A SELF-ORGANIZED COLLECTIVE FORAGING METHOD USING A ROBOT SWARM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TUGAY ALPEREN KARAGÜZEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

APRIL 2020

Approval of the thesis:

A SELF-ORGANIZED COLLECTIVE FORAGING METHOD USING A ROBOT SWARM

submitted by **TUGAY ALPEREN KARAGÜZEL** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Ali Sahir Arıkan
Head of Department, **Mechanical Engineering** _____

Assist. Prof. Dr. Ali Emre Turgut
Supervisor, **Mechanical Engineering, METU** _____

Assist. Prof. Dr. Eliseo Ferrante
Co-supervisor, **Computer Science, VU Amsterdam** _____

Examining Committee Members:

Assoc. Prof. Dr. Erol Şahin
Computer Engineering, METU _____

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU _____

Assoc. Prof. Dr. Ulaş Yaman
Mechanical Engineering, METU _____

Assist. Prof. Dr. Kutluk Bilge Arıkan
Mechanical Engineering, TED University _____

Assist. Prof. Dr. Ender Yıldırım
Mechanical Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Tugay Alperen Karagüzel

Signature :

ABSTRACT

A SELF-ORGANIZED COLLECTIVE FORAGING METHOD USING A ROBOT SWARM

Karagüzel, Tugay Alperen
M.S., Department of Mechanical Engineering
Supervisor: Assist. Prof. Dr. Ali Emre Turgut
Co-Supervisor: Assist. Prof. Dr. Eliseo Ferrante

April 2020, 72 pages

In this thesis, a collective foraging method for a swarm of aerial robots is investigated. The method is constructed by using algorithms that are designed to work in a distributed manner, by using only local information. No member in the swarm has access to global information about positions, states or environment. The environment, that robots are planned to operate in, contains a virtual scalar field which consists of grids containing constant values. The grid values indicate desired regions of the environment. By using the proposed methods, swarm forages towards the desired regions in collective manner as a cohesive and ordered group. Important point to mention is that members do not have the capability to sense the environment to do so on their own. Instead, they use information extracted from interactions with neighbouring members. This phenomenon, which has interesting examples on nature, is called collective sensing. At first, the algorithms are implemented on MATLAB environment with particle agent models and errorless movements. Later, they are implemented and tested on GAZEBO physics simulator with realistic and physics based models of robots. Finally, they are tested with real aerial robots which are modelled on GAZEBO before. The results are analyzed in terms of being a single and cohe-

sive group without any collision in addition to the success in foraging towards desired regions of the environment and stay there as long as possible.

Keywords: swarm intelligence, swarm robotics, flocking, collective sensing, collective foraging

ÖZ

BİR ROBOT SÜRÜSÜNÜN KENDİ KENDİNE MÜŞTEREK YÖNELİM HAREKETİ SERGİLEMESİ İÇİN BİR YÖNTEM

Karagüzel, Tugay Alperen
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ali Emre Turgut
Ortak Tez Yöneticisi: Dr. Öğr. Üyesi. Eliseo Ferrante

Nisan 2020 , 72 sayfa

Bu tezde, uçan bir robot sürüsü için müşterek arama ve yönelim davranışı sergilemek üzere bir yöntem araştırılmıştır. Bu yöntem, bütün sürü üyeleri üzerinde dağıtık olarak çalışan, sadece yerel olarak toplanmış bilgiler ve sadece yerel ve anonim haberleşme ile çalışmak üzere tasarlanmış algoritmalardan oluşmaktadır. Sürü içerisindeki hiçbir üye pozisyonlar, anlık durumlar veya çevre durumu hakkında genel bir bilgiye sahip değildir. Robotların çalışması planlanan ortam, robotların deney boyunca seyrettikleri rota ile karşılaştırıldığında yok sayılabilecek kadar küçük ve içlerinde sabit bir sayısal değer taşıyan kare alt alanlardan meydana gelen bir sayıl alan içermektedir. Önerilen yöntem sayesinde sürü, çalışma alanının tercih edilen (içerisinde daha küçük sayısal değerler tutan) alanlarına birleşik ve düzenli bir grup olarak müşterek şekilde yönelir. Burada üyelerden herhangi birinin bu yönelimi gerçekleştirmek için gerekli çevre algılama kabiliyetine sahip olmadığını belirtmek önem arz eder. Bunun yerine, komşuluklarındaki üyelerle aralarındaki etkileşimlerden doğan bilgileri kullanılar. Doğada da ilginç örnekleri görülen bu olgu müşterek algılama olarak tanımlanır. İlk olarak algoritmalar, noktalar olarak tanımlanmış ve hatasız hareket eden üyeler ile MATLAB ortamında gerçeklenmiştir. Daha sonra, robotların bir fiziksel benzetim ortamı olan

GAZEBO'da hazırlanmış gerçekçi ve dinamik modelleriyle test edilmiştir. Son olarak, algoritmalar daha önce GAZEBO benzetim ortamında modellenen platformların kendileriyle, gerçek dünyada test edilmiştir. Elde edilen sonuçlar, sürünün çalışma alanının tercih edilen bölümlerine doğru yönelme ve olabildiğince uzun süre orada kalma başarımlarına ek olarak herhangi bir üyeler arası çarpışma gerçekleştirmeden tek ve birleşikliği açısından da değerlendirilmiştir.

Anahtar Kelimeler: sürü zekası, sürü robotiği, sürü göç hareketi, müşterek algılama, müşterek yönelim

To waking dreams...

ACKNOWLEDGMENTS

This study was conducted at METU KOVAN Research Laboratory. Many thanks to Ali Emre Turgut and Erol Şahin for their generous support throughout my studies. I also thank B.K.A. for the knights of king. Lastly, to Suşi and Sufle, may I never be without them.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
CPU	Central Processing Unit
PC	Personal Computer
HA	Heading Alignment
SM	Speed Modulation
DM	Desired Distance Modulation
SITL	Software in the Loop
IMU	Inertial Measurement Unit
RGB	Red Green Blue
ROS	Robot Operating System
LIDAR	Light Detection and Ranging
UWB	Ultra Wide Band

CHAPTER 1

INTRODUCTION

Over a time span starting from very first ages of it's existence, humans are just little members of the big and complex biosphere of earth. As all creatures on earth must do, we have to fulfill various tasks in a daily, monthly and even yearly basis to survive. Thanks to our special distinctness inside our heads, we were always able to come up with solutions to those tasks not to just fulfill them but making them in easier, faster and even better ways. Most of the time, these solutions were nothing but the orders of mother nature whispered from hard coded parts of our brain, telling us to co-operate with others as all creatures do, since this is sometimes the only way to survive. By co-operating with the others, we become stronger, crowded and even dominant in the habitat we choose. Hereupon, human race has started to use, manipulate and shape the environment that it lives in. It was the point we realize that we do not have to do everything on our own, we can make tools to do that for us. It all started with shaping, carving or hewing stones, woods and all other materials we found. As time passed, it turned out to be a snowball effect; Every item that a human crafted, led to the other. The inevitable leap has occurred with the invention of systems of tools. Little wheels, rods and mechanisms are some examples of them.

After centuries, with the occurrence of systematic power sources, electricity and computation technologies after them, humankind discovered various materials and forms that handle tasks conveniently. The transition from tools to robots that involve silicone microchip brains, metal motor joints and glass camera eyes, has drastically changed the game for humans. A "robot" can be defined as the physical embodiment that evaluates current state of it and takes a series of actions automatically by interacting with the environment.

By developing complex and capable robots those can walk, fly, jump and do many more, human race has challenged itself for better. The answer was again, taking part in nature itself. There are animals with huge bodies, claws or sharp visions but they were still overshadowed with fascinating abilities of other small ones like ants, bees, birds or fishes. Next step was understanding these talented groups of weak and small animals. In last decades, there are numerous studies to deepen that understanding. In [1], the questions of how interactions between animals in a group effect the group behaviour and how information is shared by indirect ways between animals are answered. The animals were able to travel over long distances as a group to their destination with small portion of members having information about the destination. This study has revealed the effects of social interactions among members on the group behaviour and developed a mathematical approach to these effects.

Understanding the behaviours of individuals in an animal group and effects of social interactions on their decisions were not the ultimate aims of the conducted studies. Further effort was made to recreate those mechanisms in various fields. In study [2], the flight of a flock of birds tried to be animated by a computer. For virtual agents those are able to fly freely in void, fundamental mathematical expressions about the necessary interactions and effects between agents are implemented and it is shown that the flight of a bird flock can be reproduced by a computer animation in a realistic way. Those studies provided opportunity to use the knowledge about behaviour of animal groups and modify it in desired ways.

The current state of technology lead the way to apply the behaviours of animal groups to the human made autonomous machines, robots. Thereafter, one of the most fascinating branches of contemporary robotics has emerged, called as swarm robotics. The definition of a robot swarm can be made as; Large number of relatively simple group of robots inspired by the social insects that interacts with each other and the environment to accomplish tasks that are beyond the capability of the single one [3]. Swarm robotics has drawn attention and found itself very interesting application areas from surveillance [4] to infrastructure maintenance [5] and agricultural applications [6]. A fundamental concept that is derived from behaviours of animal groups made those applications possible; Collective behaviour. The definition of collective behaviour can be stated as the pattern of behaviours emerging at a group of simple

members completely determined and dominated collectively, without any centralized effect [7].

One of the mostly encountered collective behaviour is the foraging behaviour, which can be formally defined as the search for food or resources by an animal or a group of animals. This crucial collective behaviour of animals offer a wide range of applications for robot swarms that can be exemplified by detecting matured crops on an agricultural terrain or searching for an ore bed under the ground in addition to health-endangering ones like searching radioactive sources or monitoring wild fires.

Using a swarm of robots has many advantages over a single complex robot [3]. First one would be the robustness of the swarm over failures, losses in the group in addition to disturbances acting on the swarm. Despite them, a swarm of robots continues to do whatever it was doing owing to it's distributed and self propelled nature. The next one is the flexibility of the group, to varying environmental changes in addition to different tasks to be fulfilled. As can be seen in the examples in the nature, this talent to adapt is the key to survive despite changing conditions and requirements. One other advantage to mention would be the scalability which tells us that the capabilities of group does not decrease with changing number of agents in the group. This concept opens a window to broad range of different applications. The advantages of employing a robot swarm may become stronger with the chosen design of the robots. Different types of autonomous robots operating in various fields may offer unique advantages. A robot swarm operating under water provides the ability to access hard to reach regions and perform tasks those are impossible for human beings. Likewise, a robot swarm can fulfill dangerous duties on the ground with less effort to move and less concern for a possible collision. As another option, an aerial robotic swarm provide a new aspect for the surveillance and search tasks over broader fields by reaching highest and toughest fields on earth. With increasing momentum of developing opportunities in the computer, sensor and actuator technologies, one can gain a chance to move freely in the air while monitoring and sensing both the air sight and the ground simultaneously with an aerial autonomous robot. This robot can sense, track or search for various different targets. A foraging task may find itself endless different usages when employed on a swarm of unmanned aerial systems.

This study presents a set of algorithms to be employed on an aerial swarm of robots to achieve a cohesive and an ordered group that can forage to particular regions of a constrained environment by relying on each other's actions while being incapable of foraging on their own. This achievement produce a talented group from incapable members in terms of foraging. Developed algorithms make it possible by employing a novel method which modify the behaviour of individuals according to local state of them. As harmonious with the swarm robotics concept, algorithms are designed to work in a distributed manner such that every agent decide for only it's own actions by using the information it gathered locally without any help from a centralized more capable system. Since every assessment of the next action of the robot is calculated based on local information, group is robust to a possible loss or failure.

The presentation of the work in this study is as follows: In following section the state of the literature on collective behaviour and foraging methods are presented in both nature and artificial systems. In Chapter 3, the methodology to maintain proposed behaviours for the swarm is presented mathematically and formulations are presented with all required background information and definitions. In Chapter 4, different experimental setups are constructed and explained. In addition, it is made clear how the methodology, proposed in Chapter 3, is employed in those experimental frameworks. Later in that chapter, the results of experiments are presented with metrics which are used to evaluate group performance on various aspects. In Chapter 5, the experiments and corresponding results are discussed. Finally, Chapter 6 concludes overall work, in order to melt all of the presented information in the same pot.

CHAPTER 2

LITERATURE SURVEY

In this chapter, the current state of the subject in the literature is presented. The previous studies are grouped and presented based on their main field of research and contributions. The first group of studies are about collective motion seen at different areas from living creatures to artificial systems. In this group, studies that deal with the question of what is the underlying mathematical models of animal's collective motion and studies that tries to maintain a group of ground or aerial robotic platforms maintaining a collective motion in an agreement with swarm robotics concepts take place. The second group focuses on studies about collective sensing and foraging. To extend, the methods and applications about groups that can collectively sense their environment and take actions related to this sensing phenomenon, are mentioned. These groups are sometimes already existing animals living in the nature but sometimes artificial human made systems that either mimic the methods in the nature or employ their novel technique to sense the environment in a collective manner. Those two different kind of examples are mentioned in corresponding subsections. In addition to those, foraging is another common aspect of the studies in this group since foraging task is sometimes related with collective sensing of the group and it comes with the requirement of collective motion. Apart from that, it might be an emergent behaviour that does not necessarily requiring a collective sensing or motion.

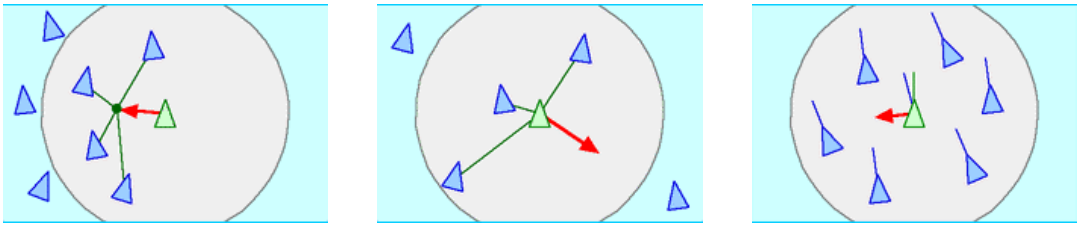
2.1 Collective Motion

2.1.1 Collective Motion in Nature

Moving as a group is a widely encountered phenomenon in the nature [8] [9]. Doing so has numerous advantages for the animals such as avoiding possible predators [10], navigating with informed members which increases the chance of finding food sources [1] in addition to other species specific advantages like increased survival chance for cannibal crickets [11] or increased chance of mating for trilobite arthropods [12]. Every animal group has its own method for maintaining collective motion depending on its member's abilities, sensing and communicating capabilities in addition the environment the group is living in. These methods of collective motion have drawn the attention of researchers from different fields. One study [13] analyzes the data collected by tracking the positions of pigeons which fly as a flock to understand interactions between them. Results reveal that their movements are directly correlated with relative positions of neighbouring pigeons at an instant. In another study [14], it is tried to be understood what are the rules behind the schooling behaviour of fishes. In order to do that, schooling behaviour is captured by a video camera and likewise to [13], the movement of each fish is analyzed to fit a mathematical model to the interactions among school. [14] suggests the existence of virtual attractive and repulsive force between fishes.

2.1.2 Collective Motion in Artificial Systems

The search for a mathematical explanation about animals that moving collectively has started decades ago and is still continuing. One of the very first studies about an approach inspired from natural systems and applied to an artificial one was [2]. In [2], Reynolds wanted to create a simulation of flying "boids". A boid resembles a simple agent freely moving in space. It was desired to have a natural scene of flying boids. So then, it has to be inspired from real flying group of animals; Birds. With his work, he introduced the very early but essential and still valid constructive elements of collective motion. There are three types of behaviours in Reynolds's work that are followed individually by boids in a distributed manner; Cohesion, separation and



(a) Cohesion (b) Separation (c) Alignment

Figure 2.1: A visual demonstration of cohesion, separation and alignment effects between voids in (a), (b) and (c) respectively. The red arrow designates the effects of those behaviours on the focal boid while the gray circle is representing its sensing range.

alignment. While cohesion designates the need of a boid to stay in a certain proximity with neighbouring ones to avoid a split from the group, the separation behaviour exists to ensure that boids should not be closer than a certain range to avoid any collision. Finally, the alignment behaviour resembles that boids should align their direction of travel according to the average direction of their neighbours in order to travel all together in the same direction as a flock. Cohesion, separation and alignment behaviours are visualized with figurative boids in Figure 2.1 .

In [15], researchers studied methods for a group of mobile ground robots to maintain a collective motion towards a certain direction. In order to achieve that, robots have infrared distance sensors to detect arena boundaries and other members. In addition, all robots are equipped with small speakers and microphones to detect each other from a wider range. The group was informed about the goal direction by a light source placed in the arena. All robots sense the light source by light sensors they are carrying. By this way, the group achieved a flocking behaviour towards goal direction in a cohesive way. All of the sensing and computation was local and distributed at the study. The underlying rules were not the same with [2], since in [15], an evolving set of neural network structures are used and every member has evolved in a different and unique way to accomplish that task. Despite, in [2], every member is identical to each other and have the same parameters on their formulations, ensuring a flexible and scalable swarm.

One study [16] shows an implementation of flocking towards a pre-defined goal direction, which is known by all members of group, with mobile ground robots using simulated sensory information. An overhead camera collects the positions of the robots and a computer sends them to robots over a communication network as it is sensed by the robot itself by considering that if there is a neighbourhood between two subjected members or not. The main purpose of group was collectively moving towards a direction in a cohesive way. An interesting approach in [16] was that robots' decision algorithms are working on a principle to calculate an instantaneous center of mass belonging to neighbouring robots and try to track that center of mass by changing their heading angles accordingly while modulating their linear speed depending on the distance to center of mass by a constant gain. The sensory system was not local in [16] and it was working on a centralized position information communication system. Also, all of the members have the information of correct goal direction. Their direction of motion was not an emergent property.

[17] is one of the most comprehensive trials about a group of mobile robot swarm that is maintaining a collective motion. In this study, a robot platform is developed that carries a capable micro controller on board. The micro controller reads and evaluates sensor signals and produce commands. The commands are sent to actuation system of the robot which moves the robot with two wheels. The sensory system consists of light emitting diodes working in infrared spectrum and light sensors to detect infrared light emitted by it's neighbours. This sensory system was a successful attempt to develop a local proximity sensing capability that does not need any global information for neighbour's relative positions. Also, robots share their heading angle information in a local range with a communicating system to employ alignment behaviour. In [17], results show that with sufficient sensory system and successfully designed algorithms, a group of simple mobile robots can move as a group to desired direction without any split, crash or confusion. Yet, this study demonstrate all of this ability on ground robots that are moving in relatively slow speed. Since all of the robots are operating at ground level, sensing system has the chance of a stable operation. Accomplishing the same behaviour with flying robots was not in the scope of this study. In addition, the group interacted with the environment only for obstacle detection via their distance sensors. There was no attracting or guiding effect coming from environment and the

goal direction of the flock was implemented in few number of members in the group. This small portion of informed members was effective on guiding whole group to goal direction.

Another study about the collective motion of autonomous robots is presented in [18]. In this study, researchers developed a group of autonomous flying robots that can move together successfully. There are quadrotors used for that task carrying single board computers to handle required computation. Although it is a good example of a collective motion, sensing is not handled locally. Instead, every agent get it's absolute position via the component that uses global positioning system and share that position information with others. In addition, members share their current velocity information involving it's direction of travel with neighbours. Every agent use this position, direction and velocity information to calculate virtual effects which represent the cohesion, separation and alignment behaviours to realize a flocking motion as a group. Despite the fact that computations are handled locally, the global positioning system employment and dependency on the communication system among robots conflicts with the very basic definition of a robot swarm. Even with this conflict, the study is a good example of collective motion realized with autonomous flying robots. Another beneficial approach of the [18] is that, researchers tune and validate corresponding parameters of flocking algorithms with simulations involving robot dynamics that accelerates the developing process by avoiding any unnecessary experiments and possible losses.

2.2 Collective Sensing and Foraging

2.2.1 Collective Sensing and Foraging in Nature

The term collective sensing defines a very unique and beneficial talent belonging to members of animal kingdom. It is the ability of a group to demonstrate an emergent behavior correlated with changes in the environment as if it is a single and complex unit while consisting of members that are incapable of sensing those changes in the environment on their own [19]. The best example to an animal group that collectively sense its environment is social insects. These insect gains this new ability purely as a

result of their collective working scheme.

Originating from various usages of collective sensing, the capabilities of social animals having this ability show a great diversity. The study [19] categorizes the most common ones as follows; Deciding by many wrongs principle, the leadership, emergent sensing, social learning and collective learning. To understand the many wrongs principle better, it can be exemplified as this case; Members in a group might have inaccurate guesses about what is better to do or where is safer to go. Yet, if all of the varying estimations about the right choice among the group are fused in a proper way, the group will collectively behave in favor of the right option [20]. The second categorized ability, the principle of the leadership in a group, might seem a bit predictable, yet it still fascinates researchers with underlying mechanisms such as the propagation speed of information among the group. Several informed members of a group in size of thousands can guide the whole group without identifying themselves as leaders to others [21]. After that, the definition of next category, emergent sensing, is highly related with collective sensing. It is the ability of a group to produce emergent behaviour correlated with environmental changes and properties. This group members have simple and insufficient sensing capability. There is no memory and communication about the measurements that members have done. Sometimes the members are not even aware of what they are doing as a group [22]. The final categories, the social and collective learning define the development of deciding mechanisms in the members according not only to individual experiences but also the experiences that other members gain. These experiences are shared over social interactions among members [23]. The development process would not be achievable with non-interacting individuals, independent of whether they are still interacting with their environment or not.

The study [19] underlines the importance of social interactions within an animal group from different aspects. The idea is experimentally supported with evidences from the nature. One interesting study [24] presents an experimental analysis and a mathematical effort to understand a crucial collective behavior of fishes. This behaviour is also a good example of emergent sensing. In [24], researchers tries to understand how the travel direction of two different fish species, named as golden shiners (Notemigonus crysoleucas) and rummy nose tetras (Hemigrammus bleheri),

is decided. From previous experiences it is known that both fish types are tend to stay in darker places of the environment. Yet it was a strange result to see although golden shiners may sense the darker direction at their current location, tetras was not able to sense the darker direction. They can only sense the point they are currently located in. In other words, they can only do a scalar measurement, can not deduct a direction. Nevertheless, tetra fishes was successful as golden shiners in the task of locating themselves in darker regions. After further analysis, it is shown that tetra school members does not have an idea about direction of darker regions. They only modulate their swimming speed according to light level and swim slower in darker regions. This individual and local behaviour which does not requires any memory, lead a tetra school to always turn towards the darker regions. Researchers believe that the reason behind the case is beside tetra's modulation on swim speeds, they also have social interactions like attraction, repulsion and alignment. Because of the social interactions, faster swimmers do not want to split the group and try to turn towards slow members when they move away. That results with a group turning towards darker regions. The study [24] demonstrates a solid example of how animals use their collective sensing ability to survive.

The speed modulation technique of the tetra fishes has inspired the speed modulation foraging method proposed by the author of current study. Yet, there exists another important but unanalyzed behaviour of fishes in [24]. Researchers state that the density of the group (mainly resembling how close the members of the group staying to each other) changes with the light level. The mathematical models for social interactions and the correlation of swim speeds with light level were constructed and formulated decently but there was no attempt to include the relation between group density and light level. The desired distance modulation method that is presented in detail at Chapter 3 is strongly inspired from that gap.

2.2.2 Collective Sensing and Foraging in Artificial Systems

There is no real world robotic application of the emergent sensing methods presented in [24] according to authors knowledge. Yet there are other algorithms that have been implemented on real robotic platforms. Unfortunately they do not exhibit properties

of collective motion and emergent sensing simultaneously. The Beeclust algorithm [25], is one of the best examples of a collective behaviour resulting with a foraged group. The algorithm shows that bees, always tend to locate themselves in regions with a specific temperature range. Individually, they are incapable of sensing the temperature difference sensitive enough to decide where to go. Acting as a swarm, they successfully forage to appropriate regions. Beeclust algorithm offers that bees are not tracking any temperature change but they modulate their waiting time in case of any encounter with another bee or a wall. In other words, a bee wait longer in the warm regions when it bump into another bee. This situation ends up with a group of bees gathered around a region with specific temperature range. [26] modifies this algorithm by adding virtual pheromones in an appropriate and controlled environment and demonstrates a successful aggregation around the chosen part of the environment. In that study, two wheeled ground robots are used.

[27] presents an attempt towards a swarm of flying robots that searches the environment. The environment has varying density of a virtual property. The swarm aims to aggregate around certain regions of the environment. In order to accomplish that, different methods are used simultaneously. The behaviour of an agent is decided by a logical priority map just like a flow chart. Every agent within the swarm can avoid an obstacle with on board sensors and this behaviour has the highest priority because of safety reasons. Moreover, every agent has a tendency to move together with other agents in a cohesive and ordered way. To obtain that collective motion, virtual effects are implemented on the controllers very similar to Reynold's rules; Cohesion, separation and alignment. Finally, every agent communicates with the ground station and deploy a volatile virtual pheromone at desirable locations. The locations and densities of virtual pheromones are stored in the ground station and shared with other agents. By using those 3 different behaviours, researchers successfully gathered agents around desirable regions of the environment. Although the study presents promising results on real platforms, the existence of a ground station that is storing virtual pheromone information and dependency on global positioning system give this method a lesser chance to be a solution for real world problems that require collective motion and emergent sensing.

The studies presented in this section has investigated the examples of collective be-

haviour in animals either to understand how animals use it in their favor or how they accomplish it. In addition, those studies clarify and present the underlying methods with mathematical expressions. Moreover, other studies are presented which aim to implement collective behaviour methods on artificial systems to solve real world problems. Different kind of sensing and motion techniques are used in those studies. None of them present a robot swarm that does not need any localization system, works without any global information about states of robots or the environment, completely relies on information gathered by local sensing capabilities, only employs a local communication system or does not employ a communication system at all and handles all computations in a distributed manner without a need for central unit to maintain a collective motion and collective foraging behaviour simultaneously. This study accomplishes some of them. In this study,

- A collective foraging method to be implemented on an aerial robot swarm is presented. This method allows the swarm to forage towards certain regions of the environment as a cohesive group and stay there.
- This method consists of algorithms those work in a distributed manner without a need for central computation or communication unit. Members of robot swarm only need the relative distance and bearing information of neighbouring robots that can be sensed locally in addition to heading angle information that every robot shares within a certain range. The information about environment properties is gathered individually, locally and does not need to be shared with others.
- The proposed method is simulated with agents which are modelled as particles with no dynamical properties. In addition, simulations with real world dynamics are done with physics based robot models. The results of both simulations are presented with metrics that evaluates group foraging performance.
- Experiments are conducted with real flying robots. The results are presented and evaluated with same metrics and compared to results of previously done simulations.

CHAPTER 3

METHODOLOGY

This section presents the methodology behind self-organized flocking and foraging proposed in this thesis. The methodology is introduced in mainly three sections. In first section, flocking behaviour is explained. Flocking behaviour ensures agents to have a collision free motion while staying cohesive and polarized as a group. In second section, fundamentals of foraging method are explained. A foraging group moves towards certain regions of the environment. These regions can be distinguished by a local value which is defined for every point of the environment and varying smoothly through different directions. In third section, method for flocking and foraging behaviours to generate motion control outputs, is explained. Upon defining different effects on an agent caused by different sources, their possible and efficient combinations are to be constructed.

3.1 Flocking Behaviour

Flocking is the ordered motion of a cohesive and polarized group without any collisions between agents while avoiding boundaries of the environment. In order to obtain a flocking group, every agent uses local information such as relative positions and heading angles of agents nearby or relative distance to a boundary if it is close enough.

In order to produce motion control outputs for agents, virtual force vectors are employed. To achieve flocking behaviour, corresponding vectors are calculated using interactive effects between agents in addition to agents and boundaries [17]. The proximal control effect between agents act as a bond such that it draws them together

if they are wide apart and it pushes them reverse if they are too close to each other. This bond is modeled through $\vec{\mathbf{p}}_i$ which is constructed in the local coordinate frame of focal agent and directed from nearby agents to focal agent itself. The avoidance effect between an agent and a boundary is only repulsive; Pushes agent if it is too close to a boundary. It is modeled through $\vec{\mathbf{r}}_i$ and also constructed in the local coordinate frame of focal agent. Heading alignment vector $\vec{\mathbf{h}}_i$ is a rotational bond between heading angle of an agent and heading angle average of nearby agents. This vector act in a way such that it orients heading angle of focal agent towards average heading angle. Heading angle of an agent indicates the direction that agent can proceed. In other words, it is current direction of motion. Resultant vector is sum of proximal control, heading alignment and boundary avoidance vectors.

The resultant virtual force vector $\vec{\mathbf{f}}_i$ on agent i is defined as;

$$\vec{\mathbf{f}}_i = \alpha\vec{\mathbf{p}}_i + \beta\vec{\mathbf{h}}_i + \gamma\vec{\mathbf{r}}_i \tag{31}$$

where $\vec{\mathbf{p}}_i$, $\vec{\mathbf{h}}_i$ and $\vec{\mathbf{r}}_i$ are proximal control vector, heading alignment vector and boundary avoidance vector for agent i, respectively. α, β and γ are the corresponding gains for these vectors to adjust the relative strength of each vector.

3.1.1 Proximal Control Vector

In order to keep the distance between the agents to the desired value, proximal control is used. It is a virtual force vector denoted by $\vec{\mathbf{p}}_i$. The vector $\vec{\mathbf{p}}_i$ is the resultant of all $\vec{\mathbf{p}}_i^m$ vectors exerted on agent i by agent m. $\vec{\mathbf{p}}_i$ is calculated as;

$$\vec{\mathbf{p}}_i = \sum_{m \in N} \vec{\mathbf{p}}_i^m \tag{32}$$

where N denotes the total number of neighbouring agents and m denotes each one of them. Neighbouring agents are the ones located in the proximal sensing range D_p of

focal agent. The constitutive vector $\vec{\mathbf{p}}_i^m$ is calculated as follows;

$$\vec{\mathbf{p}}_i^m = \mathbf{p}_i^m(\mathbf{d}_i^m)\angle e^{j\phi_i^m} \tag{33}$$

In above equation, $\angle e^{j\phi_i^m}$ denotes the angle of the virtual vector exerted by agent m on agent i and is resolved in the local coordinate frame of i. The term $\mathbf{p}_i^m(\mathbf{d}_i^m)$ determines the magnitude and calculated as;

$$\mathbf{p}_i^m(\mathbf{d}_i^m) = -\epsilon\left[2\frac{\sigma_i^4}{(\mathbf{d}_i^m)^5} - \frac{\sigma_i^2}{(\mathbf{d}_i^m)^3}\right] \tag{34}$$

In above equation, $\mathbf{d}_i^m$ designates the distance between agents m and i. Figure 3.1 contains a sample visual for three agents, their inter distances and representative proximal vectors for one of them. Term ϵ determines the strength of proximal vector and term σ_i is directly correlated with desired distance between agents in a way;

$$\mathbf{d}_{des}^i = 2^{1/2}\sigma_i \tag{35}$$

3.1.2 Heading Alignment Vector

Heading alignment vector creates an effect which aims to provide an agreement between direction of motion of an agent with direction of motion of neighbouring ones. To do so, heading angles which are measured in a common reference frame for all agents, must be shared within a certain range with others. This range is limited and it's magnitude is presented by $\mathbf{D}_a$. To calculate this vector, heading angle information of neighbours belonging to focal agent is collected, resolved on its local coordinate frame and all of the angles are averaged to extract an information about the direction of motion of group. Later, this information is used to make corrections on the direction of motion of focal agent.

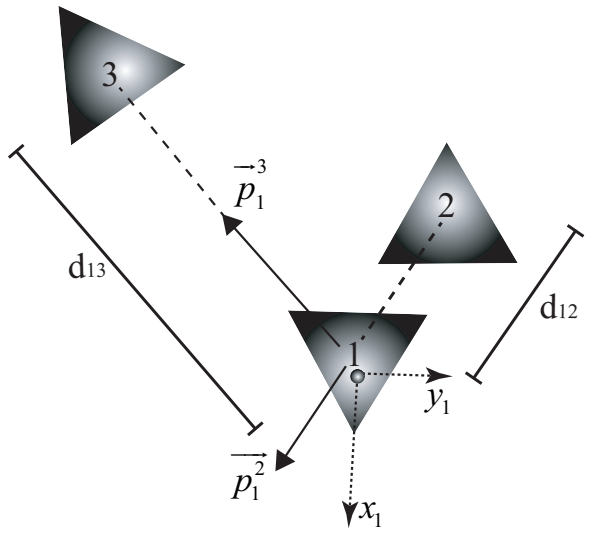


Figure 3.1: Illustration shows a focal agent with its local frame consisting of axes x_1 and y_1. $\vec{p}_1^2$ and $\vec{p}_1^3$ show the proximal control vectors originating from agent 2 and 3 on focal agent respectively. d_{12} and d_{13} are the distances to agent 2 and agent 3 respectively, relative to focal agent.

The calculation of heading alignment vector for agent i can be formulated as;

$$\vec{\mathbf{h}}_i = \frac{\angle e^{j\theta_0} + \sum_{m \in N} \angle e^{j\theta_m}}{\left|\left|\angle e^{j\theta_0} + \sum_{m \in N} \angle e^{j\theta_m}\right|\right|} \tag{36}$$

In the equation above, $\angle e^{j\theta_m}$ designates the heading angle of agent m among total N agents in the range $\mathbf{D}_a$ resolved in the common reference frame while $\angle e^{j\theta_0}$ designates the current heading angle of the subjected member itself. The result of that calculation gives a unit vector directing towards average direction of travel of the group.

3.1.3 Boundary Avoidance Vector

In a 2D environment, every individual senses the boundary of the environment within a range of $\mathbf{D}_r$. In order to be sensed by focal agent, the shortest distance between boundary and the agent must be smaller than $\mathbf{D}_r$. If the focal agent senses multiple

boundaries at the same time, the resultant repulsive force is calculated as:

$$\vec{\mathbf{r}}_i = \sum_{b \in B} \vec{\mathbf{r}}_i^b \tag{37}$$

Where B is the total number of sensed environment boundaries and b designates every one of them. The effect of single wall on the member i is calculated as;

$$\vec{\mathbf{r}}_i^b = \mathbf{k}_{rep} \left(\frac{1}{\mathbf{L}_b} - \frac{1}{\mathbf{L}_0} \right) \left(\frac{\vec{\mathbf{p}}_i^b}{\mathbf{L}_b^3} \right) \tag{38}$$

where $\mathbf{k}_{rep}$ stands for a gain term to tune strength of boundary repulsion effect, $\mathbf{L}_b$ is distance between center of agent and closest point of the boundary to that agent, $\mathbf{L}_0$ acts as a desired distance between boundary and agent. Finally, $\vec{\mathbf{p}}_i^b$ is the unit vector directed from closest point of the boundary to center of agent in order to direct virtual repulsive force.

3.2 Foraging Behaviour

Foraging is defined as searching an area for food by animals individually or as a group without any priori information about the environment. In this study, the environment consists of local values stored in grids with infinitesimal size. It can be thought as a heat map or gray scale digital image. An agent can only gather that local value stored in the current location of it. There is no information about gradient or rest of the environment.

In this study, the foraging of the group expected to be towards regions with smaller local values and these regions are referred as favorable or desirable regions occasionally. A successful foraging results not only with a group moving towards favorable regions but also staying there as long as possible. There are two approaches presented to achieve this; Speed modulation and desired distance modulation. In both approaches, agents only use instantaneously gathered information about local values and do not share it with others.

3.2.1 Speed Modulation

The linear speed of every agent is calculated by virtual force vectors at an instant. This speed is limited within an interval. The upper bound of that interval for agent i is $\mathbf{U}^i_{max}$. When speed modulation is employed as a foraging method, every agent in a flocking group changes maximum linear speed of it directly correlated with local value at current location. To clarify, agents experiencing smaller local values have smaller maximum speeds while others have the reverse situation.

When maximum speeds of agents in a group is modulated, it results as follows; The agents that are experiencing smaller local values, move slower than the rest of the group, as a result of social interactions between agents, the direction of travel of group tends to point out favorable regions of the environment which slow agents are currently located. The case is similar to a two wheel differential drive vehicle; if the wheel on the one side rotate slower than the other, in ideal case, the vehicle rotates towards the direction of slower wheel. Suchlike, the group rotates towards the favorable areas by the help of slower agents.

The modulation of maximum speed of an agent depending on the local value is as follows:

$$\mathbf{U^i_{max}} = \mathbf{U_{min}} + \frac{\mathbf{G^\circ}}{\mathbf{G_{max}}} \left(\mathbf{U_{max}} - \mathbf{U_{min}}\right) \tag{39}$$

In above equation, $\mathbf{U}_{max}$ designates maximum possible linear speed of an agent. Whereas $\mathbf{U}_{min}$ is the minimum linear speed of it. $\mathbf{G^\circ}$ term is the local value that an agent is experiencing. $\mathbf{G_{max}}$ is the maximum value of the local value defined within the environment. The essential modulating effect is not $\mathbf{G^\circ}$ but the ratio of it by $\mathbf{G_{max}}$.

3.2.2 Desired Distance Modulation

The second method to be used in order to achieve a foraging behaviour is the modulation of desired distance $\mathbf{d}^i_{des}$ which is presented in Equation 35. The desired distances are modulated in a way such that agents experiencing smaller local values have longer

desired distances while others have the reverse situation.

When the agents in a foraging group with desired distance modulation experience different local values, those agents calculate different desired distances. While the agents in more favorable regions produce virtual force vectors that are pushing them away from their neighbours to reach their desired distance values to others, agents located in less favorable regions produce force vectors pulling them towards agents which are located in favorable regions. This difference lead to a chaser evader case between them which ends up in regions with smaller local values. This fact also prevent a group already located in a favorable region from leaving there since the reverse situation occurs in that case.

The calculation of desired distance σ_i of an agent to others is as follows:

$$\sigma_{\mathbf{i}} = \sigma - \left(\frac{\mathbf{G}^{\circ}}{\mathbf{G_{max}}}\right)^{\mathbf{0.1}} \sigma_{\mathbf{var}} \tag{310}$$

In above equation, σ standing for an upper limit of $\sigma_{\mathbf{i}}$ while $\sigma_{\mathbf{var}}$ determine the interval for variation of $\sigma_{\mathbf{i}}$.

3.3 Motion Control

In order to produce motion control outputs for agents by using virtual force vectors, the resultant vector $\vec{\mathbf{f}}_i$ is resolved in local frame of agent i which is depicted in Figure 3.1. The motion control outputs are linear and angular speed values. The direction of linear speed is the current heading angle of the agent. This heading angle is defined around an axis orthogonal to local frame and angular speed is the changing rate of it. The calculation of linear and angular speeds is as follows:

$$\mathbf{U_i} = \mathbf{K_1 f_x} \tag{311}$$

$$\omega_{\mathbf{i}} = \mathbf{K_2 f_y} \tag{312}$$

Where $\mathbf{f_x}$ and $\mathbf{f_y}$ are components of $\vec{\mathbf{f}}_i$ on the respective axes of the local frame. The terms $\mathbf{K_1}$ and $\mathbf{K_2}$ are respective gains for linear and angular speeds.

The calculated linear speed is clamped between $\mathbf{U_{min}}$ and $\mathbf{U^i_{max}}$. To clarify, if the calculated linear speed in Equation 311 is higher than $\mathbf{U^i_{max}}$, it is forced to be $\mathbf{U^i_{max}}$ whereas if it is smaller than $\mathbf{U_{min}}$, it is forced to be $\mathbf{U_{min}}$.

CHAPTER 4

EXPERIMENTAL SETUP AND RESULTS

4.1 Simulations with Particle Agent Models

4.1.1 Agent Models

As the name implies, every agent is modelled as a particle in the simulations. After producing the desired linear and angular speeds, these values are directly applied to the model to calculate next position. To extend, no inertia, friction or mass are involved. The physical model or locomotion technique of the agents are ignored. Besides, the local values are sensed through the center point of the agent. Agents can only proceed in the direction of their current heading angle. The linear speed is applied at that direction whereas angular speed determines the rate of change of heading angle around an axis orthogonal to local frame of the agent.

4.1.1.1 Proximal Sensing

Agents are modelled to have distance information about nearby agents and environment boundaries. In addition to having it, distance information can be distinguished by agent about if it belongs to a neighbouring agent or a boundary. A neighbouring agent is sensed if it is located within a range of $\mathbf{D_p}$ around focal agent whereas a boundary should be closer than $\mathbf{D_r}$ to be sensed. The distance information is gathered with the relative angle information by focal agent. To clarify, the distance and the bearing of a nearby agent or a boundary is known by focal agent if they are close enough.

4.1.1.2 Sharing of Heading Angle Information

The heading angles of agents within a range $\mathbf{D_a}$ is known by focal agent as also its heading angle is known by others in that range. In order to have a convenient information sharing among agents, shared angles are resolved in a common frame.

4.1.2 Environment Model

The environment is modelled as a square field having an edge size of 20 m. Particle agents are not capable of pass or interact in any way with boundaries of that environment.

The environment includes local values stored in grids with infinitesimal size. Variation of the values are desired to be smooth in any direction all over the environment. Since there is consideration about having a non-artificial scalar field, it is produced by imitating a natural scalar quantity; temperature. In order to do this, an interactive multi physics software is used which is capable of simulate different heat transfer modes in a 2D environment [28]. Numerous heat sources with irregular shapes are created in this simulation environment and the natural heat transfer is simulated for a short period of time. Finally, couple of local and temporary heat sources (can be thought as local heat disturbances) are created to add a noise-like effect to scalar field. When scalar field seems appropriate to test, the values for a grid of 800 by 800 is collected. A sample screenshot from the heat transfer modelling software, Energy 2D, can be seen in Figure 4.1. By using these methods, 4 different scalar field models for the arena are produced. The overall looking of those scalar fields can be found at Figure 4.2. As color bars on the figures indicate, the blue regions are the regions with lowest values which is mentioned as "favorable regions" in Chapter 3. In addition, there is a noticeable difference in one of the simulation environments, the one in Figure 4.2d. The scalar field of that environment looks much more artificial and ordered when compared to the others. In fact, it is a pure and smooth gradient field with only one global minimum at its center. This environment is designed to make a comparison between group performances in a natural and irregular scalar field and an artificial and highly ordered scalar field.

In an arena with 20 m x 20 m dimensions with a 800 by 800 scalar field value matrix, the local values are stored in grids with a size of 0.025 m. This value defines the resolution of the scalar field.

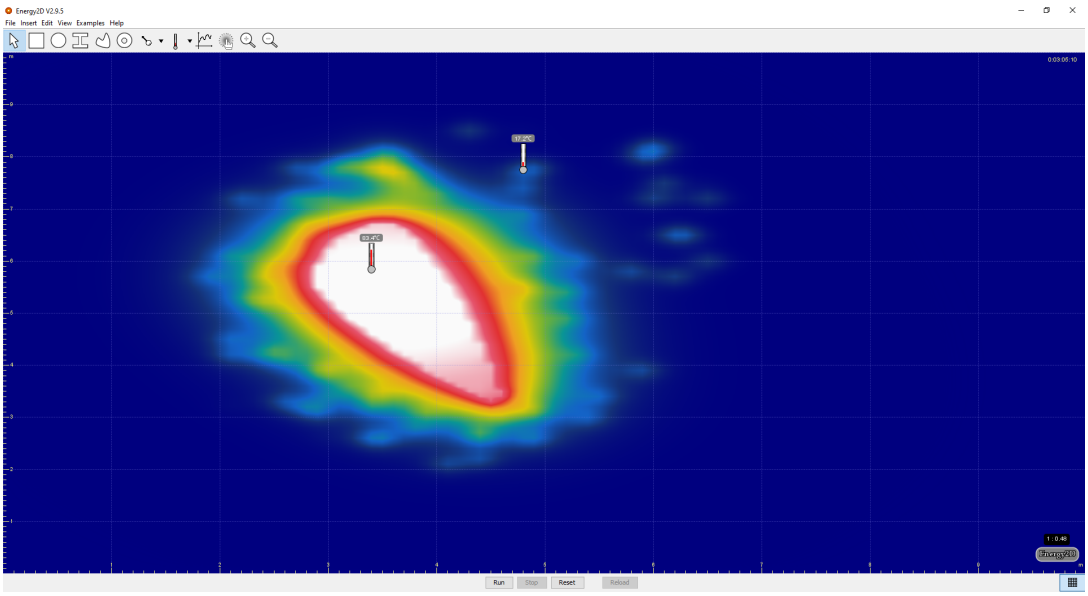
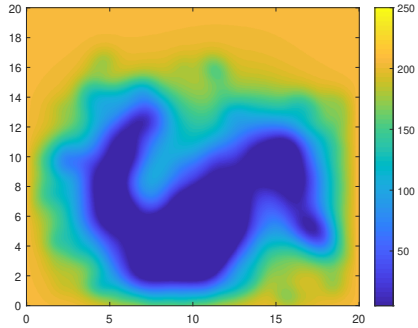


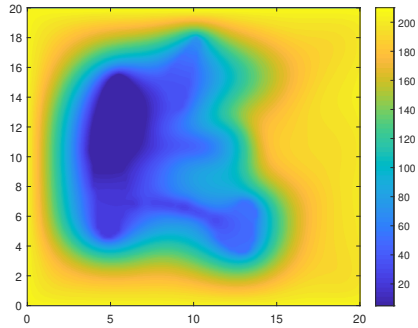
Figure 4.1: A screenshot from the Energy 2D heat transfer modelling software showing a heat source and the temperature distribution around it, visualized by a color map.

4.1.3 Implementation Details

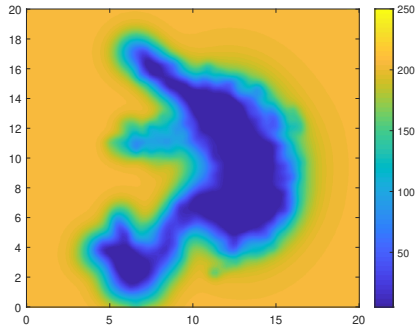
The simulations with particle agent models are conducted within MATLAB environment. Entire simulation process is carried out in a discrete manner. Throughout the simulations, time step was held constant and this time step is designated with $\mathbf{dt}$. Length of a simulation was 80000 steps for each trial. Particle agent models with first order dynamics do not involve any inertial elements. Hence, their new positions in the 2D arena and respective heading angles are calculated as a simple integration. This discrete integration is as follows;



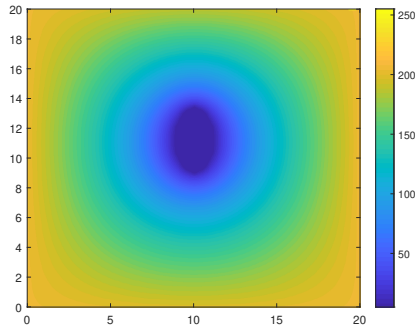
(a) Simulation Environment 1



(b) Simulation Environment 2



(c) Simulation Environment 3



(d) Simulation Environment 4

Figure 4.2: Simulation environments 1 to 4 in (a), (b), (c) and (d) respectively. The local values of the environment are mapped to different colors. Mapping of colors to local values is shown at respective color bars in figures.

$$\mathbf{X_i^{t+1}} = \mathbf{X_i^t} + \mathbf{U_i^t}\cos(\widehat{\mathbf{h_i^t}})\mathbf{dt} \tag{41}$$

$$\mathbf{Y_i^{t+1}} = \mathbf{Y_i^t} + \mathbf{U_i^t}\sin(\widehat{\mathbf{h_i^t}})\mathbf{dt} \tag{42}$$

$$\mathbf{h_i^{t+1}} = \mathbf{h_i^t} + \omega_\mathbf{i}^\mathbf{t}\mathbf{dt} \tag{43}$$

where $\mathbf{X_i^t}$, $\mathbf{Y_i^t}$ and $\widehat{\mathbf{h_i^t}}$ stand for the position components of the individuals in the 2D environment and their instantaneous heading angles.

The simulations on MATLAB demand high computational power especially for simulations with high agent numbers. In order to increase the speed of simulations, MATLAB scripts are converted into C source codes by using MATLAB C/C++ Code Generator Toolbox and these source codes are compiled in MATLAB. For every repetition, the execution of the simulation is assigned to a different core of the CPU of host PC and all cores are used simultaneously.

There are 3 different parameters changing between simulations; number of agents, the foraging method and simulation environment. The aim was to evaluate the effect of those parameters on foraging performance of the group.

In order to test foraging performance of the group with different methods, combinations of those methods are employed in simulations. There was 2 different foraging methods introduced in Chapter 3; Speed modulation and desired distance modulation. In addition to them, heading alignment is also considered to have an effect on foraging performance.

For every different choice of agent number, foraging technique and simulation environment, simulation is repeated 512 times. For each of them, the starting points of agents were different. These starting points are randomly chosen around a single initial point. For 10 agents, the starting points are placed in a square with an area of 1.5 m^2 centered at an initial point. It was 3 m^2, 4.5 m^2 and 6 m^2 for 20, 30 and 40 agents respectively. The position of the initial point is chosen randomly for each simulation.

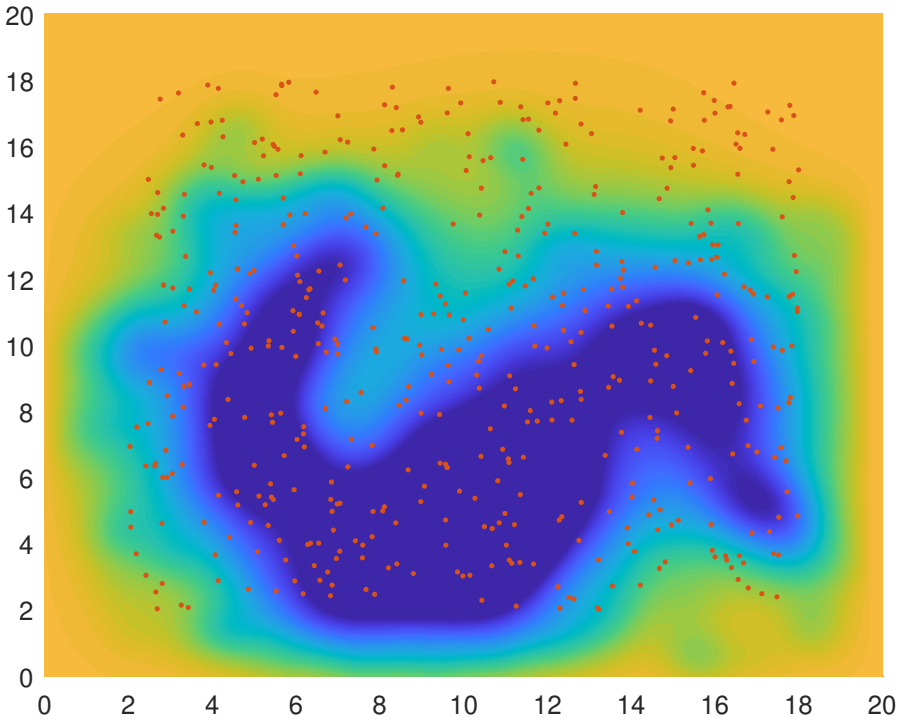


Figure 4.3: 512 randomly chosen initial points at simulation environment 1, designated with red dots. It is desired to spread the initial points over the environment because the effect of starting points of agents on foraging performance is wanted to be eliminated.

Figure 4.3 shows 512 different initial points for a chosen configuration on one of the simulation environment 1.

4.1.4 Metrics

There are two different performance metrics to compare group foraging success. They are number of groups and mean local value.

Number of groups reveals the ability of the swarm to stay cohesive during foraging. The definition of group can be made as this; A number of individuals can be defined as a single group if and only if every individual have at least one other in proximal sensing range of it. To measure number of groups, distance between any two mem-

bers are compared to proximal sensing range and then equivalence class method is applied on inter distances to identify difference equivalence classes representing different groups [29].

Mean local values are representatives of foraging performance of the group. The environment consists of grids measured by 0.025 m containing a certain scalar value. Agents can not sense any further scalar value other than the grid they are currently located in. Throughout the simulation, one in the 100 time steps, the mean of the local values from every agent in a group is calculated. Later, for the 800 measurements which are collected over 80000 simulation time steps are averaged again to reach a final and single scalar value to represent overall foraging success of the group. The calculation of the mean local values can be formulated as;

$$\overline{\mathbf{g}}_{\mathbf{t}} = \frac{\sum_{m \in N} \mathbf{G}^{\circ}_{\mathbf{m}}}{N} \tag{44}$$

$$\overline{\mathbf{G}} = \frac{\sum_{t=1}^{t_{sim}/100} \overline{\mathbf{g}}_{\mathbf{t}}}{t_{sim}/100} \tag{45}$$

In above equations, $\mathbf{G}^{\circ}_{\mathbf{m}}$ stands for the local value of agent m of all N agents in a time step, $\overline{\mathbf{g}}_{\mathbf{t}}$ for the momentary average of local values of the group, t_{sim} is the total time step number of a simulation and $\overline{\mathbf{G}}$ is the overall average of momentary group averages throughout a simulation.

4.1.5 Results

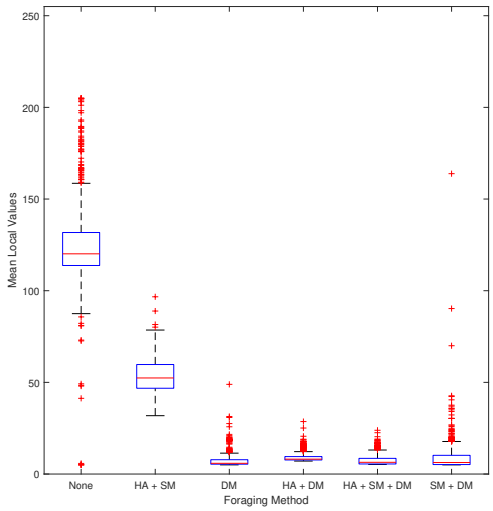
Table 4.1 lists the values of parameters used in the simulations.

Table 4.1: Parameter values or range of values used in particle agent model simulations

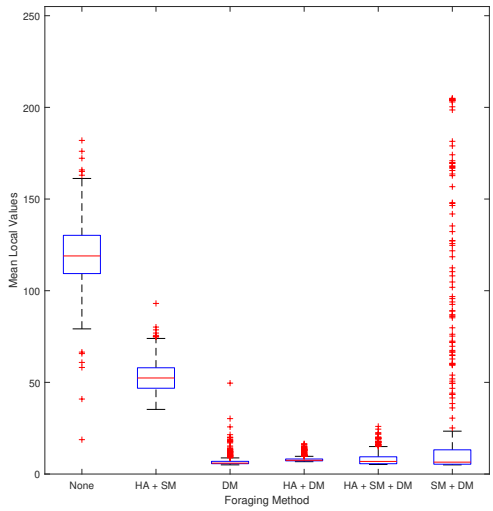
Variable	Description	Value
N	Total number of agents	10, 20, 30, 40
D_p	Proximal sensing range of an individual	2.0 m
α	Gain of the proximal virtual force vector	2.0
σ	Desired distance regulator term	0.4, 0.6
σ_{var}	The variance window of σ	0.2
ϵ	Multiplier for proximal force vector magnitudes	12.0
d_{des}	Desired distance between agents for constant σ	0.56
D_a	Heading alignment communication range	2.0 m
β	Gain of the heading alignment virtual force vector	2.0
γ	Gain of the boundary avoidance virtual force vector	1.0
k_{rep}	Multiplier for boundary avoidance vector magnitude	80
L_0	Relaxation threshold for boundary avoidance	0.5 m
K_1	Gain of linear speed	0.5
K_2	Gain of angular speed	0.06
U_{max}	Maximum linear speed allowed	0.075, 0.05 m/s
U_{min}	Minimum linear speed allowed	0.01 m/s
ω_{max}	Maximum angular speed allowed	$\pi, \pi/4$ rad/s
G_{max}	Maximum local value in the environment	255
G_{min}	Minimum local value in the environment	0
t_{sim}	Total simulation time steps	80000
d_t	Time step length	0.1 sec

The mean local values from the particle agent simulations are presented by grouping them according to the simulation environment. Under each group, there are subplots for changing number of agents from 10 to 40. In each subplot, mean local values are placed by employed foraging method combination. For each method combination, results for 512 simulations are presented.

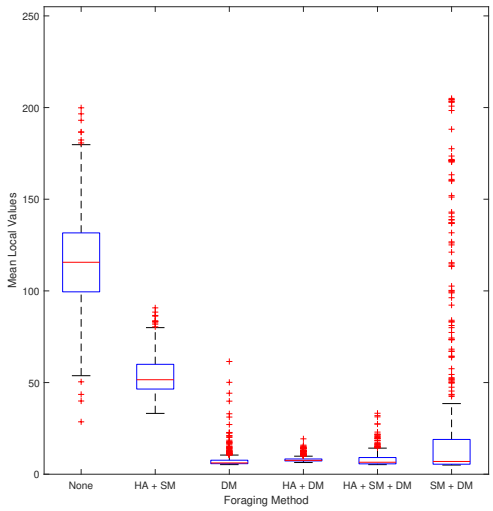
Figures 4.4 to 4.7 contain mean local values for all simulations. For each foraging method, size of group and simulation environment there are 512 results obtained. It is convenient to represent distribution of those results in a range between maximum and minimum of local values in the environments. In order to achieve that, box plots are used. In these box plots, central lines represents the median value of 512 repetitions, upper and lower edges of the boxes stand for 25th and 75th percentiles. The maximum and minimum points are represented with edge of dashed lines that are vertically crossing the boxes. The data points which can be considered as outliers are presented outside of the boxes with a $+$ sign. Further information about box plots could be obtained from [30]. In the horizontal axes of figures, the foraging method of group is labeled for each set of data. By the $None$ label, as name implies, none of the foraging methods is used, neither there is no heading alignment effect on the agents. For the rest, HA stands for heading alignment, SM stands for speed modulation and DM stands for desired distance modulation method.



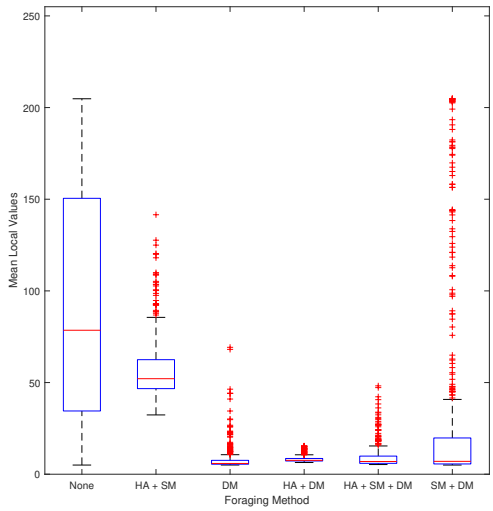
(a) Mean local values of 10 agents in simulation environment 1



(b) Mean local values of 20 agents in simulation environment 1

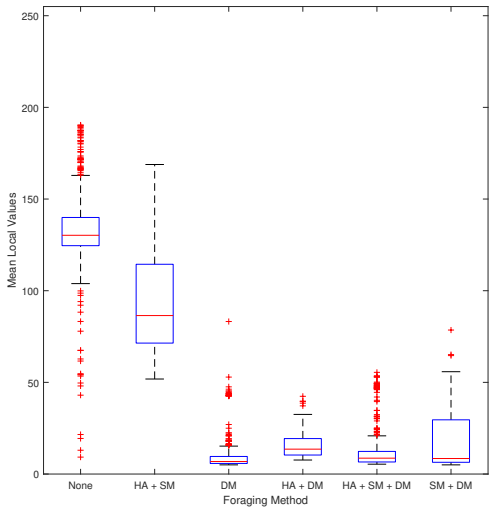


(c) Mean local values of 30 agents in simulation environment 1

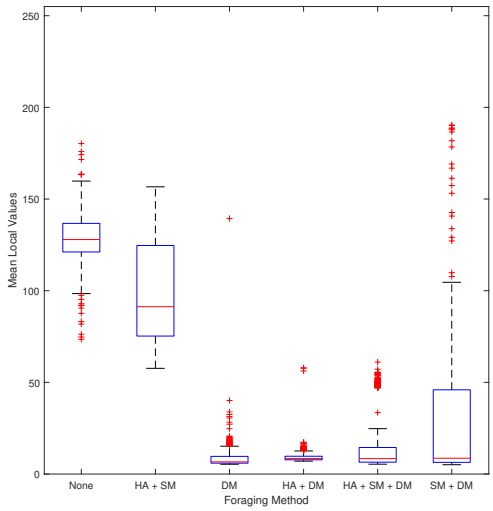


(d) Mean local values of 40 agents in simulation environment 1

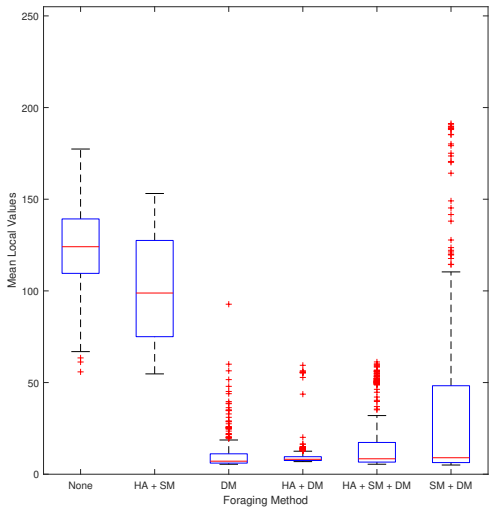
Figure 4.4: Mean local values of various number of agents in simulation environment 1 represented as box plots



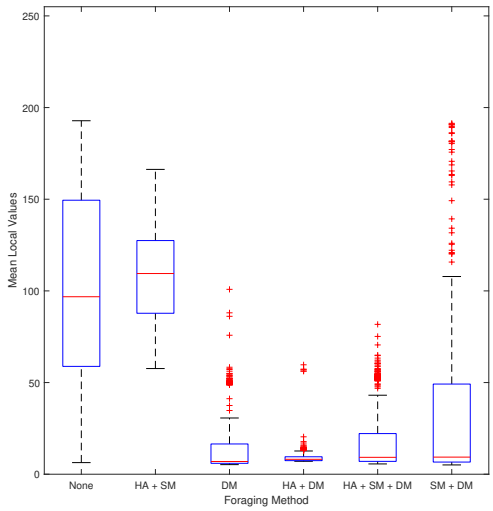
(a) Mean local values of 10 agents in simulation environment 2



(b) Mean local values of 20 agents in simulation environment 2

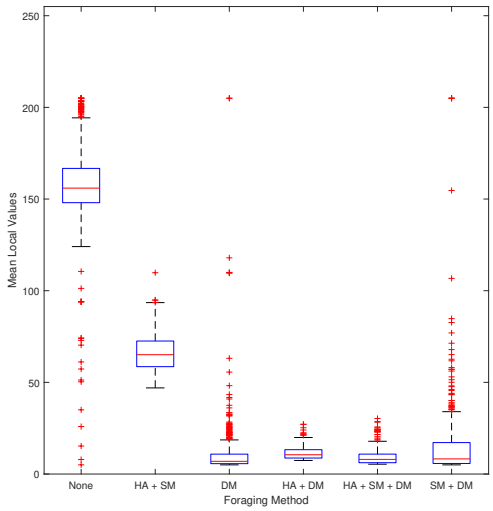


(c) Mean local values of 30 agents in simulation environment 2

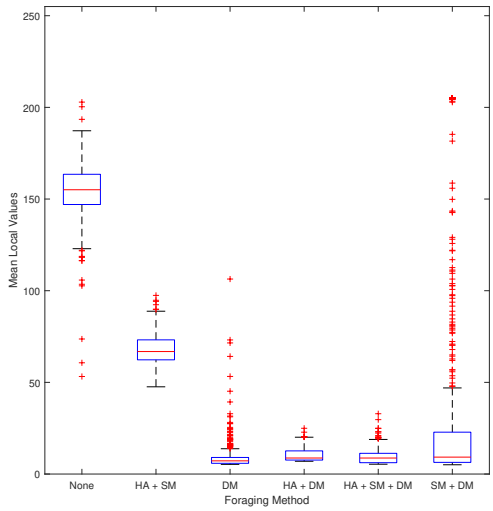


(d) Mean local values of 40 agents in simulation environment 2

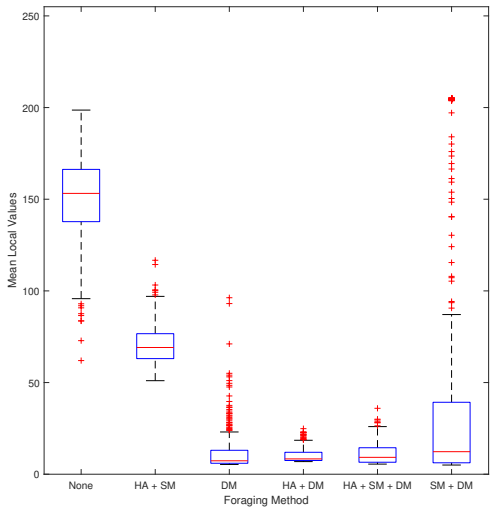
Figure 4.5: Mean local values of various number of agents in simulation environment 2 represented as box plots



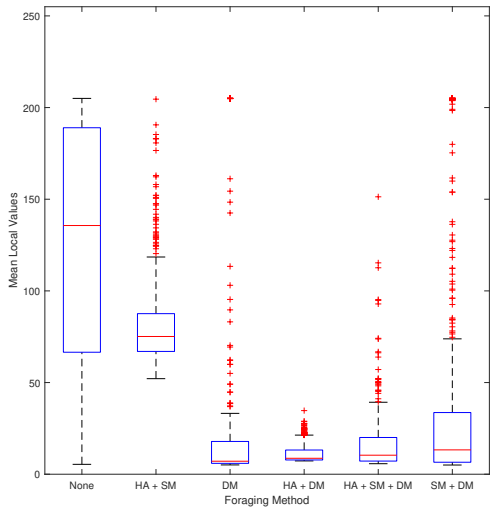
(a) Mean local values of 10 agents in simulation environment 3



(b) Mean local values of 20 agents in simulation environment 3

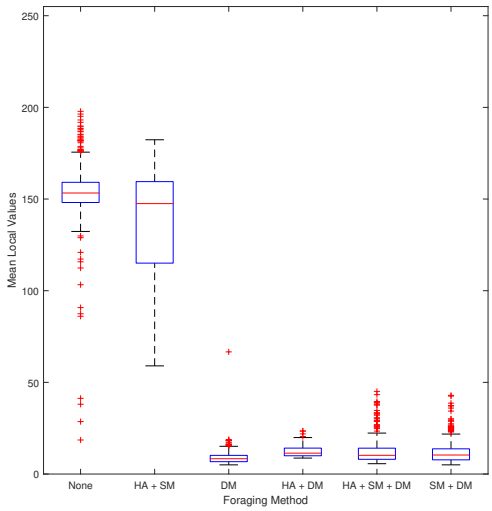


(c) Mean local values of 30 agents in simulation environment 3

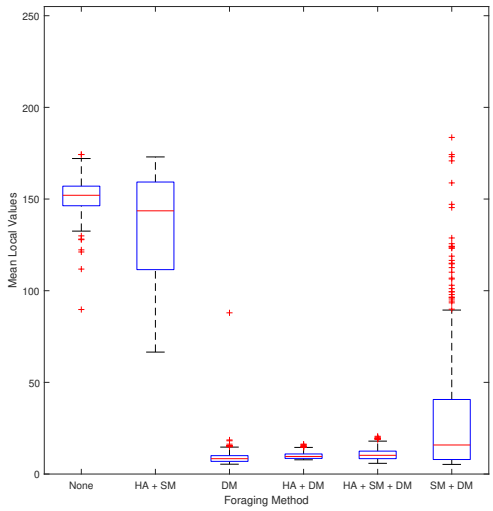


(d) Mean local values of 40 agents in simulation environment 3

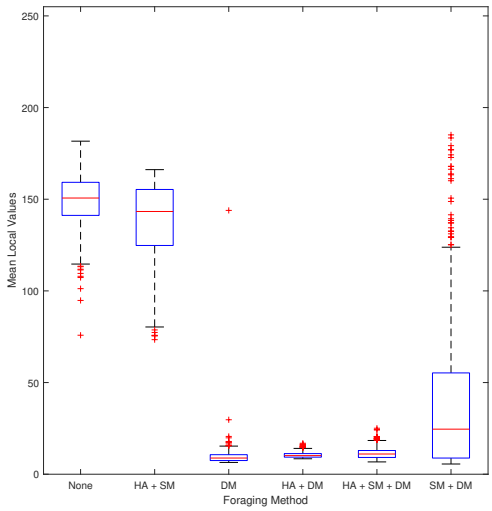
Figure 4.6: Mean local values of various number of agents in simulation environment 3 represented as box plots



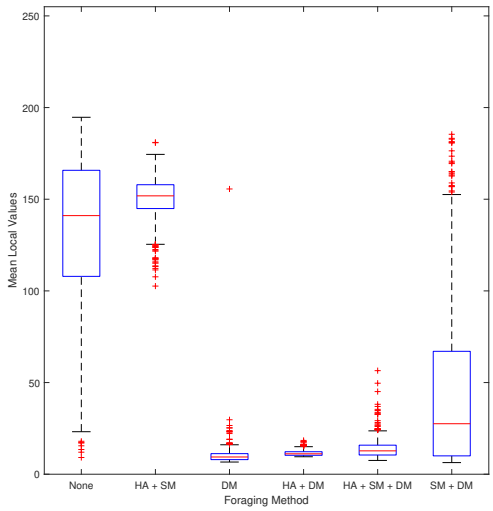
(a) Mean local values of 10 agents in simulation environment 4



(b) Mean local values of 20 agents in simulation environment 4



(c) Mean local values of 30 agents in simulation environment 4



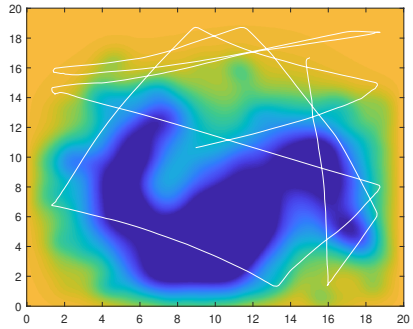
(d) Mean local values of 40 agents in simulation environment 4

Figure 4.7: Mean local values of various number of agents in simulation environment 4 represented as box plots

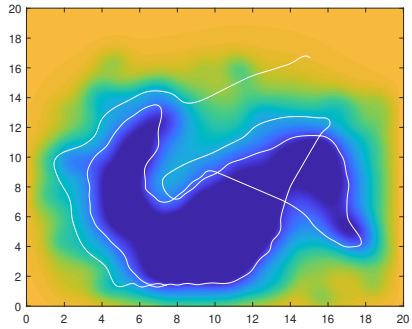
When Figure 4.4 is observed, which shows mean local values for simulation environment 1, effect of modulation methods are clear by their smaller median lines when compared to no foraging method case. Among them, the ones with desired distance modulation are demonstrating a far better foraging performance with much lower median values and smaller number of outlier points. Results of simulations for changing group size are showing similar results. The group size does not seem to be effecting foraging performance in environment 1. A slight difference can be observed for simulations with 40 agents. In this figure, the result of a group with no foraging method is spread over a wider range. The least number of outlier points can be observed for 10 agents. Figure 4.5 presents results for simulation environment 2 where desired distance modulation outperforms other methods in all group sizes. The performance of speed modulation is decreased when compared to environment 1. Increasing group size also widens the mean local value range for no foraging method case. In environment 3 where its results are presented in Figure 4.6, although it is outperformed by desired distance modulation, speed modulation method gives the best results among all environments. Environment 4 which has a distinct distribution of local values, demonstrates the worst performance for speed modulation in Figure 4.7. Its performance is getting closer to performance of no foraging method case. Desired distance modulation is outperforming others in this environment in all group sizes. In addition, the narrower ranges for no foraging method are observed in this environment.

The number of groups metric is not visualized since in total of 49152 simulations (for 4 different environments, 4 different group sizes, 6 different foraging methods and 512 repetitions for each), there was no separation and the group was a single and cohesive one from start to finish.

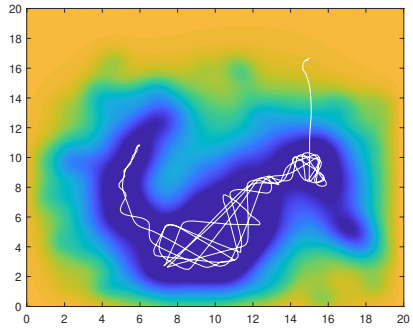
Finally, it is beneficial to present trajectory of centroid of a 20 agent group with different methods on a sample environment. The trajectories are visualized in Figure 4.8.



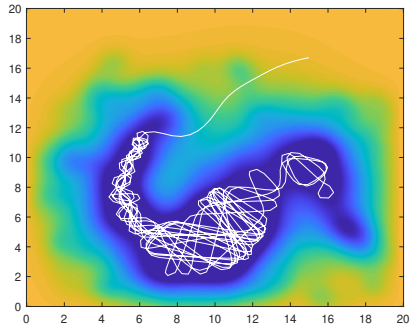
(a) $NONE$



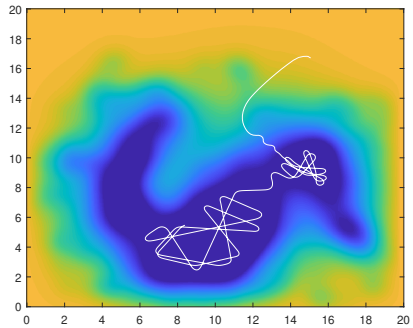
(b) $HA + SM$



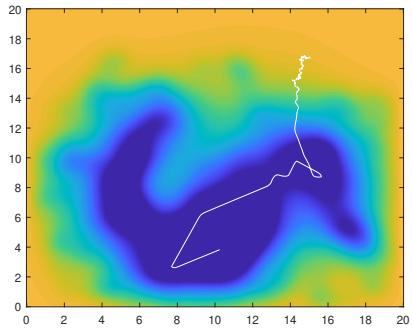
(c) DM



(d) $HA + DM$



(e) $HA + SM + DM$



(f) $SM + DM$

Figure 4.8: Trajectory of centroid of group consisting of 20 agents in simulation environment 1 with various foraging methods

4.2 Physics Based Simulations

Simulations with robot dynamics act as a bridge between a pure mathematical algorithm and real world results of it. They give opportunity to identify different source of errors through the conversion from abstract to physical.

The physics based simulation experiments are conducted with dynamical models of small sized aerial vehicles. These small flying robots are safe to use indoors and they are appropriate platforms to be used as proof of concept tools before developing methods to use with bigger flying platforms. Small aerial vehicles could be fast and agile which are beneficial properties to test a flocking or collective foraging technique. Those advantages bring complexity to control systems and intolerance to any crash on air. The simulated models must be realistic as much as possible and must carry the constraints of both physical system and its control architecture. This the only possible way to avoid any unexpected behaviour or error in real applications, yet it still does not guarantee it.

While the GAZEBO Simulator handling real world physics, the software that involves control, feedback and actuation architecture of flying robots is simulated with a Software in the Loop (SITL) firmware as shown in Figure 4.9. The SITL firmware is exactly the same with that runs in the real aerial vehicles except the sources of its inputs and destination of its outputs. In real world system, sensory devices send related signals they produce to central controller according to physical state of vehicle and that central controller produces commands and send them to components that are responsible for control of actuation systems - which are motors. For the SITL application, the same software with central controller exists but instead of real sensory devices, it accepts inputs from physics simulator as if they are real world physical states and sends actuation control signals to systems that are modelled in the same physics simulator. Thanks to powerful central processing units of personal computers these days, multiple software imitations of aerial vehicles can be run simultaneously.

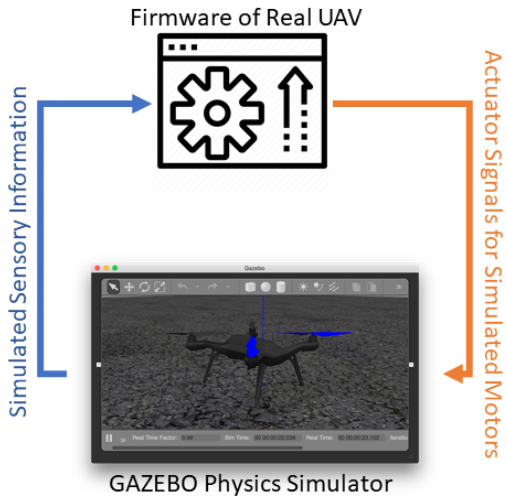


Figure 4.9: The framework of physics based simulations. The firmware of the real aerial vehicle works in SITL form and accepts simulated sensory data generated by GAZEBO physics-based simulator. Firmware produces actuator commands to be executed by GAZEBO.

4.2.1 Simulation Environment

The simulation environment is a square shaped field with 10 m edge length. The reason behind this choice is the fact that the flight speeds of real robots are limited. In order to be consistent between physics based simulations and real robot experiments in terms of flight time, the size of simulation environment is decreased.

The local values are assigned in an identical way with particle agent simulations and same values are used for 4 different environments. Since the edge lengths of the environment are decreased from 20 m to 10 m, the grids carrying local values have an edge size of 0.0125 m instead of 0.025 m in physics based simulations.

4.2.1.1 GAZEBO Simulator

GAZEBO simulator [31] is an open-source, physics-based robotic platform simulator. A robot or component model with mass, inertia or friction characteristic can be constructed in GAZEBO. It is possible to develop sensor models such as inertial

measurement units (IMU), RGB cameras or flow cameras with desired noise levels. Actuators can also be modelled such as motors and propellers. It allows users to create dynamical models of aerial vehicles. GAZEBO simulator also allows users to create visual models that is almost the same with real robots by the help of its integration level with several computer aided design tools. These capabilities of GAZEBO allow users to model a robotic platform conveniently with the least possible effort.

The platforms that are used in real robot experiments are Crazyflie 2.1 aerial vehicles. They are developed and sold by Bitcraze AB [32]. Numerous studies that employ GAZEBO simulator for research purposes can be found in literature. The tools used in those studies are mostly shared with the community in an open-source form. The RotorS project [33] is one of the most comprehensive studies and it involves sensor and actuator models related to Crazyflie. Moreover, there are other studies that are focused on modelling of Crazyflie in GAZEBO. This modelling efforts integrate those sensor and actuator models on a highly accurate Crazyflie physical model. The results are also shared as open-source [34]. Also firmware of Crazyflie platform is modified to work in the SITL method on a personal computer so that one can simulate the flight of a Crazyflie with the same firmware as in real life, with accurate physical properties. Finally, Gazebo simulator provide an opportunity to visualize models with a realistic rendering. This is quite helpful at the beginning of experiments while constructing a working simulation framework before proceeding to repetitive experiments. Figure 4.10 show a realistic rendering of a Crazyflie platform side by side with a real photograph of the platform.

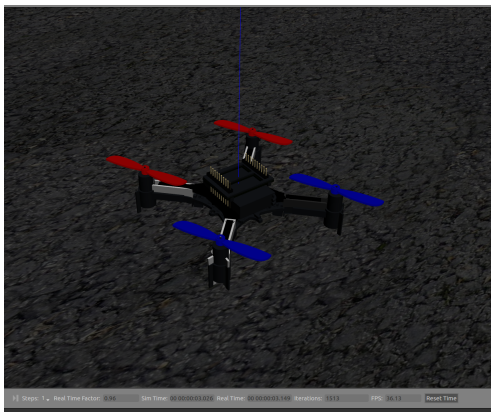
4.2.1.2 Robot Operating System (ROS)

Robot operating system (ROS) is a set of libraries, tools and programs that works under LINUX [35]. Although, it is mostly used on robotic applications, application areas are not limited with those. ROS requires programs that are written either in Python or C++ software languages with some pre-defined functions and usages that is specifically developed for ROS. As a main functionality, ROS generates an application specific communication network and a operating system working under LINUX. It handles all the synchronization, routing and communication tasks for a constructed

communication architecture and produces outputs to designated routes created for specific applications. A ROS framework on a computer placed on a real robot can accept sensory readings such as an image frame from a camera, a signal coming from a serial communication infrastructure or a piece of information from a world wide network. It can produce various outputs that user has programmed beforehand such as motor control signals, estimation about an environment or a signal that robot is programmed to actuate with. ROS provides opportunity to develop specifically ROS related libraries, programs and even algorithms that are shared among the community. This case accelerated all the studies involving ROS especially on robotics. Nowadays, when a new hardware is offered for sale, it directly comes with its related ROS libraries, drivers and tutorials to create a demand by ROS users. LIDAR's, motors, RGB and depth cameras and even aerial vehicles could be found with their corresponding libraries, drivers, applications, in other words, ROS packages.

ROS is highly integrated with GAZEBO simulator. This integration allows researchers to construct their software architecture on ROS and use this architecture to make experiments with their robot models on GAZEBO. After that, when transition from software world to real world is required, only the location of software has to be changed.

(a) A photograph of Crazyflie 2.1



(b) A screenshot from GAZEBO simulator showing Crazyflie model

Figure 4.10: Visuals showing both real looking of Crazyflie and rendering of its model in GAZEBO physics-based simulator

Same situation is applicable for Crazyflie aerial platform. One comprehensive work putting a controller ROS program and GAZEBO models together can be found at [35].

In this study, all the programs required to control Crazyflie for its flight are implemented in ROS. The high level controller programs are implemented by using Python because of it's efficient and easy to use structure. High level controller produces reference velocity or position commands in addition to commands like start, stop, take off or land. All of the low level controller programs such as position controller, velocity controller, attitude controller, motor drivers etc. are located within firmware and maintained by developers of Crazyflie. High level controller only tells what action to take, which velocity to maintain or which position to be.

The programs in ROS, which are developed for physics based simulations, can be used for real robot experiments with a slight change; Destination of commands that are produced by high level controller and source of feedback about platforms. By changing only two variables, the programs can command real robots.

4.2.2 High-Level Control Node

When a Crazyflie SITL has started on a processing unit of a computer, it can be thought of Crazyflie is turned on from it's power button. In order to obtain meaningful sensory information for initialization of low level control software, SITL should be receiving sensor feedback. Thus, first task is to start GAZEBO simulation beforehand and start the real time physics calculations for the simulated world that sensors, actuators and all other components of Crazyflie are located in. Then, SITL software starts and Crazyflie turns alive with its full functionality.

A program that is written within ROS can be called as a ROS node. The high level control node takes information from ROS communication network about current physical state of platform. A central and single controller does not fit the nature of previously proposed methods. Unfortunately both hardware and software architecture of Crazyflie do not allow a fully autonomous decision making and flight without taking commands from a central unit. Thus, the distributed nature of proposed meth-

ods will be realized by emulating it. To say, none of the platforms uses it's global position information to decide which action to take for collective foraging. Also none of them receives information about global states of other members or overall information about the environment. The functions, that make group of robots to realize a collective foraging, require information about relative range and bearing of nearby agents. They also use local value of platform's instantaneous position. Only if heading alignment technique is employed, the current heading angle of nearby agents are also taken into account by those functions. The information flow for a Crazyflie and overlook of high-level controller node can be found at Figure 4.11.

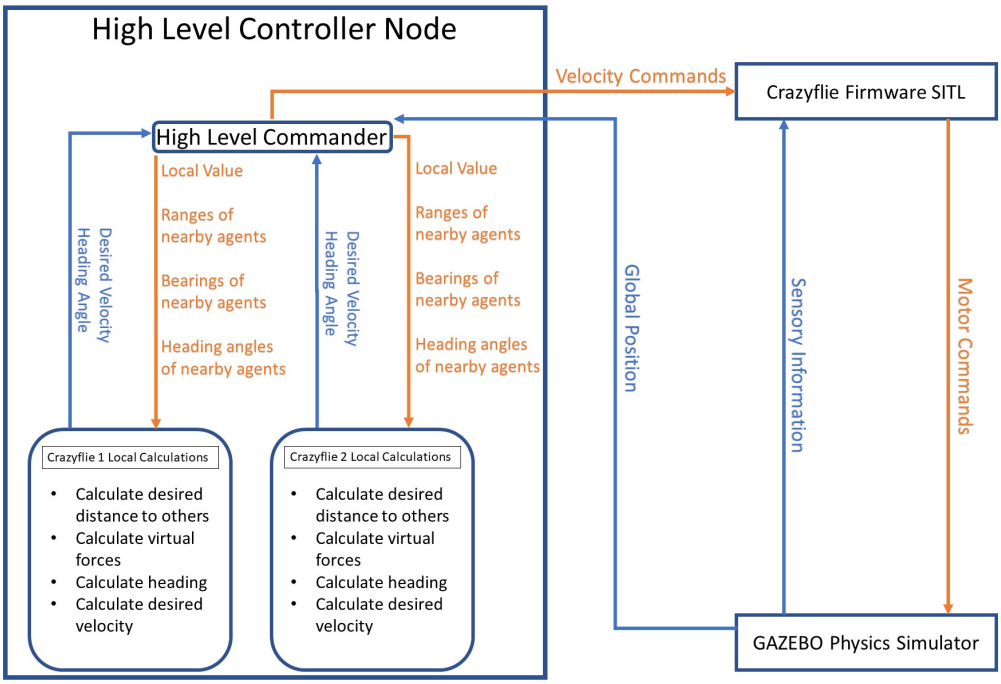


Figure 4.11: A diagram showing the information flow between high level controller node, firmware in SITL form and GAZEBO physics-based simulator

4.2.2.1 Proximal Sensory Information

In an ideal case, all of the Crazyflie's should have a sensor extension on top that can sense nearby agents and environment boundaries. It should also capable of making a distinction between agents and boundaries. Unfortunately it is not possible to have

that sensor extension for Crazyflie at the moment. In order to emulate that sensor extension, the architecture of software constructed in a special way. High level commander portion of high level controller node receives global position information of every agent. By using pre-defined parameter for maximum sensing range (D_p), the commander calculates and generates a neighbourhood for every agent by involving only the agents that are closer to the focal agent than D_p. Then it calculates the relative distances together with relative bearings for each agent and sends a message that only involves those values without any ID. This message is basically an array of consisting of relative range and bearing values. By this way, every agent do calculations with local in information. If one day, a sensor extension is developed to sense nearby agents and send that information to central control unit of Crazyflie, none of the proposed algorithms will be prone to change.

4.2.2.2 Heading Angle Communication

Ideal way of sensing instantaneous heading angles of other agents would be doing it locally with a special sensory system that does not require any communication. Less ideal but more applicable way would be sharing instantaneous heading angles with nearby agents with kind of a mesh network system with limited range. Unfortunately, none of them could be realized with current state of technology about Crazyflie. Since then, heading angle communication will be emulated in software. As in proximal sensory information case, high level commander node listens for calculated heading angles from agents, by considering pre-defined range designated for maximum heading angle communication (D_a) shares it with appropriate agents without any ID information. This way, when heading alignment technique is employed, the information sharing is done in a local and anonymous manner.

4.2.3 Implementation Details

In physics based simulations, for a foraging method in chosen simulation environment, flights are repeated 20 times. In every repetition, the start points of agents are changed. 20 different start points for each agent are chosen such that they cover the

whole field. In addition, their heading angles are chosen randomly at the beginning. In Figure 4.12, the 20 distinct initial points for simulation environment 1 is visualized. Also, corresponding start points of 6 Crazyflies are visualized around those initial points. Black dots in the arena are initial points while the white and smaller dots are start points of each Crazyflie. Figure 4.12 gives an insight about how start points of group members are spread around the arena.

The local values of simulation environment are not modeled in GAZEBO. Instead, Crazyflie models are freely flying in void without any obstacles. Their instantaneous global positions, which are resolved in a common frame, are used to find which grid the focal agent is located. The local value of this grid is calculated by high level controller.

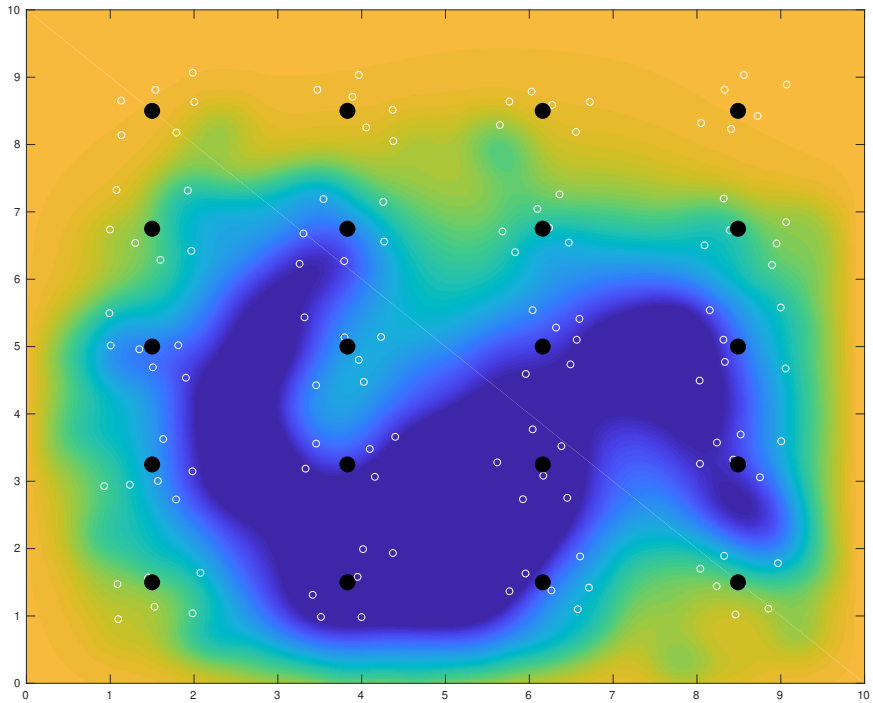


Figure 4.12: 20 initial points, designated with black dots. The white circles around them show the start points of 6 agents around a chosen initial point. It is desired to spread the initial points over the environment because the effect of start points of agents on foraging performance is wanted to be eliminated.

Although the CPU of host PC is more powerful when compared to Crazyflie's, 6 Crazyflie firmware running simultaneously with ROS and GAZEBO are enough to overload that CPU. Especially rendering tasks of Crazyflie models are heavy enough on their own to slow down the process. To solve this case, ROS and MATLAB integration is used. The Robotic Systems Toolbox of MATLAB is employed. A MATLAB script is written that listens to positions, heading angles and local value readings of agents. MATLAB only acts as a listener and data logger here to avoid rendering tasks of GAZEBO. In all simulations, the graphical user interface and visualizations of GAZEBO are turned off. The data is fed directly to MATLAB. Then, it is saved in appropriate data storage format to use it later to evaluate foraging performance of group.

At particle agent simulations, 6 different methods of foraging are used. In physics based simulations, only the most promising ones are chosen and speed modulation is never employed. When the mean local values of particle agent simulations are considered, it can be seen that the groups with only desired distance modulation are far better than others. Since, only desired distance and heading alignment options are used in physics based simulations.

4.2.4 Results

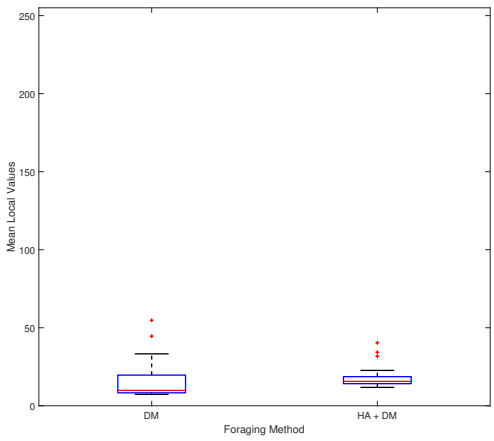
Table 4.2 shows the parameter values used in foraging methods employed in simulations. The parameters show slight deviations from values of particle agent simulations. The possible reasons are discussed in detail at Chapter 5.

Table 4.2: Parameter values or range of values used in physics based simulations

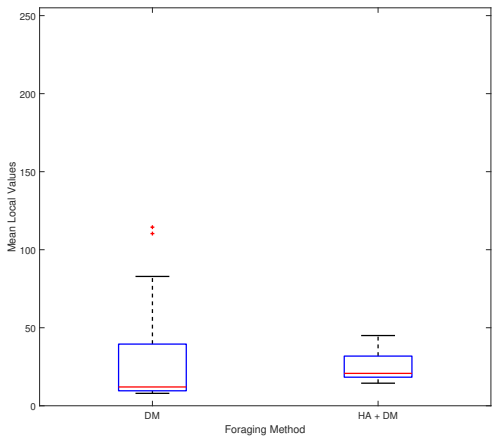
Variable	Description	Value
N	Total number of agents	6
D_p	Proximal sensing range of an individual	2.0 m
α	Gain of the proximal virtual force vector	3.0
σ	Desired distance regulator term	0.4, 0.8
σ_{var}	The variance window of σ	0.4
ϵ	Multiplier for proximal force vector magnitudes	12.0
d_{des}	Desired distance between agents for constant σ	0.56 m
D_a	Heading alignment communication range	2.0 m
β	Gain of the heading alignment virtual force vector	1.0
γ	Gain of the boundary avoidance virtual force vector	1.0
k_{rep}	Multiplier for boundary avoidance vector magnitude	50
L_0	Relaxation threshold for boundary avoidance	0.5 m
K_1	Gain of linear speed	0.5
K_2	Gain of angular speed	0.06, 0.03
U_{max}	Maximum linear speed allowed	0.25 m/s
U_{min}	Minimum linear speed allowed	0 m/s
ω_{max}	Maximum angular speed allowed	$\pi/2, \pi * 4$ rad/s
G_{max}	Maximum local value in the environment	255
G_{min}	Minimum local value in the environment	0
t_{sim}	Collected data points	1400
d_t	Time step length	0.01 sec

The mean local value and number of group metrics are used to evaluate foraging performance of group. Since there are less methods are used in physics based simulations than particle agent simulations and agent number is same for all experiments, the mean local value results are grouped for 4 different simulation environments. They are presented at Figure 4.13 with 4 sub figures. Every sub figure presents mean local values for each simulation different.

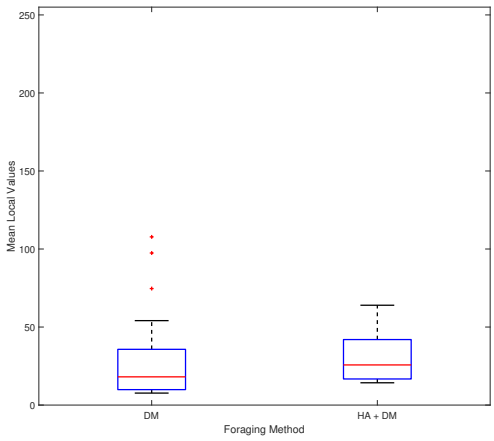
Figure 4.14 shows trajectory of centroid of a group in simulation environment 1, employing two different foraging methods.



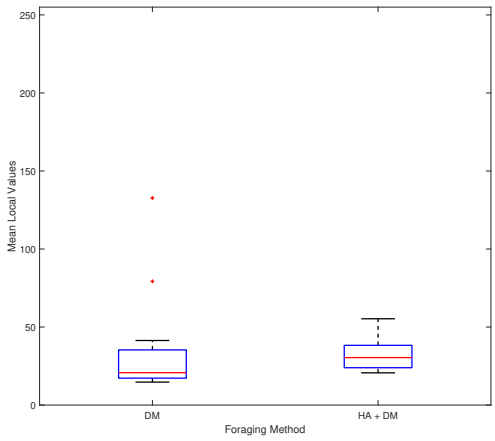
(a) Simulation Environment 1



(b) Simulation Environment 2

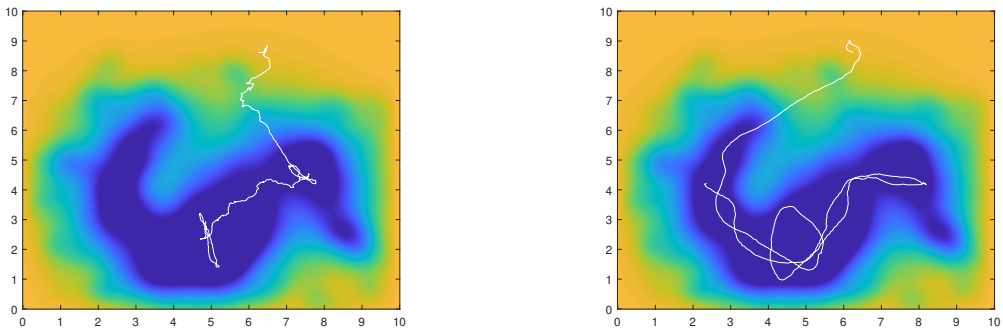


(c) Simulation Environment 3



(d) Simulation Environment 4

Figure 4.13: Mean local values of 6 agents in GAZEBO simulations on 4 different simulation environments, represented as box plots



(a) DM

(b) $HA + DM$

Figure 4.14: Trajectory of centroid of a group consisting of 6 agents in simulation environment 1 with various foraging methods on GAZEBO physics-based simulator

When Figure 4.13 is observed, it can be concluded that both foraging methods indicate comparable performances. The performances agree with particle agent model simulations. The heading alignment slightly enhances the foraging performance. Effects of simulation environments on the foraging performance are not clear to make a comparison between them.

4.3 Real Robot Experiments

The proposed collective foraging algorithms are implemented and tested with real flying robots as a final step. The experiments with real robots prove the robustness of proposed algorithms on systems with dynamical properties of real aerial vehicles and different kinds of imperfections and disturbances occurring related to real world conditions. The physics based simulation experiments have introduced the system dynamics, communication and actuation delays together with inertial effects but all of those effects were based on mathematical models. As expected, mathematical models are not perfect. Although they give an insight about what is the effect of real world conditions on foraging performance, the real world conditions will always have some unexpected or even unknown influence.

The robots were chosen to be Crazyflie 2.1 autonomous aerial platforms. The group

consists of 4 platforms flying simultaneously and they are controlled from a station PC. Although the algorithms are designed to work in a distributed manner, the physical capabilities of Crazyflie platforms does not allow it. Instead, distributed control architecture is emulated in controlling software which works on station PC. Station PC listens for instantaneous states of agents, makes calculations for agents and sends commands to them. The calculations are done as every agent uses its own processing unit, can only communicate with or sense agents in a limited range ($\mathbf{D_p}$, $\mathbf{D_a}$) and can only sense the local values of the environment in their instantaneous positions. By doing so, the fundamental assumptions made for particle agent models and physics based models are preserved.

4.3.1 Experiment Environment

The environment used for real robot experiments is an indoor flight arena covered with safety nets. The arena size is 5 m by 5 m. As same with particle agent and physics based simulations, the flight arena does not involve any obstacles. The local values that are used for foraging are assigned in the same way before and identical scalar fields are used. Since the edge size of the environment has changed, the grids carrying local values have an edge size of 0.00625 m for real robot experiments.

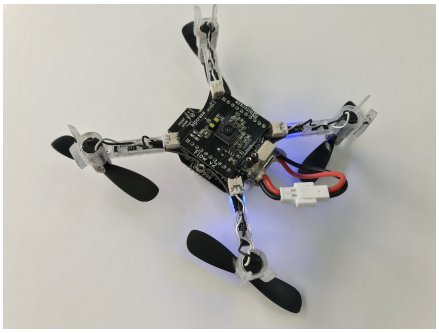
4.3.1.1 Crazyflie 2.1 and Positioning Infrastructure

The Crazyflie 2.1 is an autonomous flying robot. This flying robot has 4 rotary wings actuated by 4 coreless brushed motors. Its body consists of the electronic circuit board carrying all components such as the CPU, communication related and sensory elements. Motors and electronic components are powered by a single cell lithium polymer battery with capacity of 350 mAh. A bare minimum configuration of Crazyflie weighs 27 g but with necessary positioning hardware, the weight increases to 37 g. Carrying additional hardware, platform offers a safe flight for nearly 4 minutes. A bare minimum flying configuration of Crazyflie is not capable to perform an autonomous flight since the IMU it carries can not handle it on its own because of low precision and drifting issues. To achieve an autonomous flight, two additional hard-

ware are used: A flow deck consisting of a flow camera together with a LIDAR and UWB (Ultra Wide Band) positioning deck to use alongside with UWB nodes placed around the flight arena. The flow camera enhances the velocity estimation of the robot while LIDAR gives a precise measurement about the altitude. The UWB deck estimates the global position of the platform using the information it receives from UWB nodes. Photographs of Crazyflie 2.1 with UWB deck on top and flow deck on bottom of it can be seen at Figure 4.15. 4 Crazyflie's flying simultaneously can be seen at Figure 4.16, during an experiment.

The UWB system works very similar to a GPS system. A node estimates its global position by using information extracted from the communication it is doing with other nodes. The nodes that are placed around the arena have fixed positions. Nodes sends messages consisting of their position fused with a time stamp. The time stamp is extracted from a clock that is synchronized for all nodes. A receiver node listens all available messages and compare the time stamp of sender with time of reception. The time difference gives an estimation about how far the sender. Since the sender adds its fixed position, the receiver have an estimation about the global position. By increasing number of sender nodes, receiver improves its global position estimation. The setup used in this study has 8 sender nodes. Every robot has its own receiver node. Crazyflie platform with an UWB positioning deck can estimate the global position with an accuracy of 10 cm. In order to improve accuracy, the velocity estimation coming from flow camera is fused with global position estimation.

(a) Crazyflie 2.1 used in experiments, a view from top



(b) Crazyflie 2.1 used in experiments, a view from bottom

Figure 4.15: Photographs of Crazyflie 2.1 used in experiments, from different angles

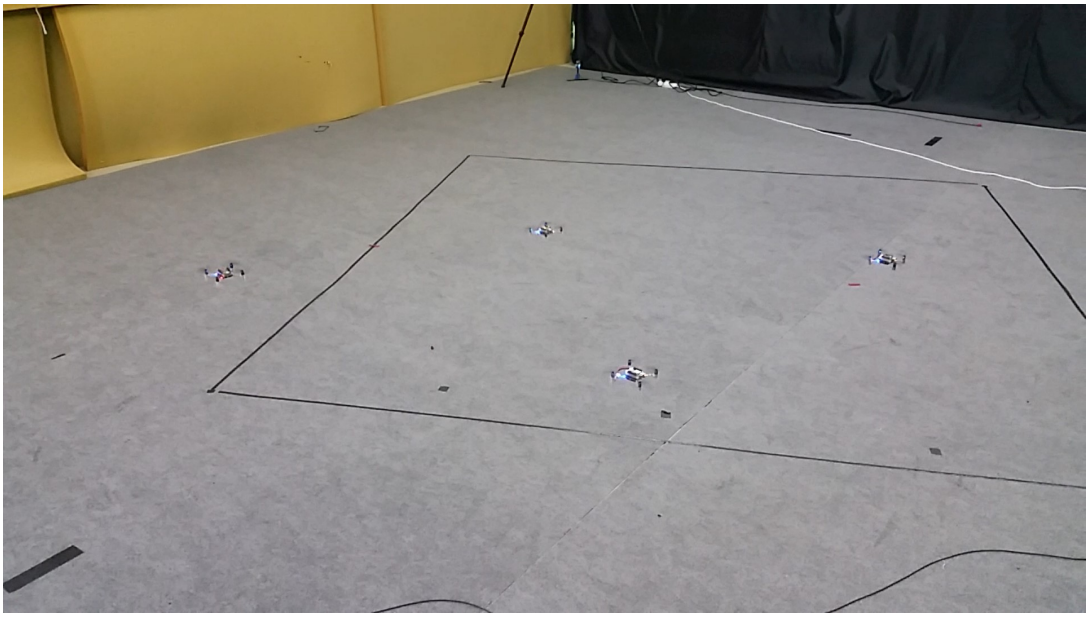


Figure 4.16: 4 Crazyflie's on the air during an experiment.

Every agent calculates its position on board. Also all low level control tasks are handled on board. Yet agents need flight commands to come from a station PC. The communication between a Crazyflie and the station PC is done by a serial radio link working in 2.4 GHz frequency band.

4.3.1.2 Control of Crazyflie 2.1 over ROS

As an advantageous outcome of the software architecture built for physics based simulations, the effort to modify this architecture to work with real robots instead of simulated ones can be kept minimum. The high level controller node used for physics based simulations is used for real robots as well with a few modifications. The first modification is to change the communication ports of the agents. In physics based simulations, the firmware of Crazyflie is used with SITL technique and the communication ports were located in the station PC itself. The high level controller node was sending flight commands and receiving state information of agents from that ports. For real robots, these ports are changed to corresponding connections of serial radio links. The other modifications are related to parameters of foraging algorithm. They are needed since capabilities and behaviours of flying robots are not exactly the same

with simulated ones.

Since the high level controller software is not changed except communication ports and parameter values, the models for proximal sensory information and heading angle communication are same with physics based simulations.

4.3.2 Implementation Details

Among 4 different environments previously used for particle agent and physics based simulations, environments 1 and 4 are used for real robot experiments. Choosing 2 distinct initial points of agents, experiments are done twice for every environment. For every trial, different foraging techniques are used. These foraging techniques are chosen as the most successful ones by considering experiments with particle agents and physics based models. For a single experiment, total flight time is 3 minutes and 30 seconds. In addition, the rate of sending velocity commands and logging state data for physics based simulations was 100 Hz whereas for real robot experiments, usage of this rate has resulted with instabilities in the flight. The rate is changed to 10 Hz. The reason behind that was inability of Crazyflie platform to track velocity commands changing 100 times a second.

For each experiment, total of 348 data points are collected for each agent. As done in physics based simulations, MATLAB and Robotic Systems Toolbox are used in order to collect and save this data.

4.3.3 Results

Table 4.3 shows the parameter values used in collective foraging methods employed in real robot experiments. The results of experiments are presented in Figure 4.17. Unlike other results presented earlier, this figure does not involves box plots. Since the number of data points is relatively low, the mean local values are presented independently. For 8 different experiment, the same number of mean local values can be observed with corresponding colors for the environment used in that experiment. In addition, Figure 4.18 shows the trajectory of centroid of group consisting of 4 agents

in environment 1 and environment 4.

Table 4.3: Parameter values or range of values used in real robot experiments

Variable	Description	Value
N	Total number of agents	4
D_p	Proximal sensing range of an individual	2.0 m
α	Gain of the proximal virtual force vector	3.0
σ	Desired distance regulator term	0.6
σ_{var}	The variance window of σ	0.4
ϵ	Multiplier for proximal force vector magnitudes	12.0
d_{des}	Desired distance between members for constant σ	0.84 m
D_a	Heading alignment communication range	2.0 m
β	Gain of the heading alignment virtual force vector	1.0
γ	Gain of the boundary avoidance virtual force vector	1.0
k_{rep}	Multiplier for boundary avoidance vector magnitude	50
L_0	Relaxation threshold for boundary avoidance	0.5 m
K_1	Gain of linear speed	0.5
K_2	Gain of angular speed	0.06, 0.03
U_{max}	Maximum linear speed allowed	0.15 m/s
U_{min}	Minimum linear speed allowed	0.01 m/s
ω_{max}	Maximum angular speed allowed	$\pi/6, 4\pi$ rad/s
G_{max}	Maximum local value in the environment	255
G_{min}	Minimum local value in the environment	0
t_{sim}	Collected data points	348
d_t	Time step length	0.1 sec

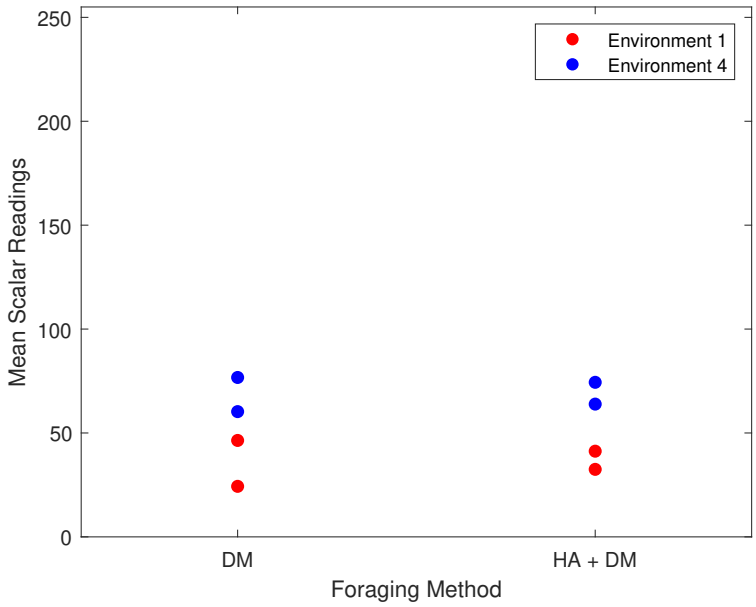
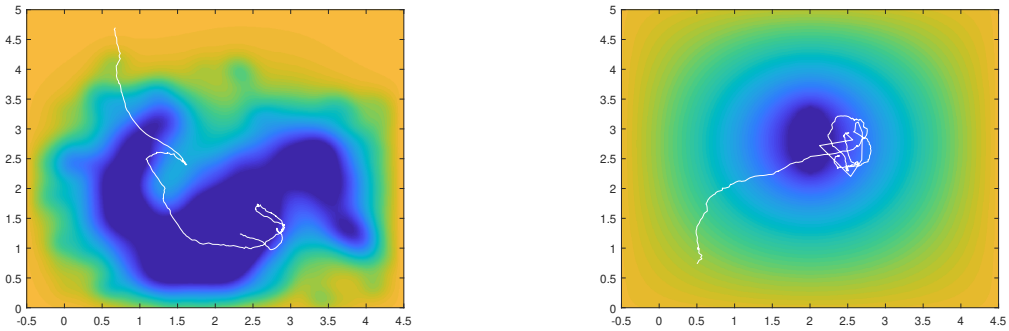


Figure 4.17: Mean local values of 4 agents in real robot experiments in Environment 1 and 4.



(a) DM, Environment 1

(b) DM, Environment 4

Figure 4.18: Trajectory of centroid of a group consisting of 4 real aerial robots, for environments 1 and 4.

When Figure 4.17 is observed, it can be concluded that for both foraging methods of DM and $HA + DM$ give results that agree with particle agent and physics based simulations in different environments. Although the mean local values are higher than physics based simulations, they still indicates that the group forage towards desired regions.

CHAPTER 5

DISCUSSION

In this chapter results of the experiments, that are conducted to validate proposed foraging methods, are discussed. The discussion focuses on the performances of groups at foraging towards desirable regions (regions with smaller local values) of the environment and the variations of them with changing group size, environment and chosen method.

The results are discussed under 3 different categories according to experimental setup they are obtained from. The first group focuses on the simulation experiments with particle agent models conducted on MATLAB while the second group focuses on physics based simulations. Finally, third group consists of discussions about results of experiments with real flying robots.

5.1 Particle Agent Model Simulations

In Figures 4.4 to 4.7, first columns consist of mean local values of groups with no foraging method. To extend, none of the heading alignment, speed modulation or desired distance modulation methods are employed. These groups have the highest median values among others. This situation is independent of the chosen environment and the group size. The only effect that holds group together is the proximal control between agents and the occasional repulsive effect acting on the group by the boundaries. The fact that groups are not separated in any of the experiments proves that agents do not need heading alignment effect to stay as a cohesive group. The trajectory of centroid of a group provides a rough estimate about foraging performance. In Figure 4.8a, if the trajectory of the group with no foraging method is considered, it can be ob-

served that the group is repeatedly reflected from arena boundaries similar to a bullet bouncing from wall to wall. The undesired effect of this pattern on mean local values is making it environment dependent. Another observation about the mean local values is that for no foraging method the difference between maximum and minimum points of the corresponding box plots is the widest for the simulation environment 4. The main reason behind is the fact that environment 4 has the smoothest transition between regions with smaller local values and higher local values. Environment 4 is like a pure sink of scalar values decreasing gradually towards the center with circular contours. The distribution of local values in environment 4 is similar to the temperature distribution of a surface with a heat sink on the center.

The foraging method with heading alignment and speed modulation ($HA + SM$) shows a very distinct trajectory around the favorable region of environment 1. When Figure 4.8b is considered, it can be observed that the centroid of group does not penetrate into the favorable region but instead, it circulates around the region. When corresponding mean local values are considered, this situation is also clear for all environments. The median values in addition to maximum and minimum points of box plots are lower than no foraging configurations but higher than the configurations with desired distance modulation. From all simulations in 4 different environments and 4 different group sizes with $HA + SM$ method, it can be concluded that this method produces a foraging behaviour towards favorable regions but it does not provide a satisfying performance. In addition, it can also be stated that in environment 1 and 3, $HA + SM$ method gives better results than others. The speed modulation method requires a sudden change between the local values experienced by different agents. When this sudden change occurs, some agents slow rapidly while others are considerably faster. Then, there is a possibility of separation. Since proximal effects on the agents do not allow it, fast agents rapidly change their directions. When the heading alignment is employed, it rapidly influence the group to turn towards the direction where slow agents are located. If agents are slower it means they are located in favorable regions. With this interpretation, it is more clear that why the smooth transition of local values in environments 2 and 4 causes worse performance for $HA + SM$ foraging method.

The foraging methods DM, $HA + DM$ and $HA + SM + DM$ can be evaluated all

together since the results they gave are similar to each other. Their median values and maximum points are below 50 in all environments and group sizes. Employing desired distance modulation obviously improves the foraging performance. The smaller numbers of outlier points are produced for $HA + DM$ while the smallest occurs at environment 3. The reason is that the transition between high local values to lowers is the sharpest at there. In other words in environment 3, there is a wider favorable region and it increases the chance of agents to be located there. In addition, heading alignment method requires an information exchange between agents but it comes with the advantage of an ordered and smooth motion. Without heading alignment, agents are still able to move as a cohesive group but the differences between direction of motions of the agents causes an oscillation since they are constantly moving in directions that are not parallel. These unparalleled motions are triggering proximal effects frequently. Because of this, $HA + DM$ gives the best result among the top 3 foraging methods which are DM, $HA + DM$ and $HA + SM + DM$. When the trajectories of groups from Figure 4.8 are analyzed, the trajectory lengths of groups can be ordered from longest to shortest as $HA + DM$, DM and $HA + SM + DM$. It is expected that $HA + DM$ to be longer than DM because of the fact that agents are not allowed to have a negative value of linear speed. To explain, if an agent has to move towards a direction that is divergent more than 90° from its current heading angle, it does not move backwards but instead, it changes current heading angle while standing still until it has a positive value of desired linear speed. Occasional pauses cause groups to have shorter trajectories. When the trajectory lengths of $HA + DM$ and $HA + SM + DM$ are compared, it is also expected that $HA + DM$ to have a longer trajectory because speed modulation changes speed of agents between pre-defined values of U_{min} and U_{max}. The U_{max} value is mostly smaller in speed modulation method when it is compared to desired distance modulation method.

The method $SM + DM$ gives a performance which is far better than $HA + SM$ but its performance is not comparable with DM or $HA + DM$. It is because of that the effect of speed modulation suppresses the effect of desired distance modulation in some cases and decreases the foraging performance. Although those two methods are not directly conflicting ones, they do not improve each other as well. The high number of outlier data points also supports the idea that they are not improving each

other and introduce uncertainty to performances of each other.

Since for all of the simulations it is the same, the number of groups metric is not presented. The group did not split in any case of 6 different methods, 4 different environments and 4 different groups sizes. The situation indicates that the collective moving ability of group does not depends on any of those variables. In addition, there is no crash between particle agents which indicates that group has travelled in a cohesive,ordered and safe manner for all options.

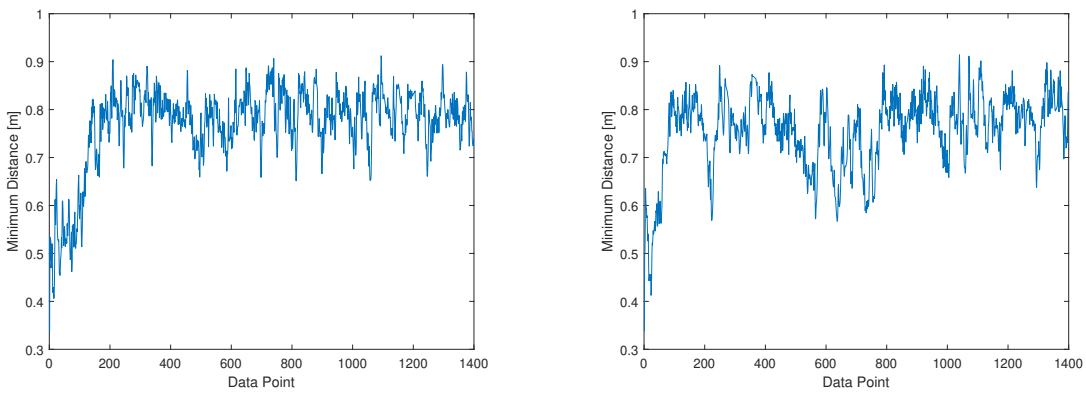
5.2 Physics Based Simulations

For the physics based simulation experiments which are conducted on GAZEBO simulator, the Figure 4.13 presents mean local values for 2 foraging methods DM and $HA{+}DM$. These two methods are chosen because they were the options with highest performances among all 6 methods in particle agent simulations. DM and $HA{+}DM$ have smallest median and maximum values in addition to relatively less number of outlier data points in all environments. The agent number for all simulations was the same and chosen to be 6. For a chosen environment and method, simulations are repeated 20 times with different starting points demonstrated as in Figure 4.12.

When Figure 4.13 is considered, it can be observed that the mean local values, their median and maximum values are in an agreement with the particle agent model simulations. The results are not exactly the same. It was expected since physical effects are involved. To explain, the positions of particles in MATLAB simulation environment are updated according to desired velocities and chosen time step by using a linear integration in a discrete way as described in Equation 41, 42 and 43. In physics based simulations, when the desired velocities are sent to corresponding SITL firmware, the velocity controller tracks these references as close as possible. Yet, it is an expected behaviour that controllers can not perfectly follow the references which are changing 100 times a second. This situation introduces error between desired and current velocities. Despite the errors in desired velocities, the median values of box plots for both methods are below 50 as in particle agent simulation results. It is a promising similarity. In addition, when Figure 4.14 is observed, the trajectories are

shorter than particle agent simulation trajectories. It also worth noting that there are more corrections occur in the heading angles of agents which results with decreased values of linear speed. The reason is the same again, desired velocities are not perfectly tracked and this introduces oscillations on trajectories of robots. Another point to observe from mean local values is that the difference between maximum and minimum points of box plots are greater for DM when compared to $HA + DM$. This is because of the fact that $HA + DM$ results with more ordered collective motion with less oscillations that is disturbing foraging behaviour.

For all of the physics based simulations, there was no group split or crash between robots. The number of groups was 1 for all trials. The probability of a crash between robots is much more dangerous than particle agent simulations since it is expected from GAZEBO simulations to be closely related with real robot experiments. This situation is closely observed and it can be concluded that no dangerous situation has occurred in any of the experiments independent of the oscillations in the minimum distance between members. Figure 5.1 demonstrates the minimum distance for an experiment in environment 1 with both DM and $HA + DM$. The minimum distance plot presents the minimum value of inter agent distances for every step in the simulation.



(a) DM (b) $HA + DM$

Figure 5.1: Minimum distances in meters between 6 agents simulated at GAZEBO physics based simulator in environment 1 for DM and $HA + DM$ foraging methods. Every data point shows the minimum of inter agent distances at that time step.

Finally, it is worth it to mention again about the reason why there is no foraging method with speed modulation in physics based simulations. The first reason is that speed modulation gave results that are much less promising than ones with desired distance modulation. The second reason is that the performance of speed modulation method is closely related with how the desired velocities are tracked. The platforms that are inaccurately following velocity references, because of physical constraints, are demonstrating much worse performances.

5.3 Real Robot Experiments

The mean local values for 2 different foraging methods, DM and $HA+DM$, tried in 2 different simulation environments are calculated. The results are presented in Figure 4.17. In these experiments, 4 agents are used. By using 4 agents, the foraging and flocking capabilities can still be proved in a flight arena with sufficient size despite the agent number is less than physics based simulations. In addition to mean local values, the trajectory of centroid of groups can be observed from Figure 4.18. By doing so, it can be said that the proposed methods make the group successfully forage towards desired regions of the environment.

In Figure 4.17, the mean local values can be seen individually for each experiment. When they are observed, it can be seen that these values are higher than their similar configurations at physics based simulations. There are several reasons behind the situation. Firstly, the real platforms are worse at tracking velocity commands. This incapability introduces errors at both individual and group level which results with more oscillatory behaviour. This oscillations distract group from foraging towards a direction in a stable manner and group spends time to converge to a direction. Secondly, the incapability of tracking velocity commands introduces a risk of on air collision. To avoid this, the maximum linear velocities of agents are decreased for real robots. Hence, group forages slower. When it is considered that the flight times for real robot groups and simulated robot groups are close to each other, it is expected to have higher mean local values from real robots. Lastly, the desired distance values are almost equal for modelled and real robot groups while the flight arena is smaller for real robot groups. This fact leads to a group of real robots covering a wider section

of total area at a time instant. Since only some regions of that area are desirable with lower local values, there is a higher chance for agent in real robot group to be located outside.

When mean local values are compared in their selves, there is not a significant difference among methods or environments. Despite this observation, increasing number of experiments with real robots may reveal effects of different foraging methods or environments.

CHAPTER 6

CONCLUSION

This study investigates collective foraging methods for a group of autonomous aerial robots. This group can be named as a robot swarm because they interact with each other, with the environment and correlate their own actions with these interactions to forage towards desired regions of the environment as a cohesive group.

The methods used to realize collective foraging behaviour are presented in Chapter 3. In the presentation of formulations, there is no detail or constraint about the agent model. This provides a flexibility to algorithms to implement and test them on various kind of systems. In experiments of this study, 2D and constrained environment models are used. Environments contain local scalar values in a certain range and these values are stored in grids with negligible sizes. Methods in Chapter 3 assume that agents are in a similar environment model.

There are two novel foraging methods introduced. The first collective foraging method proposes to change linear speeds of the agents based on local values of the environment they are experiencing. By doing so, there exists a variation in the speeds of agents. Since agents also move as a cohesive and polarized group, this speed variation results with the group turning towards locations of slower agents which are desirable regions of the environment. The proximal effects between agents keeps them at a desired distance. Second method proposes to change this desired distance for the agents depending on local values. Some agents have longer desired distances than others when they are experiencing smaller local values. Varying desired distances between agents causes a group motion towards desirable regions.

Proposed methods are implemented on MATLAB with particle agent models with

no physical properties. Positions and velocities of particles are updated with a linear discrete integration. The collective motion and foraging behaviours are observed in 4 different simulation environments with different local values. The foraging performance is evaluated with mean local values of groups during experiments. For 4 different environments, 4 different group sizes and 6 different method combinations, 512 simulation experiments for each are conducted to make sure there is no randomness in the results. The mean local values of groups with speed modulation are smaller than the groups with no foraging methods. This indicates that modulating the speeds of agent according to local values is successful at foraging a group towards desirable regions. The mean local values of groups that are modulating their desired distances are remarkably smaller than others. The two best methods are chosen to be DM and $HA+DM$ which are desired distance modulation and its variation including heading alignment.

The best two methods are tested with physics based simulation experiments. To achieve this, realistic dynamical models of robots are used in GAZEBO physics based simulator. In addition, the firmware that is working on real robots is used in a software in the loop (SITL) form as parallel to GAZEBO. The algorithms are implemented on ROS environment which is highly integrated with GAZEBO simulator. To make robots able to sense local values, there was not a sensory system model. Instead, it is emulated by sending robots the corresponding local values on their instantaneous locations. By this way, the algorithms are used without any modification. Also a system to make robots share their heading angles is emulated with appropriate software architecture instead of modelling a communication system from scratch. For every simulation, 6 agents are used. 4 different simulation environments are employed. They are identical to the ones in particle agent model simulations in terms of local values. Every method in a chosen environment is tested 20 times with changing starting positions of agents. The mean local values obtained from groups working with DM and $HA + DM$ methods are similar to particle agent models and show satisfying performances in terms of collective foraging. Physics based simulation experiments has proved that chosen methods are applicable to a swarm of autonomous aerial vehicles.

As a last step, proposed methods for collective foraging are tested with real aerial

robots. These robots are chosen to be Crazyflie multi rotor platforms. They were modelled before at GAZEBO for physics based simulations. The same environment models with previous simulations are used for real robot experiments. Since proper on board communication and sensing devices do not exist, these concepts are simulated without conflicting with proposed methods and corresponding assumptions. The experiment results proved that the collective foraging methods are successful with a swarm of robots consisting of 4 agents. Despite slight differences in related metric values, demonstrating a collective foraging behaviour with 4 autonomous aerial platform is sufficient to plan future developments and applications involving proposed methods.

In future, implementing and testing proposed algorithms on bigger aerial platforms, possibly carrying a sensory system and local communication system, will open endless doors for different applications. Instead of virtual local values, these bigger platform may sense temperature to detect a wild fire or radiation level to detect a possible radio active source. In addition, algorithms can be modified to adapt changing environment conditions. A dynamical environment with changing local values can be a good example. Finally, algorithms can also be adapted to a 3D environment in order to demonstrate a collective foraging behaviour in 3D which provides an opportunity to use all of the potential that autonomous aerial vehicles have.

REFERENCES

[1] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, p. 513–516, 2005.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, p. 25–34, Jan 1987.

[3] E. Şahin, S. Girgin, L. Bayindir, and A. E. Turgut, *Swarm Robotics*, pp. 87–100. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[4] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles," *Journal of Intelligent Robotic Systems*, vol. 84, p. 469–492, Oct 2016.

[5] T. Zahariadis, A. Voulkidis, P. Karkazis, and P. Trakadas, "Preventive maintenance of critical infrastructures using 5g networks drones," *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[6] F. G. Costa, J. Ueyama, T. Braun, G. Pessin, F. S. Osorio, and P. A. Vargas, "The use of unmanned aerial vehicles and wireless sensor network in agricultural applications," *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2012.

[7] T. Vicsek and A. Zafeiris, "Collective motion," *Physics Reports*, vol. 517, no. 3, pp. 71 – 140, 2012. Collective motion.

[8] J. Krause and G. D. Ruxton, *Living in groups*. Oxford Univ. Press, 2008.

[9] I. D. Couzin and J. Krause, "Self-organization and collective behavior in vertebrates," *Advances in the Study of Behavior*, p. 1–75, 2003.

[10] R. S. Olson, A. Hintze, F. C. Dyer, D. B. Knoester, and C. Adami, "Predator confusion is sufficient to evolve swarming behaviour," *Journal of The Royal Society Interface*, vol. 10, p. 20130305, Jun 2013.

[11] S. J. Simpson, G. A. Sword, P. D. Lorch, and I. D. Couzin, "Cannibal crickets on a forced march for protein and salt," *Proceedings of the National Academy of Sciences*, vol. 103, p. 4152–4156, Mar 2006.

[12] J. Vannier, M. Vidal, R. Marchant, K. E. Hariri, K. Kouraiss, B. Pittet, A. E. Albani, A. Mazurier, and E. Martin, "Collective behaviour in 480-million-year-old trilobite arthropods from morocco," *Scientific Reports*, vol. 9, no. 1, 2019.

[13] G. D. Kattas, X.-K. Xu, and M. Small, "Dynamical modeling of collective behavior from pigeon flight data: Flock cohesion and dispersion," *PLoS Computational Biology*, vol. 8, no. 3, 2012.

[14] K. Suzuki, T. Takagi, and T. Hiraishi, "Video analysis of fish schooling behavior in finite space using a mathematical model," *Fisheries Research*, vol. 60, no. 1, p. 3–10, 2003.

[15] G. Baldassarre, S. Nolfi, and D. Parisi, "Evolving mobile robots able to display collective behaviors," *Artificial Life*, vol. 9, no. 3, p. 255–267, 2003.

[16] A. T. Hayes and P. Dormiani-Tabatabaei, "Self-organized flocking with agent failure: Off-line optimization and demonstration with real robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, pp. 3900–3905 vol.4, 2002.

[17] E. Ferrante, A. E. Turgut, A. Stranieri, C. Pinciroli, M. Birattari, and M. Dorigo, "A self-adaptive communication strategy for flocking in stationary and non-stationary environments," *Natural Computing*, vol. 13, no. 2, pp. 225–245, 2014.

[18] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, 2018.

[19] A. M. Berdahl, A. B. Kao, A. Flack, P. A. H. Westley, E. A. Codling, I. D. Couzin, A. I. Dell, and D. Biro, "Collective animal navigation and migratory culture: from theoretical models to empirical evidence," Jul 2017.

[20] K. O. D. Goran Bergman, *An analysis of the spring migration of the common scoter and the long-tailed duck in southern Finland*. Societas pro Fauna et Flora Fennica, 1964.

[21] T. Mueller, R. B. Ohara, S. J. Converse, R. P. Urbanek, and W. F. Fagan, "Social learning of migratory performance," *Science*, vol. 341, no. 6149, p. 999–1002, 2013.

[22] R. M. Holdo, R. D. Holt, and J. M. Fryxell, "Opposing rainfall and plant nutritional gradients best explain the wildebeest migration in the serengeti," *The American Naturalist*, vol. 173, no. 4, p. 431–445, 2009.

[23] J. I. Macdonald, K. Logemann, E. T. Krainski, Sigurðsson, C. M. Beale, G. Huse, S. S. Hjøllo, and G. Marteinsdóttir, "Can collective memories shape fish distributions? a test, linking space-time occurrence models and population demographics," *Ecography*, vol. 41, no. 6, p. 938–957, 2017.

[24] J. G. Puckett, A. R. Pokhrel, and J. A. Giannini, "Collective gradient sensing in fish schools," *Scientific Reports*, vol. 8, no. 1, 2018.

[25] T. Schmickl and H. Hamann, "Beeclust: A swarm algorithm derived from honeybees derivation of the algorithm, analysis by mathematical models and implementation on a robot swarm," 2011.

[26] F. Arvin, A. E. Turgut, T. Krajnik, S. Rahimi, I. E. Okay, S. Yue, S. Watson, and B. Lennox, "ϕ clust: Pheromone-based aggregation for robotic swarms," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[27] A. L. Alfeo, M. G. C. A. Cimino, N. D. Francesco, A. Lazzeri, M. Lega, and G. Vaglini, "Swarm coordination of mini-uavs for target search using imperfect sensors," *Intelligent Decision Technologies*, vol. 12, p. 149–162, Jul 2018.

[28] "Energy 2D." `https://energy.concord.org/energy2d/`. Accessed: 2020-04-20.

[29] W. V. W. Press, S. Teukolsky and B. Flannery, *Numerical Recipes in C*. Cambridge University Press, 1992.

[30] R. Mcgill, J. W. Tukey, and W. A. Larsen, "Variations of box plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, Sep. 2004.

[32] "Crazyflie 2.1." `https://www.bitcraze.io/products/crazyflie-2-1/`. Accessed: 2020-04-20.

[33] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. Cham: Springer International Publishing, 2016.

[34] "Gazebo Simulation for Crazyflie CRTP." `https://github.com/wuwushrek/sim_cf`. Accessed: 2020-04-20.

[35] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.