority cluster is formed. The forming of such is illustrated in the top right hand corner of the alignment heatmap, where paths of lower counts join to produce an area of higher density. High grid counts are a result of numerous agents close together as opposed to time spent within that area. Agents tend to spend less time in a single grid, adjusting quickly to their new desired bearing. Contrarily, anti-alignment swarms disperse slowly and evenly, making more drastic alignment adjustments. This, whilst within their neighbours' detection radius, keeps them from travelling large distances in any single direction. Maximum counts are approximately double that seen in alignment, suggesting individual agents spend more time in a smaller area.

These observations are continued in figure 16. The plot illustrates the distribution of headings seen across the swarms. Alignment creates a majority alignment that all agents tend to follow, as expected by the Vicsek model. Meanwhile, anti-alignment maintains its inherent randomness, with an average heading of 3.14, being the expected value of the uniform distribution in which it is drawn upon.

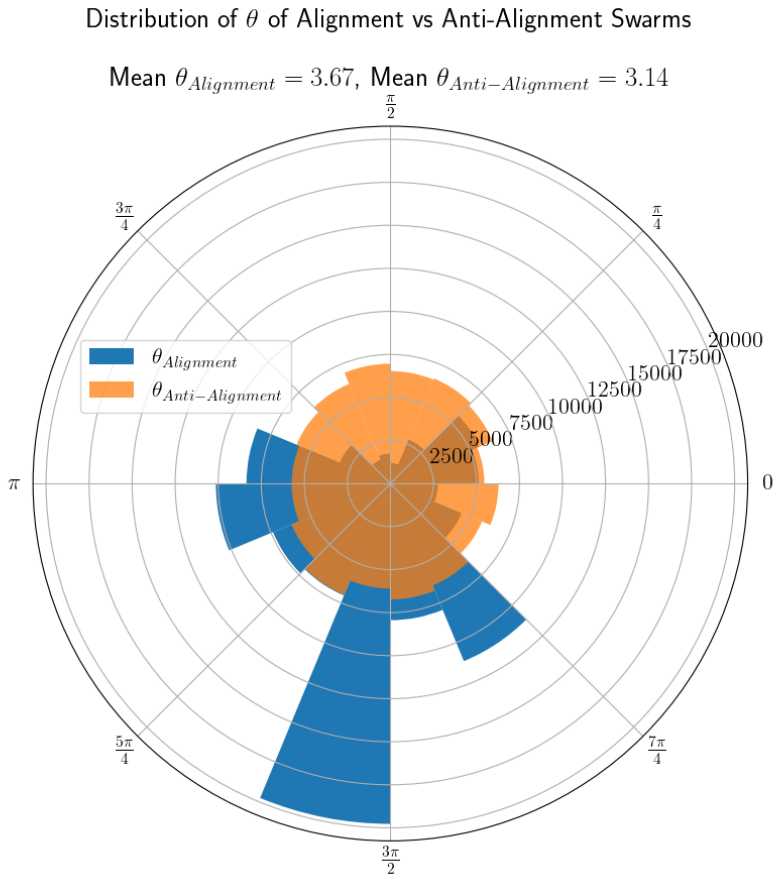


Figure 16: Histogram showing heading distribution of agents in alignment and anti-alignment swarms. Height of bar indicates total count of headings within $\pi/8$ bands across all timesteps (every 100ms) and agents of the simulation. Alignment swarms tend to converge onto a single common heading, whereas anti-alignment retains a uniform distribution between 0 and 2π.

An alternative approach to investigate this is to use the Vicsek order parameter as outline in equation (4). Figure 17 displays the time series of the Vicsek order for alignment and anti-alignment swarms over the course of the simulation. The results are congruent with that seen in figures 15 and 16, indicating a tendency towards a common order for alignment, and the opposite for anti-alignment.

However, figure 18 suggests that although alignment agents tend to order themselves at the microscopic level, the macroscopic movement of the swarm is still random. The result shows how the co-ordinates of the centre of mass of the respective systems displace over time. Alignment shows how the underlying run-and-tumble random walk re-emerges in the swarm dynamics. Anti-alignment on the other hand centralises around its starting location for the duration of the simulation.

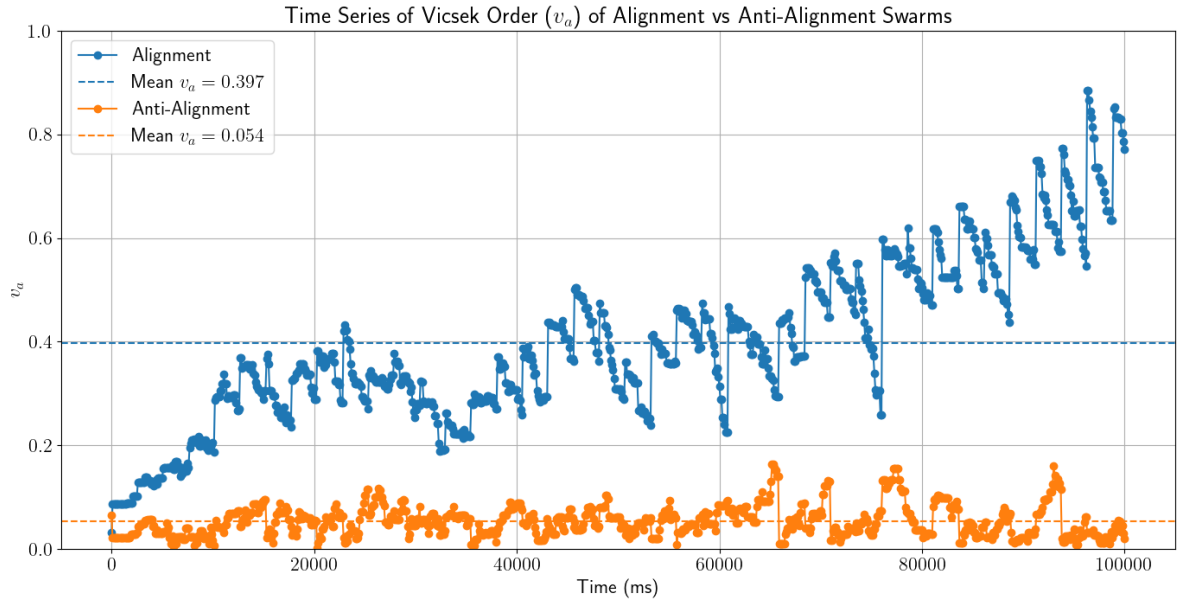


Figure 17: Time series of Vicsek order parameter for alignment and anti-alignment swarms. Order of the system is evaluated every 100ms using equation (4). Alignment order tends to increases over time, from a highly disordered state to that of coherent motion. Anti-alignment remains un-ordered throughout the length of the simulation.

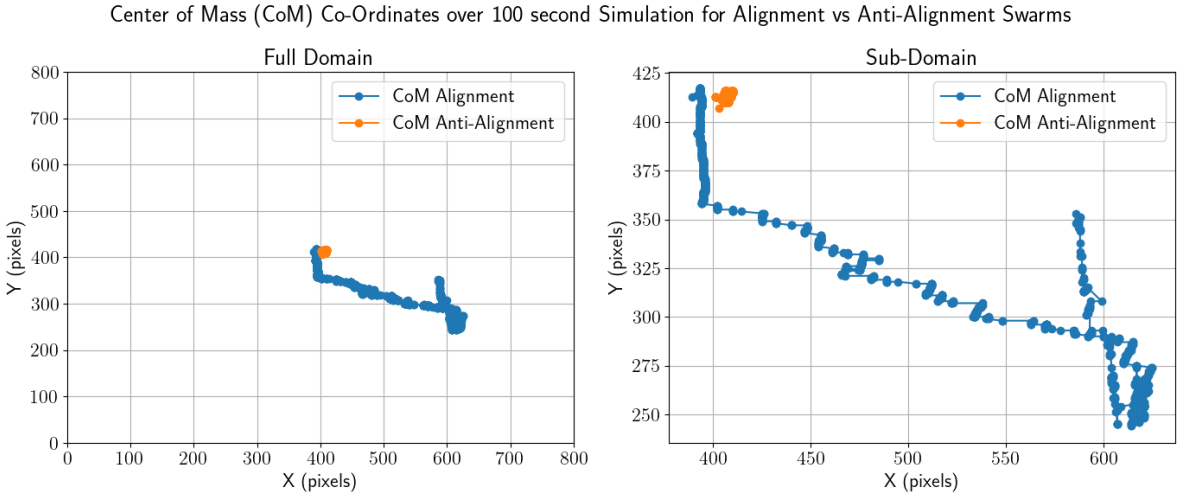


Figure 18: Plot of centre of mass (CoM) location every 100ms for alignment and anti-alignment swarms. Anti-alignment CoM remains steadily around its starting location. Alignment CoM produces a collective run-and-tumble random walk dynamic.

5.1.2 Adjustment Rate

The combination of run-and-tumble with anti-alignment introduces a new system parameter compared to classical alignment swarms. The disconnect between heading adjustment and forward propulsion does not allow continuous updating of alignment. As a result, agents anti-align at a specified adjustment rate interval. In theory, the adjustment rate could be set to zero as a form of continuous adjustment.

However, there are two lower bounding limits of the system. Firstly, the lower the adjustment rate, the lower the time spent running, hence, the distance travelled by any given run becomes negligible with respect to the domain. Secondly, the adjustment rate cannot be shorter than the time taken to execute an adjustment. Although in simulations instantaneous heading updating can be achieved, this is not feasible in practice. Consequently, if the adjustment rate is lower than the average time taken to turn, agents will become stuck in place. This is due to the agents detecting a new heading alignment before the previous adjustment has finished. The effects of this parameter are outline in the heatmaps displayed in figure 19. As the adjustment rate is increased, the rate of dispersion of the swarm increases, with lower maximum counts seen in any single grid.

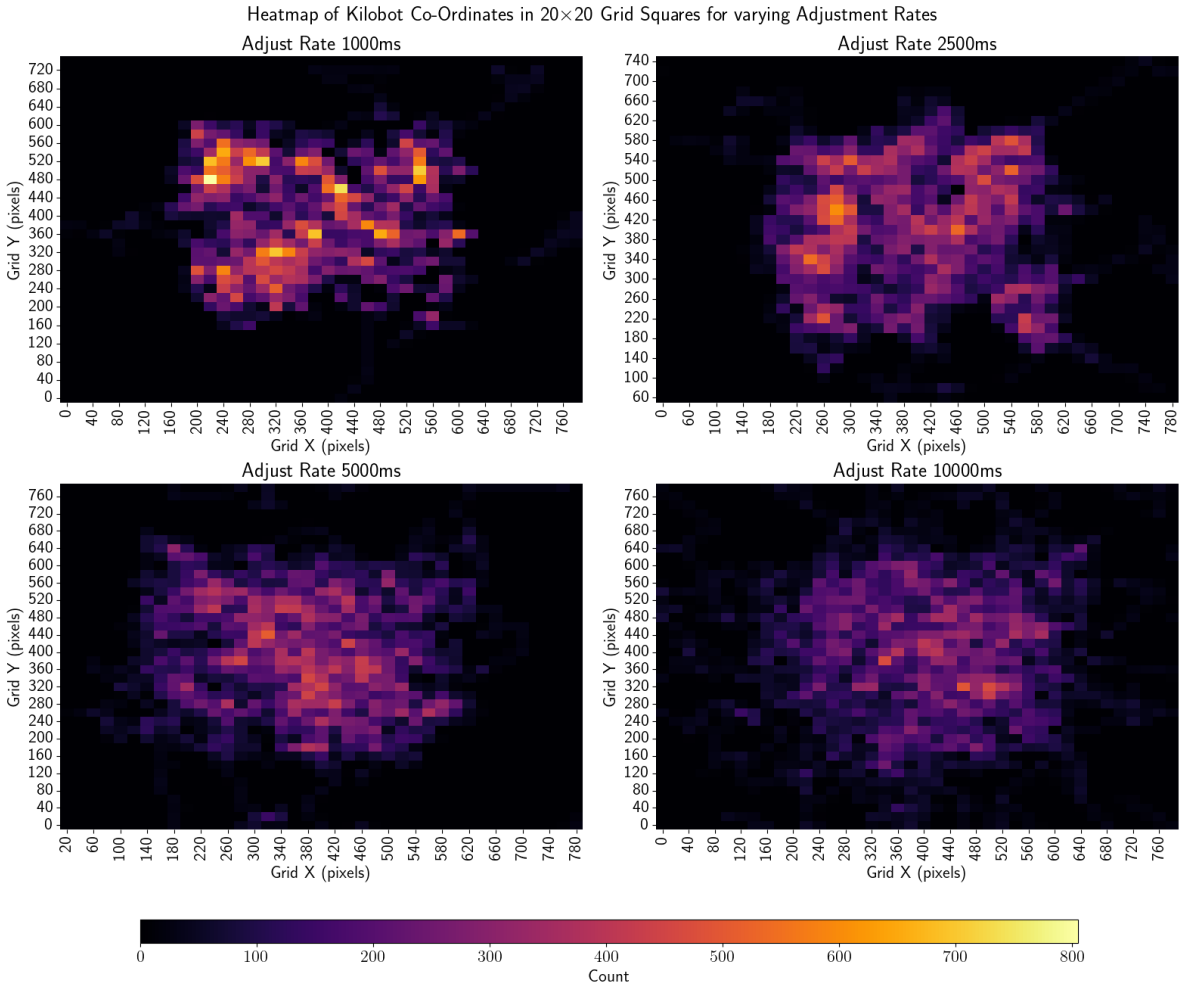


Figure 19: Agent locations for varying adjustment rate anti-alignment swarms are recorded every 100ms. Agents within a 20×20 px grid space are added to the count within that square and plotted as heatmaps. As the adjustment parameter is increased, it is seen that the dispersion of the agents increases, taking less time to reach further from their initial location.

5.1.3 Anti-Alignment vs No Alignment

When comparing anti-alignment to purely run-and-tumble swarms, the distribution of headings, order parameter and centre of mass co-ordinates remain very similar. This is expected due to the general similarities in simulation initialisation. No alignment systems spread out evenly from their starting location based on their random distribution of initial headings. However, the swarm structure degrades

quicker over time as opposed to anti-alignment. As figure 19 demonstrated, the cohesion of the group reduces with increasing adjustment rate. In the case of no alignment, this adjustment rate can be deemed as infinite, hence the swarm will tend towards the highest rate of loss of cohesion. This result is resonated in figure 20, showing the relationship of neighbour detection over time for both anti-alignment and random/no alignment swarms. It is noted how the rate of loss of neighbours is significantly higher for no alignment systems as opposed to anti-alignment. This suggests that anti-alignment swarms, although slower, more thoroughly and cohesively work together to explore the domain. Furthermore, the number of zero neighbour counts is also shown, depicting the number of agents who have no neighbours within their detection radius. Zero neighbour counts tend to arise quicker and to a higher maximum in systems without anti-alignment.

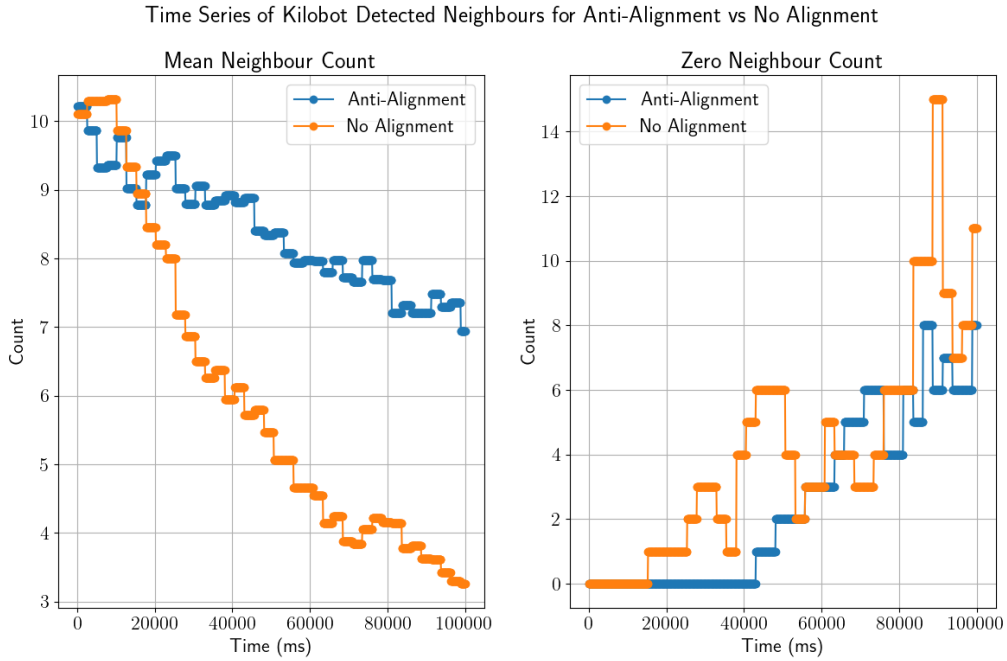


Figure 20: Time series of neighbourhood mean and zero counts for anti-alignment and random motion swarms. Anti-alignment agents have a lower rate of loss of neighbours, as well as less agents with zero neighbours, as opposed to its random motion counterpart.

5.1.4 Pattern Heading Estimation

Before experimental testing, the theoretical error for pattern heading estimation is evaluated within the simulation. Both patterns provide their own advantages and disadvantages during implementing; pattern A requires less computation to compute headings although more unique colour detection is required, meanwhile pattern B offers smaller approximate increments however is dependent on grid size. Figure 21 demonstrates how the average error of each pattern varies over the course of a short simulation. For the first 5 seconds, pattern B, outperforms pattern A. However, pattern A quickly lowers its average error at this point, remaining consistently lower than pattern B for the remainder of the simulation. This is due to 2 main reasons; NaN count and grid size. Figure 21 further demonstrates the NaN count for each pattern, being the total number of agents unable to determine their heading. Pattern A requires more distance to be travelled by an agent due to its 3 colour sequence technique before assigning its first heading. Additionally, not all sequences are encoded, including those beginning on a circle, hence no estimation is provided until a valid sequence is detected. As a result, the total time for all agents to receive an

estimated heading is approximately 8 times longer for pattern A than pattern B. However, as the number of approximations rises, the error rate falls below that of pattern B. Although pattern B can theoretically predict smaller changes in heading, it is limited by grid size. Velocity estimates are calculated based on time taken to traverse from one grid point to another, using the grid size as the displacement. Although only requiring a single colour/intensity change, short timings over edge borders will be estimated as large velocities in the respective direction, providing up to a $\pi/2$ error in either direction. Speed and accuracy of estimations can be improved for both patterns by reducing their projected size. However, individual space sizes must be large enough to accommodate a full agent as to ensure sensors adequately detect the displayed light.

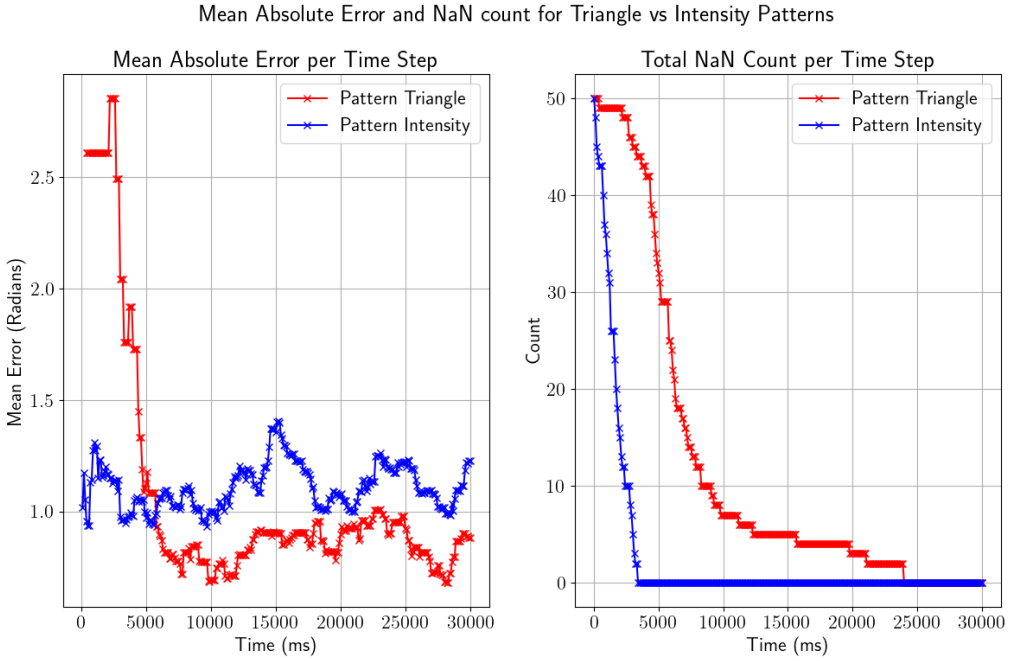


Figure 21: Simulation of a run-and-tumble swarm with 50 agents with pattern A and B projected for 30000ms each. The estimation error and NaN count time series are plotted. Estimation error is calculated from the absolute difference between the true heading and pattern estimation. NaN counts represent the number of agents who have yet to estimate their first heading.

5.2 Experiments

5.2.1 Kilobot Testing

Each of the rotational estimation methods were tested on a small swarm of Kilobots to investigate their respective viability. Limitations of the ambient light sensor made it difficult to accurately distinguish between similar colours and brightness's. Tests were conducted through setting equally bounded thresholds across the range of possible values, 0 to 1024. Pattern A, as described in section 4.4.1, requires 7 unique colours, hence the range was split into bands of length of approximately 150. Upon immediate inspection, all colours fell within the bottom 3 bands. Therefore, to increase the variability and sensitivity of the results, the sensor reading is multiplied by 10 and the upper most threshold reduced to approximately 10% of the maximum value. The results of the subsequent test can be seen in table 1. It is observed that on average the most consistently distinguishable colours are Blue, Cyan/Yellow and White. However, using this method a viable solution for pattern A was not found. Colours such as Magenta, Red and

Green were highly variable within the same ranges making it difficult to consistently discern them across the swarm.

Threshold	Displayed LED Colour	Projected Colour
0-149	Off	N/A
150-349	Blue	N/A
350-499	Magenta	Blue
500-649	Red	Blue, Magenta, Red
650-749	Cyan	Magenta, Red, Green
750-899	Yellow	Cyan, Yellow
900+	White	Cyan, Yellow, White

Table 1: Results from pattern A testing. Kilobots change their onboard LED colour depending on the detected ambient light sensor value (multiplied by 10). Projected colour indicates the area of the pattern where Kilobots are placed, with the resulting colour being placed in any band observed across the swarm. For example, Kilobots placed on Magenta tiles indicated LED colours of Red and Cyan, indicating its detected range of values between 500 and 750.

Using the gathered results, Pattern B was updated and tested with Blue, Yellow and White as its set colours, as seen in figure 22. 3 thresholds were therefore required, these being under 650 for Blue, over 1000 for White and Yellow in-between. Figure 23 shows a single Kilobot detecting each colour as it moves across the pattern. However, changes in intensity on the y-axis are harder to detect accurately. Additionally, noise from external light sources and variability across Kilobot sensor capabilities affect the stability of colour detection (see Appendix A figure 27).

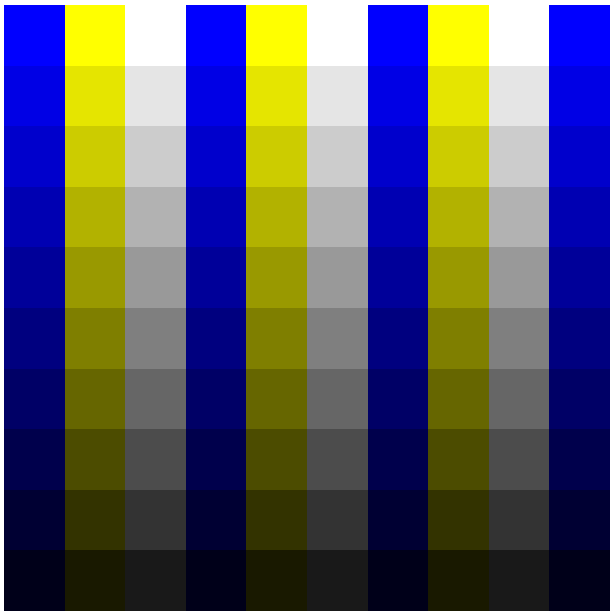
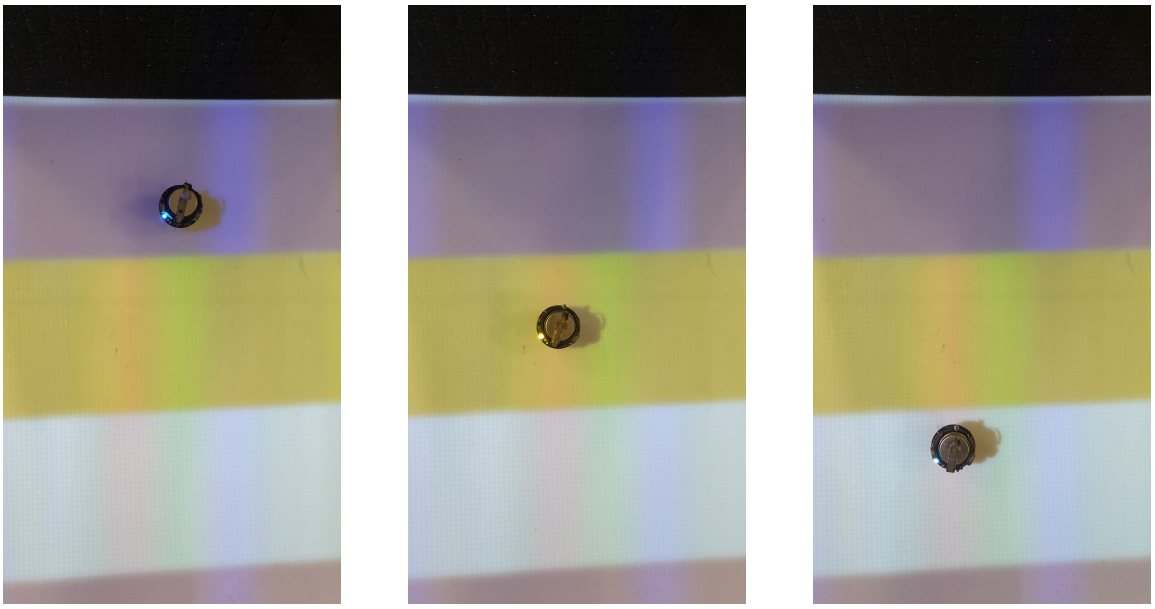


Figure 22: Pattern B updated to include most distinguishable colours (Blue, Yellow and White), as determined by pattern A testing.



(a) Blue section correctly detected, displayed by Blue LED.

(b) Yellow section correctly detected, displayed by Yellow LED.

(c) White section correctly detected, displayed by White LED.

Figure 23: Moving Kilobot detecting Blue, Yellow and White using ambient light sensor. Colours are projected using overhead projector, displaying updated pattern B image (figure 22).

Kilobots will therefore employ the method of rotational estimation based off average angular speed. During calibration, each Kilobot is timed during right and left motor tuning and adjusted to complete one full rotation approximately the same time as the previous. The average time to rotate 2π radians was estimated to 11 seconds. Kilobots will be placed in the same orientation, defining the zero heading with respect to the arena, followed by an initialisation phase to induce random headings.

5.2.2 Full Swarm

Figure 24 illustrates the implementation of anti-alignment on a real-world swarm of 50 Kilobots. Each Kilobot has been individually calibrated with a unique ID number and motor power levels for turning left/right and forward propulsion. The experiment begins by mass programming the agents with the infrared controller, each of which have been placed with aligned orientation as required for heading estimation. Sub-figure A shows the setup of the Kilobots, indicated by their white light. Here, each robots' random number generation is seeded based on their unique ID, giving unique behaviours to each Kilobot, that may also be replicated if necessary. Timers for appropriate features, such as alignment adjustment and neighbourhood array resetting, are also configured here. An 11 second initialisation period (to allow full rotational adjustments) is begun immediately after setup. This phase randomly adjusts Kilobot headings before the main algorithm begins, as to replicate simulation agent generation. This phase is indicated by a yellow light, as seen in sub-figure B. Sub-figures C through I display states in 30 second intervals of the experimental swarm.

The result presents some key findings of simulation vs experimental implementation. Firstly, placing the Kilobots within too close of a proximity of one another tends to result in mass clustering due to frequent collisions. The effect of which can therefore not be ignored at this scale. However, when arranged spaciously, as such in sub-figure A, Kilobots generally are more free to move, resulting in fewer un-resolvable clusters. Collision physics should therefore be modelled and including to investigate

at what initial proximity does the swarm fail. Although at larger scales the effects are seen to be negated over time, knowing pre-experiment the minimum distance to place agents would aid in future implementations. Secondly, Kilobots generally rotate around the axis of the leg of the respective direction, and not directly in place. Although having little effect on the overall result, tumbling and adjusting visuals differ substantially between experimentation and simulation. It is expected and obvious that experimental results contain sufficiently more noise and error than simulated robots. Agents should therefore contain a small noise parameter for co-ordinate and heading updates. These findings can be used to improve the representativeness of future simulations and explore their effect on theoretical models.

The aforementioned findings reflect fundamental differences in the way both systems perform, those of which cannot be easily negated experimentally. On the other hand, further instances are a result of an uneven mapping of parameter values when exposed to real-world implementation. An example is the ratio between rotational speed and running speed being significantly higher experimentally as opposed to simulation, i.e Kilobots are able to turn much more effectively than they run. This produces the effect of shorter run lengths between tumbling. This effect may, however, be mitigated by adjusting the tumble rate parameter. Simulation and experiments may therefore be used hand-in-hand to tune one another to the respective physical system. One additional aspect overlooked in simulations is the distribution of tumbling and adjusting timings. Although a small amount of time was allocated to performing these actions, simulations did not vary this time across the agents. This presents two possible fixes, neither more correct than the other. Simulations may be updated to include rotational dependant adjustment timings, i.e. the more an agent is required to turn the longer it takes to complete the action. Contrarily, Kilobots may be modified to include a uniform adjustment time, maintaining their state for an additional delay to ensure all other robots have completed their action before proceeding. Proof of this concept was illustrated in the initialisation phase. Although each Kilobot alters its heading by a random amount, hence requiring different adjustment times, the collective behaviour only begins once all bots have completed this action. This alternative dynamic would give the effect of that seen in simulations. The appropriate change would require further investigation as to which is the most suitable for the desired task.

To further analyse the results, video tracking is required to plot the co-ordinates of the swarm agents. However, the previously used video tracker is limited in its ability to differentiate objects accurately at close proximity, producing data too noisy for effective analysis. Therefore, a computer vision model may be trained using this video data as an effective method to capture the required data. However, due to the time limitations of this project, this was unable to be investigated in the time frame.

Visual results however are concurrent with that of simulations. Agents can be seen tumbling randomly and detecting their neighbours at regular intervals, adjusting accordingly. The final result produces a swarm that has sufficiently explored the domain, remaining generally within their neighbourhood regions. An additional result, not seen in simulations, is robustness to failure. Two Kilobots fail to initialise, as seen in sub-figure A by those without white lights displayed, showing little impact on the overall emergent behaviour of the swarm.

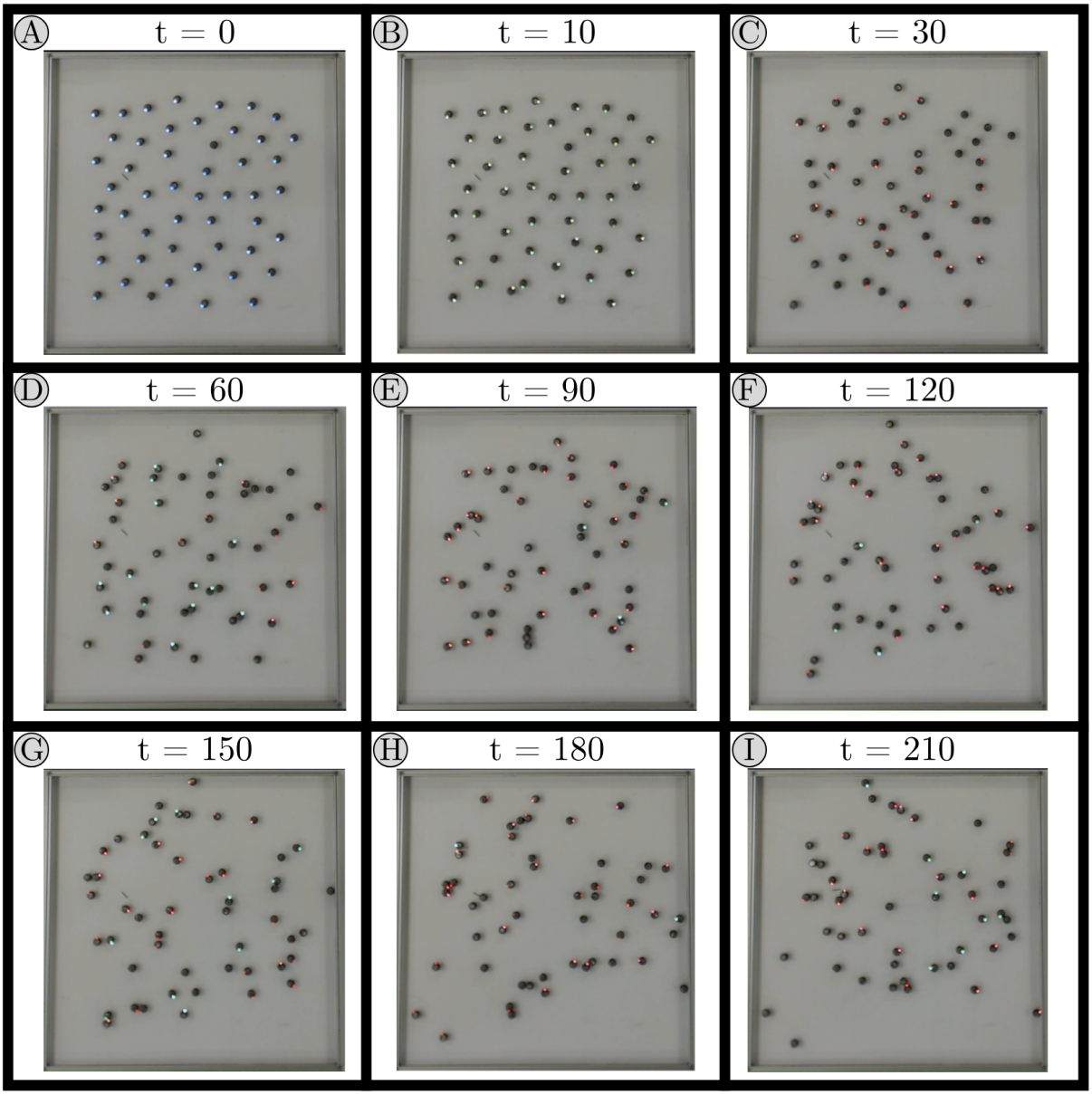


Figure 24: Diagram showing overhead camera recording of the movement of experimental swarm using 50 Kilobots. (A) Starting placement and initialisation of algorithms. (B) End of initialisation phase, producing random start headings. (C)-(I) 30 second intervals showing progression of swarm over time. Average full turn time was estimated to be 11 seconds. Experimental parameters: $\lambda = \frac{1}{50000}, DETECT\ RADIUS = 100\text{mm}, MAX\ NEIGHBOURS = 20, t_{2\pi} = 11\text{s}, t_{adjust} = 20\text{s}$.

6 Conclusion

This report has investigated the simulation and experimentation of an anti-alignment swarm. The aims pose how the combination of active matter and swarm dynamics may be utilised as an effective development strategy for intelligent collective behaviour. Anti-alignment was achieved through the modification of the classical alignment model, also known as the Vicsek model. Simulations highlight the effect seemingly innocuous changes can have on the emergent behaviours of swarms. Analysis of both models demonstrates these differences through co-ordinate heatmaps, heading distributions, system order and centre of mass tracking. Anti-alignment swarms tend to slowly and evenly disperse around their central starting position, whereas their alignment counterpart cover larger distances through a collective central mass random walk. Furthermore, the combination of anti-alignment and the underlying run-and-tumble dynamics presents an additional tuning parameter, being the adjustment rate of agents. This is shown to alter the rate of dispersion of the swarm, dispersing quicker with increasing adjustment rate. The anti-Vicsek model further presents coherent behaviours above that of random motion. By analysing the average and zero neighbour counts, it is seen how anti-alignment retains cohesion amongst the swarm as agents distribute across the environment.

Experimental testing displayed the challenges involved with the real-world implementation of this collective behaviour on a swarm of Kilobots. The simplistic nature of the Kilobots presented limitations on the capabilities and accuracy to achieve the desired dynamics. Firstly, each Kilobot required careful calibration to effectively achieve run-and-tumble motion; this process was highly time consuming and ineffective for some agents. Additionally, Kilobots cannot directly monitor their position or heading in space, hence requiring a supplementary process to do so. Three different methods were tested; two involving overhead projectors and ambient light sensors, one using average rotation timings. Theoretical errors for light pattern projection methods were revealed through simulation testing, presenting unique advantages to both techniques. However, timing estimation was implemented in the final swarm due to the limitations of the ambient light sensor; the produced sensor value limits the range of possible colours/intensities able to be uniquely identified.

A swarm of Kilobots was, however, implemented and shown to achieve similar behaviours to those of simulations. 50 agents were calibrated and programmed with the run-and-tumble and anti-alignment algorithms, displaying coherent neighbourhood detection and appropriate adjusting behaviours. Kilobot states were illustrated through their on-board LED, displaying the appropriate colour while performing an action. The swarm demonstrated how the collective behaviours are additionally robust, with the fault of two Kilobots having negligible effect on the dynamics.

The results of both simulations and experiments elevate the importance of both approaches. Simulations were utilised to quickly and easily investigate parameter changes, expected behaviours and theoretical limits of the models. Additionally, they provide visual and numerical aid when comparing to experiments to evaluate the implementation. Experiments illustrate the limitations of both the applied hardware and the simulation software. It was seen that not only can simulation representativeness of the real-world implementation be improved, e.g. through the addition of noise, but experimental algorithms may alternatively be tuned to better re-create simulations. Both representations may be used conjunctively to provide the most effective tuning and execution for the desired task.

6.1 Potential Applications

The findings of the anti-Vicsek model present some interesting potential applications for this collective behaviour. It is clear that small perturbations of the system parameters exhibit wildly different outcomes. However, this variability can be utilised as a method to induce multiple swarm behaviours within a single algorithm. By dynamically modifying the system's parameters, the algorithms can maintain memory and computational efficiency whilst boosting their capabilities.

An example application would be search and rescue using drone swarms. Similar to birds that use flocking during migration, swarms of drones could be deployed from one location and utilise alignment to travel together to remote areas. Once in place, the same algorithm, with the alignment parameter switched to negative, may be used to effectively explore the area whilst remaining within effective communication ranges of one another. Once the person or object of interest has been found, the drones can quickly communicate this to the rest of the swarm, ending the search.

It was shown how these algorithms can exist in decentralised swarms without any global positional tracking required. The simplicity of this solution therefore makes it a viable option for resource mapping in signal-obscured area. Long range communication may be impaired by the local environment, e.g. caves and oceans, making it difficult for geo-data collection and mapping. By taking advantage of the robustness and low range communication needs of anti-alignment swarms, these systems may be deployed in place of specialised expedition's to explore these remote locations.

The anti-alignment algorithm may be utilised to create temporary communication networks. By deploying a swarm centrally between multiple locations, the agents may connect these hubs together by sending signals across the swarm. Due to the maintained connectivity between the agents, it would be possible to create long range communication by utilising the many local connections, similar to the approach of packet routing within remote networks.

Lastly, with the addition of an outward bound (ensuring agents don't disperse indefinitely), these swarms may be employed for continuous monitoring in a wide array of applications. Periodic sensing, communication and random movement would allow swarms to efficiently scan large environments automatically and continuously. Low flying drones may be used for agriculture and wildlife monitoring, military reconnaissance and extraterrestrial exploration. Subaqueous swarms may be established within coral reefs, an increasingly at risk habitat due to environmental changes causing bleaching events. Smaller motorised units may be utilised for infrastructure inspection and intruder security systems.

6.2 Ethics

As mentioned in section 1.5, it is important that research in the field of swarm robotics outlines any ethical implications the results may induce. The findings of this report do not pose any ethical concerns for the implementation of anti-alignment on a swarm of Kilobots.

7 Further Work

System noise is currently drawn from a uniform distribution between 0 and 2π, taking effect as tumbling. Future work would explore alternative noise distributions and their effects on the anti-alignment swarm. An example could include the von Mises distribution, also known as the circular distribution. This distribution has the advantage of being limited between $-\pi$ and π, making it well-suited to generate random rotations. Additionally, it provides an extra parameter, κ, which can be used to alter the measure concentration over time, providing further potential behaviours to the swarm.

Future research into the modification of the underlying movement mechanics of swarm agents holds promising potential for uncovering further novel insights into collective behaviours and optimisation strategies. Such investigations range from simply choosing an alternative random walk strategy, to the development of adaptive algorithms that allow swarm agents to dynamically adjust their movement patterns. System parameters, such as the rate of tumble, may also be linked to these algorithms to provide intelligent behaviours based on environmental changes e.g. varying light intensity.

The introduction of additional parameters poses difficulty in system fine tuning, exponentially increasing the available state space. One approach to aiding in these investigations would be the incorporation of machine learning, more specifically genetic algorithms (GAs). GAs offer the ability to explore large parameter spaces automatically in search of local maxima for a given task, such as search and rescue, resource gathering etc. GA's could be an effective tool in optimising swarm behaviours at both the microscopic and macroscopic level.

Further testing involving projected colour patterns for heading estimation could be done using an alternative light sensing technique. In the paper by Alhafnawi, Hauert and O'Dowd, swarms of Kilobots were used to create a type of robotic canvas [21]. The report introduces the method as "a dynamic and interactive visual art medium, capable of displaying static images or video feeds, upon which humans can paint with their physical gestures or dedicated robots" [21]. The technique takes advantage of the waveform makeup of different colours displayed by a digital light projector and correlated to the ambient light sensor reading. As a result, the researchers were able to quickly and accurately detect 3 distinct colours; Magenta, Yellow and Blue. This method could therefore be used in conjunction with a modified pattern B (section 4.4.2) as a potential viable solution to estimating Kilobot headings.

Finally, alternative robotic agents may be used to further investigate the practicality of experimental implementation. More sophisticated robots may prove better equipped at re-creating simulation results, and hence more suitable for deployment in the real-world. Further capabilities may also be developed through modifications to the currently used hardware, such as the addition of accelerators/gyroscopes/magnetometers for position tracking. Multiple Kilobots may also be physically combined to act as a single larger agent, giving access to multi-sensor readings, granting increased intelligence at no additional cost. This could most effectively be used to sense environmental gradients pre-exploration.

References

[1] MarketsAndMarkets. (2023) Swarm robotics market. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/swarm-robotics-market-208095486.html?utm_source=Globenews&utm_medium=referral&utm_campaign=paidpr

[2] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.

[3] O. Peleg, J. M. Peters, M. K. Salcedo, and L. Mahadevan, "Collective mechanical adaptation of honeybee swarms," *Nature Physics*, vol. 14, no. 12, pp. 1193–1198, 2018.

[4] S. Ramaswamy, "The mechanics and statistics of active matter," *Annu. Rev. Condens. Matter Phys.*, vol. 1, no. 1, pp. 323–345, 2010.

[5] M. R. Shaebani, A. Wysocki, R. G. Winkler, G. Gompper, and H. Rieger, "Computational models for active matter," *Nature Reviews Physics*, vol. 2, no. 4, pp. 181–199, 2020.

[6] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical review letters*, vol. 75, no. 6, p. 1226, 1995.

[7] G. Baglietto and E. V. Albano, "Finite-size scaling analysis and dynamic study of the critical behavior of a model for the collective displacement of self-driven individuals," *Physical Review E*, vol. 78, no. 2, p. 021125, 2008.

[8] G. Grégoire and H. Chaté, "Onset of collective and cohesive motion," *Physical review letters*, vol. 92, no. 2, p. 025702, 2004.

[9] T. Vicsek and A. Zafeiris, "Collective motion," *Physics reports*, vol. 517, no. 3-4, pp. 71–140, 2012.

[10] A. W. Eckford, "Nanoscale communication with brownian motion," in *2007 41st Annual Conference on Information Sciences and Systems*, 2007, pp. 160–165.

[11] M. E. Cates and J. Tailleur, "Motility-induced phase separation," *Annu. Rev. Condens. Matter Phys.*, vol. 6, no. 1, pp. 219–244, 2015.

[12] H. C. Berg, *E. coli in Motion*. Springer, 2004.

[13] R. Bearon and T. Pedley, "Modelling run-and-tumble chemotaxis in a shear flow," *Bulletin of Mathematical Biology*, vol. 62, no. 4, pp. 775–791, 2000.

[14] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 3293–3298.

[15] Kilobotics. (2014) Kilobot documentation. [Online]. Available: https://kilobotics.com/documentation

[16] I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert, and J. Sharpe, "Morphogenesis in robot swarms," *Science Robotics*, vol. 3, no. 25, p. eaau9178, 2018.

[17] M. Alhafnawi, S. Hauert, and P. O'Dowd, "Self-organised saliency detection and representation in robot swarms," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1487–1494, 2021.

[18] S. Javed, A. Hassan, R. Ahmad, W. Ahmed, R. Ahmed, A. Saadat, and M. Guizani, "State-of-the-art and future research challenges in uav swarms," *IEEE Internet of Things Journal*, 2024.

[19] B. Saintyves, M. Spenko, and H. M. Jaeger, "A self-organizing robotic aggregate using solid and liquid-like collective states," *Science Robotics*, vol. 9, no. 86, p. eadh4130, 2024.

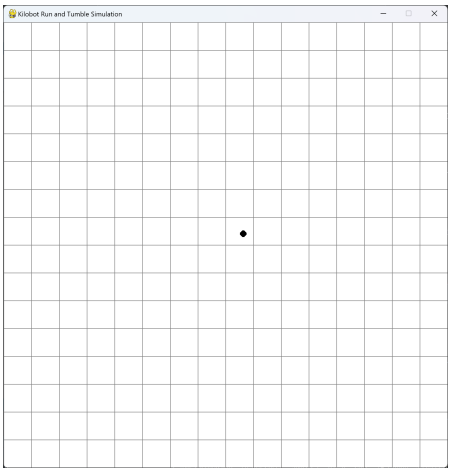
[20] H. Levine and D. I. Goldman, "Physics of smart active matter: integrating active matter and control to gain insights into living systems," *Soft Matter*, vol. 19, no. 23, pp. 4204–4207, 2023.

[21] M. Alhafnawi, S. Hauert, and P. O'Dowd, "Robotic Canvas: Interactive Painting onto Robot Swarms," ser. Artificial Life Conference Proceedings, vol. ALIFE 2020: The 2020 Conference on Artificial Life, 07 2020, pp. 163–170.

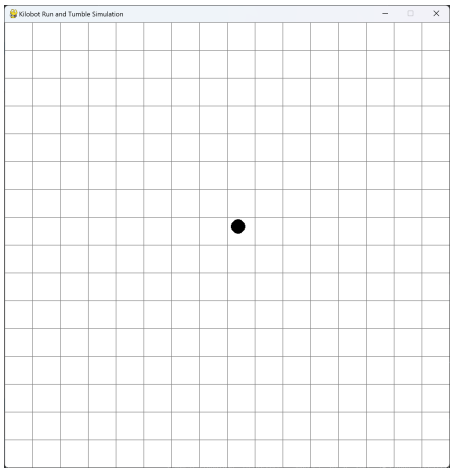
A Appendix

STATE	COLOUR (SIMULATION)	LED (EXPERIMENT)
Running	Black	OFF
Tumbling	Red	Red
Adjusting	Green	Green

Table 2: Colour representations of different states during simulations and experiments



(a) Scale factor = 5, $d = 2.6\times$ Scale = 13px. Equivalent of a Kilobot being placed in an arena of size 1.6×1.6 meters.



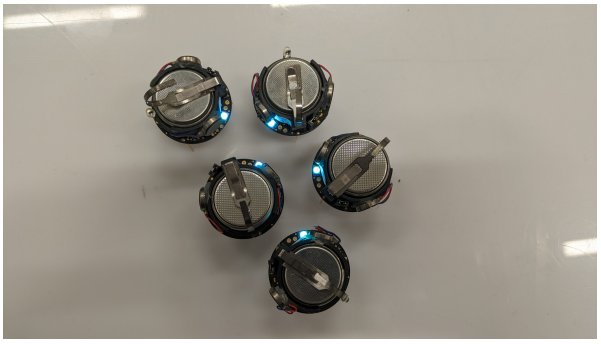
(b) Scale factor = 10, $d = 2.6\times$ Scale = 26px. Equivalent of a Kilobot being placed in an arena of size 0.8×0.8 meters.

Figure 25

(a) Single Kilobot detecting no neighbours indicated by red LED light.

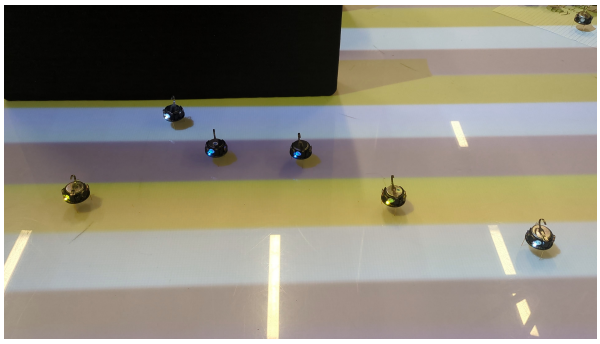
(b) Two Kilobots detecting a single neighbour indicated by green LED lights.

(c) Three Kilobots detecting two neighbours indicated by blue LED lights.

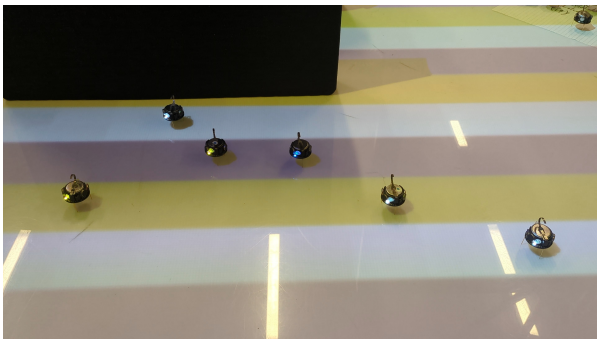


(d) Five Kilobots detecting three or more neighbours indicated by white LED lights.

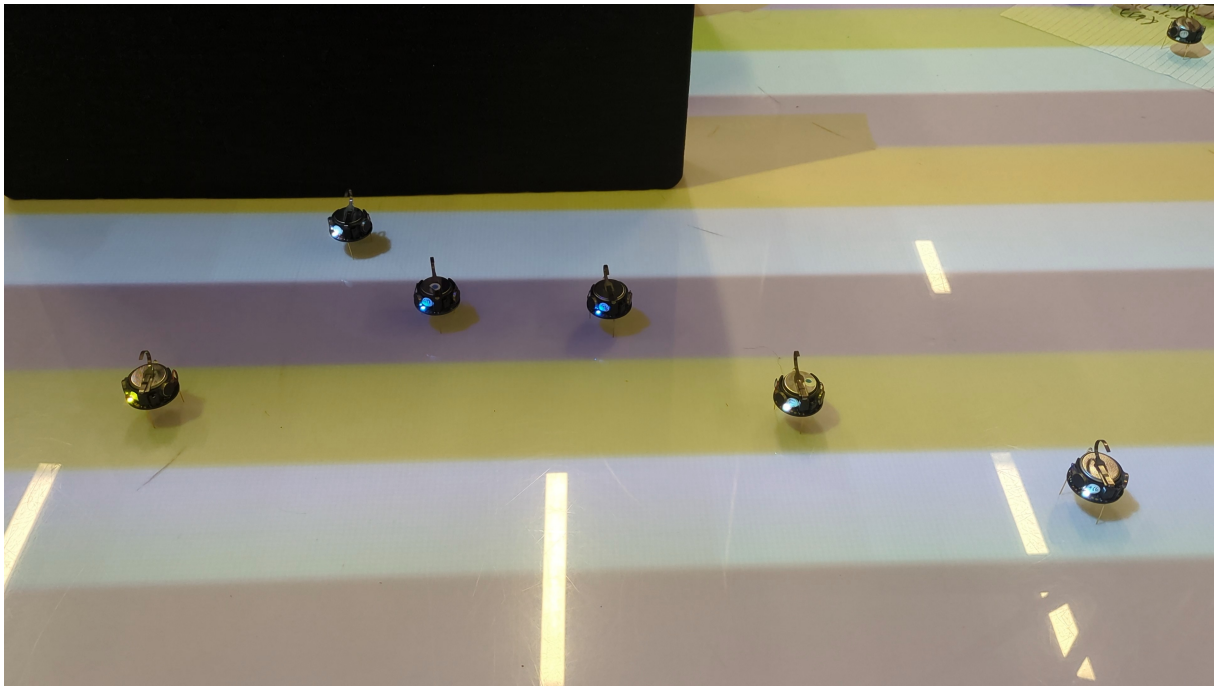
Figure 26: Neighbourhood detection count testing. LED colour indicates the number of detected neighbours; Red for none, Green for 1, Blue for 2 and White for 3 or more.



(a) Time = 0. All Kilobots correctly determining projected colours.



(b) Time = 1 second. Kilobots detecting incorrect colours.



(c) Time = 2 seconds. External light sources affecting 2 right most Kilobots.

Figure 27: Swarm of stationary Kilobots showing variation in detected colours over time and effects of external light sources. 4 left most Kilobots are covered from external light sources by black object, the remaining 2 are exposed to overhead lights.