Design and Density Control of a Swarm of Bimodal Particles

by

Justine Shaw

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

© Justine Shaw 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapters 1, 2, and 5 were solely authored Justine Shaw. The work presented in these chapters including formations of the introductory material, literature reviews, methodologies and result analysis.

Chapters 3 and 4 are based on a collaborative research project conducted with supervisor Gennaro Notomista. The content in these chapters has been developed from a paper co-authored by Justine Shaw and Gennaro Notomista, which is scheduled for publication in the proceedings of the Symposium on Distributed Autonomous Robotic Systems (DARS) later in 2024. Justine Shaw was primarily responsible for the conceptualization, experimentation, result analysis and initial drafting of the paper, while supervisor Gennaro Notomista provided guidance in form of MATLAB modeling, critical revisions, and editorial support.

Abstract

In this thesis, we present the design and control of a swarm of bimodal particles that switch their geometric shape between two modes. The particles are designed and 3D printed using layers of thermoplastic polyurethane(TPU) and polylactic acid(PLA) materials and are shaped like dodecahedrons. The switching control of the geometric shape is due to the particles reaction with an external stimuli of temperature, inducing switching between a open mode classified as mode 1 and an closed mode classified as mode 2. To enact the changes in geometric shape various temperature based hot and cold programming methods were conducted using water and artificial and natural heat sources. To quantify the aggregation of the swarm and control the switching motion we utilize the metric of the Motility-Induced Phase Separation (MIPS) index. To control the particles using the MIPS index we identify the motion model of the swarm as a Brownian particle or a noisy unicycle, having different parameters for both modes. In this research we experimentally validate the noise parameters that directly affect the motion of the particles thus affecting the MIPS index, and allowing us to control the swarm more directly. Simulations in MATLAB were conducted to characterize this switching behavior using the identified noise parameters and using the identified noise parameters the aggregation of the swarm in both modes were identified. Our simulations demonstrate that with identified noise parameters that affect the particles motion, desired swarm aggregation can be achieved using simple robots that are capable of changing their geometric shape. This research highlights how the simplicity of hardware design of a single agent can achieve aggregations for swarms which enable various environmental sensing tasks to be achieved.

Acknowledgments

I would like to thank everyone who contributed to this thesis in small and large ways alike. To my supervisor Professor Gennaro Notomista, I would like to extend my thanks for his support and guidance throughout my Master's program and writing of this thesis. To the members of my examination committee I would like to express my gratitude for taking time to evaluate the work presented in this thesis. I would also like to thank my supporting family and friends, who without, this thesis and journey would not have been possible. Finally, I would like to thank God for providing me with the strength and abilities to complete this work.

Dedication

I dedicate this thesis to my friends and family. Thank you for all your unfailing love and support.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgments v

Dedication vi

List of Figures ix

List of Tables xii

1 Introduction 1

2 Related Work 4

2.1 Swarms . . . 5

2.1.1 Swarms for environmental monitoring . . . 9

2.2 Safe and ecological robot design . . . 10

2.3 Morphing matter . . . 11

2.4 Swarm Control . . . 12

2.4.1 Density Control . . . 14

2.4.2 MIPS Index . . . 14

3 Design **16**

3.1 Mechanical Design 17

3.2 Description of modes 28

3.3 Model Parameters Identification 29

4 Density Control **35**

4.1 Density Control using DBSCAN 35

4.1.1 DBSCAN 35

4.1.2 Open Loop Control 38

4.1.3 Closed Loop Control 42

4.2 Aggregation Control using the MIPS index 43

4.2.1 MIPS Index 45

4.2.2 Open Loop Control 50

4.2.3 Closed Loop Control 52

5 Conclusion and Future Work **58**

References **60**

List of Figures

2.1 Example of a simple robotic swarm. Source:[57] 6

2.2 Naturally occurring biological swarms. Source:[52] 7

3.1 3D CAD model for the tetrahedron(Left) and 3D printed PLA tetrahedron(Right) 18

3.2 3D CAD model of dodecahedron (Right), 3D printed dodecahedron (Left) . 19

3.3 Size comparison for 3D printed dodecahedrons with the largest printed dodecahedron with surface area approximately 4.83 in^2 to the furthest right, the second largest printed dodecahedron in the center and the final and smallest printed dodecahedron with surface area of approximately 3.45 in^2 to the furthest left. 20

3.4 Energy vs Temperature graph for the state of a polymer as a polymer is heated above it transition temperature(T_g) and melting temperature(T_m). Source:[4] 21

3.5 The shape representation and the molecular mechanism of a polymer as it undergoes the process of being programmed moving from its permanent shape to a temporary shape then back to its permanent shape. Source:[4] . 21

3.6 Method of cold and hot programming for 3D printed polyhedrons. 23

3.7 Self folding 3D printed rectangular strip, with one layer of TPU and 3 layers of PLA. 25

3.8 Flat 3D printed circular strip, with one player of TPU and 3 layers of PLA. 26

3.9 Self folding of 3D printed circular strip, with one player of TPU and 3 layers of PLA after being hot programmed. 26

3.10 Self folding process of 3D printed cylinder PLA showing the initial print on the left, the straightened print in the middle and the reformed self folded print on the right. . . . 27
3.11 Closed 3D printed cylindrical strip made of PLA. . . . 27
3.12 3D CAD models of the dodecahedron model (top) and the actual printed versions of the dodecahedrons (bottom). On the left is Mode 1 (open) and on the right is Mode 2 (closed). . . . 29
3.13 Ball and ramp experiment showing the side and top view of the ramp and base where the balled 3D printed dodecahedrons are rolled to determine the model parameters. . . . 30
3.14 Overlayed image of printed dodecahedron particles after 1 second (top). Overlayed image with identified points of selection (bottom). . . . 31
3.15 Ball and ramp experiment depicting the calculation of $\Delta\theta$, by determining the angle between the initial starting point red circle on the farthest right and the 3D printed dodecahedron's red circle to the farthest left within the base. . . . 32

4.1 Spread of Clusters for 125 points displaying cluster one in red, cluster two in blue and outlier points with an "x". . . . 36
4.2 Clustering for 500 points after 200 seconds has elapsed using DBSCAN . . 37
4.3 Max Cluster Size vs Time for changes in Velocity . . . 40
4.4 Max Cluster Size vs Time for changes in Theta . . . 41
4.5 Trapezoidal Trajectory following for Max Cluster Size vs Time for changes in Velocity . . . 42
4.6 Trapezoidal Trajectory following Max Cluster Size vs Time for changes in Theta . . . 43
4.7 Direct comparison of clustering data for robots in the largest cluster for DBSCAN vs MIPS Index. Source: [27] . . . 44
4.8 MIPS index computation using the convolution of varying window sizes, computation of variances of density fields and normalized variances over the window sizes. Source: [27] . . . 47
4.9 Clustering for 1024 particle points in MATLAB with corresponding MIPS index values of 0.89 on the left and 0.19 on the right. . . . 48

4.10 Density distribution for a collection of 1024 particles of radius 1 generated in MATLAB with uniformly distributed collection corresponding to a low MIPS index of 0 (left) and densely distributed collection corresponding to a high MIPS index of 1 (right). Source [48] 49

4.11 Graph showing the MIPS index vs the number of iterations for a simulation of 1024 dodecahedron particles where mode 1 (open) is represented on the top and mode 2 (closed) represented on the bottom. 53

4.12 Graph showing the MIPS index vs the number of iterations for a simulation of 1024 cylinder particles where mode 1 (open) is represented on the top and mode 2 (closed) represented on the bottom. 54

4.13 Graph showing the MIPS index vs iterations of the dodechedron's closed loop controller that switched between mode 1 and mode 2 at a desired MIPS index value of 0.5 (blue dashed line). 56

4.14 Graph showing the MIPS index *vs* iterations for the cylinder's closed loop controller that switched between mode 1 and mode 2 at a desired MIPS index value of 0.6 (blue dashed line). 57

List of Tables

2.1 Applications of robotic swarms 8

3.1 Table of results for various 3D printed objects under cold and hot programming. 24

3.2 Comparison of Identified and Calculated Parameters for Dodecahedrons and Cylinders 34

4.1 Table showing differences for calculated values of I_{max}, I_{min} and the Average MIPS index in both modes based on the normalized MIPS indices for the dodecahedron and the cylinder. 52

Chapter 1

Introduction

Environmental modeling requires the representation of processes occurring in the real world within time and space [14]. To correctly represent these processes large amounts of data is needed for environment models. In the emerging field of robotics, many researchers have found that robot agents are capable of acquiring useful data when deployed within the environment. These robotic systems utilize various sensing and kinematic capabilities to accurately measure physical quantities such as temperature, humidity, airflow velocity, [55], and various gases like CO2, H2 and NH4 [12]. Other robotic systems use the collection of these physical quantities and engage in monitoring the worlds marine life [21], as well as monitoring homes using IoT (Internet of Things) devices [56],[64]. One challenge that exists in the usage of robots for environmental monitoring is the need to cover large spatial domains. In recent times, robot swarms have allowed for sensing and actuation in large environments because robot swarms consist usually of a large number of robots that work together to achieve distributed sensing and coordination [39]. Swarm robotics comprises of multiple individual agents that have their own processors, sensors and communication on-board that are interconnected allowing them to react autonomously to the environment [44]. Main characteristics of swarm robotics include autonomy, local communication and sensing, decentralized control, and cooperative task completion [11]. These characteristics stem from natural occurrences of this behavior in nature as swarms utilize the principle of bio-mimicry, which is the study of social animals like, ants, fish, birds and bees, and applies their behaviors to solve real world challenges [65]. These motivations among others allow for swarm robotics experts to design robots for environmental monitoring and sensing, performing dangerous environmental tasks like mine field clearing and adaptive scalability tasks like exploring ship wrecks and oil spills. Swarms are typically designed such that their collective behavior is achieved from interacting with members of the swarm and the

environment [65]. With the deployment of large number of robots within the environment, the impact of these systems must be taken into consideration. Traditional robotic systems and some swarms can be equipped with IoT devices, active sensors, processors, actuators, batteries. These devices are typically made of materials that are not biodegradable leading to several environmental concerns. To keep the environment safe it is important that when swarm robotics moves from inside controlled environments, where there is little variation in the physical quantities, to uncontrolled environments, swarms are designed with not only the protection of humans and wildlife that exist within the environment but also designed to manage electronic waste (e-waste), soil and water pollution. This consideration motivates the need for robotic swarms to be simple units that do not consist of harmful materials to the environment.

This thesis will address the aforementioned concerns for environmental monitoring using environment friendly design methods for robotic swarms by designing simple single units for a swarm of particles and a swarm level density controller. The main aim of this thesis is to design and control an environmental sensor as a swarm of particles, which are observable over time. This design should be a simple, battery-less, and microchip free particle that is potentially biodegradable. For the units to be simple and free to move within the environment without traditional actuation, we introduce a particle that is bi-modal in nature, that switches its geometric shape between two modes of operation, *open* and *closed*. To switch between the two modes of operation the particles when deployed should react to an external stimuli of temperature changes. With environmental safety as a leading factor these bi-modal particles are actualized by 3D printing the designs out of polylactic acid (PLA) and thermoplastic polyurethane (TPU), with further studies from various bio-degradable materials that are able to perform similarly to ensure geometric shape changing in various temperatures. Consequently after designing such particles it is important to explore how these particles will react with other members of the swarm and also the environment. To study these behaviors the need for a swarm level density controller is required. Swarm level density controllers are algorithms and methods that regulate the spatial distribution, and level of aggregation of agents within a swarm to achieve various task operations. Swarm level density controllers can be used for formation control [53], obstacle avoidance[43], as well as dispersion [30]. To control a swarm considering the design challenges' omission of swarm communication, sensing microchips and micro controllers requires accurate aggregation quantification. To quantify and control the bi-modal particles we leverage the aggregation metric of the Motility-Induced Phase Separation (MIPS) index. Using the MIPS index data we are able to deduce the particles status within the environment, specifically as they move and form clusters. Based on the metric we can determine how the particles move and examining the final distribution of

the particles we can infer environmental temperature changes.

The remainder of this thesis is structured as follows:

- Chapter 2: In this chapter we present the background and related works that are crucial in the design and control of the each particle and the swarm at large. The design section will focus on principles from soft robotics, morphing matter controlled by temperature changes, morphing matter for geometric changes, swarm design and lastly biodegradable and compostable design methods. The control section will focus on biomimcry patterns that influence the control of swarms, previous works within the field for density control methods and strategies, and previous works with MIPS index.

- Chapter 3: In this chapter we detail the mechanical design of the particle. This includes the description of the two modes of operation for the particle, and methods used for identifying the model parameters that are used within the simulation to mimic the particle swarm. Important contributions from this section are from [48], which is to be added to the International Symposium on Distributed Autonomous Robotic Systems (DARS) in 2024.

- Chapter 4: In this chapter we focus on background information that captures definitions of the MIPS index and the mathematical principles behind how the index is capable of determining aggregation. We also describe how the designed particles are controlled using the MIPS index. Lastly we demonstrate the results obtained from simulations, which used the identified model parameters to achieve open and closed loop control. Important contributions from this section are from [48], which is to be added to the International Symposium on Distributed Autonomous Robotic Systems (DARS) in 2024.

- Chapter 5: In this chapter we give the concluding thoughts and future works that can be presented based on the experiments and information presented within this thesis.

Chapter 2

Related Work

In this chapter we present the background and related works that are crucial in the design and control of each particle and the swarm at large.

To build a stable foundation, we first introduce the concepts and motivations behind swarms and swarm robotics and look to expand these motivations to their use in environmental monitoring. Within robotics it is widely noted that swarm robotics stems from swarm intelligence - where Gerardo Beni et. al [9], states that systems of non-intelligent or simple robots can work in unison to create organized patterns or structures within their environment. Common characteristics from [45] that describe swarm intelligence include:

- Flexibility - the ability for a swarm to adapt to different environments while solving different actions.
- Error tolerance / Redundancy - the ability of the swarm to perform corrective actions and complete the given tasks irrespective of a singular agents failure to complete an action.
- Scalability - the ability of a swarm to perform its given tasks independent of the number of agents present within the swarm.

Abualigah et al. explains in [1] that a main goal of swarm intelligence is to replicate decentralized organisms that are naturally intelligent, where the collective intelligence of the group is greater than that of individual agents within the group. Naturally occurring organisms that swarm intelligence is based on include bacteria, ant colonies, bees, birds and fish among others which will be further expounded upon in section 2.1. These

species display naturally intelligent behaviors which are studied and incorporated into various swarm intelligence algorithms, methods and techniques. Using inspiration from these natural organisms, many swarm intelligence techniques exist which utilize stochastic methods, in which simple individuals interact with their environment and produce some form of collective behavior. Swarm intelligence then seeks to implement an advanced heuristic approach to practically, optimally and strategically solve complex problems. Common swarm intelligence applications and algorithms include:

- Artificial Bee Colony(ABC) and Ant Colony Optimization (ACO), which are meta-heuristic optimization algorithms based on the forging of a bee colony and ant colonies food searching methods respectively [45].
- Firefly Algorithm (FA), inspired by the light intensity and flashing of fireflies [45].
- Bacterial Foraging Optimization Algorithm (BFOA), inspired by E.Coli baterium's ability to forge together [45].
- Particle Swarm Optimization (PSO) which is an optimization algorithm to solve problems that have a solution which is best represented as a point or surface in an n-dimensional space [28].

From these swarm intelligence principles it is determined that swarm robotics applies swarm intelligence methods to analyze the activities of physical robotic devices, actuating and operating under various conditions, effecting changes within the environment based on various inputs [28].

2.1 Swarms

This section will detail the definition, characteristics, historical background, applications, challenges, limitations and recent advances as it relates to robot swarms.

A robot swarm is a collection of three or more robots that cooperatively perform tasks, which operate with minimal or no human control [5]. Figure 2.1, shows an example of a robot swarm with multiple different agents, that are capable of cooperatively performing a task with minimal human control. As stated previously robotic swarms are influenced by the natural occurrence of swarm intelligence in nature. Using bio-mimicry many robot swarms exist that operate cooperatively to perform various tasks. In figure 2.2, we see the natural occurrence of swarms or groupings biologically in nature like ants, birds, fish

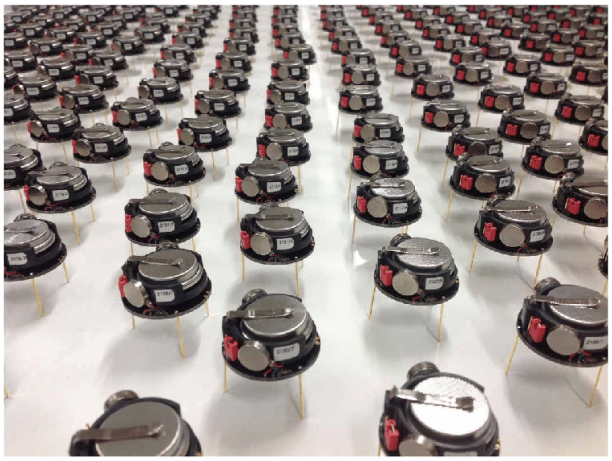


Figure 2.1: Example of a simple robotic swarm. Source:[57]

and mammals from [52], that can be implemented into robotic swarms. Observing various literature to support such claims we note that in [32], the authors propose a distributed bees algorithm for task allocation for robot swarms based on the foraging behavior of colonies of bees in nature. The authors in [35] describe a decentralized transport system based on the cooperative transport that ants display by changing position and alignment until the prey is moved. [42] uses the flock formation of birds to inspire a behavioral flight model based on avoidance, copying, centering and viewing. Other organisms also exhibit swarm behaviors such as active and passive Brownian particles where particles gain energy from their environment and converts this energy into motion [7]. Key features of robot swarms include, scalability, flexibility, redundancy and robustness [65]. These key features enable the robot swarm to solve various tasks in different relative swarm sizes while functioning in diverse environments compensating for individual agent failures where applicable. Other highly important features of swarms usually include the robot agents and swarms at large being simple, decentralized, self organized and distributed. Those features allow each individual robot agent that make up the swarm to be able use local information alongside interactions with the environment to make decisions about order, structure, task completion, motion and actuation. In unison all features mentioned give various advantages within the field of robotics. These advantages are comparable to single individual robots and also other multi-agent / multi-robot systems. In terms of single robots the authors in [52], note that swarm robotics allow for the following advantages:

- Parallelism - Due to the number of robots present in the swarm many tasks can be done in parallel thus saving a significant amount of time in comparison to a single



Figure 2.2: Naturally occurring biological swarms. Source:[52]

robot agent.

- Scalability and Stability - The swarms ability to adapt to changes in size to complete tasks, without disturbing the entire swarm or requiring alterations to the hardware or software. This feature allows for more stability within a swarm of robots if one or more agents are not functioning as required. This is vastly different to singular systems as, if the singular robot is incapable of completing the task there is no other capable agent, thus impacting stability and requiring changes to either the hardware or software.

- Economical and Energy efficient - The nature of swarm robotics dictates that individual agents within the swarm be simple, thus the cost of each individual robot agent in a swarm is less when compared to one single robot. These swarms can be mass produced and still be more cost effective and also energy efficient as any onboard sensing or power is smaller than that of a single complex robot with high powered sensors and IoT devices.

These advantages prove useful in the field of robotics where swarms of robots are capable of performing tasks that cover large areas like environmental monitoring, tasks too dangerous for humans or singular robots like mine rescuing, tasks that requires scalability like oil leakages and tasks that require redundancy [52]. Using these advantages are important when moving from simulation to real world applications and as such it is important to review robotic swarms applications that have been invented and used. Table 2.1 lists a few swarms created for specific applications.

Application	Description	Source
Oil spill cleanup	Uses a swarm of sea robots to collect oil spills reducing human intervention and enhancing accuracy.	[62]
Agriculture	This particular implementation looks into using swarm robotics for weed control problems using a swarm of unmanned aerial vehicles.	[3]
Space exploration	Uses the particle swarm optimization(PSO) algorithm to plan paths for space exploration.	[26]
Traffic monitoring	Uses an aerial swarm and a control algorithm to continuously monitor traffic in a simulation SwarmCity which models cars and drones realistically.	[22]
Indoor navigation extended to environmental monitoring and disaster mitigation	Introduced a swarm navigation algorithm to search indoor environments	[50]

Table 2.1: Applications of robotic swarms

Though there has been great advancements in robotic swarm technology and algorithms numerous challenges and limitations still exist for the successful deployment of these systems in real life. Namely some of these challenges and limitations include, difficult coordination and communication within swarms, as well as interference between robot agents resulting from collisions and occlusions [39]. We seek to address the coordination limitation as the proposed solution seeks to control the density of the designed swarm.

The following section 2.1.1 will detail robot swarms that exist for environmental monitoring as we seek to use these swarms as inspirations and motivations for the solution presented in this thesis.

2.1.1 Swarms for environmental monitoring

This subsection will focus on using the principles outlined in the previous section and applying them to robot swarms for environmental monitoring. It will also focus on previous research in the field for environmental monitoring using robot swarms and its relevance and limitations as we build our solution.

The design and technology of robots and robot swarms have been mostly confined to research laboratories due to the unpredictable nature of some environments, however these experiments provide a good baseline to start from when designing robotic swarms and robots for environmental monitoring. Research for swarms in environmental monitoring includes [12], where a wheeled rover is used to measure various pollutants within the environment, [19], where research is conducted to test the feasibility of robot swarms in marine environments, and even [3] which targets a road map for weed control in the agricultural sector by means of a swarm of unmanned aerial vehicles(UAV's). Due to the fact that robot swarms are able to cover large spatial domains, have redundancy and robustness, operate in remote, hazardous and inaccessible areas, as well as posses local sensing capabilities which can influence collective behaviors, this makes it a viable candidate for environmental monitoring. Drawing inspiration from swarm intelligence and social animals, early works in technology with multi-agent robotic systems have influenced their use in the environment. The application of environmental monitoring for robot swarms has made vast improvements over the years, where common uses include agriculture, air and water quality monitoring, wildlife monitoring, climate change data and disaster response. For inspiration for our specific solution, we turn to research done by [29], where Hird *et al.*, describes a robot swarm design that uses preloaded elastic energy and mechanical latches to randomly jump around a large environment. The jumps are triggered by some environmental stimulus determined by the user and continue by simulating 500 robots to cover an area. Robots within the swarm are designed without any electronics, which has capabilities to be fully biodegradable. This work is relevant for our solution as we seek to create a robot swarm lacking electronics, capable of giving feedback about the environment based on their morphing capabilities due to an external stimuli of temperature.

The next section 2.2 will detail related works for safe and ecological robotic design as we seek to design a robot agent that is environmentally friendly.

2.2 Safe and ecological robot design

For robots to be safe and ecological it is important that robots be designed, developed and deployed in such a way that it does no harmful impact to the humans and species that use the environment but also minimizes the impact the technology has ecologically. For robot swarm safety we look to [31], where it details 10 checklist items to consider when determining the safety of a robot swarm. The ten factors to check includes the following:

- Ethical usage.
- Legality.
- Accountability.
- User interaction.
- Deduction of individual agents causing potential physical harm to humans, animals or the environment.
- Deduction of robot swarm causing potential physical harm to humans, animals or the environment.
- Behavioral harm from individual robots.
- Behavioral harm from the swarm of robots.
- Security of individual robots.
- Security of the swarm of robots.

The authors after developing these questions then applies them to a swarm of 100 small floating robots that are deployed to monitor water pollutants.

We can also look to principles within soft robotics for safety as soft robotics allow robots to change its shape based on external stimuli by having bodies designed with materials that are flexible. We see such developments in research like [61], [34] and [46], where soft robotics are developed for drug delivery operations in the biomedical field, development of soft actuators and shape-changing robots to achieve locomotion over flat and inclined surfaces respectively. These principles are crucial to consider in our design factors as we see development of robots that are electronic-less but are still able to achieve motion and though they might not all be bio-degradable they are still in many ways better for the

environment by designing without electronics and batteries. As society continues to evolve over time it is important to practice reducing the ecological footprint to aid in the protection of natural eco-systems, supporting sustainability goals as well as prevention of pollution of land and marine life. We now focus on developments in robotics that practice ecological design. For a practical and safe and ecological robot for environmental monitoring we look to the E-seed self-drilling mechanism described by [38]. The mechanism is a three-tailed seed carrier platform made from wood veneer, which is a stiff biomass. The solution is biodegradable and is able to self drill into soil by reacting to moisture fluctuations in the environment. This is also an important inspiration for our work as this work performs actuation based on moisture changes within the environment - which is an external stimuli, whilst also being biodegradable. Some challenges for ecological environmental monitoring still remain like trade-offs in performance resulting from the prioritization of safety and environmental concerns, scalability and cost implementations that might be difficult for sourcing ecological materials and also possible regulatory setbacks in order for reliable research and regulatory acceptance to use these sustainable practices.

The next section 2.3 will focus on works related to morphing matter that are important for our study due to our design constraints.

2.3 Morphing matter

This section will detail morphing matter as it relates to robotic swarms and detail research in the field that is applicable for our research.

Like the principles of swarm intelligence morphing matter gains its inspiration from naturally occurring phenomena in nature. Morphing matter refers to materials that adapt their shape or position in reaction to some external stimuli [33]. In relation to robotic swarms, morphing materials allow for greater adaptability as the robots are able to fit various tasks in complex environments, they also can improve efficiency as their morphing capabilities can enable them to optimize their configurations, and finally it increases flexibility of the robots as shape changing allows the swarm to complete more tasks without needing specialized robots. Some applications of morphing matter include environmental monitoring, agriculture, medical field, and also for search and rescue missions. Morphing materials can exist in a few formats, namely soft robotics [46], shape memory polymers, and 3D/4D printing [4]. Existing research shows how these shape changing robots can be applied to these industries for example in [51], we see a design for a swarm of robots called ShapeBots which individually and collectively change their shape displaying information or performing tasks that require some form of actuation. In [4], we note the usage

of 3D printing and shape memory polymers to create self-folding 3D polyhedrons. The authors use layering of PLA and TPU having shape memory characteristics and various directional patterns which allow for actuation from the various 2D shape to create a 3D shape if required. One advantage of this morphing materials is the usage of shape memory polymers. Shape memory polymers are materials which can change their flexibility and thus shape based on various forms of thermal stimuli. This is crucial to our research as we look to use the shape memory and programming techniques detailed within [4] to further our research. These techniques allow for morphing robots to be simple, accessible and easily replicated. It is noted that [4] uses morphing mechanics principles to achieve its shape changing capabilities. Morphing mechanics principles govern how robots are designed and they include understanding materials as well as strain- mismatch, elastic instability, origami and kirigami principles, morphogenesis-inspired design and assembly of discrete elements [58]. For our purposes we determine that origami and kirigami principles are required as our two modes being *open* in a 2D structure and *closed* in a 3D shape thus using folding principles from origami. Though these works exist it is noted that morphing materials have the following limitations and challenges: control complexity, material durability, general acceptance into existing swarm robotics. Our research will seek to improve on the swarm control and will further expound on these concepts in 4. As it relates to our work we will use the principles outlined from [4] and [58] and expound on our solution in Chapter 3.

In the next section 2.4 we will focus on related works for swarm control and swarm density control.

2.4 Swarm Control

This section will detail swarm control, various swarm control methods including density control and the aggregation metric of the MIPS index and its relevance for our proposed solution.

After looking in depth into robot swarms and various design methods for environmental monitoring it is important to then understand and learn to control these swarms. Swarm control is the usage of methods and algorithms to effect changes on how the agents within the swarm aggregate or disperse into various formations and spatial configurations to complete the given tasks. With original influences from swarm intelligence techniques we recognize that many control algorithms mimic that of swarm nature. In [10], the authors introduce a honeybee inspired control algorithm BEECLUST to study robot swarm interaction. The research operates by allowing the swarm to be influenced by illuminance,

having sets of the swarm aggregate at points of high and low illuminance. This influences our research solution as we notice the usage of external stimuli to influence aggregation behavior in swarms. Various swarm control methods exist within the robotics field such as:

- Formation control: This is controlling the pattern or various sizes of robotic swarms to traverse and interact with their environment. Common formation control methods are leader follower [37], virtual structure [53], potential field approach [59], . In the various methods we see how either the swarm follows a leader robot for position and trajectory information, or the swarm uses a virtual structure to maintain specific position based on neighbouring robots, or the swarm using potential fields, attractive and repulsive forces to either go towards a goal or repel obstacles.
- Dispersion: This is controlling how the robot swarm is spread out across the environment. [30]
- Task Allocation, coordination and obstacle avoidance [30]. These methods control which tasks the robots decide to do in specific orders of priority and coordinate with each other whilst avoiding various obstacles within the environment.
- Aggregation : This is controlling how the robot swarm form groups/clusters by interacting with other agents within the swarm and the environment [6].
- Switching control: This form of control includes allowing the robots to switch between various modes of operation to complete a task. In [54], we see a switching control method that is implemented when a distance threshold is met.
- Trajectory Tracking: This is controlling the swarm by allowing the robots to follow specific trajectories. Authors for [2], use a combination of virtual vehicles, a leader follower robot and geometric trajectory tracking to maintain a specific formation along a geometric path.

While these methods are suitable for many applications such as search and rescue, transport, environmental monitoring, joint task completion our solution requires us to control aggregation and more specifically the density of our swarm. The following section 2.4.1 will detail density control methods and previous works for density control.

2.4.1 Density Control

To understand the nature of robot swarms and their purpose within an environment it is important to have a measure that quantifies the aggregation being displayed by the swarms. Density control refers to the spatial distribution of robots within an environment. This is useful for task allocation, collision avoidance, and adequate coverage of an area. The advantages that stem from density control include efficiency in terms of the swarm being able to cover and monitor large areas if necessary, increased scalability as the size of the swarm can be increased or decreased based on need or robot availability and also increasing the swarm ability to adapt to various changes within the environment. Various aggregation metrics exist within literature for us to study. In [20], the authors use density, voronoi tessellation and centroids which contain attractive and repulsive forces, to better identify characteristics of the swarm namely, expansion, stability and contraction. [17], uses cluster analysis algorithm called Density-based spatial clustering of applications with noise (DBSCAN) and inter-robot distance for its density control. The algorithm uses variables like minimum points and distance and differences between the various points to identify density regions, marking non-uniform clustering data. This density based control is further expounded upon in Chapter 4. Though these methods have been proven within simulation and lab setting another novel metric exists to quantify the degree of clustering within a robot swarm which is independent of the choice of parameters. This metric is known as the Motility Induced Phase Separation Index (MIPS) index and will be focus of the following section 2.4.2.

2.4.2 MIPS Index

This section details the origins of the MIPS index and expands on the notion and its relevance to this work.

The Motility Induced Phase Separation (MIPS) index is a metric that stems from density variations in self-propelled particles, where high density particle regions slow down due to collisions and clusters being formed. The metric uses the physics principle of phase separation to quantify and determine the degree of aggregation with an identified swarm [27], [24]. The main contribution to this research stems from [27], where we seek to expand and further the research presented in this paper. The authors of [27], detail the use of the MIPS index to quantify aggregation of their micro-bristle robots within an identified region by varying characteristics known as motility characteristics. Motility characteristics are parameters that affect the motion of particles. In this case they use vibration-driven robots and vary vibration amplitude and frequency which also varied the behavior of the

swarm. To further this work we can use their findings that the MIPS Index is able to quantify the aggregation of swarms to control a different swarm of robots using a more direct control approach. Further details and explanations for the usage of this metric will be provided in Chapter 4.

The following Chapter 3 will detail the design of the particle swarm including the description of the two modes and the identification of the modes model parameters for simulations.

Chapter 3

Design

This chapter will focus on the mechanical design of a swarm as well as the parameters identified for the simulation of the agents within the swarm.

One of the primary goals of this thesis is to design a particle that has the capabilities of changing its shape between two modes, *open* and *closed*. This change in shape should be a result of temperature changes in the environment enabling the particle to traverse its environment and give feedback to the user based on its present shape and distribution within the environment. The switching capability of the particles between two modes labels each particle as bimodal and it is the bimodality of each particle that allows them to traverse the environment giving the deployment team information about a given area over time. The designed robots should be electronic free and have the ability to be biodegradable. To achieve this feat it is necessary to adopt principles from morphing matter, where materials change their shape due to reactions with external stimuli like temperature, pH, moisture, electric and magnetic fields [41]. The morphing principles in unison with origami, kirigami and morphogenesis-inspired design creates a good framework to build our solution. The origami and kirigami principles utilized stem from materials using crease and cut patterns to fold their planar structure to create 3D structures. To jump start our solution for bimodal particles we make use of 3D printing and shape memory techniques. Previous research by [4] shows how the authors manipulate the shape memory retention characteristics of PLA and TPU to allow for various self folding patterns and 3D polyhedrons.

The following section 3.3 details the mechanical design of the bimodal particles and focuses on various experiments completed to choose our final solution.

3.1 Mechanical Design

This section will detail the mechanical design of the bimodal particles, shape memory principles involved in designing such particles, as well as various experiments to test the shape memory characteristics of the 3D printed designs.

Bimodal robotics is the study and design of robots that have the ability to change features like locomotion, sensing, actuation among others that are embedded into the robot. One main advantage of a bimodal robot is its ability to adapt to diverse scenarios by switching between its modes [18]. Among other advantages include robustness due to switching capabilities providing the opportunity to switch to a different mode if one encounters issues and greater optimality in that the robots can be equipped with capabilities to determine the most optimal mode based on sensing. For the works in [18], we note that they design a snake-worm bi-modal robot called Snake-Worm Locomotion Robot (SWL-Robot) which switch between two modes of snake-like locomotion and lateral undulation. This bimodality is said to increase terrainability and efficiency and was tested using simulation of transitioning between a peristaltic motion inside and tube and planar motions on a flat surface. For the works in [15], the authors uses principles in self burying robots to design a bimodal robot that is equipped to switch between one mode of travelling above the ground and the other of burying itself in the ground. The robot labelled the Quad Digger experimentally successfully buried itself in sand and though with adjustments successfully traversed a light soil environment surface. The authors also note that a possible expansion for this design is to develop and unassisted mode of switching between the two modes of operation.

Using inspiration from these two papers we seek to design an unassisted bimodal robot that is capable of switching between two modes of *closed* and *open*. From the principles of origami, kirigami and morphogenesis-inspired design the two modes are to be actualized by crease deformation of a designed 3D printable 2D shape which will morph edges to close forming a ball-like shape and open forming a flat 2D structure. As we also utilize morphing principles of designs changing based on external stimuli, we aim to trigger the aforementioned shape change with a chosen stimuli of temperature. The considered design should use changes in temperature within the environment to either open when the temperature is low and close when the temperature is high. With these shape changing effects we also intend and expect forms of motion to accompany them. Due to its design simplicity we consider the motion of the*closed* mode of operation as rolling with a specific direction and orientation along a surface and the *open* mode is considered not to be moving in any specific direction and orientation. This choice of design parameters enable the particles to traverse the environment in its closed state by forming a ball shape, possibly

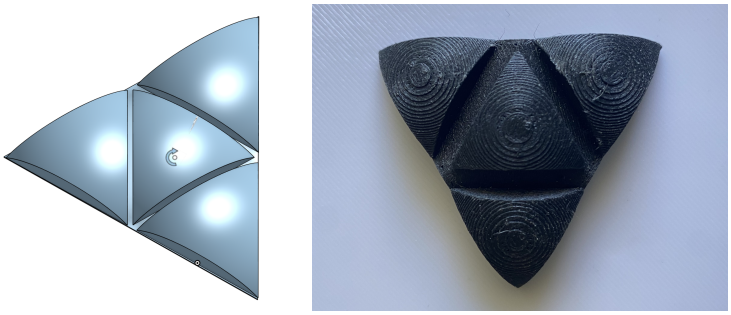


Figure 3.1: 3D CAD model for the tetrahedron(Left) and 3D printed PLA tetrahedron(Right)

indicating higher temperatures and in its open state be moving freely possibly indicating lower temperatures in the environment.

The design of our solution was created using online 3D modelling software Onshape[1] and printing was completed using the following two type of printers: Original Prusa i3 MK3S+ 3D printer, Original Prusa MINI+. For our first design it was decided to use a geometric shape that was simple and has a small number of faces, to study the bimodal behavior. This shape was determined to be a spherical tetrahedron, having four spherical faces and six arced edges. Figure 3.1 displays the design of said tetrahedron in Onshape and also the 3D printed soft PLA actualized version of the tetrahedron. Upon doing various tests with flexibility, physically forcing the tetrahedron in a closed shape and testing the locomotion of the shape when closed it was determined that a new design was required for the objectives to be met. This design failure was mainly due to the excessive rigidity of the tetrahedron shape and lack of continuity, stability and smoothness in rolling.

The new main design was determined to be a rounded dodecahedron which has 12 rounded faces with a pentagonal base, 30 rounded edges, 20 rounded vertices and each interior angle on the pentagonal face of 108°. When doing initial flexibility tests and forced ball shape testing, it was found that the dodecahedron had a smoother ball formation and also a smoother rolling motion along a surface. The more favorable results from initial testing is as a result of the dodecahedrons more planes of symmetry passing through the center, greater number of faces, more points of contact with the ground and a larger dihedral angle of approximately 116.57° to a tetrahedrons 70.53° approximately. The dihedral angle provides information about the angled nature of a shape at the edges where faces meet or the angle between two intersecting planes. Smaller dihedral angles result in

[1] www.onshape.com

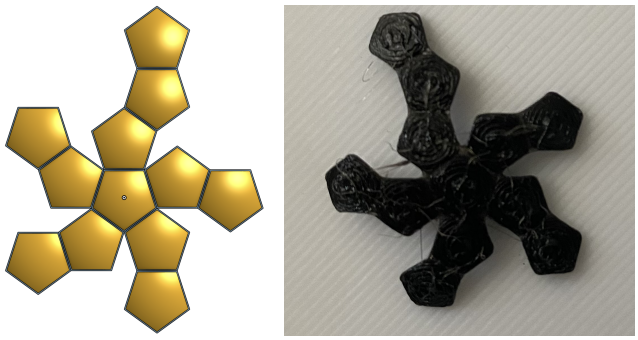


Figure 3.2: 3D CAD model of dodecahedron (Right), 3D printed dodecahedron (Left)

more rigid and sharp bending at the edges whereas larger dihedral angles result in less rigid and sharp bending. These factors develop a better rolling object as the ball formation is less geometrically rigid, smoother and more continuous for rolling. Figure 3.2 depicts the designed 3D printable dodecahedron and the printed dodecahedron models.

The final design was determined to be a rounded dodecahedron with a surface area of approximately 3.45 in^2. Initially the original surface area of the printed dodecahedron was approximately 4.83 in^2, however during testing of flexibility and shape retention the size of the dodecahedrons was reduced. Figure 3.3 shows the changes in size of the printed dodecahedrons over time as we sought to build our ideal solution.

To keep in line with the design constraints, another important factor to consider after noting the shape of the designed particles, is the switching factor or external stimuli that will enact the shape change. Based on the 3D printable nature of the designed particles, the external stimuli chosen to enact shape changes is temperature. A common 3D printed filament, like PLA, is known as a shape memory polymer(SMP), which means the material is capable of returning to an original permanent shape from a deformed temporary shape when exposed to an external stimuli like temperature [36]. For 3D printed objects to posses the ability to deform and return to its original shape we note that authors in [36] and [4] detail what is known as shape memory programming. A polymer can be affected by two main temperatures, transition temperature (T_g) and its melting temperature(T_m). Shape memory programming involves manipulating a polymers transition temperature using either hot or cold stimuli to deform the polymer and then manipulating the temperature once more to allow the polymer to return to its original shape. The transition temperature is the temperature at which the polymer changes from a rigid glassy state to a soft rubbery state, if heated well above its transition temperature it reaches its melting temperature and becomes liquid in state. Figure 3.4 shows the energy vs temperature graph for a poly-

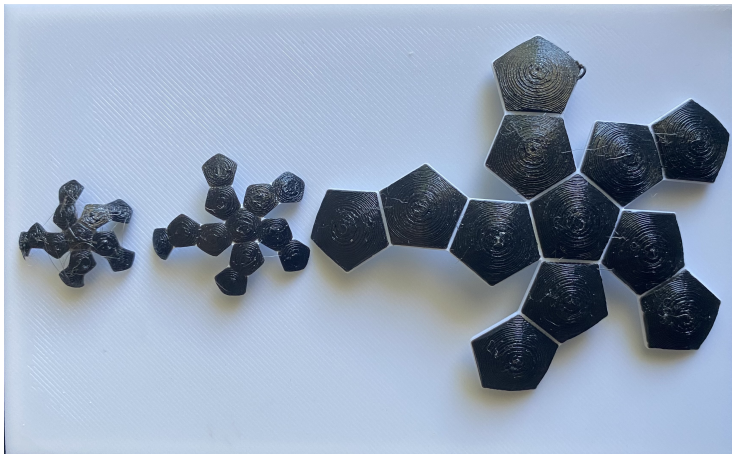


Figure 3.3: Size comparison for 3D printed dodecahedrons with the largest printed dodecahedron with surface area approximately 4.83 in^2 to the furthest right, the second largest printed dodecahedron in the center and the final and smallest printed dodecahedron with surface area of approximately 3.45 in^2 to the furthest left.

mer that is heated beyond its transition and melting temperatures, showing the state it achieves at various stages. A polymer can either be hot or cold programmed depending on the particular application. Hot programming involves heating the polymer above its T_g but below its T_m to obtain a temporary shape while cold programming involves cooling the polymer below its T_g. When the polymer is deformed into its temporary shape and the stimuli is removed the polymer becomes relaxed, rigid and maintains the temporary shape. For the polymer to return to the original permanent shape the polymer should be exposed to either heating or cooling depending on the type of programming involved. Figure 3.5 shows the process of the shape memory programming at the molecular and shape level taken from [4]. It is seen in this figure that once step one is completed which is programming a polymer the temporary shape has molecules which are more relaxed in nature and then once step 2 is enacted by heating or cooling above or below a determined T_g the polymer regains its original shape and the molecules are less relaxed. For PLA the T_g is approximately (60 to 65°C) and its T_m is (173 to 178°C). For TPU T_g is (-50 to -10°C) and T_m is (150 to 230°C). For Soft PLA T_g is (25 to 40°C) and T_m (140 to 170°C).

We seek to implement the hot programming method outlined in [4], to induce the shape change of our designed particle, which will visibly represent switching between two modes. Another implementation from former research done in the paper was found that two main factors can influence the shape retention capabilities of a polymer which are

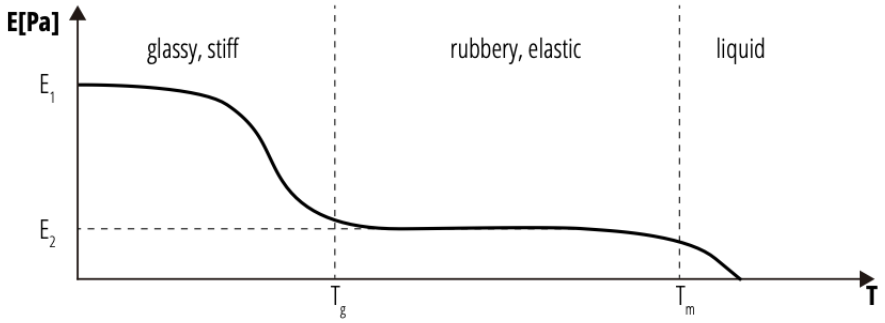


Figure 3.4: Energy vs Temperature graph for the state of a polymer as a polymer is heated above it transition temperature(T_g) and melting temperature(T_m). Source:[4]

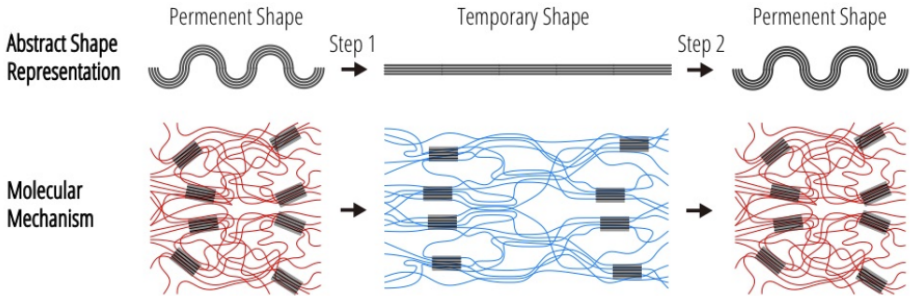


Figure 3.5: The shape representation and the molecular mechanism of a polymer as it undergoes the process of being programmed moving from its permanent shape to a temporary shape then back to its permanent shape. Source:[4]

the type of material/polymer and the 3D printing speed of the printer when printing the material. Findings showed that better shape retention occurred when there was a blend of PLA and TPU layers in the printed shape as well as increasing the printing speed for the PLA layers. In the infancy of this research we 3D printed one tetrahedron and various dodecahedrons of different sizes, with and without a blend of PLA and TPU layering, at different printing speeds and performed cold and hot programming methods to verify the findings and find a working solution for ourselves. For testing the shape memory characteristics we conducted the experiment detailed in Figure 3.6. The figure outlines the cold programming method we used for our 3D printed polyhedrons and two hot programming methods. The methods utilized a commercial freezer for cold programming with temperatures typically ranging anywhere between ≈0°C to -18°C and boiling water for hot programming with temperatures ranging from ≈80°C to 100°C, which is above the T_g but lower than the T_m for PLA material. For the varied prints each polyhedron was printed using either the Original Prusa i3 MK3S+ 3D or the Original Prusa MINI+ printers at the following nozzle temperatures PLA 210-215°C, Soft PLA and TPU 230°C. Table 3.1 shows the results of the cold and hot programming experiments on all prints with varied polyhedrons, polyhedron sizes, printing speeds and layering. The most favorable result with the best shape retention of this experiment was the hot programmed dodecahedron printed at 40% of the original size with PLA, printed at 500% of the original speed on the bottom layer and TPU on the top layers. It should be noted that the height for one single layer for the utilized 3D printers is approximately 0.2mm.

Taking the results from the final row of table 3.1 we see that the dodecahedron can be influenced by hot/boiling water to change its shape from closed to open using the hot programming 1 method described above. This shows that temperature changes influence the designed material and can cause shape changing effects. This form of shape changing however is not enough to allow for the switching between the two modes of open and closed. The shape changing effect described in the results represents direct or one-way shape memory characteristics. Ideally due to our design considerations, we would like the designed solution to not just open but to also have the capability to close. As we seek to close our designed print, the need for additional design and print manipulations are required. From [4], we recognize that printing the desired shape and manipulating the printers characteristics to print the filament in varying directions can greatly influence 3D printed mechanisms ability to close when exposed to stimuli like temperature. Within the paper the authors detail the various shapes of the prints, printing direction/angle, speed and length, for the printed shapes to self fold. Using inspiration from Table 1. Geometric primitives in [4], we take a mixture of actuator layer instructions from the straight and angled fold section to test various methods to get a desired printed shape to close. To

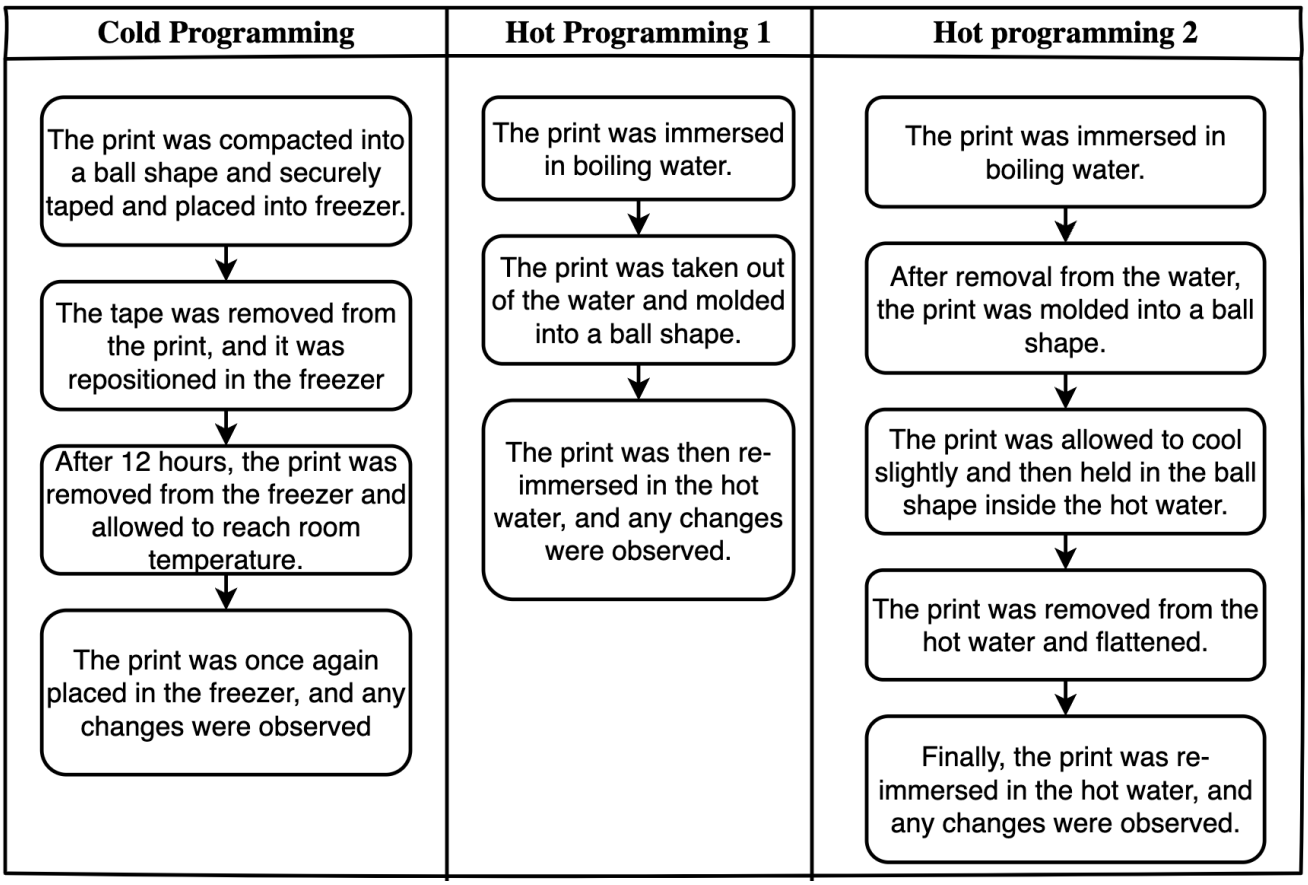


Figure 3.6: Method of cold and hot programming for 3D printed polyhedrons.

Print	Print Speed	CP	HP	Result
Tetrahedron Soft PLA	100	Yes	No	CP - Did not close shape after re-placed in freezer.
Dodecahedron Soft PLA	100	Yes	Yes	CP - Did not re-form full ball shape after re-placed in freezer. All edges were more curved than original. - HP 2: Noticeable curved edges
Dodecahedron PLA	100	Yes	Yes	HP worked as expected. When placed in hot water, the material became flexible and was able to be temporarily shaped into a ball. Shape returned to original flat shape as printed when re-placed in hot water. End result had minimal curved edges.
Dodecahedron TPU	100	No	Yes	Due to its transition temperature, TPU was not cold programmed. HP: No notable shape retention in the print
Double Layer Full Size Dodecahedron	100	No	Yes	More flexible than single-layered PLA. Better shape retention than TPU and soft PLA for HP, however print did not maintain or form a fixed ball shape when placed in hot water.
Double Layer 40% of original size dodecahedron	400	No	Yes (HP1, HP2)	HP1 - Print was successfully hot programmed and reformed its open shape when re-placed in water. HP2- the print had better shape retention than with speed 100 when re-placed in water.
Double Layer 40% of original size dodecahedron	500	No	Yes (HP1, HP2)	HP1: Starting from a balled shape, when placed in hot water the ball formation opened more readily than all previous prints. HP2: Maintains the most curved edges, however, no full shape retention of ball formation.

Table 3.1: Table of results for various 3D printed objects under cold and hot programming.

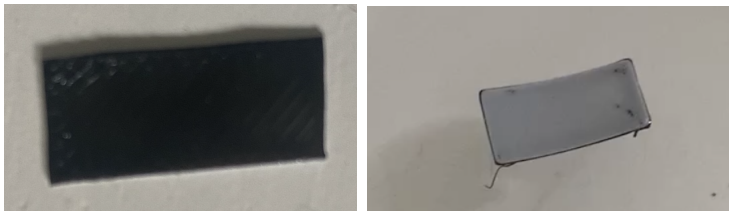


Figure 3.7: Self folding 3D printed rectangular strip, with one layer of TPU and 3 layers of PLA.

demonstrate a simple proof of concept for self folding we printed an angled fold similar to the design in the aforementioned table, a rectangular strip, housing 3 layers of PLA of 0.6mm in height and one layer of TPU of 0.2mm in height. The strip was printed with the PLA and TPU printing orientation of 45°, with all layers being in the same direction. To achieve this feat the following print settings were changed:

- Within the Infill settings of the printer the fill density was changed to 100%
- Within the Infill settings of the printer the fill pattern was changed to Aligned rectilinear
- Within the Advanced Infill settings of the printer the fill angle was changed to 45°.
- Within the Layers and perimeters settings of the printer the layer height was fixed to 0.2mm, Perimeters were set to 1, and Solid Layers Top and Bottom both set to 0.

To determine the self folding capabilities to close, we used hot programming method 1 described within 3.6. Starting from a flat straight state we placed the rectangular strip in boiling water and observed the changes. From figure 3.7 it is observed from the image on the right that the ledges of the rectangle curve over as expected. Though this proves the ability of a printed object to self fold, we seek a more drastic shape closure. To achieve a more drastic fold using other simple shapes we designed another shape, a cut cross-sectioned circle with one layer of TPU and three layers of PLA printed with the same printing instructions as the rectangular strip depicted in 3.8. Similar actions of placing the flat printed cut circle within boiled water and the corresponding Figure 3.9, shows how the shape folded on itself forming a dome-like shape.

Using these results we seek to design a shape that upon interacting with external stimuli closes. The initial proposed design was a cut cylinder printed with PLA and hot programming method number 2 from 3.6 was conducted on the cylinder. Figure 3.10 shows



Figure 3.8: Flat 3D printed circular strip, with one player of TPU and 3 layers of PLA.

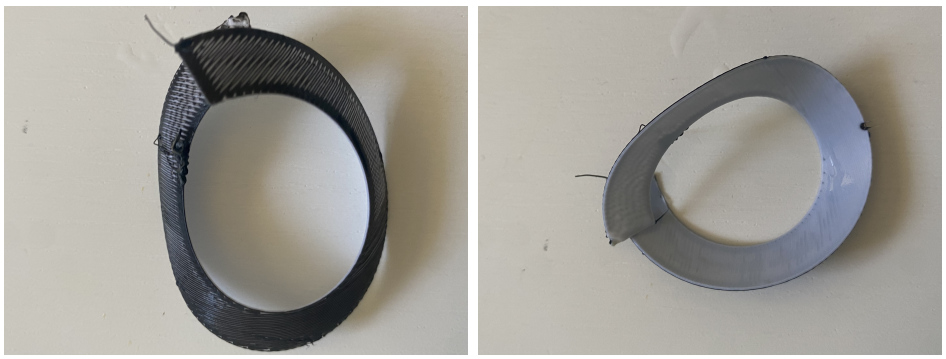


Figure 3.9: Self folding of 3D printed circular strip, with one player of TPU and 3 layers of PLA after being hot programmed.



Figure 3.10: Self folding process of 3D printed cylinder PLA showing the initial print on the left, the straightened print in the middle and the reformed self folded print on the right.



Figure 3.11: Closed 3D printed cylindrical strip made of PLA.

the aforementioned print and the results from the programming process. These results are crucial in our design findings as it now shows that if 3D printed in a closed cylindrical form, a shape can return to it closed form as printed when exposed to an external stimuli. These results motivate us to further study the cylindrical shape and manipulate the structure for our overall bimodal robot units. To further study the cylindrical shape, we then used the 3D printer to print wider cylinder shown in figure 3.11, and conducted the same hot programming experiment as the cylinder shown in 3.10. The results of the experiment were similar in that the cylinder when placed in hot water of $\approx$ 80°C-90°C and then flattened when re-placed in hot water closed and returned it its original shape.

To summarize the findings in this section we note that to achieve bimodality of our designed particle we need a structure that can open and close when exposed to an external stimuli of temperature. The findings above detail that when the double layered dodecahedrons are balled they have capability of opening when placed within boiled water of $\approx$80°C - 100°C. We also notice that when a simple cylinder printed in the closed shape

with PLA, flatted using high temperatures of boiled water between ≈80°C - 100°C , the shape closes when re-placed into the water. Further design decisions then are required to determine what the switching factor will be, in terms of requisite temperatures. If we have two shapes each capable of performing one of the two required modes, the next logical step is to determine how to combine these two designs to achieve one final design that will open and close based on the temperature changes and not nullify each other.

The next section 3.2 describes the two modes that the particles are designed to switch between,detailing the mathematical models for motion.

3.2 Description of modes

This section provides the details for the two modes that each particle should switch between. The details include the motion models for both modes and provides the various equations to supplement the models that will be used in the simulation of the particles in Chapter 4.

The two modes of operation for the particles are *closed* and *open*, with mode 1 being *open* and mode 2 being *closed*. Figure 3.12, depicts both the designed Onshape dodecahedrons and the actual printed dodecahedrons in *closed* and *open* modes.

As previously stated mode 1: *open*, represents a flat 2D object that moves within the environment without specific direction and orientation while mode 2: *closed*, represents a rolling 3D object that moves within the environment in a specific direction and orientation. For modelling purposes the open mode is modelled using Brownian motion that is random in x, y, and θ:

$$\begin{cases} \dot{x} = \delta_{x,1}, \\ \dot{y} = \delta_{y,1}, \\ \dot{\theta} = \delta_{\theta,1} \end{cases} \tag{3.1}$$

where $\delta_{x,1}$, $\delta_{y,1}$, and $\delta_{\theta,1}$ are random Gaussian noise terms representing Brownian motion in each direction.

The motion of the closed mode was modelled using a noisy unicycle:

$$\begin{cases} \dot{x} = v\cos(\theta) + \delta_{x,2}, \\ \dot{y} = v\sin(\theta) + \delta_{y,2}, \\ \dot{\theta} = \delta_{\theta,2} \end{cases} \tag{3.2}$$

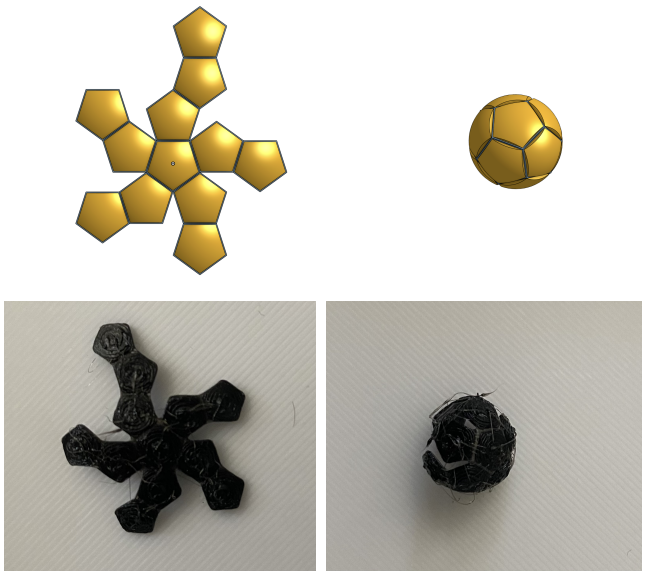


Figure 3.12: 3D CAD models of the dodecahedron model (top) and the actual printed versions of the dodecahedrons (bottom). On the left is Mode 1 (open) and on the right is Mode 2 (closed).

where x and y are position coordinates, θ is the orientation angle, v is the linear velocity, $\delta_{x,2}$ and $\delta_{y,2}$ represents translational noise, and $\delta_{\theta,2}$ is the rotational noise both Gaussianly distributed.

The next section 3.3, specifies how the model parameters were determined for the two modes described above.

3.3 Model Parameters Identification

This section will determine the parameters for both modes to be used in the model simulation for the control of the designed agents.

After designing the bimodal particles and determining the motion models for both modes it is essential to pinpoint the model parameters that will be required to run a realistic simulation. Using the actual 3D printed versions of the dodecahedrons and observing their motion in a controlled environment we seek to obtain the simulation model parameters for MATLAB which are essential to the two modes of the particle. Since both modes utilize a noise factor within its motion model an experiment was designed to correctly

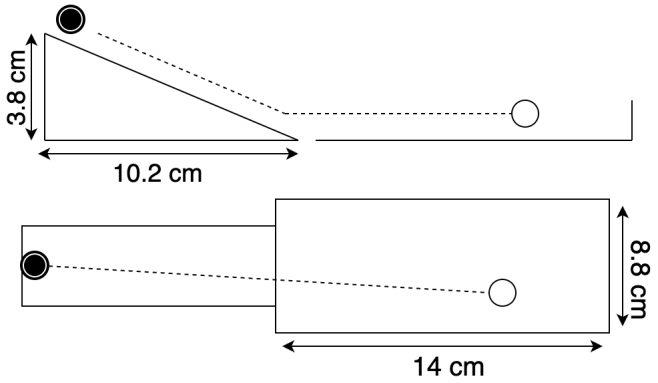


Figure 3.13: Ball and ramp experiment showing the side and top view of the ramp and base where the balled 3D printed dodecahedrons are rolled to determine the model parameters.

model the noise parameters for the motion of the dodecahedron. The main purpose of the experiment is to obtain the model parameters for the translational and rotational noise for the system. Using 10 double layered dodecahedrons and forming them into its mode 2 shape, a ball, we perform a ball and ramp experiment, which captures the rolling motion of the dodecahedrons and are able to use various calculations to determine the noise present in the system. Using mode 1, *open* was not deemed to be necessary to obtain the model parameters required as mode 1 represents limited motion and also motion is random in all its variables. The ball and ramp experiment features a 3D printed wedged ramp, and rectangular base made of PLA along with the 10 dodecahedrons. Figure 3.13 shows the setup, dimensions and expected motion for the ball and ramp experiment setup where the transparent and opaque circular dot represents a single dodecahedron in mode 2, a ball. The method for the ball and ramp experiment is as follows:

1. Set up the ramp and base as shown in Figure 3.13
2. Place the balled dodecahedron at an initial potion denoted as θ_i notably at the center of the ramp.
3. Release the ball from the θ_i and take snapshots of the final position of the ball after 1 second of time has elapsed.
4. Remove the ball from the base and repeat steps 2 and 3 until snapshots of all 10 dodecahedrons have been taken.

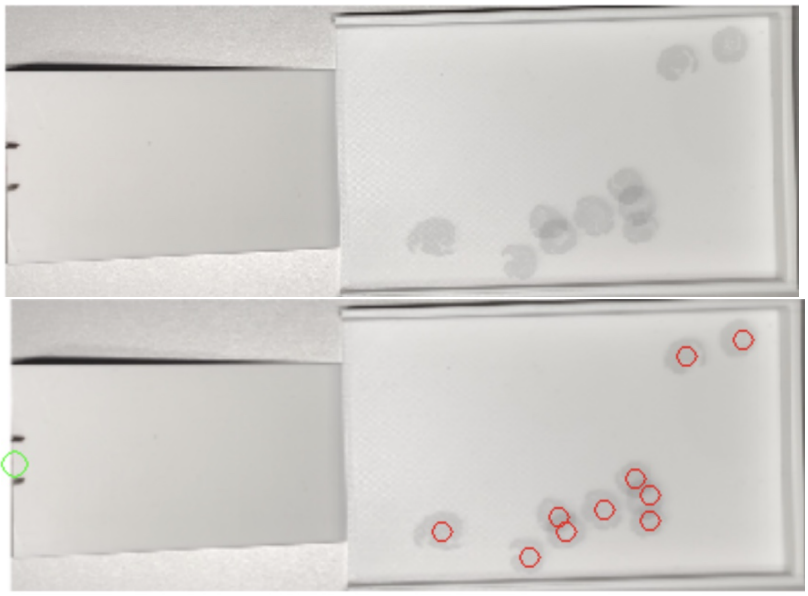


Figure 3.14: Overlayed image of printed dodecahedron particles after 1 second (top). Overlayed image with identified points of selection (bottom).

After the experiment was completed for all 10 dodecahedrons an overlayed image in Fig. 3.14 was created in MATLAB with 10 dodecahedrons in the base. The image features 10 dodecahedrons after 1 second overlayed in the 3D printed base at the top and the bottom image is the same image of the overlayed dodecahedrons with the a green circle at the top of the ramp denoting the initial position of each ball, θ_i, while each red circle depicts the final position of each ball.

Using the image in Fig. 3.14a and selecting the initial point θ_i, and then selecting each balls final position, we then seek to do calculations for a change in theta, $\Delta\theta$, in degrees by calculating the angle between the initial point and each of the final selected points, denoted as $\Delta\theta_n$, where n represents the ball number. This is depicted by the Fig. 3.14b which shows the initial point θ_i with a highlighted green circle and each selected point as a red circle within the bounds. Fig. 3.15 visually shows the angles direction and the change in theta, $\Delta\theta$, between the initial point and one selected final point. To calculate the angular velocity $\dot{\theta_n}$ we use the following equation: $\dot{\theta}_n = \frac{\Delta\theta_n}{\Delta t}$ where Δt equals 1 second. After obtaining a range of $\dot{\theta}'_n s$ for the 10 dodecahedrons we then calculate the sample mean and covariance matrix over all the particles to obtain the translational and rotational noise for each particle. To calculate the translational noise we created a function in MATLAB that generates a matrix where each column represented a random vector sampled from a

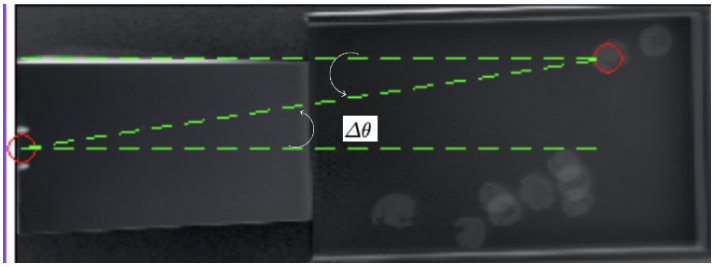


Figure 3.15: Ball and ramp experiment depicting the calculation of $\Delta\theta$, by determining the angle between the initial starting point red circle on the farthest right and the 3D printed dodecahedron's red circle to the farthest left within the base.

multivariate normal distribution with a mean vector X and Y and a covariance matrix. To calculate the rotational noise we calculated the sampled mean and standard deviation of $\dot{\theta}_n$ added to random Gaussian noise. Algorithms 1 and 2 detail the functions used to generate the noise parameters identified. With the creation of these two functions in MATLAB, the dynamics of the particles at the next iteration can be determined by adding the previous particle with position and orientation $p = [x, y, \theta]$, calculating the rotational and translational noise shown below.

```
p = p + [gen_trans_noise(N, p, dt, f, mobility); ...
    gen_rot_noise(N, D_r, dt)];
```

where p is the current position of the point p, N is the number of points, dt is the change in time, f is the repulsion forces and D_r is the rotational coefficient detailed in Chapter 4.

From section we see that, in addition to the printed dodecahedrons we also 3D printed cylinders to experiment with. Similarly we performed the ball and ramp experiment for the closed cylinders and identified their model parameters. Table 3.2 shows the calculated and identified model parameters for both the printed dodecahedrons and the cylinders, where σ, is the standard deviation, $\bar{\dot{\theta}}$ is the mean theta value, $\bar{x}$ is the mean x values and $\bar{y}$ is the mean y values.

In Chapter 4 we use these model parameters to simulate the particles and look to control the particles using the MIPS index for both open and closed loop control. Previous density methods are also reviewed within the section along with a detailed analysis of the MIPS index.

Algorithm 1 Generate Translational Noise

global MODE1
Define the change in time dt, mobility m and repulsion force f variables
Define mean values $\bar{x}$ and $\bar{y}$
Define Cholesky decomposition matrix $cholXY$
Calculate Cosine and Sine trigonometric functions $c = \cos(p(3,:))$ and $s = \sin(p(3,:))$
Calculate Cosine (c) and Sine (s) Squared trigonometric functions and cosine multiplied by sine $c2 = c^2$, $s2 = s^2$, $cs = c \times s$
Generate random Gaussian noise $\delta_1 G$ and $\delta_2 G$
Calculate local X and Y variances $M11$, $M12$, $M21$, $M22$ using Cholesky decomposition matrix and trigonometric functions.
Calculate noise terms δ_1 and δ_2 using the X and y variances and the random Gaussian noise $\delta_1 G$ and $\delta_2 G$.
if MODE1 **then**
    $n = [\delta_1; \delta_2] + m \times f$
else
    STATE $n = [c \times \bar{x} - s \times \bar{y}; s \times \bar{x} + c \times \bar{y}] + [\delta_1; \delta_2] + m \times f$
end if
Translational Noise $= n \times dt$
return Translational Noise

Algorithm 2 Generate Rotational Noise

Define the number of points N, and the change in time dt
Define Mean, $\bar{\dot{\theta}}$ and Standard deviation, σ
global MODE1
if MODE1 **then**
    $\dot{\theta} = 0 + \sigma \times \text{randn}(1, N)$
else
    $\dot{\theta} = \bar{\dot{\theta}} + \sigma \times \text{randn}(1, N)$
end if
Rotational Noise$= \dot{\theta} \times dt$
return Rotational Noise

Shape	**High MIPS (I_max)**	**Low MIPS (I_min)**	$\bar{x}$	$\bar{y}$	$\bar{\bar{\theta}}$	σ
Dodecahedron	0.0375	0	17.621	5.6722	-2.763	6.9727
Cylinder	0.072	0	20.6108	4.9658	-1.1915	6.7946

Table 3.2: Comparison of Identified and Calculated Parameters for Dodecahedrons and Cylinders

Chapter 4

Density Control

This section will detail the control of density of the designed swarm. We will first detail previous experiments that used DBSCAN to quantify aggregation of clusters as a density based control method and then moving toward our solution of using the MIPS Index to quantify aggregation and control the particle swarm using open and closed loop control.

4.1 Density Control using DBSCAN

4.1.1 DBSCAN

Density Based Spatial Clustering of Applications with Noise otherwise known as DBSCAN is a popular clustering algorithm used to identify clusters of points within a given dataset that has no specific shape. This algorithm works by taking three input parameters, epsilon(ϵ), the minimum number of points required to form a dense region (MinPts) and the data set. Taking the parameters the algorithm the selects a random data point from the dataset whilst checking the number of data points that are within ϵ distance from the selected point. After doing this check, the algorithm then seeks to create a core point by checking if the number of points are greater than or equal to the MinPts variable, creating a new cluster. A cluster is formed when the algorithm adds any point within ϵ distance from the identified core point. After creating the cluster the algorithm then adds neighbouring data points that are core points to the same cluster. Any point not identified as a core point and does not have any core points within ϵ neighborhood is considered as noise or an outlier and given a value of -1. The algorithm is completed once every point within

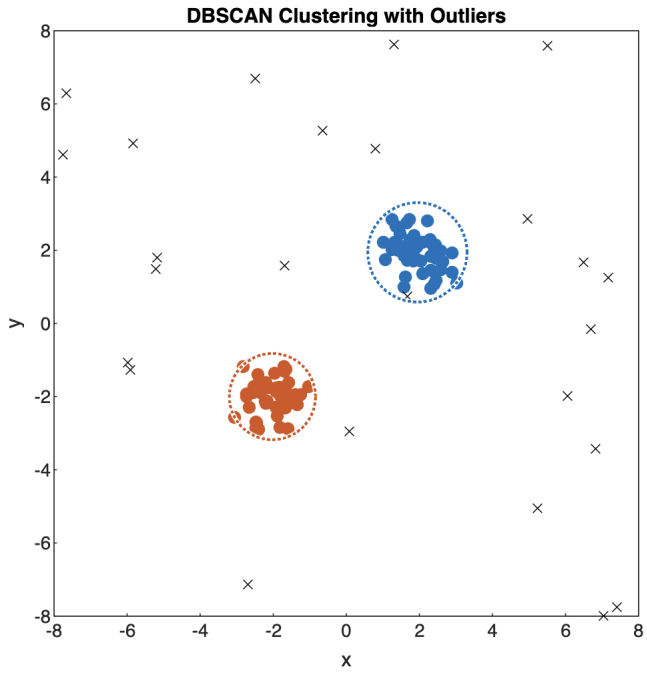


Figure 4.1: Spread of Clusters for 125 points displaying cluster one in red, cluster two in blue and outlier points with an "x".

the given dataset has been assigned to a cluster, marked as an outlier or as noise. Figure 4.1 displays a spread of 125 points within a rectangular space with x and y limits of -8 and 8 in MATLAB. The parameters given to the algorithm had a minimum distance (ϵ) of 0.5 and the minimum points value of 5. The figure shows two clusters clearly defined in colors red and blue enclosed within a circle of corresponding cluster color. Each point within the each cluster has points that are within an (ϵ) of 0.5 of other points within the cluster. Any point not colored by the red or blue is defined with an x showing that they are outlier points and do not belong to a cluster. It is seen that though outlier points are bunched together in groups they do not meet the required minimum point parameter of 5 to be considered a cluster and thus are outside of a cluster. Within the data displayed for the clusters obtained from the function $[cluster, \ ] = dbscan(data, epsilon, minPts)$;, which represents the function in MATLAB used to evaluate DBSCAN of data points, we see that the cluster parameter can either have a value of 1, 2 or -1. Any point evaluated and contained with the red cluster has a value of 1, likewise any point evaluated and contained within the blue cluster has a value of 2 and all points labelled with the x have a value of -1.

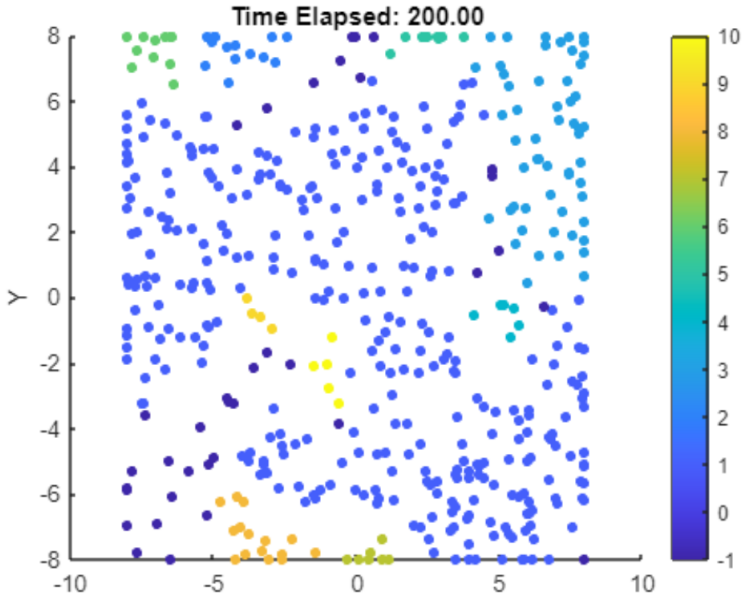


Figure 4.2: Clustering for 500 points after 200 seconds has elapsed using DBSCAN

This algorithm is universally chosen for density based clustering because the algorithm is simple in nature and also it has the ability to identify clusters of varying shapes based on the density of the dataset [60]. DBSCAN has also been proven to discover dense and sparse areas given a specific data set [17]. Figure 4.2 represents the MATLAB graph results of a spread of 500 points having initial random x,y and θ, bounded between -5 and 5 and being moved by the simple kinematic model of a unicycle : $\dot{x} = v\cos(\theta)$, $\dot{y} = v\sin(\theta)$, and $\dot{\theta} = \omega$, where:

- x and y are the coordinates of the points in the 2D plane,
- θ is the orientation angle of the points,
- v is the linear velocity,
- ω is the noise.

The figure shows various colors of points, each of which represents a cluster characterized by the DBSCAN function. Dark blue points however correspond to a value of -1 which represents an outlier point and not a member of any cluster.

To understand further the concepts of DBSCAN with respect to bimodal robots we seek to simulate an open loop experiment to determine firstly switching parameters based on cluster size by varying two parameters velocity and an angle θ, followed by a closed loop experiment which uses cluster-based trajectory following. The aim of the experiment is to study robots, represented by points, that are capable of switching between two modes, changes in velocity and also changes in the rotation angle (θ). The mode switching represents a change in the robots shape within the environment. By switching the modes whilst the points are moving the points form clusters within the environment and the DBSCAN algorithm is then used to identify cluster sizes. The maximum cluster size is then taken and used to thoroughly control the maximum cluster size to follow a predefined trapezoidal trajectory. Trajectory following is useful in that it is a simple method of control which helps to infer data about a robot's state and environment. The algorithm titled Generate and Analyze Clusters in 2D Space 3 details the method for the overall controller we seek to build.

4.1.2 Open Loop Control

To design the open loop controller we utilized the MATLAB software to simulate a dataset of points moving within an environment with a kinematic motion model of a regular unicycle like that of the points shown in Figure 4.2. The design of the controller is such that it follows steps 1-4 in the cluster analysis algorithm 3, by generating points, allowing them to move around and vary velocity and theta depending on the desired mode of operation and plotting the maximum cluster size vs time. Figures 4.3, 4.4 show the Max cluster size vs Time graph results taken from Part 4 of the algorithm 3. Figure 4.3 displays graphs from a 500 point dataset, over a time of 200 seconds, with velocity one equal to 5 and velocity two equal to 0.5. Keeping the theta value constant, the points have a mean of 0 and a standard deviation of 10. The graphs show that at time t=100 seconds all max cluster sizes reduced in value. At time t=100 seconds the mode was switched from mode 1 velocity of 5 to mode 2 velocity of 0.5. This indicates that when the particles are moving slower the maximum cluster size decreased. In Figure 4.4 we depict 4 graphs of max cluster size vs time for 500 points over 200 seconds, varying theta(θ). To vary theta we used the mean and standard deviation of the points, making theta one have a mean of 0 and standard deviation of 35 and theta two having a mean of 0 and standard deviation of 5. Analyzing the graphs it is noted that the max cluster size was greater when theta was equal to theta two indicating that a smaller standard deviation produced larger cluster sizes for this set of data.

In summary for the open loop controller we note that for lower velocity values, the maximum cluster size is reduced while for lower standard deviation values that affect theta

Algorithm 3 Generate and Analyze Clusters in 2D Space

Part 1: Generate Points
Define space boundaries and number of points.
for each point **do**
  Generate random x, y, θ.
end for

Part 2: Simulate Motion
Define max time and Δt.
Determine mode (1 or 2).
if mode 1 **then**
  Update points with velocity 1 for half the time.
  Update points with velocity 2 for the other half.
  Add random noise to θ.
else
  Set constant velocity.
  Update θ to value 1 for half the time.
  Update θ to value 2 for the other half.
end if

Part 3: Cluster Analysis
Define time steps for max clusters.
Compute max clusters at specific times.
Determine cluster sizes using histcounts function.
Find max cluster size.

Part 4: Plotting
Plot Max Cluster Size vs Time.

Part 5: Analysis
Run algorithm multiple times and analyze variations.
Define trapezoidal trajectory path.
Adjust velocity and θ to follow the trajectory.

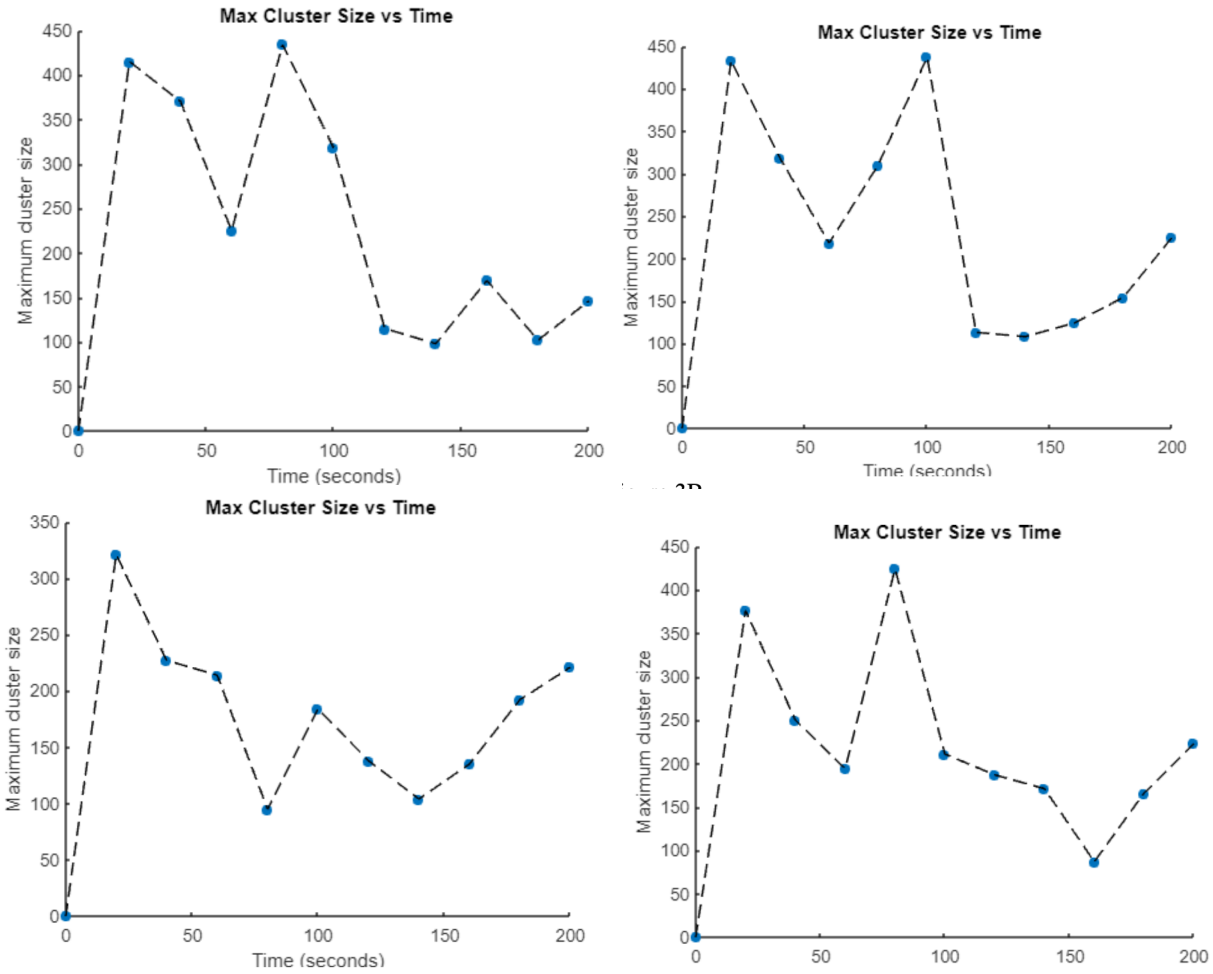


Figure 4.3: Max Cluster Size vs Time for changes in Velocity

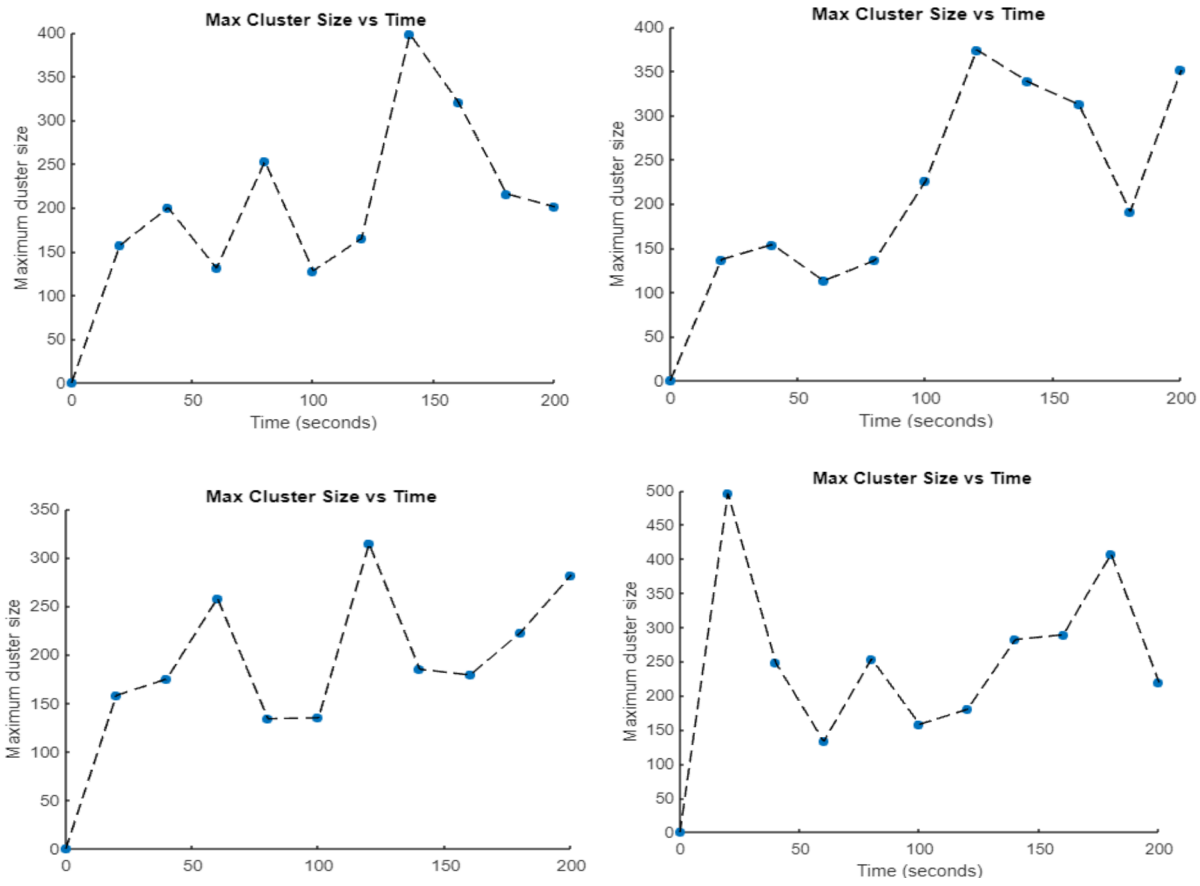


Figure 4.4: Max Cluster Size vs Time for changes in Theta

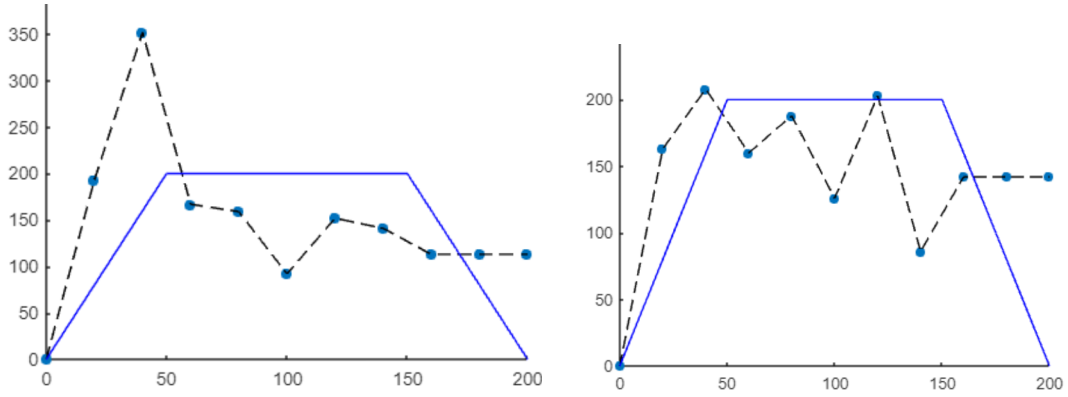


Figure 4.5: Trapezoidal Trajectory following for Max Cluster Size vs Time for changes in Velocity

the maximum cluster size is increased.

4.1.3 Closed Loop Control

Using the open loop results we then seek to perform Part 5 of the cluster algorithm 3. From the analysis of the velocity and theta changes for mode 1 and 2 we then design a controller to vary the maximum cluster size to follow a trapezoidal trajectory. The design for the trapezoidal trajectory was simple by having the max cluster size to rise to 200 in 50 seconds, remain steady at 200 for 50 more seconds, and then falling until the simulation time has elapsed at 200 seconds. Figure 4.5 shows the results for manipulating the velocity switching between velocity 1 and 2 to follow the trajectory. From the previous results we used velocity 1 to rise the max cluster size and velocity 2 to reduce the cluster size. The graph shows great promise as for both simulations the the rise occurred until reaching the 50 second mark and tried to remain steady until 150 seconds had elapsed and varying off slightly until the end of the simulation at 200 seconds. Figure 4.6 shows the results for switching between theta 1 and theta 2 where theta 1 is used to reduce the max cluster size and theta 2 used to increase the max cluster size. Analysing the graphs we note that both graphs rise to the 200 mark of the max cluster size and decrease significantly when the time t = 50 seconds, however it failed to maintain steady at the max cluster size of 200 dipping well under the proposed trajectory line.

These closed loop results show that using bimodal principles of velocity and theta switching we can control clusters of particles to adjust their size to follow a predefined

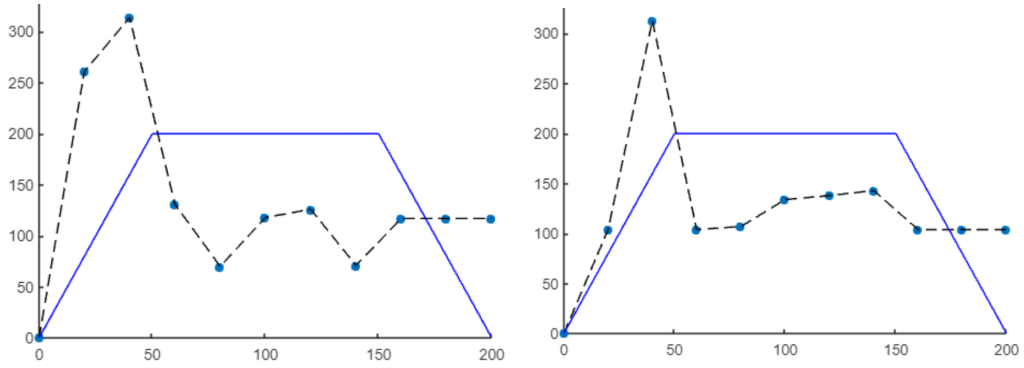


Figure 4.6: Trapezoidal Trajectory following Max Cluster Size vs Time for changes in Theta

trajectory. Although this method is capable of analyzing and identifying clusters we seek to have a more detailed metric to determine the aggregation of points within our simulation. This is why we turn to the metric of the MIPS index to quantify the degree of aggregation for a dataset. In section 4.2 we seek to highlight the advantages of using the MIPS index over DBSCAN, detail mathematical definitions for the MIPS index and display the designed open and closed loop controllers that are operated by the usage of the MIPS index.

4.2 Aggregation Control using the MIPS index

This section details the study of aggregation control as it relates to the metric of the MIPS index.

The term aggregation in the context of our research refers to a set of particles that are grouped together resulting from collisions between members within a particle swarm and between particles and the bounds of a given environment. This section seeks to use the MIPS index to control the aggregation of particles within a swarm. The MIPS index is a metric for determining the degree of aggregation of a swarm [27]. The governing principles behind the MIPS index stems from the physics principle of phase separation, where self propelled particles form varying density regions within an environment. Due to collisions with the environment and with other particles within the swarm, particles that have higher regions of density slow down and form clusters [24]. This aggregation metric is useful as it accurately quantifies and characterizes phase separation in scenarios that are diverse.

Motivation to use the MIPS index stems from previous research done by [27] which validates the usage of the MIPS index over DBSCAN for capturing global density variance.

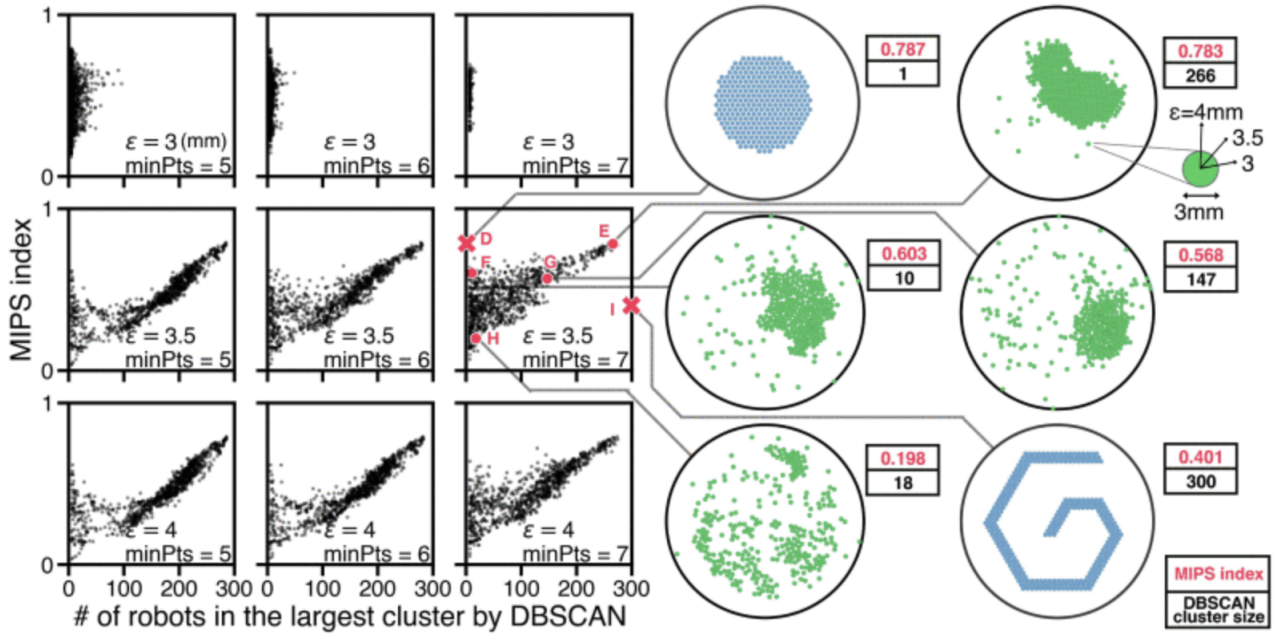


Figure 4.7: Direct comparison of clustering data for robots in the largest cluster for DBSCAN vs MIPS Index. Source: [27]

Figure 4.7 shows a comparison of DBSCAN vs MIPS index generated in a CUDA simulation, for position data with 300 robots. Varying the input parameters of ϵ and Minimum Points(minPts) for the DBSCAN algorithm calculations of clustering were performed and analysed and for the same swarm the MIPS index was calculated also and analyzed and compared. The results show a selection of green points which represent regions of high, medium and low MIPS indices with the DBSCAN results to the right. The image also presents two extreme scenarios which show that for a large cluster of points the DBSCAN result is 1 verses the MIPS index value of 0.787 which would correspond to a high density region and secondly a scenario where the cluster forms an irregular shape similar to that of a snake. The DBSCAN result characterizes this with a high density region giving a result of 300 whereas the MIPS index value has a result of 0.401, indicating the region is not as dense as the DBSCAN result presented. These two extremes show us that for more diverse scenarios the MIPS index is a better quantifier for the level of aggregation present within a cluster of points for a simulation. For this reason we seek to use the MIPS index to not only quantify the extent of the aggregation for our designed particle swarm but also to build a controller for the designed swarm using the MIPS index.

4.2.1 MIPS Index

This section will describe the mathematical and other findings of [27] as it relates to the MIPS index and provide a suitable framework for how the MIPS index will be used for the aggregation control of the designed particle swarm.

Looking further into the MIPS index and how it operates as an influential factor to the variation of density for a swarm of particles we notice the term motility characteristics. Motility characteristics are attributes that influence the motion of a particle. Motility characteristics of particles can be varied to examine the clustering and extent of aggregation of a swarm, quantifying them using the MIPS index. In [27], we note that a suitable property to affect motility is the activity parameter. Activity quantifies the motion of a particle relative to another. It is defined by a the particles radius, velocity and rotational diffusion coefficient. The activity parameter is represented by the following equation:

$$A = \frac{v_0}{2rD_r}, \tag{1}$$

where A is the activity parameter, v_0 is the velocity, r is the radius and D_r is the rotational diffusion coefficient. Velocity V_0, represents the particles speed in a given direction linearly. The radius r is the radius of the particle which details the physical size or interaction range of the particle. The rotational diffusion coefficient D_r quantifies the changes of a particles orientation and directional movement due to random rotational movements. Controlling a particle swarm using the MIPS index can be achieved by performing a combination of varying the activity and velocity of a particle or by controlling the rotational diffusion coefficient directly. To control a particle swarm using D_r, directly it is important to understand how this parameter affects the swarm. If a swarm has a higher D_r, this will enable the particles to separate more easily from its clusters and break formations with greater ease. This phenomena if able to be controlled for a particle swarm would affect the motion and aggregation of the particles, which in turn affects the value of the MIPS index. Pairing the rotational diffusion coefficient with rotational and translational parameters which affect the motion of of a particle can allow us to control a particle swarm.

To obtain and understand values for the MIPS index, we look further into how the MIPS index is calculated from [27]. For interrobot collisions, it is understood that robots slow down and form high-density aggregate formations. Considering the local density characterized by a smooth field for a square region centered around a grid point (p, s) with a specific side length w known as the window size $W(p, s, w)$, the local density is characterised by the following equation:

$$[\lambda_w(t)]_{p,s} = \frac{|\{z_i(t) : z_i(t) \in \mathcal{W}(p,s,w)\}|}{|\mathcal{W}(p,s,w)|} \tag{2}$$

.

For simplicity the numerator of this equation represents the total number of robots within the given window size and the denominator is the window area. The MIPS index is a novel metric for its counterparts in that it requires no posteriori information and is also independent of the given window size. Using the local density calculations in (2) we then characterize the variance of the density distribution as

$$\begin{aligned}\mathrm{Var}(\lambda_w) =& |\mathcal{D}|^{-1} \sum_{(p,s)\in\{0,1,\ldots,l-1\}^2} [\lambda_w]^2_{p,s} \mathrm{d}p\mathrm{d}s \\ &- \left(|\mathcal{D}|^{-1} \sum_{(p,s)\in\{0,1,\ldots,l-1\}^2} [\lambda_w]_{p,s} \mathrm{d}p\mathrm{d}s \right)^2 \end{aligned} \tag{3}$$

where $\mathrm{Var}(\lambda_w)$ is the variance of the function λ_w given in 2. This calculates the difference between the mean square values and the square of the mean value of the function for a 2D region $|\mathcal{D}|$, which is the area of the set $\mathcal{D} \subset \mathbb{R}^2$.

The next stage is to calculate an average of the variance of densities over an interval of window sizes $[w_{min}, w_{max}]$ making the resulting metric independent of the window size.

$$I = \frac{\int_{w_{\min}}^{w_{\max}} \mathrm{Var}(\lambda_w) \mathrm{d}w}{w_{\max} - w_{\min}} \tag{4}$$

The final value for the MIPS Index is then calculated by normalizing I, in equation (4) to correspond to values between 0 and 1.

$$\text{MIPS index} = \frac{I - I_{\min}}{I_{\max} - I_{\min}}, \tag{5}$$

where $I_{\max}$ and $I_{\min}$ correspond to the highest and lowest possible density distribution of the particles respectively.

Figure 4.8 shows how the MIPS Index is computed using convolutions with varying window sizes with different density fields, depicting the calculations presented in equation 2. After which the variances of the density fields are computed and using the variances the

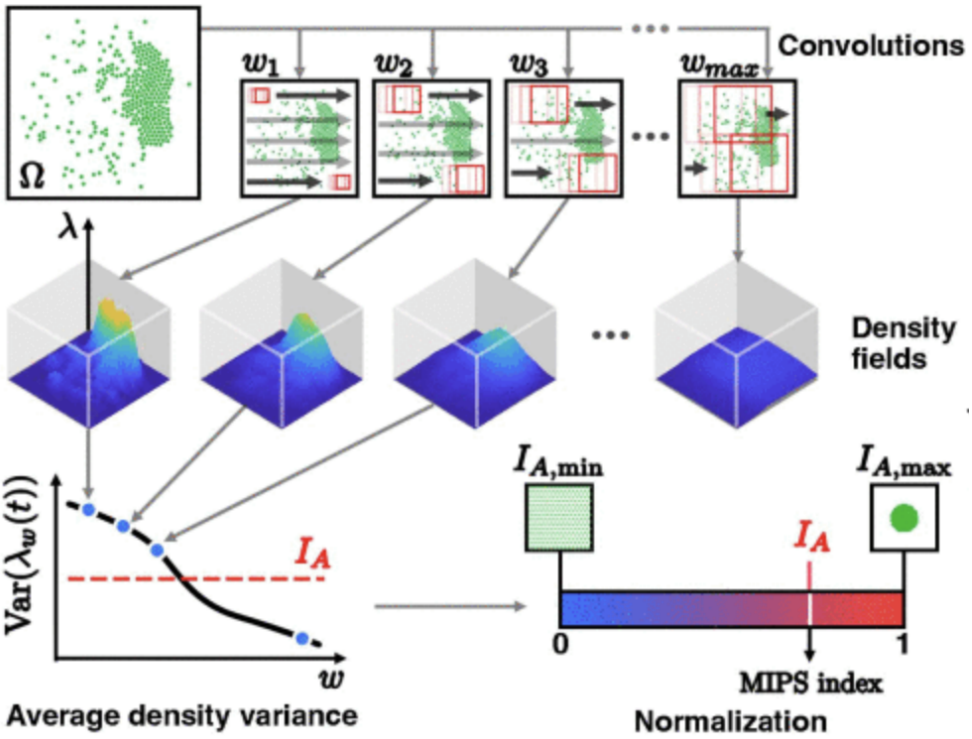


Figure 4.8: MIPS index computation using the convolution of varying window sizes, computation of variances of density fields and normalized variances over the window sizes. Source: [27]

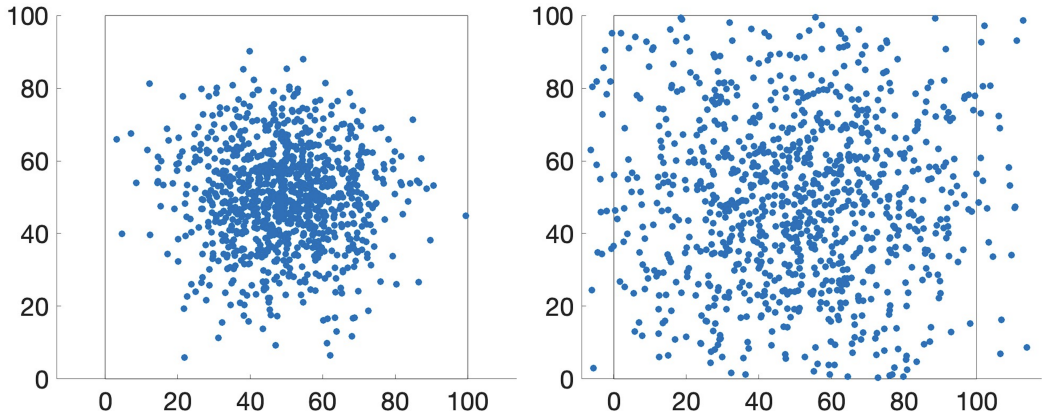


Figure 4.9: Clustering for 1024 particle points in MATLAB with corresponding MIPS index values of 0.89 on the left and 0.19 on the right.

MIPS index is calculated using the normalized average of the variances over the window sizes.

In general the clustering of the particles and their corresponding MIPS index can be visualized by Fig 4.9, which shows clustering for two sets of 1024 particle points that are clustered closest to the center of a 100x100 region in MATLAB. The left graph displays the 1024 particles heavily clustered within the center of the region while on the right the graph displays a more scattered dispersion of the 1024 particles with even some being outside of the created region bounds. This difference in clustering is not only visual but seen in the corresponding MIPS index values for each graph. The graph on the left corresponds to a MIPS index value of 0.89 which is logical as most of the points are closely clustered together within the center. The corresponding MIPS index value for the graph on the right is 0.19, which accurately portrays the dispersion of the particles and less clustering in general. It should be noted that these particles are allowed to overlap each other and have no accommodation for repulsive forces that are taken into account during the actualized MATLAB simulation and figures presented in Fig 4.10.

To characterize and calculate the MIPS index for our designed swarm of particles we now look to a simple experiment to quantify our $I_{\max}$ and $I_{\min}$. Consider 1024 particles of radius 1 within a 100×100 square region, designed with the parameters identified in Chapter 3 and simulated in MATLAB. Figure 4.10 shows the resulting density distribution which shows the particles having varying levels of distribution. The graph displayed on the the left depicts a uniformly distributed set of particles corresponding to a MIPS index of 0 and characterizing $I_{\min}$, while the graph on the right displays a densely distributed set

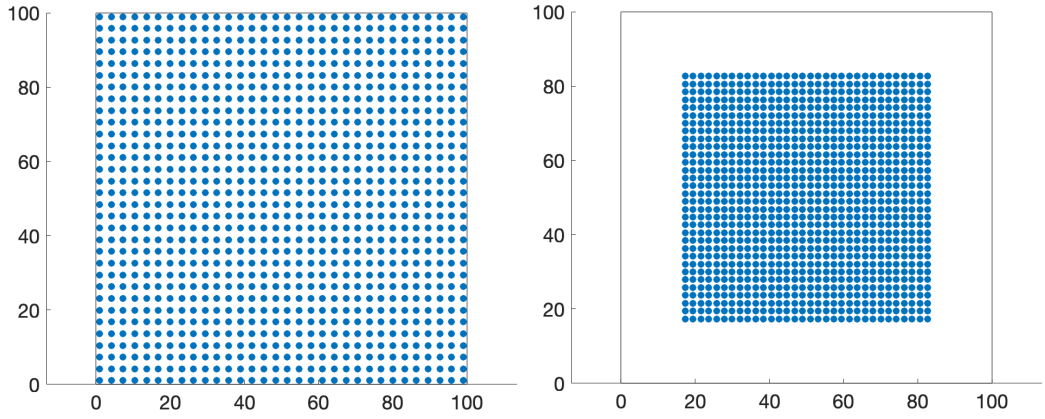


Figure 4.10: Density distribution for a collection of 1024 particles of radius 1 generated in MATLAB with uniformly distributed collection corresponding to a low MIPS index of 0 (left) and densely distributed collection corresponding to a high MIPS index of 1 (right). Source [48]

of particles corresponding to a MIPS index of 1 and characterizing $I_{\max}$. Calculations in MATLAB resulted in $I_{\max} = 0.3097$ and $I_{\min} = 0.0037$. This indicates that if the received MIPS index value of the aggregation is equal to $I_{\max} = 0.3097$, then the metric will produce a 1 and if the calculated MIPS index is equal to $I_{\min} = 0.0037$ the MIPS index value will be 0. The method to calculate and characterize the $I_{\max}$ and $I_{\min}$ are detailed in the algorithm pseudocodes 4 and 5.

Algorithm 4 Obtaining $I_{\max}$

1: Generate 32 linearly spaced points l between 1 and 99 using LINSPACE
2: Create a grid p of all possible combinations of points in l using COMBVEC
3: p is a 2x1024 matrix, where each column represents a point in 2D space
4: Calculate a new set of points b by scaling p by 2/3, shifting it, and storing the result in b
5: **for** $i \leftarrow 1$ to 1024 **do**
6: Set the configuration of the MIPS object to the i-th pair of points in b
7: Calculate the MIPS index and store the value in $I_{\max}$
8: **end for**
9: Plot the points in b as a scatter plot within a region defined by *region* and *bounds*

The subsection bellow will explore the use of these results to control the designed particle swarm.

Algorithm 5 Obtaining $I_{\min}$

1: Generate 32 linearly spaced points l between 1 and 99 using LINSPACE function
2: Create a grid p of all possible combinations of points in l using COMBVEC function
3: $p \leftarrow p$ as a 2x1024 matrix, where each column represents a point in 2D space
4: **for** $i \leftarrow 1$ to 1024 **do**
5: Set the configuration of MIPS object to the i-th pair of points in p
6: Calculate the MIPS index and store the value in $I_{\min}$
7: **end for**
8: Plot the points in p as a scatter plot within a region defined by *region* and *bounds*

4.2.2 Open Loop Control

As previously stated one of the aims of this thesis work is to control a swarm of bimodal particles using the MIPS index. With the aim of using a direct control of D_r the rotational diffusion coefficient by controlling the rotational and translational noise identified in Chapter 3. To simulate the use of the identified parameters, we created two open loop controllers to determine the switching capabilities of the particles. The two open loop controllers are used to determine the exact capabilities of mode 1 and mode 2 to determine what levels of aggregation is achieved by the swarm overtime. To determine the mode operations for the open loop controller we simulate a swarm of 1024 particles over 200000 iterations in MATLAB. All simulations initially begin with the particles randomly distributed and then are analyzed after applying the motion models outlined in Chapter 3. In this simulation we utilized the mean of the particles overtime to determine the translational noise of the particles, with the mean value to zero in mode 1, *open*, see algorithm 1.

Algorithm 6 details the general outline of the code for obtaining the open loop results of mode 1 and mode 2. Within the algorithm we see that I_{max} and I_{min} are calculated from the dataset of obtained MIPS indices to normalize the received MIPS index. This calculation takes the maximum and minimum value obtained from the entire dataset of MIPS index values for the entire simulation. This is to account for the received negative MIPS indices when the points were sparsely distributed and had a lower I_{min} that the original calculated I_{min} displayed from 4.10.

Figure 4.11 displays the results of the open loop controller for mode 1 on the top and mode 2 below for dodecahedrons. By analyzing these two graphs we see that in mode 1 the average MIPS index aggregation was calculated to be 0.36 and for mode 2, 0.49 represented by the blue dotted line. These aggregation values show that for the respective I_{max} and I_{min} calculated the average actual MIPS index values for mode 1 were not close to the largest

Algorithm 6 Open Loop Algorithm

1: Initialize variables and parameters $SPEEDUPFACTOR$
2: Set $MODE1 = true$ or $MODE1 = false$
3: Update the state of the points by obtaining the dynamics from the dynamics function which utilizes algorithms 1 and 2
4: **if** $mod(iterations, SPEEDUPFACTOR) == 0$ **then**
5: Set the configuration of MIPS object
6: Calculate the MIPS index
7: Display values
8: **end if**
9: Calculate the I_{max} and I_{min} from MIPS index values generated
10: Calculate the normalized MIPS index
11: Calculate the average MIPS index from normalized MIPS index values
12: Update plot
13: Plot Normalized MIPS Index vs iterations

possible aggregation that could be achieved, while for mode 2 the aggregations were closer to the highest possible value the particles were able to achieve. By observing the graphs closely we notice that the graph displayed by mode 1 shows minimal oscillatory spikes but steadily decreases overtime and then oscillates towards the end while the graph displayed by mode 2 extremely oscillatory in nature having very sharp spikes throughout. Other differences between the two modes were found that mode 1 obtained its highest MIPS index value at 0.46 whilst mode 2's highest MIPS index value was 0.8. The difference in the nature of the graphs tells us that for mode 2 the particles within the simulation were moving rapidly, forming and breaking clusters swiftly while for mode 1 the steady decrease in the index for more than half of the entire simulation suggests that the particles were moving much slower. The differences observed support the motion models of both modes where mode 2's motion of rapid movement mirrors that of the noisy unicycle representing a ball formation and mode 1's motion resembles that of a Brownian particle where the motion is random and the accompanying shape is flat. The open loop controller shows that we can possibly rise the MIPS index values by putting the controller in mode 2 and reducing the MIPS index value by putting the controller in mode 1. Similarly in Figure 4.12 for the printed cylinders we note resembling looking graphs to that of the dodecahedrons with minor differences. For mode 1, we notice the average MIPS index was 0.38 whilst for mode 2 the average MIPS index was 0.57. These values also support a similar notion to that of the dodecahedrons where mode 2's average MIPS index was higher than that of mode 1, and also mode 1 achieved its highest MIPS index value of about 0.5 to mode 2's 0.8. Mode

2 for the cylinder shows that the particles were rapidly forming and breaking clusters while mode 1 for the cylinder shows a more stable and steady decrease for more than half of the simulation and then having some oscillations towards the end of the simulation. The table 4.1 below shows a direct comparison of the results for the simulations mentioned above for the dodecahedrons and the cylinders.

Print Type	**Mode 1**			**Mode 2**		
	I_{max}	I_{min}	**Average MIPS index**	I_{max}	I_{min}	**Average MIPS index**
Dodecahedron	0.08	0	0.36	0.08	0	0.49
Cylinder	0.08	0	0.38	0.08	0	0.57

Table 4.1: Table showing differences for calculated values of I_{max}, I_{min} and the Average MIPS index in both modes based on the normalized MIPS indices for the dodecahedron and the cylinder.

Using these insights we can then design a closed loop controller, allowing particles to follow desired aggregation at the specific MIPS index value by switching between mode 1 and 2 for the dodecahedrons and the cylinders.

4.2.3 Closed Loop Control

The open loop control provided insight to determining which modes are suitable for increasing and decreasing the MIPS index for the particles. We now use these insights to design a closed loop bang-bang type controller capable of switching between two modes to follow a specified MIPS index value trajectory. The design of the controller takes into account the results that to achieve higher MIPS indices for the specified iterations, we should use mode 2 and to achieve steady decreases in the MIPS index the better option is mode 1. This conclusion is drawn from the difference in the mode 1 and 2 averages over the iterations showing that mode 2 had a greater overall MIPS aggregation value for its average than for mode 1. The closed loop controller design is to have the normalized MIPS index be increased to a desired MIPS index target value. The particles should then maintain that desired MIPS index for the duration of the simulation. The bang-bang nature of this controller is such that if the MIPS index falls below the desired value the controller should increase the MIPS index by switching its mode and once above it should decrease

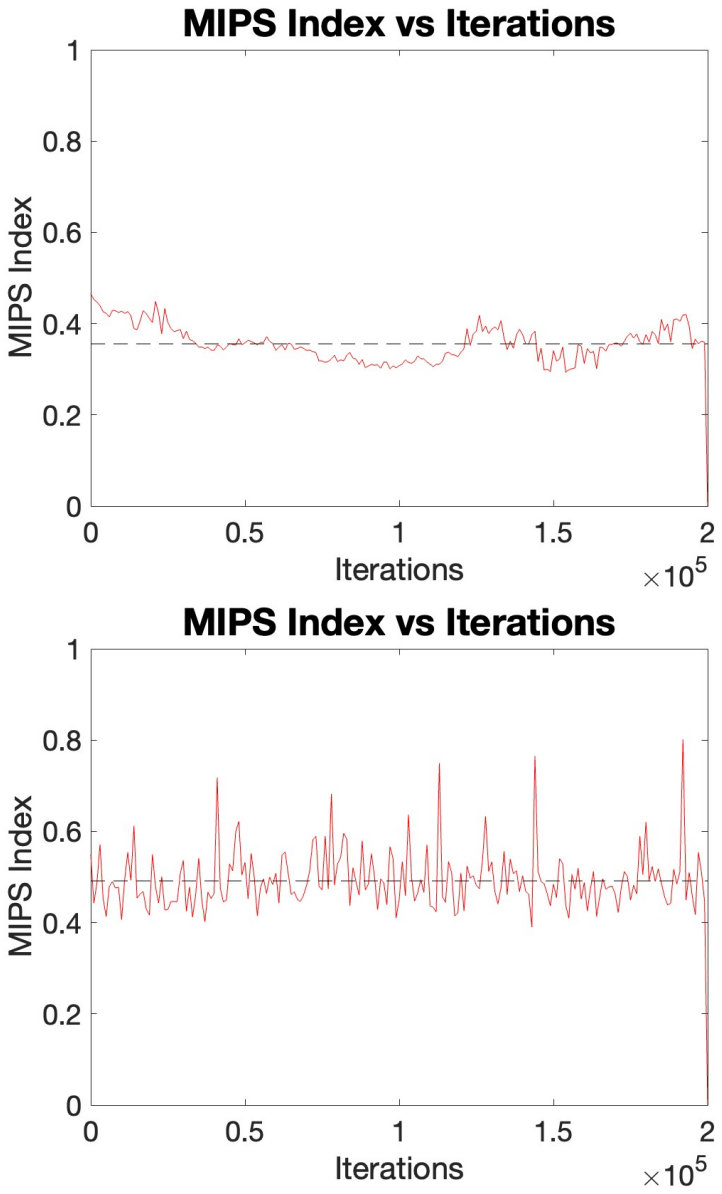


Figure 4.11: Graph showing the MIPS index vs the number of iterations for a simulation of 1024 dodecahedron particles where mode 1 (open) is represented on the top and mode 2 (closed) represented on the bottom.

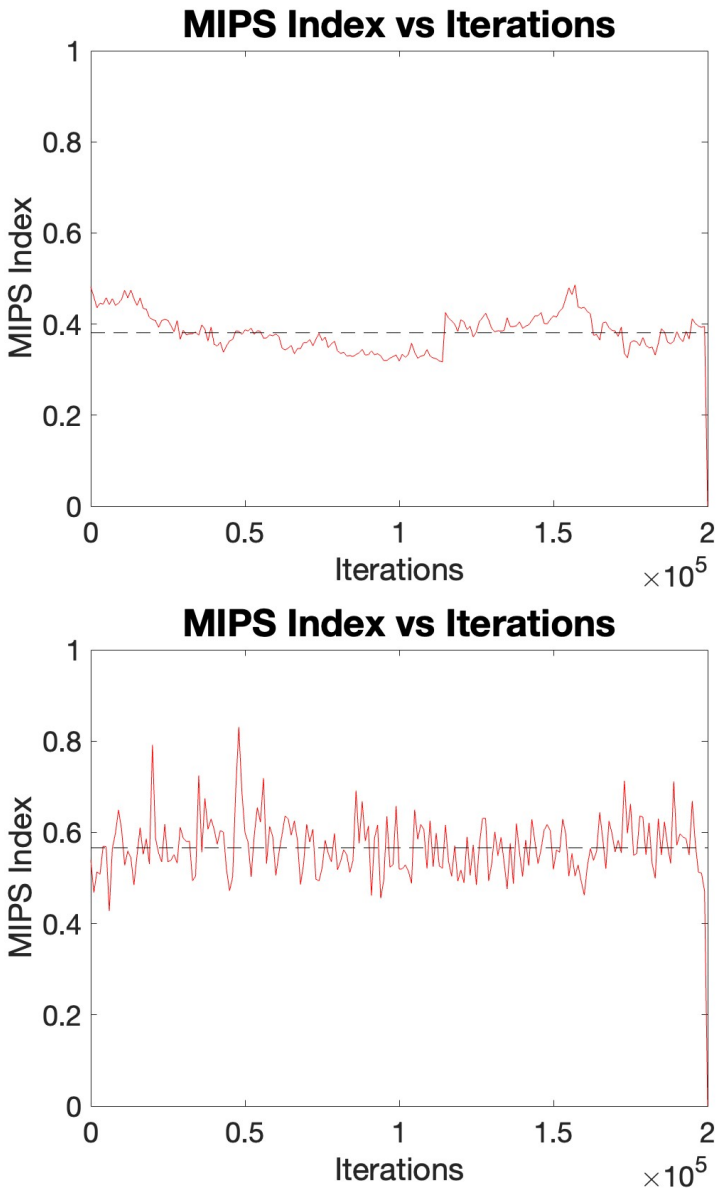


Figure 4.12: Graph showing the MIPS index vs the number of iterations for a simulation of 1024 cylinder particles where mode 1 (open) is represented on the top and mode 2 (closed) represented on the bottom.

the MIPS index by switching to a different mode. Algorithm 7 shows the outline of the design of the controller with a desired target value.

Algorithm 7 Closed Loop control for MIPS index

```
Initialize variables and parameters
for each iteration from 1 to the maximum number of iterations do
    Update the state of the system using the dynamics function with the current state and parameters
    if the current iteration is a multiple of the speed-up factor then
        Set the configuration of the MIPS calculation object using the current state
        Calculate the MIPS index
        Normalize the MIPS index to be between 0 and 1
        Increment the index for storing MIPS values
        Calculate the average MIPS index up to the current iteration
        Display the current MIPS index and the average MIPS index
        if the normalized MIPS index is less than a minimum target value then
            Set the mode to Mode 1
        else
            Set the mode to Mode 2
        end if
    end if
    Update the plot with the new state data and refresh the display
end for
Plot MIPS index values vs the iteration numbers
```

To display the results of the algorithm we performed two closed loop controllers and designed the desired MIPS index target value based on the average of the MIPS index for the dodecahedron and cylinder. For the dodecahdron the target MIPS index value was determined to be 0.5, while for the cylinder the target MIPS index value was determined to be 0.6. These chosen values suggest that for both simulations we expect the graph to show sharp peaks for switching motions around the chosen target values. Figure 4.13 shows the results of the algorithm displaying a graph showing the designed controller that attempts to maintain a MIPS index of 0.5. The controller started in mode 2 resulting in a quick rise in the MIPS index, then once the particles were allowed to move we observed the changes. We notice that if the controller reduces below 0.5 then the graph instantly has a sharp increase in its MIPS index, indicating a switch of mode which increases the value of the MIPS index and then switches back once lowered below 0.5 again. The oscillatory nature of the graph shows the controller switching between mode 1 and mode 2 to maintain the



Figure 4.13: Graph showing the MIPS index vs iterations of the dodechedron's closed loop controller that switched between mode 1 and mode 2 at a desired MIPS index value of 0.5 (blue dashed line).

target value, whilst having an average MIPS index value of 0.5 which is indicated on the graph with a black dotted line. We note that the average and the target value were the same indicating that the designed closed loop controller for the dodecahedrons was capable of allowing the particle swarm to maintain a desired MIPS index over a period of time.

Figure 4.14 shows the results of the closed loop controller for the designed cylinders that tries to maintain its MIPS index at a value of 0.6. By observing the results we see that the MIPS index rises above 0.6 and then has a sharp decrease in its MIPS index followed by a rise above 0.6 again. This oscillatory nature of the graph shows that each time the MIPS index was lowered below 0.6 the mode was switched. The average MIPS index for this simulation was determined to be 0.7, just above the desired MIPS index target value, but can be expected due to the nature of the controller. These results also show that for the printed cylinders with the identified model parameters it is possible to design a simple bang-bang like controller to control the MIPS index values of a swarm of particles moving

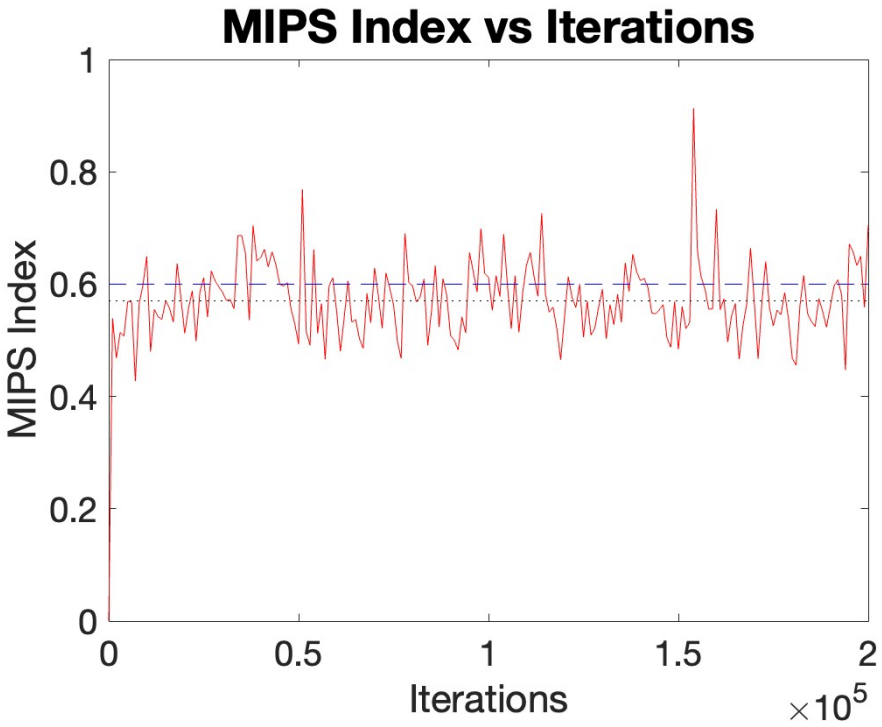


Figure 4.14: Graph showing the MIPS index *vs* iterations for the cylinder's closed loop controller that switched between mode 1 and mode 2 at a desired MIPS index value of 0.6 (blue dashed line).

within a simulated environment.

Both of the designed controllers demonstrates significant potential for the control of bimodal particles using the MIPS index.

Chapter 5

Conclusion and Future Work

This thesis has expanded on the works of [27] and [48] by presenting the design and control of a swarm of bimodal particles using the MIPS index. The aggregation control metric was utilized by controlling motility characteristics directly thus influencing the clustering nature of particles which in turn influenced the MIPS index. This thesis explored the field of swarm intelligence, swarm robotics, morphing matter and aggregation control to aid in the successful creation of a bimodal particle swarm that was capable of being controlled. The bimodal particles were designed to change its shape based on temperature changes within the environment, between an open and closed mode representing rolling and being flat. The design of the particle swarm was a 3D printed dodecahedron using a combination of layering of PLA and TPU to increase its shape retention capabilities. Using principles from morphing matter we were able to design a particle that was simple, electronic-free, and capable of being biodegradable having the capabilities to be opened or closed separately. By conducting the ball and ramp experiment we sought to identify the parameters required to build the open and closed loop controllers. To control the particles we sought to control the rotational diffusion coefficient D_r directly thus affecting the kinetics and clustering of the particles. The clustering of the particles was then successfully quantified using the aggregation metric of the MIPS index. We also designed open and closed loop controllers from the identified parameters with both closed loop controllers being equipped with the ability to successfully switch its modes between 1 and 2 representing open and closed modes of operation. The switching control opens doors for swarm monitoring schemes that are capable of giving feedback to the user by its shape formations and possible aggregation over time. This research brings contributions to the field of swarm robotics as to the best of our knowledge is the only controller that utilizes switching control of particles in a swarm by directly controlling the parameters directly affecting the motion of the particles

thus affecting the MIPS index. It also makes contributions to studies involving morphing matter as it provides another geometric design of a robot that maintains ecological safety, simplicity and repeatability.

The following list show future additions to this work that could result in an all-encompassing and real world result:

- Adjusting the design of the bimodal particle by combining the two designs of the dodecahedron and the cylinder, without nullifying their actuation effects on the particle.
- Experimenting with the designed particles ability to open and closed using natural sources of heat such as the sun.
- Designing the particles in a more dynamic way to apply to environmental scenarios where the ground is unknown and also complex.
- Deployment of the designed swarm of particles in a known environment and observing their behavior over a period of time.
- Further research into printing the particles from biodegradable and compostable materials and after deployment, observing the changes over time.

Future applications from this work could be environmental monitoring scenarios which can use temperature sensitive bimodal particles and their aggregations in various modes to determine temperature variations on land or oceans and lakes. Another useful scenario for this type of particle swarm behavior is in robotics where simple robots can be created to mimic these shape changing abilities using bio-degradable materials for camouflage adaptive tasks. By designing robots to adopt these bimodal particle swarm behaviors the applications can extend to biomedical engineering, material science and more complex environmental monitoring scenarios.

References

[1] Laith Abualigah, Deborah Falcone, and Agostino Forestiero. Swarm intelligence to face iot challenges. *Computational Intelligence and Neuroscience*, 2023(1):4254194, 2023.

[2] A. Ailon and I. Zohar. Control strategies for driving a group of nonholonomic kinematic mobile robots in formation along a time-parameterized path. *IEEE/ASME Transactions on Mechatronics*, 17(2):326–336, April 2012.

[3] Dario Albani, Joris IJsselmuiden, Ramon Haken, and Vito Trianni. Monitoring and mapping with robot swarms for agricultural applications. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017.

[4] Byoungkwon An, Ye Tao, Jianzhe Gu, Tingyu Cheng, Xiang'Anthony' Chen, Xiaoxiao Zhang, Wei Zhao, Youngwook Do, Shigeo Takahashi, Hsiang-Yun Wu, et al. Thermorph: Democratizing 4d printing of self-folding materials and interfaces. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–12, 2018.

[5] Ross Arnold, Kevin Carey, Benjamin Abruzzo, and Christopher Korpela. What is a robot swarm: A definition for swarming robotics. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pages 0074–0081, 2019.

[6] Levent Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[7] C. Bechinger, H. Löwen, C. Reichhardt, G. Volpe, and G. Volpe. Active particles in complex and crowded environments. *arXiv.org*, 2016. Available online: https://arxiv.org/abs/1602.00081.

[8] Giovanni Beltrame, Ettore Merlo, Jacopo Panerati, and Carlo Pinciroli. Engineering safety in swarm robotics. RoSE '18, page 36–39, New York, NY, USA, 2018. Association for Computing Machinery.

[9] G. Beni and J. Wang. Swarm intelligence. In *Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, 1989.

[10] Michael Bodi, Ronald Thenius, Martina Szopek, Thomas Schmickl, and Karl Crailsheim. Interaction of robot swarms using the honeybee-inspired control algorithm beeclust. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):87–100, 2012.

[11] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, et al. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[12] Marco Carpentiero, Luca Gugliermetti, Marco Sabatini, and Giovanni B. Palmerini. A swarm of wheeled and aerial robots for environmental monitoring. In *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, pages 90–95, 2017.

[13] Michael E. Cates and Julien Tailleur. Motility-induced phase separation. *Annual Review of Condensed Matter Physics*, 6(1):219–244, 2015.

[14] S. K. Chaulya and G. M. Prasad. Sensing and monitoring technologies for mines and hazardous areas. In *Sensing and Monitoring Technologies for Mines and Hazardous Areas*. Elsevier, 2016.

[15] Carl Darukhanavala, Andrew Lycas, Arpit Mittal, and Ashwinram Suresh. Design of a bimodal self-burying robot. In *2013 IEEE International Conference on Robotics and Automation*, pages 5600–5605, 2013.

[16] Jean-Louis Deneubourg, Simon Goss, Nigel Franks, Ana Sendova-Franks, Claire Detrain, and Laeticia Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *From animals to animats: proceedings of the first international conference on simulation of adaptive behavior*, pages 356–365, 1991.

[17] D. Deng. Dbscan clustering algorithm based on density. In *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, pages 949–953, 2020.

[18] Zhenhua Du, Hongbin Fang, and Jindong Xu. Snake-worm: A bi-modal locomotion robot. *Journal of Bionic Engineering*, 19:1272–1287, 2022.

[19] Miguel Duarte, Jorge Gomes, Vasco Costa, Tiago Rodrigues, Fernando Silva, Víctor Lobo, Mario Monteiro Marques, Sancho Moura Oliveira, and Anders Lyhne Christensen. Application of swarm robotics systems to marine environmental monitoring. In *OCEANS 2016 - Shanghai*, pages 1–8, 2016.

[20] Neil Eliot, David Kendall, and Michael Brockway. A new metric for the analysis of swarms using potential fields. *IEEE Access*, PP:1–1, 10 2018.

[21] Fausto Ferreira, Anja Babić, Martin Oreč, Nikola Mišković, Corrado Motta, Roberta Ferretti, Angelo Odetti, Simona Aracri, Gabriele Bruzzone, Massimo Caccia, Federica Braga, Giorgia Manfè, Giuliano Lorenzetti, Gianmarco Scarpa, and Francesca De Pascalis. Heterogeneous marine robotic system for environmental monitoring missions. In *2023 IEEE Underwater Technology (UT)*, pages 1–5, 2023.

[22] Pablo Garcia-Aunon, Juan Jesús Roldán, and Antonio Barrientos. Monitoring traffic in future cities with aerial swarms: Developing and optimizing a behavior-based surveillance algorithm. *Cognitive Systems Research*, 54:273–286, 2019.

[23] Veysel Gazi, Barıs Fidan, Lino Marques, Raul Ordonez, EF Kececi, and M Ceccarelli. Robot swarms: Dynamics and control. In *Mobile robots for dynamic environments*, pages 79–107. ASME New York, NY, USA, 2015.

[24] Giuseppe Gonnella, Davide Marenduzzo, Antonio Suma, and Adriano Tiribocchi. Motility-induced phase separation and coarsening in active matter. *Comptes Rendus Physique*, 16(3):316–331, 2015.

[25] H. Hamann. Introduction to swarm robotics. In *Swarm Robotics: A Formal Approach*. Springer, Cham, 2018.

[26] W. Hao and S. Qin. Multi-objective path planning for space exploration robot based on chaos immune particle swarm optimization algorithm. In H. Deng, D. Miao, J. Lei, and F. L. Wang, editors, *Artificial Intelligence and Computational Intelligence. AICI 2011*, volume 7003 of *Lecture Notes in Computer Science*, pages 42–49. Springer, Berlin, Heidelberg, 2011.

[27] Zhijian Hao, Siddharth Mayya, Gennaro Notomista, Seth Hutchinson, Magnus Egerstedt, and Azadeh Ansari. Controlling collision-induced aggregations in a swarm of micro bristle robots. *IEEE Transactions on Robotics*, 39(1):590–604, 2023.

[28] Michael G. Hinchey, Roy Sterritt, and Chris Rouff. Swarms and swarm intelligence. *Computer*, 40(4):111–113, 2007.

[29] J. Hird, A. T. Conn, and S. Hauert. Stochastic jumping robots for large-scale environmental sensing. *Advanced Intelligent Systems*, 4:2100219, 2022.

[30] T. R. Hsiang, E. M. Arkin, M. A. Bender, S. P. Fekete, and J. S. B. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In Jean-Daniel Boissonnat, Joel Burdick, Ken Goldberg, and Seth Hutchinson, editors, *Algorithmic Foundations of Robotics V*, volume 7 of *Springer Tracts in Advanced Robotics*. Springer, Berlin, Heidelberg, 2004.

[31] Edmund Hunt and Sabine Hauert. A checklist for safe robot swarms. *Nature Machine Intelligence*, 2, 07 2020.

[32] Aleksandar Jevtic, Álvaro Gutierrez, Diego Andina, and Mo Jamshidi. Distributed bees algorithm for task allocation in swarm of robots. *IEEE Systems Journal*, 6(2):296–304, 2012.

[33] Dejin Jiao, Qing Li Zhu, Chen Yu Li, Qiang Zheng, and Zi Liang Wu. Programmable morphing hydrogels for soft actuators and robots: from structure designs to active functions. *Accounts of Chemical Research*, 55(11):1533–1545, 2022.

[34] S. Kim, E. Hawkes, K. Choy, M. Joldaz, J. Foleyz, and R. Wood. Micro artificial muscle fiber using niti spring for soft robotics. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2228–2234, St. Louis, MO, USA, 2009.

[35] C Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1-2):85–101, 2000.

[36] G. Li and A. Wang. Cold, warm, and hot programming of shape memory polymers. *J. Polym. Sci. Part B: Polym. Phys.*, 54:1319–1339, 2016.

[37] A. Loria, J. Dasdemir, and N. Alvarez Jarquin. Leader–follower formation and tracking control of mobile robots along straight paths. *IEEE Transactions on Control Systems Technology*, 24(2):727–732, March 2016.

[38] Danli Luo, Jianzhe Gu, Fang Qin, Guanyun Wang, and Lining Yao. E-seed: Shape-changing interfaces that self drill. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 45–57, 2020.

[39] Iñaki Navarro and Fernando Matía. An introduction to swarm robotics. *International Scholarly Research Notices*, 2013:10 pages, 2013.

[40] K. Nayanasitachowdary and M. Padmaja. A real and accurate gps based environmental monitoring robotic system using iot. In *2021 Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 242–247, 2021.

[41] Dinesh K. Patel, Ke Zhong, Haiqing Xu, Mohammad F. Islam, and Lining Yao. Sustainable morphing matter: Design and engineering practices. *Advanced Materials Technologies*, 8:2300678, 2023.

[42] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 25–34, New York, NY, USA, 1987. Association for Computing Machinery.

[43] Hamed Rezaee and Farzaneh Abdollahi. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. *IEEE Transactions on Industrial Electronics*, 61(1):347–354, 2014.

[44] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7:36, 2020.

[45] B. Septfons, A. Chehri, H. Chaibi, R. Saadane, and S. Tigani. Swarm robotics: Moving from concept to application. In Alfred Zimmermann, Robert J. Howlett, and Lakhmi C. Jain, editors, *Human Centred Intelligent Systems*, volume 310 of *Smart Innovation, Systems and Technologies*. Springer, Singapore, 2022.

[46] D. S. Shah, J. P. Powers, L. G. Tilton, et al. A soft robot that adapts to environments through shape change. *Nature Machine Intelligence*, 3:51–59, 2021.

[47] Dhruv Shah and Leena Vachhani. Swarm aggregation without communication and global positioning. *IEEE Robotics and Automation Letters*, 4(2):886–893, 2019.

[48] Justine Shaw and Gennaro Notomista. Design and density control of a swarm of bimodal particles, 2024. In preparation for the International Symposium on Distributed Autonomous Robotic Systems (DARS).

[49] Michael Shell. IEEEtran homepage on CTAN. *test journal*, 2002.

[50] Timothy Stirling, James Roberts, Jean-Christophe Zufferey, and Dario Floreano. Indoor navigation with a swarm of flying robots. In *2012 IEEE International Conference on Robotics and Automation*, pages 4641–4647, 2012.

[51] Ryo Suzuki, Clement Zheng, Yasuaki Kakehi, Tom Yeh, Ellen Yi-Luen Do, Mark D Gross, and Daniel Leithinger. Shapebots: Shape-changing swarm robots. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 493–505, 2019.

[52] Ying Tan and Zhong yang Zheng. Research advance in swarm robotics. *Defence Technology*, 9(1):18–39, 2013.

[53] T. H. A. van den Broek, N. van de Wouw, and H. Nijmeijer. Formation control of unicycle mobile robots: A virtual structure approach. In *Proc. 48th IEEE Conf. Decision Control 28th Chin. Control Conf. (CDC/CCC)*, pages 8328–8333, December 2009.

[54] J. Wu and Z. Jiang. On the switching control of multiple mobile robots formation. In *2009 IEEE International Conference on Robotics and Automation*, pages 2711–2716, 2009.

[55] Junjun Wu, Zhonghui Huang, Yisheng Guan, Chuanwu Cai, Qinghui Wang, Zhiguang Xiao, Zhifang Zheng, Hong Zhang, and Xianmin Zhang. An intelligent environmental monitoring system based on autonomous mobile robot. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 138–143, 2011.

[56] Chenhui Xiang and Junkai Yang. Design guidelines for indoor intelligent environmental monitoring and control system. In *2012 IEEE International Conference on Computer Science and Automation Engineering*, pages 737–740, 2012.

[57] Guang-Zhong Yang, Jim Bellingham, Pierre Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, Brad Nelson, Brian Scassellati, Mariarosaria Taddeo, Russell Taylor, Manuela Veloso, Zhong Wang, and Robert Wood. The grand challenges of science robotics. *Science Robotics*, 3(14):eaar7650, 2018.

[58] X. Yang, Y. Zhou, H. Zhao, W. Huang, Y. Wang, K. J. Hsia, and M. Liu. Morphing matter: from mechanical principles to robotic applications. *Soft Science*, 3:38, 2023.

[59] Yongming Yang and Yantao Tian. Swarm robots aggregation formation control inspired by fish school. In *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 805–809, 2007.

[60] Fatma Günseli Yaşar and Gözde Ulutagay. Challenges and possible solutions to density based clustering. In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, pages 492–498, 2016.

[61] S. Yim and M. Sitti. Shape-programmable soft capsule robots for semi-implantable drug delivery. *IEEE Transactions on Robotics*, 28(5):1198–1202, Oct 2012.

[62] E. M. H. Zahugi, M. M. Shanta, and T. V. Prasad. Oil spill cleaning up using swarm of robots. In Natarajan Meghanathan, Dhinaharan Nagamalai, and Nabendu Chaki, editors, *Advances in Computing and Information Technology*, volume 178 of *Advances in Intelligent Systems and Computing*, pages 181–190. Springer, Berlin, Heidelberg, 2013.

[63] Matt Zarek, Michael Layani, Ido Cooperstein, et al. 3d printing of shape memory polymers for flexible electronic devices. *Advanced Materials*, 28(22):4449–4454, 2016.

[64] D. Zhang, W. Kong, R. Kasai, Z. Gu, Y. Minami Shiguematsu, S. Cosentino, S. Sessa, and A. Takanishi. Development of a low-cost smart home system using wireless environmental monitoring sensors for functionally independent elderly people. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 153–158, 2017.

[65] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2005.