Priority-Based Human-Swarm Interaction Applied to a Foraging Application

by: ©Dalia Said

A thesis submitted to the School of Graduate Studies

in partial fulfillment of the requirements

for the degree of

Master of Science

Department of Computer Science

Memorial University of Newfoundland

October 2016

St. John's Newfoundland and Labrador

Abstract

A robot swarm employs a large number of robots to facilitate sophisticated problem-solving as well as improved load-balancing and time efficiency compared to single robot systems. Swarm Intelligence is based on the local interactions between agents of the swarm that enables the emergence of a desired global behaviour, thus allowing the swarm to be autonomous. While autonomy is efficient for straightforward applications, in complex problems and environments human intervention may be more efficient. Human control of a swarm remains an open problem with multiple approaches proposed, each designed for a specific type of application. This work suggests a priority-based approach inspired by well known Human-Swarm Interaction techniques. The approach aims to serve as a high-level guide for the agents of a swarm, allowing them to use Swarm Intelligence on a low level. It also allows the division of the swarm into subswarms that can be easily controlled separately.

Before experiments could be carried out to validate the proposed approach, the robots used in our experiments had to be put together, and their software needed to be designed to put to use their various components. A vision system upon which their sensing is dependent needed to be established, with defined visual markers and obstacle detection.

This thesis tests the proposed priority-based Human-Swarm Interaction system by implementing a simple foraging application, using simulated and real robots, and studying the

effects of introducing such a system to a group of robots that use simple Swarm Intelligence.

Results show that the proposed approach does succeed in dividing the swarm into sub-swarms and increasing the efficiency of the foraging solution, however, some drawbacks manifested themselves throughout the process. We discuss these advantages and issues as well as future work.

Acknowledgments

First off, I would like to thank my supervisor, Dr. Andrew Vardy, without whom I would not have developed an interest in Swarm Robotics. His guidance, support, and patience throughout this process have made this thesis possible. I would also like to thank my colleagues in the research group at the Bio-Inspired Robotics lab for their helpful and friendly input.

My sincerest gratitude goes to my parents for their continued and unconditional love and encouragement through the ups and downs of the past two years, and my brother without whom I would not have come to Newfoundland to begin with. I would also like to thank my friends, Reem, Noha and Rawdat for always being there for me, and lifting my spirits when I needed it most.

Contents

List of Figures

List of Algorithms

List of Tables

Introduction

Robots are deployed in the field to perform tasks that are too dangerous or tedious for humans. The aim is to take advantage of the reduced cost of operation and risk of human injury, and to automate the more mundane and dangerous tasks that can be delegated to a machine. As a result of this, applications previously deemed impossible due to human limitations are becoming possible [43].

The type of robot system being used depends on the nature of the problem that needs to be solved. The most well-known type of robot systems, the single-robot system, consists of one robot, that is either controlled partially by a human or is autonomous [35]. Compared to human-controlled robot systems, fully autonomous robot systems are much more complex to design and implement. As a result, successful fully autonomous systems are extremely rare commercially [56], compared to systems that have some form of human control or interaction.

Despite being predictable and easy to maintain, single-robot systems have their drawbacks. Most importantly, the robot is a single point of failure; if it becomes trapped or malfunctions in any way it will not be able to accomplish its goal. There is also the issue of cost; the more a robot can do, the more expensive it will be.

To solve these problems, more than one robot/agent can be deployed as a multi-robot system (MRS). The robots should be fault tolerant and cheap. The cheaper the robot, the simpler it will be, but also the more limited its capabilities are. The robots could be ho-

mogeneous, each of them identical to the rest, or heterogeneous, with different robots best suited for different tasks. The idea is that the synergy produced by the larger number of robots working together, even if they are modestly equipped to tackle the problem, should provide a boost in performance when compared to a single-robot solution. This also means that MRSs must know how to work together to accomplish a common global goal efficiently, by collaborating and not getting in each other's way. A fully autonomous MRS solution is, therefore, more complicated to formulate than a fully autonomous single-robot solution.

In general, the use of an MRS allows more sophisticated problem-solving as well as improved load-balancing and time efficiency compared to a single robot system [56]. A robot swarm is one such MRS that is characterized by the cooperation of its agents based on local interactions they have with themselves or with their environment. The agents are often homogeneous, acting asynchronously in parallel. They function without centralized control to produce useful behaviours through stigmergic communication, whether they are aware of it or not [27]. Similar to single-robot solutions, there are methods to include partial human control in MRSs and robot swarms. As the fields of MRS and swarm robotics research grow, so too does the interest in examining the effects of adding human control, and determining its limitations and benefits.

1.1 Swarm Intelligence

Swarm intelligence (SI) is a relatively new approach for solving problems [11, 29]. It places an emphasis on distributiveness, with direct or indirect interactions between simple agents [19]. SI is difficult to program, given that the solution it provides is not predefined, rather it emerges as a result of local interactions between the agents and themselves and the agents and their environment [19, 43, 60, 28, 11]. This resulting global behaviour is usually

referred to as an "emergent behaviour".

Typically, the emergent behaviour is robust to changes in the environment and robot failure. The emergent behaviour makes it seem as if the swarm itself, as a unit, is much smarter and more useful than an individual agent in it [27].

Work on SI trails back to 1987, when Reynolds' pioneering work was the first to simulate bird flocking behaviour by implementing three simple rules that governed a simulated bird's motion, leading to a global flocking behaviour [54]. The term "Swarm Intelligence" was introduced by Beni and Wang. Their work in [18] explores the ability of autonomous, non-synchronized, and non-intelligent "Cellular Robotic Systems" to achieve global goals through cooperation.

There are many advantages to using SI, chief of which is the system's robustness to failure. Given that there is no single point of failure and no centralized control upon which the entire process is dependent, an agent can malfunction or fail without breaking down the entire system [27, 20, 29]. The other side of that coin is the system's scalability; just as removing agents does not cripple the system, adding more agents does not require any adjustments. The system is made up of simple, inexpensive robots that can solve complex problems by implementing a small and simple set of rules that leads to a complex emergent behaviour. Finally, there is an element of redundancy, meaning that an agent may repeat a task that another agent has performed, thus reducing the margin of error [29].

SI should not be considered for all applications and has its disadvantages. There is no specific method of designing an SI approach for a given problem. There is no blueprint to follow when specifying the local rules that the agents need to follow. Therefore, the outcome of an SI solution may be unpredictable and there are no guarantees that the local rules followed by the agents will produce the desired global behaviour exactly. SI cannot compete with deterministic optimization methods when it comes to static problems with non-varying characteristics and conditions, but it is more suited to dealing with uncertainty

in problems than deterministic approaches due to its adaptive nature [11].

1.2 Human-Swarm Interaction (HSI)

Kolling et al. argue in [43] that while the use of completely autonomous robot swarms is much more effective when the swarms operate in open environments, human controlled swarms are better at adapting to complex environments where obstacles present a challenge. It is infeasible to assume that one human operator can process feedback from every single entity in a swarm that they control [39], even if reliable communication can be assumed. Full and complete control by humans is also not practical; it could lead to incoherent acceleration that in turn would break up the swarm, or state transitions that are highly inefficient in comparison to autonomous control [44].

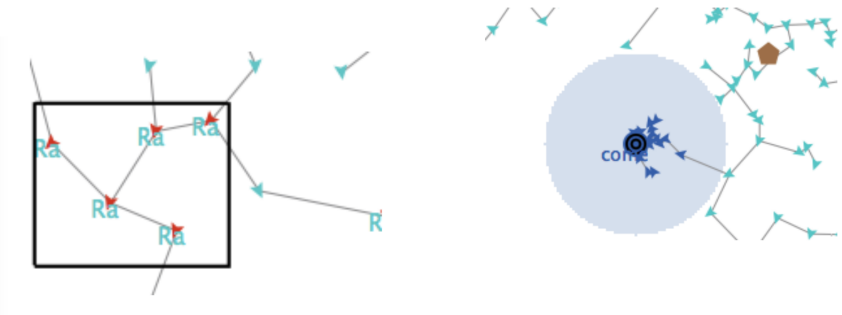
Cavallin and Svensson present teleoperation in [24]; an area of robot intelligence that deals with the cooperation of humans and robots, also referred to as semi-autonomy. It is becoming a more popular field of study in industry, the military and the service sector. The idea is that robots manage the tasks that they excel at: dangerous, repetitive and dull tasks that humans tend to avoid. Humans are then free to concentrate on the tasks that they are better at, like finding patterns, and making quick conclusions and decisions, the sort of reasoning that robots have not mastered entirely. This gives the robots room to focus on their sub-goals. By allowing both parts to excel at their strengths, each part compensates for the other's shortcomings.

The field of HSI is relatively new and studies show humans can achieve efficient cooperation between themselves and robotic swarms [43, 49]. Humans need to be able to interact effectively with the robots in a way that is intuitive but simple enough to keep every agent simple and inexpensive.

Ideally one person would be able to control a swarm intuitively. The idea of assigning

one person to control one robot so that each robot is operated by a human becomes infeasible given the large number of robots per swarm and the near impossibility of coordinating such a large number of people [43].

There are many different methods with which a human can control a swarm, four of which have been the most inspiring to the approach presented in this thesis. The first is intermittent interaction, where an operator selects a part of the swarm whose behaviour they wish to change. It is similar to the control interfaces of many computer strategy games where commands are issued to a selected group of agents. This type of interaction is often referred to as selection control. The second type of interaction is environmental interaction [26]. The operator does not select any subgroup, but manipulates the environment that the robots react to. This type of control attempts to mimic the indirect communication by pheromones between social insects like ants, and is usually referred to as beacon control. Figures 1.1a and 1.1b are taken from the work of Kolling et al. in [43]. In Figure 1.1a, a sub-group of the agents, represented by the light green triangles, is selected by drawing a black rectangle around them. The selected agents are coloured red. In Figure 1.1b, the dark blue circle represents the beacon, while the light blue area represents the area of effect of that beacon. Agents affected by the beacon are coloured in dark blue while those that are out of the beacon's area of effect are green.



(a) Selection Control (b) Beacon Control

Figure 1.1: Types of HSI

The third type is persistent control. In this type of control, continuous input is provided either by the operator, or by a leader or predator that is deployed to provide the input in the field as a proxy. The final type of control is parameter setting. In this type of swarm control certain parameters that affect the swarm's emergent behaviour are set, changing to allow a dynamic range of possible behaviours. An example of these parameters would be the distances that the robots would attract or repel each other while swarming/flocking [43].

1.3 Foraging

In nature, foraging is the task of searching for food. In the case of some social insects like ants and honeybees, finding the food is usually followed by delivering it to a destination or nest. There is a growing interest in drawing inspiration from natural phenomena to solve problems in robotics; the study of foraging behaviours, particularly those of social insects, has also grown rapidly in recent years [34]. This has led to an increased interest in the behaviours that social insect colonies employ to forage and their possible applications to swarm robotics [13]. Foraging techniques can be adapted to solve a variety of problems including toxic waste clean-up, collection of terrain samples, and demining [22]. Campo et al. describe robot foraging in [23] as a generalization of problems like search and rescue, mining, agriculture, and exploration of unknown or hostile environments.

Multi-foraging is a variation of the foraging problem, where there is more than one type of resource that the robot can retrieve and must decide on which resource is the most worthwhile to pick up [22].

Foraging is therefore the application that this thesis will use to compare between a standard SI approach vs. an HSI-assisted priority-based SI approach.

1.3.1 Foraging in Insect Swarms

There are a multitude of strategies that social insects employ to facilitate foraging depending on the type of insect, the habitat that it forages in, and the type of food that is being foraged for. Factors such as the density of the food sources, path selection decisions, and the cost of reaching them based on their distance to the nest and the quality of food to be extracted, all impact the behaviours of insects [38]. Insects have evolved foraging strategies to increase their efficiency. They employ specific, non-random strategies that increase the efficiency of their foraging, like systematic search and trap-lining their routes [34].

For navigation and localization in foraging several strategies are used to extract directional information, depending on the insect. Some insects use sound waves, or direct visual localization. Some types of ants and termites use trail-laying and trail-following without a central leader, or knowing their own location or the location of their food [40]. Other types deposit pheromones and navigate based on the chemical gradients of pheromones or food odors. Insects that control their orientation using non-directional information (i.e. when they cannot perceive any information from their environment) may switch to directional information upon its perception [17]. Some insects can orient themselves based on the earth's magnetic field [34], others use the north star and celestial objects to localize and navigate.

In order to find food sources ants tend to communicate with each other by leaving the aforementioned pheromones, forming trails to achieve stigmergic communication. Stigmergy is the local communication achieved when agents alter their environment or deposit pheromones as a means to communicate with other agents indirectly. The information conveyed by stigmergy does not specify a particular recipient, and while it is helpful it is not necessary for the entire swarm to receive that information[40]. When ants stumble upon a viable source of food, they deposit a trail of pheromones that attracts other ants, leading them to the source they found. As a larger number of ants favours a trail, more pheromones

are deposited along it, and thus the trail draws more ants to follow it, they themselves depositing more pheromones. This process of reinforcing the trail is an example of an autocatalytic process. Pheromone-laying is already being used to solve problems in computer science such as the Traveling Salesman Problem, Group Shop Scheduling, Routing Problems, and Area Coverage using robots [13].

Foraging honeybees have a different approach. The honeybees recruit other members of their colony by performing a "waggle dance". This dance, described by some as the language of bees, consists of a succession of right and left looping movements that the bee performs interrupted for a time by waggling its abdomen from left to right. The duration of the waggle is to indicate the distance to the food source and the pattern of loops indicates its direction. Honeybees do not use pheromones for navigation, instead opting for the use of Path Integration (PI); they continuously update the vector to their destination by integrating all the angles they turned and the distances they traveled. By doing so, the bees can broadcast the final vector to a food source to the others in the colony [13].

1.3.2 Foraging in Swarm Robotics

As previously stated, there has been an increased interest in natural phenomena as sources of inspiration for solutions to problems in computer science and robotics [13]. To that end, some swarm robotics research attempts to mimic the behaviour of social insects, which is no easy task given that simple, inexpensive robots of the swarm have limited sensing and computing capabilities. Leaving temporary marks that decay in the environment, to emulate pheromone-depositing, is difficult to implement physically, though easier to simulate [41, 13]. The process of interpreting a waggle dance performed by another robot is challenging, considering the limited sensing capabilities of robots in a swarm.

The foraging task can be sub-divided into two sub-tasks: exploration and transportation [22]. The robots explore the environments to locate one of several target objects which

they transport to a pre-defined destination. A very relevant type of foraging to this thesis is central place foraging. It is the type of foraging where the robots return the gathered objects to a central location, known in biology as a "nest".

1.4 Motivation

A fully autonomous robot swarm attempting to solve a foraging problem, whose robots have no information regarding their location in the environment, can encounter several issues that reduce the efficiency of its solution. The area around the resource being foraged can become overcrowded, leading to the application of implicit race conditions between the robots as they compete for a chance to retrieve the resource. In a charted environment, i.e. one where a human operator is informed of the whereabouts of the resources in the environment, a fully autonomous solution cannot make use of the operator's knowledge, nor is there any way for the operator to communicate with the swarm. This can lead to a lot of time needlessly wasted on exploration. The foraging problem is, as previously explained, divisible into two subtasks: exploration and transportation. Thus, agents of the swarm are usually in one of two states: either actively exploring and picking up resources, or actively transporting and depositing resources. A solution that takes advantage of this nature and divides the foraging subtasks evenly across members of the swarm should increase the efficiency with which the swarm solves the problem. The solution must reliably allow the swarm to be split up into more than one subswarm, and each subswarm should be able to function independently, in accordance with a human operator's commands. Ideally, this solution should be applicable to other problems and not tailor-made for the foraging problem.

This work argues that a robot swarm that functions according to simple rules and that is not fully autonomous, but controlled indirectly by a human operator, making use of the operator's knowledge of the environment, and utilizing the priority-based beacon control

HSI technique proposed in this thesis, will function more efficiently than a swarm that is completely autonomous, operating under a similar set of simple rules. The proposed HSI approach facilitates splitting up the swarm into smaller subswarms, each of which can be controlled independently to accomplish a specific task. To showcase this approach and to support our argument, a simple foraging application is solved once with an autonomous swarm of robots and again with a swarm that can be influenced by a human operator that uses our prioritized, beacon-controlled HSI technique.

1.5 Organization

The literature review of works relevant to this thesis is presented in Chapter 2. The robots used in simulations and in real-world experiments are introduced in Chapter 3. Chapter 4 details the vision system designed for the robots and the experiments conducted to design a suitable visual marker. Chapter 5 is dedicated to the methodology, discussing in detail the tested approaches, the algorithms used, and the experimental setup. Chapter 6 presents and discusses the results of all conducted experiments, and Chapter 7 presents the conclusion and suggestions for future work.

1.6 Associated Publications

A paper covering the research presented in this thesis has been submitted to the Journal of Human-Robot Interaction (JHRI) and is currently under review.

Literature Review

The works discussed in this chapter encompass the fields of Swarm Robotics, Human-Swarm Interaction and Foraging.

2.1 Swarm Robotics (SR)

Swarm robotics is considered to be a subfield of swarm intelligence [48]. SR research tends to focus more on the interaction and cooperation between robots to accomplish a task and less on the details of how the task is to be executed [21].

Bayındır does an in-depth review in [15] of the following SR tasks: aggregation, flocking, foraging, object clustering and sorting, navigation, path formation, deployment, collaborative manipulation and task allocation. All these tasks are discussed, and an overview is given of their experimental methods, algorithms, and results analysis. In [59], Wright et al. discuss swarm robotic platforms, swarm implementations, and the emergent behaviours that they were able to produce. They discuss swarm missions including search and rescue, area survey, and area setup. In his work [46], Mataric continued the trend of being inspired by behaviours of species in nature, specifically the behaviours of safe-wandering, aggregation, homing, dispersion and avoidance. He shows how simple actions can combine to form high-level group behaviours and how agents can cooperate and exchange information. In [55], Rubenstein et al. use robots that communicate locally, within an are three times

their diameter wide. They built a thousand-robot swarm that can self-assemble into complex two-dimensional shapes. Their algorithms is robust to error, handling situations where some of their robots might wander away or become slowed down.

In [42], Junior and Nedjah use a type of distributed algorithms called wave algorithms. They divide complex tasks into sequential subtasks. A robot can communicate by passing messages in its neighborhood to mark the start and end of a subtask. Robots are recruited into groups, forming clusters, each of which tackles a subtask. They achieve collective navigation, i.e. moving the swarm across the environment without losing swarm agents, using a process of recruitment, alignment and movement to accomplish tasks such as foraging and collaborative transport. In [48], Mondada et al. also achieve collective navigation, however, they link their robots physically. They introduce the SWARM-BOTS project, in which agents of a swarm of robots, called s-bots, connect to each other via grippers to create a formation called the swarm-bot. The swarm-bot formation is self-reconfigurable, and can traverse obstacles and move in narrow passageways.

Brutschy et. al explore the abstraction of complex tasks for robot swarms in [21]. They use a task abstraction module (TAM) into which a robot of the swarm can enter and be considered working on an abstract task. They divide complex multi-robot tasks into simpler single-robot subtasks and these subtasks are abstracted in the TAM while their logical interrelationships are maintained. This allows the team to focus on the global behaviour itself without having to worry about the details of executing its subtasks or whether their robots are equipped to carry them out.

In a more direct application of insect-inspired SI, Chatty et al. present in [25] a clustering algorithm inspired by the cemetery organization of ants, showing how the behaviour can emerge from the application of simple, local rules. In [52] Purnamadjaja and Russell explore the use of chemicals to mimic a queen bee's use of pheromones. A leader robot is assigned the task of releasing the pheromones that trigger light seeking and congrega-

tion behaviours in the robot swarm. They continue their work in [51] where they explore the use of pheromones in necrophoric bee behaviours, i.e. the behaviours that bees adopt to dispose of dead bees in the nest. Disabled robots emit a pheromone-like chemical that other robots in the swarm can identify as a distress signal and use to locate and rescue them. Garnier et al. explore in [32] the applicability of incorporating ant behaviours for ant-like robots. Their robots use LEDs and light sensors; a robot with its LED light on is in a trail laying state that other robots track and follow, while a robot with its LED light off is in an exploration state without trail laying.

Hamann and Wörn in [37] discuss building a framework that supports both a direct and indirect way to design emergent algorithms, so that the impact of the behaviour of an individual robot on the swarm's emergent behaviour is easier to predict and the amount of time spent testing an SI algorithm in simulations and experiments can be cut down.

2.2 Human-Swarm Interaction

In some cases, research is concerned with how an operator can directly relay their command to a swarm. For example, Nagi et al. use gesture recognition to select robots from a swarm by using teleoperation in [49]. In [14], Alonso-Mora, Siegwart and Beardsley compare three different methods of communication: real-time drawing where robots adopt a physical configuration as a representation of the drawing, by using a user interface such as a tablet, and using an RGB-D sensor to recognize gestures.

In other cases, research is concerned with indirect communication between an operator and a swarm, where the operator does not need to be visible to the entire swarm. In [43], Kolling et al. present a study of HSI methods, including those mentioned in Section 1.2, with a comparison between the execution and performance of intermittent and environmental control. Agent dynamics in a swarm are discussed by Goodrich, Kerman, and Jung in

[33]. They explain the different styles of influence that agents can have on each other and compare between agents that are controlled by humans and those that are not, and agents that are aware of the types of other agents and those that are not. In [20] Brown, Kerman, and Goodrich investigate influencing the swarm by utilizing high level attractors. They introduce biologically inspired quorum sensing as a means to allow human input to be dispersed in the swarm without a single point of failure. Lewis et al. in [44] explore controlling swarms by manipulating control laws, highlighting that the roles of control laws are mainly keeping the swarm coherent and completing desired tasks by producing certain behaviours. Cooperative control and the challenges of HSI and its future are also discussed. Becker et al. in [16] control a large swarm of robots using a global broadcast signal to which all robots respond uniformly. An obstacle is placed in the environment and as the global signal moves all the robots, the obstacle moulds the swarm into the required shape or leads them to the desired destination.

These approaches generally study one direction of communication: human-to-swarm. McLurkin et al. study the other direction in [47]. They provide a physical infrastructure, utility software and methods for the robots to communicate with humans using lights, sounds and movements.

Sources of uncertainty need to be taken into account when implementing any of the above approaches and Hayes and Adams present their main types of uncertainty: physical state, virtual state and compound state in [39].

2.3 Foraging

As was previously mentioned, swarm robotics' foraging solutions tend to be inspired by ants' pheromone-laying behaviour. Some approaches do attempt to mimic the behaviours of foraging honeybees, despite the difficulty of their implementation.

2.3.1 Virtual Ant-Inspired Approaches

In [23] Campo et al. use virtual ants that implement artificial pheromone laying techniques in a network of robots to study path selection in robot foraging. They argue that path selection focuses the swarm on the most profitable resource. Their virtual robots are able to stop their path maintenance duties and switch to other tasks. Also using simulated robots and pheromones are Hamann and Wörn in [36]. They apply an analytical model to a foraging task that uses simulated robots that deposit pheromones. The robots detect pheromone gradients in two mutually orthogonal directions. The model was used with a large swarm of 100-150 agents and from it qualitative measures pertaining to the stability of the behaviour, the flow of food that the behaviour achieves and the robot density while executing the behaviour can be extracted. In his dissertation [40], Hoff presents three algorithms for solving the foraging problem, also using a swarm of simulated robots with simple sensing and actuation. The first two algorithms are the Virtual Pheromone (VP) algorithm and the gradient algorithm. In both of them he generates a field of beacons, composed of stationary robots. These robots carry information that other robots that decide not to become beacons (referred to as "walkers") can access and read. The beacons act as an emulation of pheromones left by ants by storing data. The difference between them is the type of data that is being stored and how the walkers choose to react to that data. The third algorithm presented, suggests forming a line of robots to sweep the area being foraged in in its entirety. In [41], Hoff et al. continue to use the robots themselves as beacons, deciding to rely on local, low-bandwidth, direct communication between their robots. They state their opposition to using permanent or temporary marks in the environments, due the difficulty of implementing such systems. They also argue against robots that deploy beacons and have to intelligently retrieve them, opting to use the robots themselves as beacons. They present the aforementioned VP algorithm and a cardinality algorithm both of which rely on

a robot being either a wanderer or a beacon that emits different types of data depending on the algorithm implemented.

2.3.2 Other Approaches

In [13], Alers et al. implement a foraging and coverage application. The former inspired by the foraging behaviour of bees and the latter by the stigmergic exploration of ants. To emulate pheromone trails they use robots equipped with UV light emitters and glow-in-the-dark foil. Bee behaviour is emulated by having the robots wander randomly in a search for food, transporting it back using the shortest route when it is found and communicating the food's location to their peers. They present two approaches that utilize two robot swarms, each with its own set of capabilities. One robot swarm has extended capabilities, i.e. advanced cameras and general purpose computers that provide high computational power. The other swarm consists of e-puck robots [6], whose sensing and computational abilities are limited in comparison, and are much simpler. They discuss different techniques of overcoming the limitations of the e-puck swarm. Using simulation, Campo and Dorigo in [22] attempt to develop an efficient central place multi-foraging behaviour. They define efficiency as a function of the amount of energy a robot needs to exert in order to retrieve an object, referred to in their work as prey, and to explore its surroundings. The robots decide whether the cost of retrieving their prey is worth the amount of energy spent to retrieve it.

Walker et al.'s work in [58] is focused on the effects of HSI in swarms of foraging robots. They examine the effects of "neglect benevolence" on the performance of a foraging robot swarm. Neglect benevolence is a concept that suggests that a swarm is to be left to carry out its tasks, grow, and stabilize its emergent behaviour before any human interference is made. They also study the effects of communication latency between a swarm and the humans controlling it. Human control consists of giving out instructions via a graphical user interface, to change the swarm's heading and flocking constraints.

The Robots

The two types of robots used in our experiments are the Bupimo, developed at our Bio-Inspired Robotics lab [1] for swarm robotics research, and the SimBot, a robot used in simulation that is similar to the Bupimo in functionality, but whose design is simplified to allow for faster simulation. This chapter details the hardware specifications of the Bupimo, and provides an overview of the software that it runs. It will also provide information regarding the SimBot and how it works, as well as the simulated beacon, the "BeaconBot".

3.1 The Bupimo

The Bupimo, shown in Figure 3.1, was so named because it is made up of three parts: Bu, as in Bubblescope, pi, as in the Raspberry Pi 3, and mo, as in the Zumo.

3.1.1 Hardware

The Bubblescope [8] is a camera attachment that allows the capture of 360-degree panoramic images. The camera used is the Raspberry Pi Camera, which is capable of capturing images with a resolution of up to 2592×1944 pixels. It connects to the Raspberry Pi 3 [9], which is equipped with a 1.2GHz 64-bit quad-core ARMv8 CPU and 1 GB of RAM. The Pi also has 802.11n Wireless LAN and Bluetooth 4.1. All image processing and behaviour logic is performed on the Pi. Powering the Pi is an Insignia NS-MB7800 power bank [2] with a

capacity of 7800 mAh.

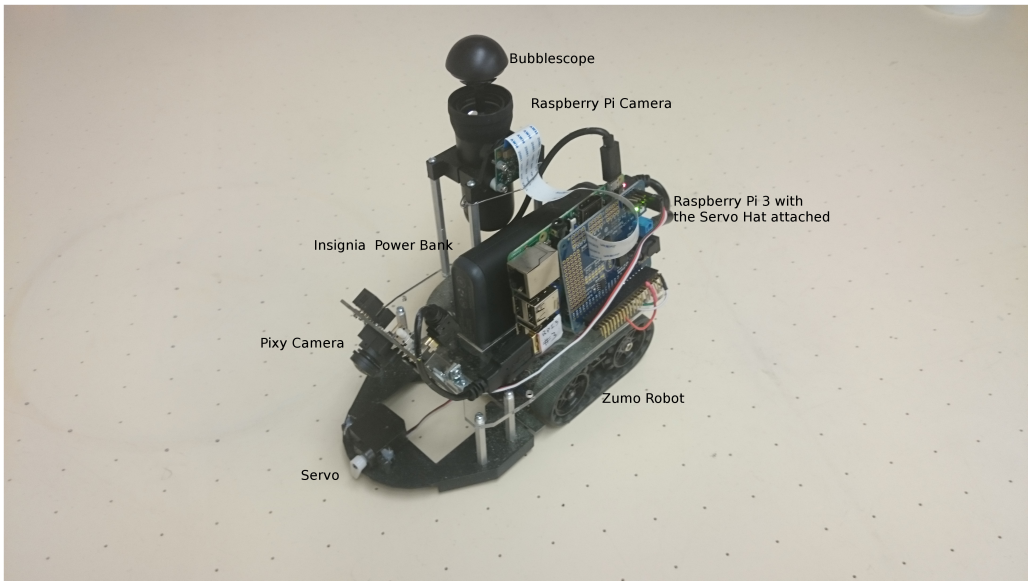


Figure 3.1: The Bupimo. The cylinder at the top is the Bubblescope camera attachment. At the front is the Pixy camera. To the right is the Raspberry Pi 3 behind which is the Insignia power bank. Below the Pixy camera is the gripper, to which is attached the Servo Motor

The base of the robot, to which the platform that holds the camera and Bubblescope is affixed, is the Zumo 32U4 [5] with a gripper attached to it. It is equipped with an Arduino-compatible ATmega32U4 microcontroller that enables it to be programmed as if were an Arduino Leonardo. The Pi and the ATmega 32U4 are connected via a serial connection, and a logic level converter is used to convert from the 5v serial ports of the ATmega32U4 to the 3.3v serial ports on the Pi. The Zumo makes use of proximity sensors to detect when the robot is in possession of a puck in its gripper.

On top of the gripper, a DC Servo Motor is attached. The Servo Motor rotates to open and close the mouth of the gripper to enable the robot to retain a collected puck when it moves. The Servo Motor is connected via 3-pin female connector to an Adafruit 16-Channel PWM/Servo HAT, which is itself mounted on top of the Pi. The Servo HAT is also powered by the Insignia.

The Pixy camera [4] is a camera that detects blobs of colour and that can be trained to

detect and track up to seven colour signatures. It is mounted at the front of the robot, at a slight downwards angle that may vary slightly from robot to robot, due to the camera's mount being adjustable. It is connected to the Pi via mini-USB cable. The inclination is useful since the Pixy is used when the robot picks up pucks or tries to steer towards a certain colour. The Pixy has proven in early experiments to be more precise when picking up colours if they are close to the robot and more accurate at estimating where a colour is relative to the robot's position than using the image provided by the Bubblescope-equipped Pi Camera. Objects closer than 15 cm to the robot are in the Bubblescope's blind spot, thus the Pixy is also helpful at picking them up. The idea is to detect markers using the Pi Camera, specifically those composed of only colours or a combination of colours and April Tags (see section 4.6), approach them, and then accurately steer towards them using the Pixy.

3.1.2 Software

The robot operating system (ROS) [53] was installed on the Pi as the main software controlling the robot. ROS nodes are processes that communicate together while performing computations. Nodes communicate with each other by publishing data onto "topics". In order to receive data from a certain topic a node subscribes to that topic. The application running on the Bupimos in our experiments required building seven nodes that run simultaneously. Figure 3.2 illustrates all the nodes and topics and their relationships with devices they communicate with. The "camera_publisher" node publishes two raw images: one at a resolution of 1024×768 published to the topic "image_topic" and another at half that resolution published on the topic "small_image_topic". The "apriltags_ros" node subscribes to the "small_image_topic" and detects April Tags in the provided raw image, publishing the detection information to the "aprilTags" topic. The "colour_detector" node subscribes to the "image_topic" topic and detects the colours green and blue in the image, publishing

their detection information to the “coloursDetected” topic. The “pixy_node” node provides a means to communicate with the Pixy camera and publishes the detected coloured blob information to the topic “block_data”. The “zumo_comms_node” node communicates with the Zumo over the serial connection. It sets up the “cmd_vel” topic that transmits the wheel speeds, that are to be assigned to each wheel, to the Zumo. It also reads the proximity sensor data from the Zumo and forwards it to the “zumo_data” topic. The “servo_control_node” node listens to the “servo_control” topic and, according to the data published to it, controls the Servo Motor. Finally, the “main_node” node is the thread that ties all the nodes together and is the basis of our application. It subscribes to the data provided by the aprilTags, coloursDetected, block_data and zumo_data topics, and processes that information in accordance with the algorithms discussed in Section 5.2, to move the robot accordingly by publishing to the “cmd_vel” topic and to manipulate the Servo Motor by publishing to the “servo_control” topic.

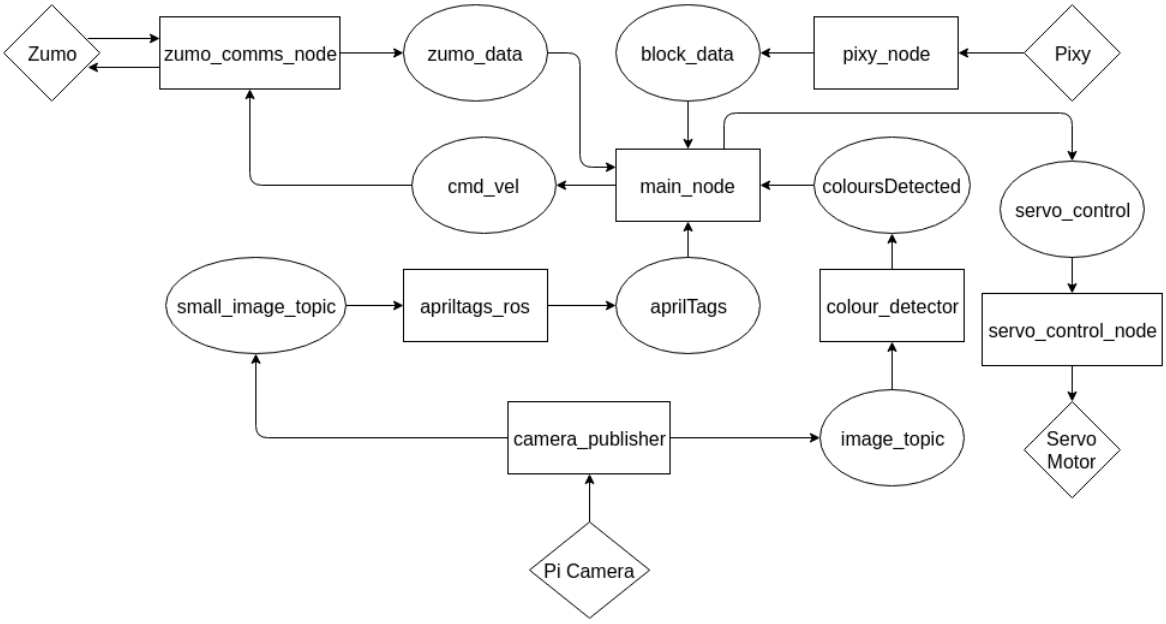


Figure 3.2: System Architechture Flowchart. Hardware is represented by diamond shapes, nodes are represented by rectangles and topics are represented by ellipses. An arrow going from a topic to a node indicates that the node is subscribed to that topic. An arrow going from a node to a topic indicates that the node publishes to that topic. An arrow from a device to a node or from a node to a device indicates that the node interfaces directly with the device, reading its data and providing it with data respectively.

3.2 Simulated Robots (SimBots)

The SimBot is displayed in Figure 3.3. The SimBot is a differential drive robot like the Bupimo. The two cuboids in front of it act as its gripper. A cuboid was added between them to shift the collected puck forwards into view of the spherical vision sensor. Consequently, the SimBot does not need a proximity sensor to sense that it has a puck.

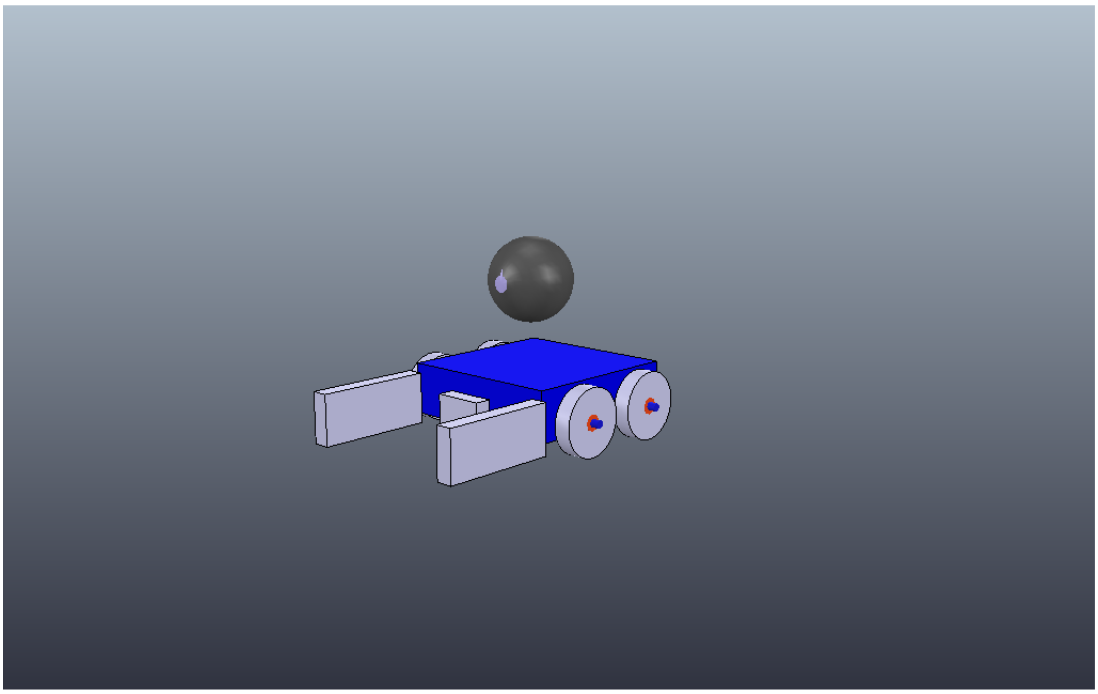


Figure 3.3: SimBot Model. The black orb is a spherical vision sensor. Both cuboids at its front act as a gripper and the cuboid in the middle props a collected puck into view of the spherical vision sensor.

In order to reduce the overhead time that would occur when images and other messages are transmitted along ROS topics, the tasks of analyzing the input images provided by the spherical vision sensor to extract colour detection information, and adjusting the wheel speeds accordingly, fall to one node: "MainBehaviour". The typical flow of information in the simulations after everything has been set-up is shown in Figure 3.4. For each SimBot, an instance of this node is run on an independent thread when the simulation starts. Those instances connect to V-REP [7]. V-REP launches these instances of the node, then it provides, as arguments, the SimBot's in-simulation I.D., and the handles of the SimBot's vision sensor and motors. Each node is given a unique name starting with the string "Robot", and ending with the SimBot's I.D. Afterwards, each SimBot's MainBehaviour node thread requests that V-REP creates a topic whose name is prefixed with "/vrep/sphericalVisionSensorData" followed by the SimBot's I.D., creating a unique topic for each SimBot to publish

its input images on. Each node thread creates its own topic to publish the motor speeds on. That topic's name is the name of the node thread, with the string "/wheelSpeeds" appended to it. The thread requests that the SimBot associated with the same I.D. in V-REP subscribes to this topic. After analyzing the images, the MainBehaviour node thread publishes motor speed values to this topic to actuate the SimBot's wheels.

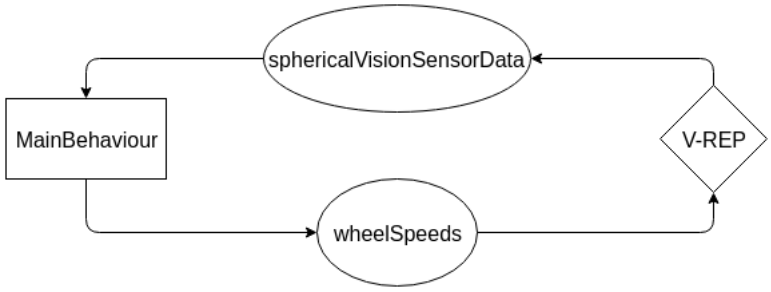


Figure 3.4: SimBot Simulation Data Flowchart. The rectangle represents a node, the diamond is V-REP and ellipses represent topics.

Figure 3.5 shows the BeaconBot. It is the same model as the SimBot, but without the gripper and the spherical vision sensor. The BeaconBot carries a colour visual marker flat on its body and is controlled by an instance of the node "beaconControl". Figure 3.6 shows the flow of data in the process of controlling the BeaconBot. The instance of this node is initialized similar to how the MainBehaviour node is. It subscribes to only one topic: the "keys" topic. Another node publishes to the keys topic, the "keyboardPublisher" node. It listens to all keypresses performed in the terminal that launches it and publishes all the keys pressed to the keys topic. depending on what keys are pressed the BeaconBot moves, as is explained in sub-subsection 5.3.1.2.

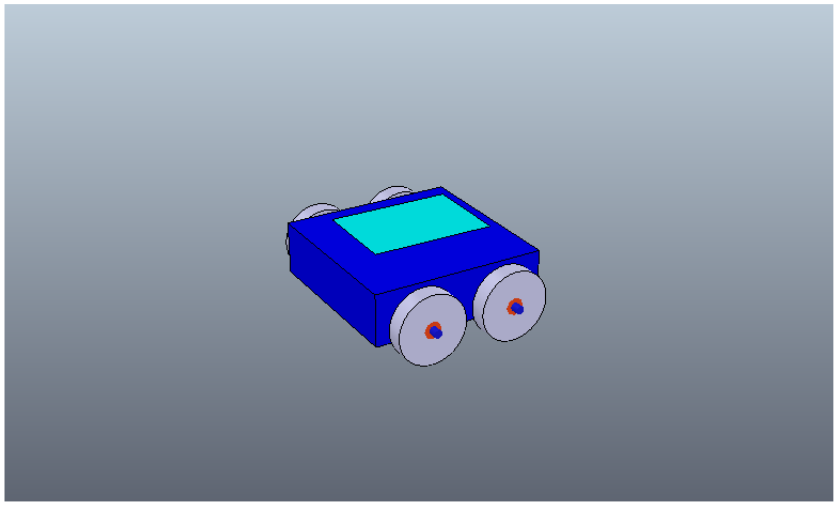


Figure 3.5: BeaconBot Model. The colour visual marker rests flat on its surface.

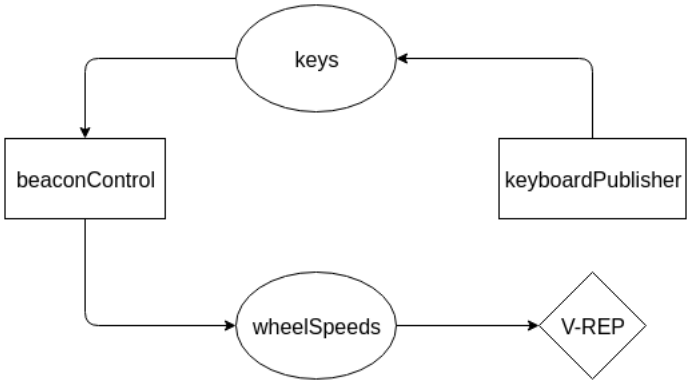


Figure 3.6: BeaconBot Simulation Data Flowchart.

Robot Vision

This chapter explores methods with which the raw images captured by our robots can be used to provide information about their surroundings. Since the robots are used to implement an SI approach, that information will be used to trigger behavioural responses to the environment and to other robots. A robot must, therefore, be able to detect other entities in its environment, such as other robots, walls, obstacles, pucks, as well as any other visual cues. We examine the use of colours and investigate some popular types of pattern-based markers that encode data in order to design our own visual markers. These markers will be used for the identification of other robots or objects of interest, determining their range and bearing relative to the observing robot, identifying obstacles, and as triggers for certain behaviours.

4.1 Overview

The Bupimos' cameras are equipped with catadioptric camera attachments, the Bubblescopes, that enable them to capture 360-degree panoramic images of their surroundings. The system works by using a curved, mirrored surface that bends incident light to produce a warped, "doughnut" shaped image. The wide field of view helps to keep the robots informed of their immediate surroundings without the need to turn in a circle. Typically, catadioptric systems do not utilize all the pixels that the camera sensor provides. The im-

age shown in Figure 4.1 is a raw output image of one of our robots' vision sensors with the unused pixels to the left and the warping effect showcased.

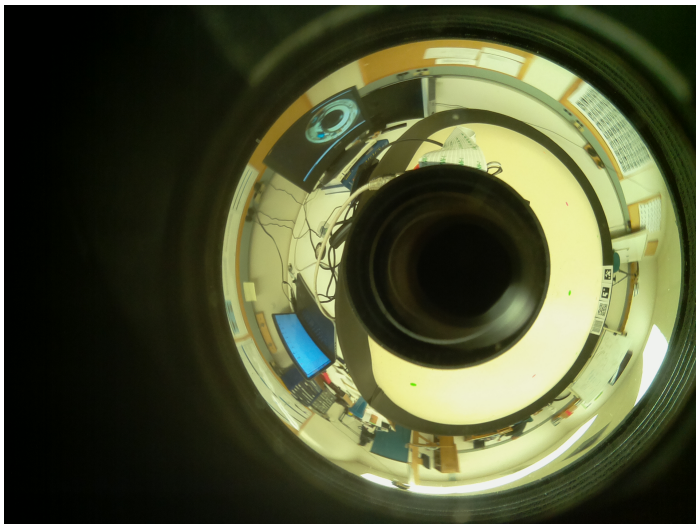


Figure 4.1: Raw Image Sample

In simulations, V-REP provides a "spherical vision sensor" that combines the images from six other 90-degree-field-of-view vision sensors. The sensor was configured so that the resulting image captures a 360-degree horizontal field-of-view and a 60-degree vertical field-of-view. The output image provides an image that is unwrapped, but which does not preserve straight lines and angles. Figure 4.2 shows an example output image of the simulated spherical vision sensor. In the simulations, the output image's dimensions are 512×256.

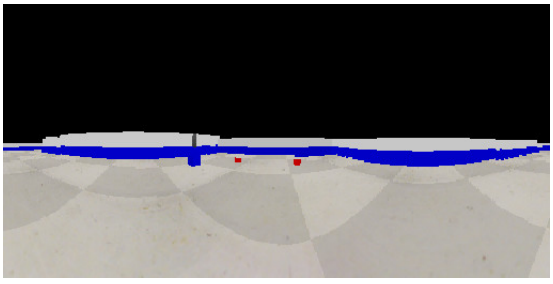


Figure 4.2: Simbot Raw Image Sample

The image's center depicts the environment ahead of the robot. By comparing that part

of the image to a snapshot of the area of the scene that it depicts, shown in Figure 4.3, the extent of the warping distortion is clear. A minor issue that is taken into consideration by the implemented vision system is that the image is mirrored, i.e. it is flipped horizontally. Objects to the right of the robot's forwards heading in the scene's 3D space, appear to the left in the image and vice versa.

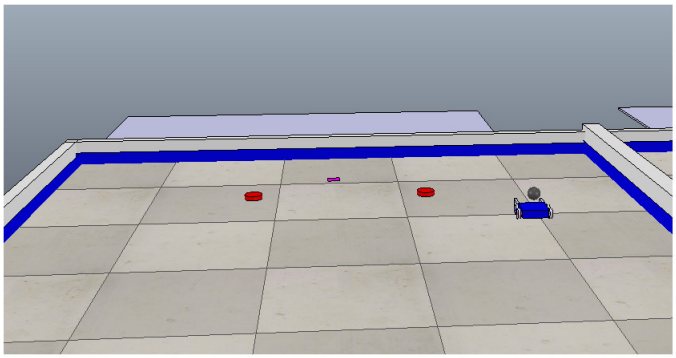


Figure 4.3: Snapshot of the Actual Scene.

V-REP's environment offers the unrealistic advantage of being uniformly lit. Regardless of a robots' position or orientation relative to an object, that object's colour will never change in the spherical vision sensor's output image. This leads us to consider colour detection as the most suitable method for extracting information from the images in simulation. It is also a simple and efficient process compared to the process of detecting and decoding pattern-based markers. Our simulations will, therefore, use colour-only markers, and will not use any other type of labels to encode information.

In real-world environments, however, colours can become washed out or darkened significantly, depending on the incident angle of light, reflections, and shadows cast by other objects in the environment. All these factors can contribute to colours not being detected at all, or being falsely detected as other, similar colours. Consequently, only the primary colours of Red/Bright Pink, Blue and Green, in addition to black were considered in the real-world experiments, given the variance in their values in most well-known colour

spaces. Since this is a rather limited selection, the work detailed in the following sections was carried out to determine what other labels could be used in conjunction with colours to offer more options.

4.2 Related Work

In [30], Fiala uses the ARtoolkit to detect unique markers in a panoramic image. Fiala then moved away from that approach in [31] to produce more reliable results and limited space consumption; the proposed navigation system uses vertically encoded linear marker patterns that resemble vertical barcodes. They discuss the methods of their detection and decoding in the warped image. While the results are promising, this approach is not applicable to our robots and applications, given that the minimum distance required to detect a code was 83 cm and the codes were designed to serve as large location markers mounted on walls. The codes we require should be small enough to mount on a robot without obstructing its field of view and should be visible when the robots are close to each other. Objects that are closer than 15 cm tend to be out of the field of view provided by the Bubblescope. Accordingly, that is the minimum distance at which a marker is to be readable.

The approach utilized in this paper relies on the recognition of both colours and another type of marker that can encode data, i.e. a pattern. This approach is similar to the one proposed by Alers et al. in [12]. Their method depends upon the placement of cylindrical features in the environment. Each cylinder is painted with a purple colour at the top and carries an EAN-8 barcode or QR code at the bottom, depending on the density of data that they need to encode. The idea is that the colour would be visible from afar, attracting the robot until it is close enough to detect and decode the barcode underneath it. As for information about other robots, all of their robots are equipped with LEDs that emit a coloured light, functioning much like the purple colour to attract other robots. Each robot

is provided with a body pattern in the form of a black triangle with two black lines on top of it. That pattern is processed to determine the robot's orientation and distance once it is within range. All the processing described before is optimized to make up for the limited processing and memory capabilities of the E-Puck robots that they use, whose memory is limited to 8 Kb. In this case, the robots are using simple cameras without an extended field of view. They claim that an experimental framework that provides the capabilities of their approach is enough for any kind of swarm experiment using swarm robots.

4.3 Approach

While the robots being developed at our lab have more memory and processing capabilities than the E-Puck robots, significant slowdown is experienced depending on the type of image processing library being used and the resolution of the captured image being processed. The distortion, caused by the wide field of view, to most well-known markers such as barcodes and QR codes is significant, rendering some unreadable, therefore we start out by selecting a suitable type of marker that can be reliably detected in the raw image.

To determine the best type of marker, the experiment detailed in section 4.3.2 was carried out. The marker that we decided upon using is the one that proved to be the most detectable at the largest distance. Markers are placed directly on the environment and on the robots. The functionality of colours and markers depends on the application being implemented and is not limited to the uses detailed in this thesis. The robots are to be encased in a colour that all robots identify as a robot-specific colour, to ensure their detection when they are too far, and to avoid collisions when they are too close.

A colour detection algorithm is needed to detect the colours in the image. The shapes of coloured objects are not considered in the detection process; the only information needed is what colours they are, where these colours are in the image, and how large they appear

to be in the image.

Markers that are composed of patterns like barcodes and QR codes cannot be detected at distances as large as those that colours can be detected at, however, colour detection can be unreliable due to changes in lighting and viewing angles. Thus, a new marker type is to be made that combines both colours and patterns, similar to the one proposed by Alers et al. [12]. Each pattern marker is to be accompanied by a band of colour at its top, to alert the robots that are too far to detect the pattern.

4.3.1 The Robot

The robot used to develop the markers is the BuPiGo [57]. It is the previous iteration of the Bupimo and was used in this experiment because the Bupimos were still in development at the time. The BuPiGo uses the same vision system as the Bupimo; it is also equipped with a Raspberry Pi Camera that is affixed to the Bubblescope lens attachment. The added height of the Bubblescope enables the robot to see farther and avoid unnecessary occlusion of the image because of its body getting in its way. An early model can be seen in Figure 4.4.

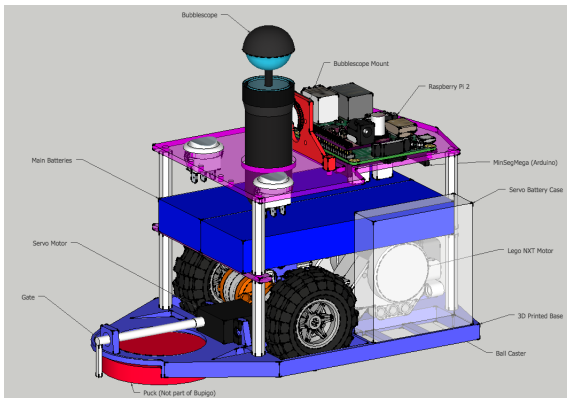


Figure 4.4: BuPiGo model.

The BuPiGo's Raspberry Pi Camera connects to the Raspberry Pi 2 [9], which is equipped with a quad-core ARM processor and 1 GB of RAM. Similar to the Bupimo, all image pro-

cessing and behaviour logic is performed on the Pi. In terms of software, the robot operating system (ROS) was installed on the Pi as the main software controlling the robot.

4.3.2 Choosing a Suitable Marker

In order to decide upon which type of pattern marker, i.e. code, to use, this experiment was carried out to test the codes that have proven most readable with a camera whose output image is not warped. The codes also have pre-existing, optimized libraries available for their detection and decoding. UPC-E barcodes, QR codes and April Tags were tested.

Of the April Tag families, two were tested: the 16h5 family and the 36h11 family. The naming convention is explained in Section 4.3.2.3. The "zbar" barcode reading library [10] is used to read UPC-E barcodes and QR codes while the "apriltags" library [50] developed by the APRIL lab at the university of Michigan is used to read the April Tags. An acceptable marker type is one that, at an unobtrusive size, i.e. not larger than the side of the robot, is read reliably with a detection range between 15 and 100 cm. Attempting to increase the detection accuracy of these markers by implementing our own detection library or by heavily editing the tested libraries is beyond the scope of this thesis and thus was not attempted. The best performing library was to be used without alteration to save development time.

4.3.2.1 UPC-E Barcodes

These barcodes are the most dense and least wide 1-D barcodes that the zbar library can identify. The relatively small size of each barcode and the thickness of each bar relative to the barcode's width make it the most readable type of 1-D barcodes in this setting, i.e. the bars do not blur together as much in the warped image as compared to other barcode types whose bars are thinner and more densely packed. An example UPC-E barcode is shown in Figure 4.5.

Figure 4.5: UPC-E Barcode

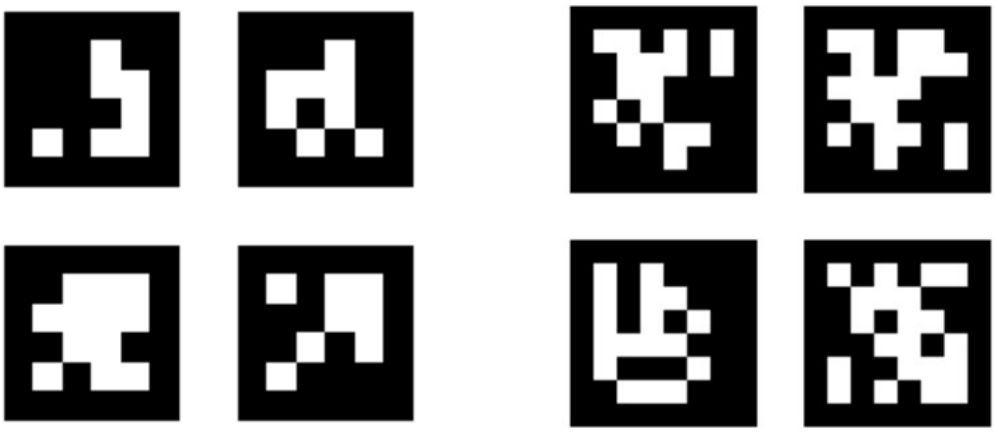
4.3.2.2 QR Codes

Quick Response (QR) codes are two dimensional barcodes widely used due to their easy, fast and accurate detection and decoding. They are robust to changing lighting conditions, and are still detectable if parts of the codes are damaged or occluded [45]. An example QR code is shown in Figure 4.6.

Figure 4.6: QR Code

4.3.2.3 April Tags

These tags are usually used in Augmented Reality for video tracking and overlaying virtual objects on the real world, as well as in robotics and computer vision applications. Each tag family is identified by a code in the form n^2hm where n^2 is the number of bits the tag encodes in the form of an $n \times n$ array, m is the minimum Hamming distance between codes and h is a separator. For example, the 36h11 is a family of April Tags that encodes 36-bit values, at least a Hamming distance of 11 apart, encoded as a 6x6 grid of black and white cells, as is shown in Figure 4.7b.



(a) April Tags - 16h5 Family

(b) April Tags - 36h11 Family

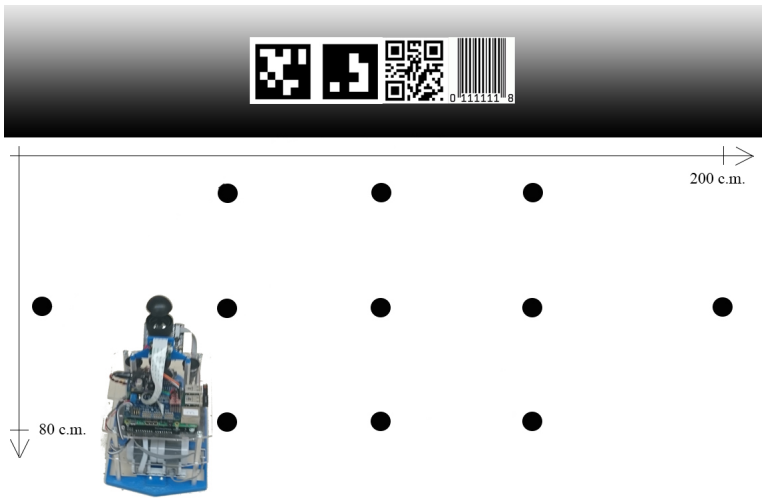
Figure 4.7: April Tags

The 16h5 family is restricted to encoding 30 values, with simpler and less detailed patterns than the 36h11 family, which makes the 16h5 family easier to detect. The 36h11 family, on the other hand, can encode 587 values. They were designed to be robust to various distortions [50], which should theoretically facilitate their detection after the image is warped.

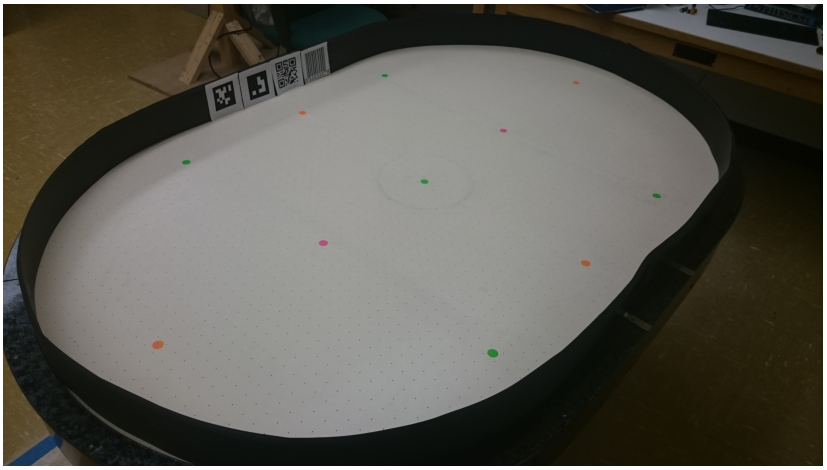
4.4 Experiment

4.4.1 Setup

The experimental setup is shown in Figure 4.8. The circles, black in Figure 4.8a and of various colours in Figure 4.8b, represent locations at which the BuPiGo is placed to capture an image. At each capture location 4 images are taken; once with the BuPiGo facing the codes, once with the codes to its right, once when they are to its left and once when they are behind it. The tests are performed at various orientations to ensure that any asymmetry or non-uniformity in the way the image is warped does not factor into the results.



(a) Sketch of the Experiment Layout. Black circles represent capture positions.



(b) The Arena. Coloured circles represent capture positions.

Figure 4.8: The Experimental Setup.

The codes' dimensions are 6.5 x 7 cm. From right to left the markers to be tested are the 1-D barcode, the QR code, an April Tag of the 16h5 tag family and an April Tag of the 36h11 family. The distances between codes and the capture positions range from 101 cm at the farthest to 10 cm at the closest.

4.4.2 Testing Markers - Phase 1

44 images were taken, covering each spot on the capture grid in the arena. The images are then provided as input to the detection libraries, and the best marker type is decided upon by how consistently it is detected and correctly decoded.

4.4.3 Results

The chart shown in Figure 4.9 gives an overview of the results.

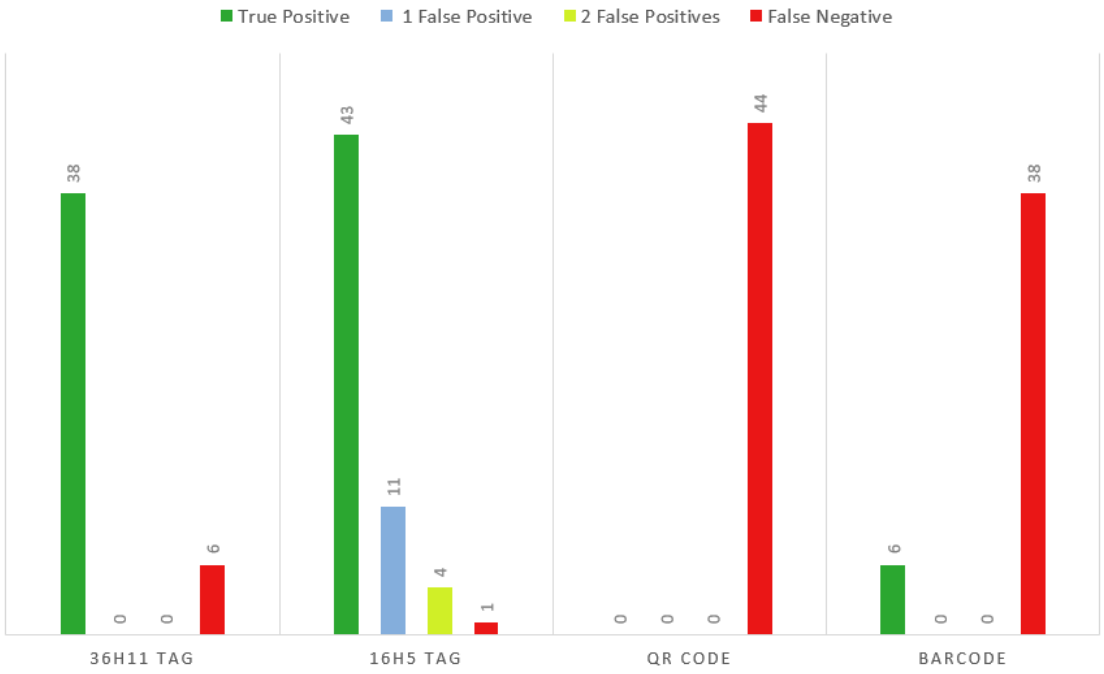


Figure 4.9: Detection stats for all 4 marker types

4.4.3.1 UPC-E Barcodes

The barcode was only occasionally detected at the closest capture positions to it. Out of 44 images, six of them resulted in a true positive barcode identification. It was most reliably detected when the images were taken at the capture position at a distance of 20 cm to the codes. At a larger distance, it was not reliably detected and its bars began to blur together.

4.4.3.2 QR Codes

The QR code was not detected at all. It was not detected even when the capture position allowed a clear view of the code and with minimal distortion.

4.4.3.3 April Tags

The most readable of the codes were the April Tags. The 36h11 family resulted in 38 true positive detections and six false negatives. The 16h5 family resulted in 43 true positives and one false negative. Of the 43 true positives 11 identifications of the tag were accompanied by false negative identification of another 16h5 tag that was not present in the arena. Similarly, four true positives were accompanied by two false positives. The only instance of a false negative identification, i.e. the tag was not detected at all, was accompanied by a false positive identification of another non-existent tag in another part of the image.

The best performing markers according to the requirements specified in subsection 4.3.2, i.e. the April Tags, were tested again in subsection 4.4.4 to determine their maximum distances and angles of detection. When the detection libraries were run, detections took an average of 3-5 seconds to be made. Performance is important in our case, because the robot needs to be able to react to moving objects and other robots in its environment.

4.4.4 Testing Markers - Phase 2

The detection rate in a constant stream of images proved too slow for real time operation, and transmitting the images through ROS had an added delay due to the high resolution. Therefore, the images are captured at a resolution of 1024×768, reduced from 2592×1944, with an unchanged aspect ratio, to speed up the detection process and reduce the lag ROS presents. At this new resolution, the tags were tested to determine the range of distances and angles at which the tags can be detected. The maximum distance and angle at which an

April Tag can be detected are measured according to Figure 4.10.

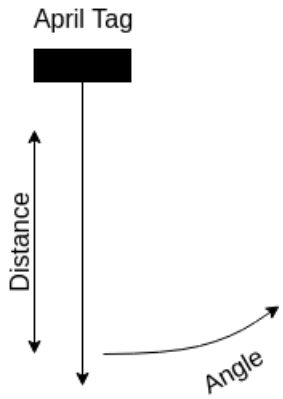


Figure 4.10: Detection Distance and Angle Calculation. The black rectangle represents the April Tag in a top-down view of the arena.

4.4.4.1 Detection Ranges

The 16h5 tags yielded more true positives, though false negatives were still an issue. They were detected at a maximum distance of 75 cm and a minimum distance of 15 cm. The 36h11 tags yielded no false negative detections, though they were detected at a maximum distance of 62 cm and a minimum distance of 20 cm. The 16h5 tags were also detected at a wider angle of 65 degrees, whereas the 36h11 were detected at a maximum angle of 55 degrees.

Based on these results the 16h5 April Tags were selected as the type of marker used to encode data. The "apriltags" library provides the x and y pixel coordinate values of the center of the detected tag. These values are used to determine the position of the tag relative to the robot.

4.5 Colour Detection

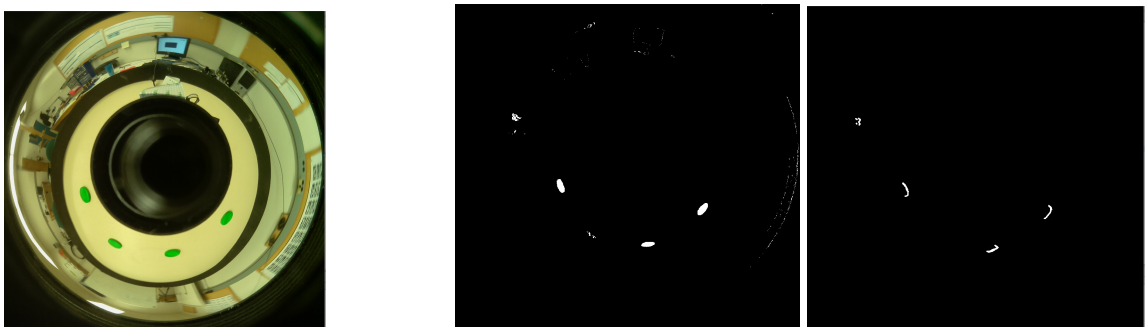
We distinguish between two main motivations for colour detection. The first is to detect and classify the objects in the robot's environment. The second is to steer away from other

robots and the boundary of the environment so as to avoid collision. At first, the same technique for colour detection was used for both purposes, but it proved lacking in the case of collision avoidance. In practical and simulated experiments, the robots were unable to accurately estimate the locations of walls and large objects around them using the method specified in Subsection 4.5.1. Therefore, the method detailed in Subsection 4.5.2 was used, to implement a more robust visual component of the collision avoidance system.

4.5.1 Detection for Object Recognition

Basic colour detection is performed using the OpenCV library [3]. The image is converted to the HSV format and filtered based on the colour that needs to be detected by applying a threshold. It should be noted that in an attempt to remove fine, unwanted details from the thresholded image, morphological opening and closing operations were performed in earlier stages of development. They proved to be too computationally taxing and slowed down the detection process enough to warrant doing away with them completely, without any negative effects on the experiments, simulated or not.

The result is an image with various white blobs representing the required colour. The contours of these blobs are extracted and their spatial moments are evaluated, providing the pixel locations of the blobs' centers, and their areas. Figure 4.11 shows an example of the different phases that the image goes through in the process of detecting the colour green, which is associated with pucks in our application.



(a) Original Image (b) Step 1 - Image Thresholding (c) Step 2 - Contour Extraction

Figure 4.11: Steps to detect the colour green

4.5.2 Detection for Collision Avoidance

Calculating the center of the detected blob works well when the object is small or when the goal is to steer towards the detected blob. When the robot is faced with a large obstacle, like a wall or a nearby robot close enough to occupy a large portion of its field of view, using the center of the blob has proven problematic in early experiments and has not been an accurate predictor of where a large blob is relative to the robot, or how far it is from the robot. To overcome this issue, we filter the image for the colour associated with an obstacle (black for the Bupimos and blue for the SimBots). The colour detection is performed the same way that is described in Subsection 4.5.1. We then examine the areas in the thresholded image that represent the robot's immediate surroundings in the three directions: forwards, right and left as is shown in Figures 4.12 and 4.13. The green-coloured areas in the figures are associated with the areas to the immediate right of the robots, the ones in purple corresponds to the immediate left and the one in white is the area straight ahead. Recall that the simulated images are mirrored. All the pixels in those areas in the thresholded image are examined. If any of them are white after thresholding, i.e. part of a detection blob, then there is an obstacle in that direction that needs to be avoided. The robot interprets that information by updating boolean variables representing whether or not it can turn right,

left or move forwards. The robot uses this information in Algorithm 5.1 to determine its steering path.

Figure 4.12: Obstacle Detection Regions in Simulation.

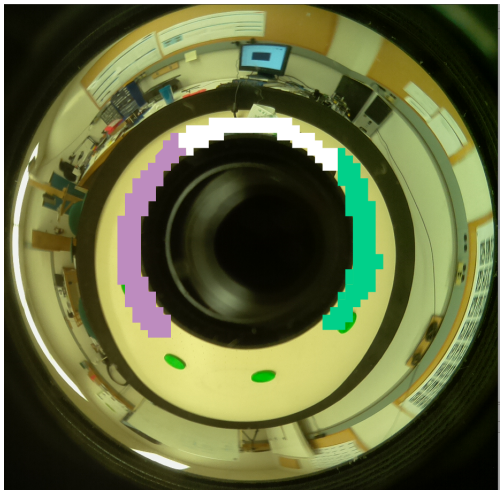


Figure 4.13: Obstacle Detection Regions for the BuPiMo.

4.6 Markers

Having implemented both April Tag detection and colour detection, three options present themselves as potential markers as shown in Figure 4.14.

(a) Combined Tag and Colour

(b) Colour Only

(c) April Tag Only

Figure 4.14: Marker Options

4.7 Determining distance and bearing

Calculating distance and bearing to objects in our implementation is limited to the image plane. The use of image distance as a proxy for spatial distance was sufficient for this foraging application. In simulation, the y-coordinate value of a detected blob's center is inversely proportional to its distance to the robot. Higher values are associated with closer objects. The x-coordinate value of the detected blob's center represents its bearing relative to the robot as is shown in Figure 4.15. The accuracy of the value of both the distance and bearing depends on the accuracy of the detected marker.

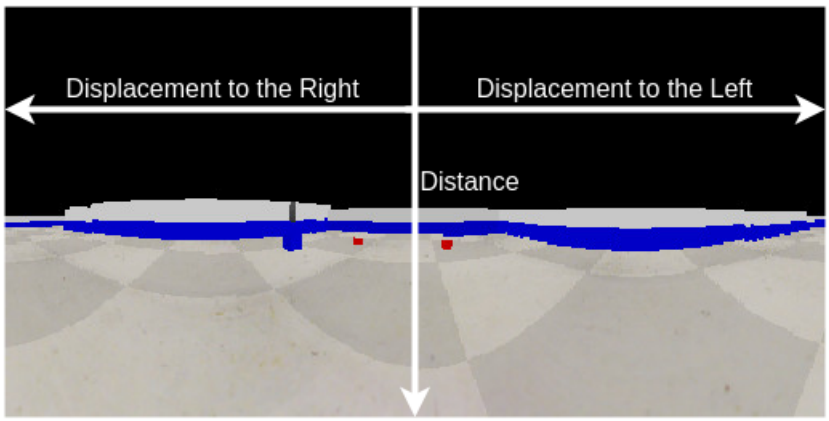


Figure 4.15: Sample Simbot Captured Image. All objects in the environment are assumed to be lying on the ground plane. Detections closer to the Simbot are closer to the bottom of the image, while those farther away are closer to the top of the image. The image is flipped so objects in the left half of the image are actually to the Simbot's right and vice versa.

In case of the Bupimos, and as seen in Figure 4.16, the Euclidean distance to the center of the "doughnut" image is calculated for each detection, colour or April Tag, and the angle between the detection and the image's x-axis, that runs through the center of the doughnut, determines its bearing.

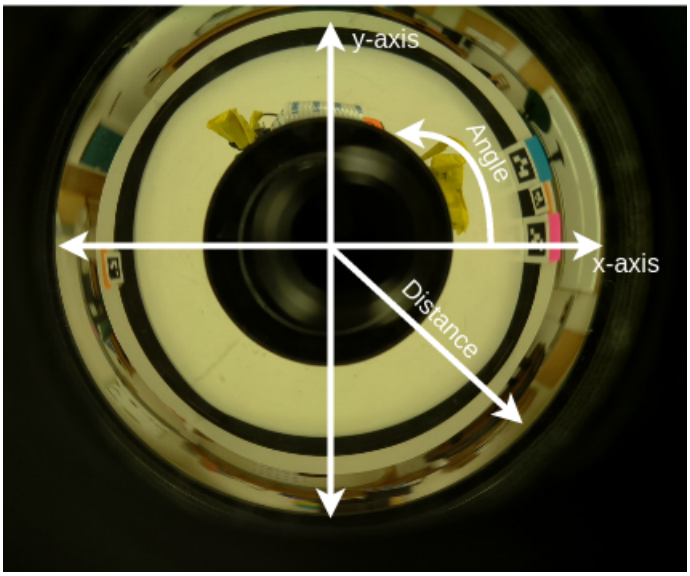


Figure 4.16: Sample Bupimo Captured Image's Angle and Distance Reference Axes. Detections made at a point farther from the origin, i.e. with larger Euclidean distance, are farther away. The positive side of the x-axis points to the direction ahead of the robot, i.e. forwards.

Methodology

5.1 Overview

To test the effects of adding HSI elements to an SI Foraging application, a simple solution utilizing SI only and a solution that uses priority-based HSI as well as SI were implemented and data describing the performance of both approaches was collected. A number of robots representing our swarm starts out near a location marked with a visual marker as a nest. Their task is to forage for, i.e. find and collect, wooden pucks that are green in physical robot experiments and red in simulations. The pucks in this case represent the resource that is being foraged. Pucks can only be found and collected at locations marked with a "source" visual marker. Pucks that are found anywhere other than near a source marker are ignored so as not to confuse robots near the nest that will try to pick up these pucks. The experiment ends when the robots have successfully found and retrieved a pre-set number of pucks from the source, and deposited them at the nest. Simulations were used as a prototyping tool before final implementation began on the Bupimo robots, and allowed the approach to be tested on two more robots, in a larger environment.

Before the approaches can be described, the following concepts need to be established.

5.1.1 Beacons

In the context of this thesis, a beacon refers to any visual cue that the robot recognizes as a trigger for an action to begin or an indication that a certain action should persist. For example, a robot might detect a visual marker that is associated with the "follow" action, i.e. it is an attracting beacon. The robot would then move towards that marker as long as it is detected.

Beacons can be either stationary or mobile. Martinoli refers to mobile beacons as active seeds in [26], however in this thesis we do not use that terminology. A mobile beacon is any visual cue, i.e. marker, whose position is intentionally altered. It can be a human operator holding up the visual cue or another human-controlled mobile agent with the cue mounted on top of it. We limit the use of the term beacon to a human controlled agent carrying one or many markers, or a marker that the human operator holds, whose position is changed to alter the flow of the experiment by affecting the behaviour of the robots. This implies that static markers that indicate the location of a nest or a source will not be considered beacons in this thesis, because they are not tools with which a human can interfere with the robots' behaviour. The robots do not identify a beacon as either stationary or mobile, the terms are used primarily to indicate to a human operator whether or not a beacon's position can be altered during an experiment.

The operator chooses which command a beacon communicates by choosing which marker is visible. The operator also decides where all the beacons are to be deployed, whether they are static or mobile, and where to move them if they are mobile beacons.

5.1.2 The Swarm

The type of swarm we use is homogeneous in terms of physical structure and software, only varying in a certain subset of parameters. All the robots in our swarm can perform

the same behaviours, however they are different in how they react to beacons. As with parameter control, each robot is assigned a parameter pertaining to beacon priority. If multiple beacons are detected, the beacon ranked highest on its priority list is the one that the robot reacts to. The priority list varies from robot to robot depending on the task that the swarm needs to perform. For example, if there are two beacons, one red and one blue, and we need half our swarm to prioritize the red and the other to prioritize the blue. Half the swarm's priority list would simply be "{red, blue}" and the other half's list would be "{blue, red}". The priority list is in order of descending priority, so the first item on it is the beacon of highest priority.

The swarm is human-blind, since it is unaware that the environmental control is operated by a human. If we consider mobile beacons to be a different type of agent, but an agent of the swarm nonetheless, then the swarm can be considered type-aware, given that the robots identify beacons as visual cues that are to be reacted to differently than other robots [33].

5.1.3 Common Behaviours

These are some common and useful behaviours used in implementing the approaches with and without HSI.

5.1.3.1 Moving and Collision Avoidance

The robots can either move forwards, or turn left or right. Robots do not move backwards unless they are depositing a puck or attempting to avoid collision with an obstacle. To move to a detected marker, the SimBots use the marker's coordinates in the image plane to steer towards it, while the Bupimos use the distance and bearing information detailed in section 4.7 to determine the steering direction according to where the detected marker is in the image. Algorithm 5.1 explains the movement and steering strategy of our robots.

Without a collision avoidance strategy the robots tend to run into walls and into each

other. Collision avoidance is not the focus of this thesis, however, it needed to be implemented to allow the robots to move as fluidly as possible and to avoid getting stuck. Consequently, a simple collision avoidance algorithm was implemented. Using the image processing technique described in subsection 4.5.2, we extract information on whether the robot can turn right, left or move forwards. The robot can move in a certain direction as long as its visual input indicates that there is no obstacle in its way. If there is, the robot attempts to move in a different direction. If it cannot move forwards or turn left or right, then it moves backwards. This behaviour is explained in detail in Algorithm 5.1.

Algorithm 5.1 Moving to a Target and Collision Avoidance Algorithms

```
1: procedure AVOID COLLISION
2:     if can turn right then
3:         hard turn right
4:     else
5:         if can turn left then
6:             hard turn left
7:         else
8:             if can move forwards then
9:                 move forwards
10:            else
11:                if has a puck then
12:                    stop
13:                else
14:                    move backwards
15:                end if
16:            end if
17:        end if
18:    end if
19: end procedure
20: procedure MOVE TO(destination coordinates)
21:    if destination is ahead then
22:        if can move forwards then
23:            move forwards
24:        else
25:            avoid collision
26:        end if
27:    else
28:        if destination is to the right then
29:            if can turn right then
30:                turn right
31:            else
32:                avoid collision
33:            end if
34:        else
35:            if destination is to the left then
36:                if can turn left then
37:                    turn left
38:                else
39:                    avoid collision
40:                end if
41:            end if
42:        end if
43:    end if
44:    return
45: end procedure
```

Typically, turning right or left is accomplished by stopping the rotation of one wheel while rotating the other. That method of turning, as opposed to rotating the wheels in opposite directions, allows the SimBot to rotate with a puck in its gripper without the puck sliding out. When the robot decides it should avoid an obstacle, one wheel is set to a speed and the other wheel is set to a speed that is two thirds that speed, in the opposite direction. This allows for a harder turn, while still maintaining the puck in the SimBot's gripper. Bupimos do not face this problem due to their Servo Motors' locking the pucks in their grippers when they have a puck. The harder turn is still useful for both the Bupimos and SimBots because it makes collision avoidance easier.

5.1.3.2 Wandering

Simple random wandering is used. A random number between 0 and 2 is generated. The value of that number corresponds to a direction; 0 corresponds to the left, 1 corresponds to the right and 2 corresponds to forwards. Algorithm 5.2 details the wandering process.

Algorithm 5.2 Wandering Algorithm

```
1:  procedure WANDER
2:      if wanderCounter > 0 then
3:          wanderCounter = wanderCounter - 1
4:          if randomNumber == 0 then
5:              if can turn left then
6:                  turn left
7:              else
8:                  avoid collision
9:              end if
10:         end if
11:         if randomNumber == 1 then
12:             if can turn right then
13:                 turn right
14:             else
15:                 avoid collision
16:             end if
17:         end if
18:         if randomNumber == 2 then
19:             if can move forwards then
20:                 move forwards
21:             else
22:                 avoid collision
23:             end if
24:         end if
25:     else
26:         let randomNumber = generated random number %3
27:         let wanderCounter = 3
28:     end if
29: end procedure
```

5.1.3.3 Picking Up and Depositing the Pucks

Pucks are considered colour-only markers and are detected as such. To pick up a puck, a robot must be within the pre-defined range to a source marker as mentioned in sub-Subsection 5.3.1.1. The robot then chooses to pursue the puck that it is closest to. In the SimBot's case, that is the marker with a red colour indicating that it is a puck, that has the highest y-coordinate value, i.e. is closest to the robot. In the Bupimo's case, it is the

corresponding marker detection, i.e. green, with the lowest "distance" value. The robot rotates and centers the detection in its image in order to align itself so that it faces the puck. That is to say, a robot is considered facing a puck if the puck is placed in its gripper when it moves forwards. The SimBots use their spherical vision sensors for alignment, while the Bupimos start by using the Pi Camera and then switch to the pixy when the puck enters its field of view. The robot then moves forwards until it detects that it is in the state "hasPuck". That state is accomplished in the SimBot's case by having the puck's red colour detected in a certain location in the image, indicating that the puck is now resting in the robot's gripper. The Bupimo detects that it has a puck by reading its proximity sensor values.

To deposit a puck, the robot moves backwards for 1.5 seconds and then the avoid collisions function is used to steer it away from the puck. Letting the robot move freely after moving backwards proved unsuccessful, especially in the SimBots' case, given that the robot would reliably pick the puck up again and start depositing it repeatedly until it randomly chose to steer away.

5.1.3.4 Creating Subswarms

Next is the process of dividing the swarm up into subswarms, much like selection control, but by only using the tools described so far. The operator should be able to issue their commands to each subswarm individually or to all agents in all subswarms, i.e. the entire swarm. It should be noted that no agent of the swarm is aware of which subswarm it belongs to or which subswarm the command is being issued to; an agent is only aware of the priorities of the beacons that it detects.

Splitting up the swarm is accomplished by using attracting beacons of varying priorities. We prioritize some beacons in a subset of robots, and prioritize other beacons in other subsets of robots. This will result in the agents following their highest priority beacons, thus, by sending each beacon in a different direction, the swarm splits up. In order to

reunite the swarm, all but one beacon are removed and the remaining beacon is deployed to attract all agents. Another method to the same end would be to use an attractor beacon whose priority exceeds all other beacons.

As an example, consider the swarm in Figure 5.1. The red agents are those that prioritize beacons with a red marker, and the blue ones are those that prioritize beacons with a blue marker. This is the start position of the swarm. In this example, if there are no beacons, the agents are expected to wander around.

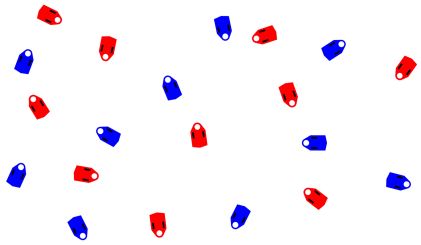


Figure 5.1: Random Swarm Distribution

In order to be able to separate the swarm in Figure 5.1, the operator deploys the two beacons described above. By turning on the beacons, represented by the red and blue circles in Figure 5.2, each agent is attracted by the highest priority beacon and we get the following separation. The priorities are set by the operator before the application is run according to the desired behaviour. For example, an uneven split can be achieved by having one attracting beacon prioritized by the majority of the swarm.

Figure 5.2: Separated Swarm

In this case, combining the swarm can be done by leading both subswarms closer together. This provides us with a swarm that can be divided, combined and deployed anywhere.

5.2 Approach

In the approach that uses HSI as well as the one that does not, the robots depend on sight and whether or not they carry a puck to determine their course of action. Not having a puck drives the robot to seek out a source and having one compels the robot to look for its nest. Both the nest and the source are represented by visual markers. The SimBot evaluates whether or not it is "at" a certain marker by examining the marker detection's y-coordinate value and whether or not it exceeds a certain threshold. The Bupimo evaluates whether or not the marker is located at a small enough distance to consider itself "at" the marker. The types of markers used and the distances between them and the robot are detailed in section 5.3. The robots do not keep odometry information nor do they localize in any way. A robot does not have any bias to a direction and looks for the nest and source by wandering randomly.

The systems implemented can be described as reactive. The presence of certain mark-

ers, or lack thereof, in the environment is the trigger of the robots' actions. The design of the system suggests an implicit hierarchy to the actions given the availability of their corresponding markers.

The robot is driven by being in one of two states; it either has a puck and is looking to deposit it, or it does not and is looking to pick one up. To avoid unwanted behaviours, like a robot attempting to pick up a deposited puck, or going after another robot's puck, pucks can only be picked up when the robot is at a source and can only be deposited when the robot is at a nest.

5.2.1 First Approach - Without HSI

The flowchart in Figure 5.3 describes the flow of conditions that the robot is constantly checking to make a decision on which course of action to follow, and Algorithm 5.3 shows the algorithm that implements it.

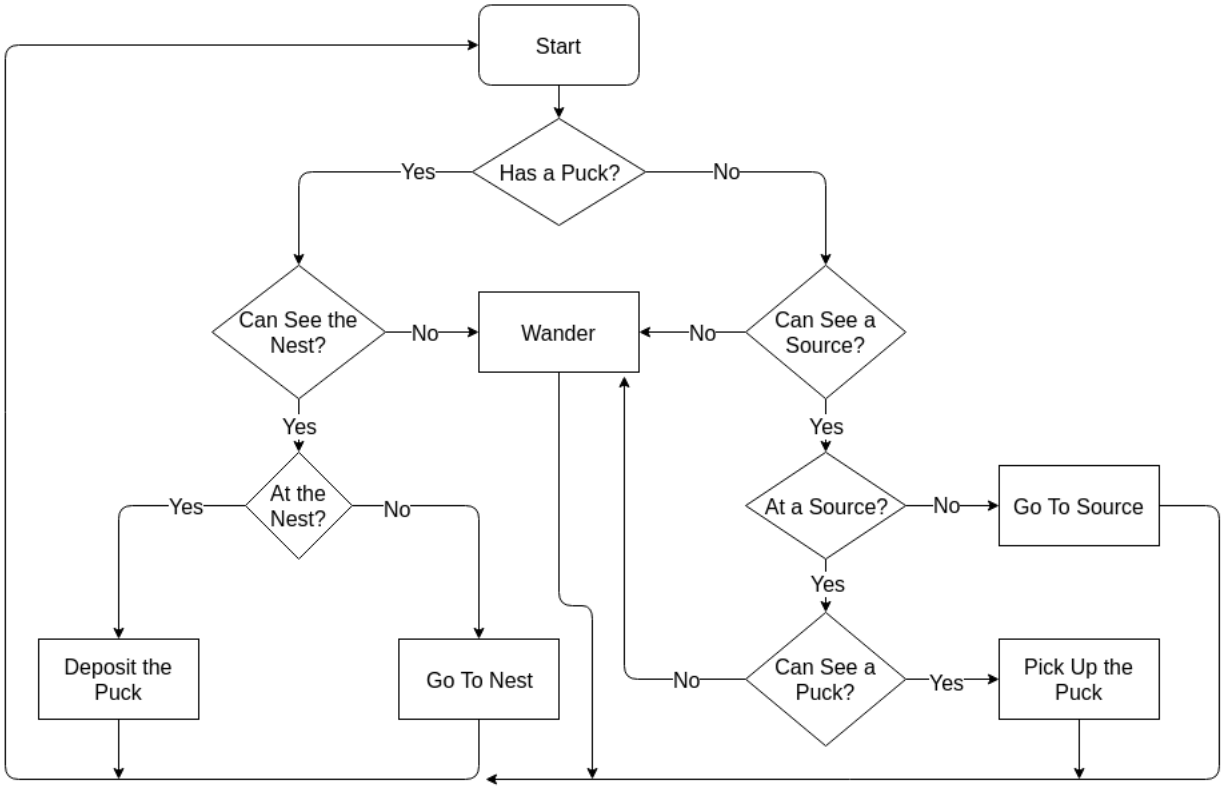


Figure 5.3: SI-only Behaviour Flowchart.

Algorithm 5.3 SI-only Foraging Algorithm

```
1:  procedure FORAGE(destination coordinates)
2:      if has a puck and at the nest then
3:          deposit the puck
4:          return
5:      end if
6:      if has a puck and can see the nest then
7:          go to the nest
8:          return
9:      end if
10:     if at a source and does not have a puck and can see a puck then
11:         pick up the nearest puck
12:         return
13:     end if
14:     if can see a source and does not have a puck then
15:         go to the source
16:         return
17:     end if
18:     if (does not have a puck and cannot see a source) or (has a puck and cannot see the
    nest) then
19:         wander
20:         return
21:     end if
22:     if at a source and does not have a puck and cannot see a puck then
23:         wander
24:         return
25:     end if
26: end procedure
```

5.2.2 Second Approach - With Priority-Based HSI

Figure 5.4 and Algorithm 5.4 introduce human-controlled, prioritized beacon awareness to the first approach. If the robots cannot find the nest to deposit their pucks or cannot find a source to pick up a puck from, and they observe a beacon that they are familiar with, they follow that beacon until they determine that they are close enough to that beacon. When they are close enough to a beacon they start wandering around it, maintaining a minimum distance of approximately 20 cm.

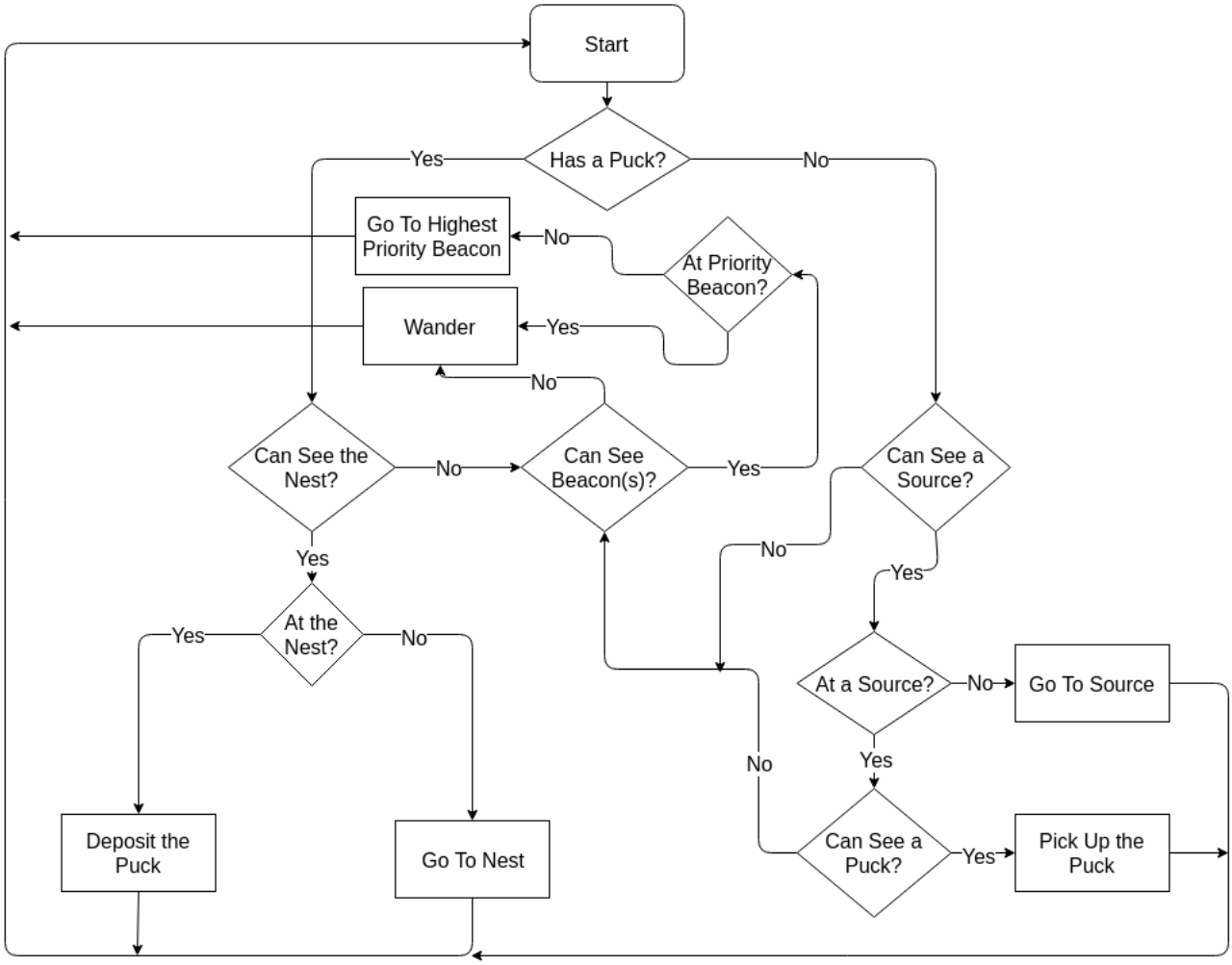


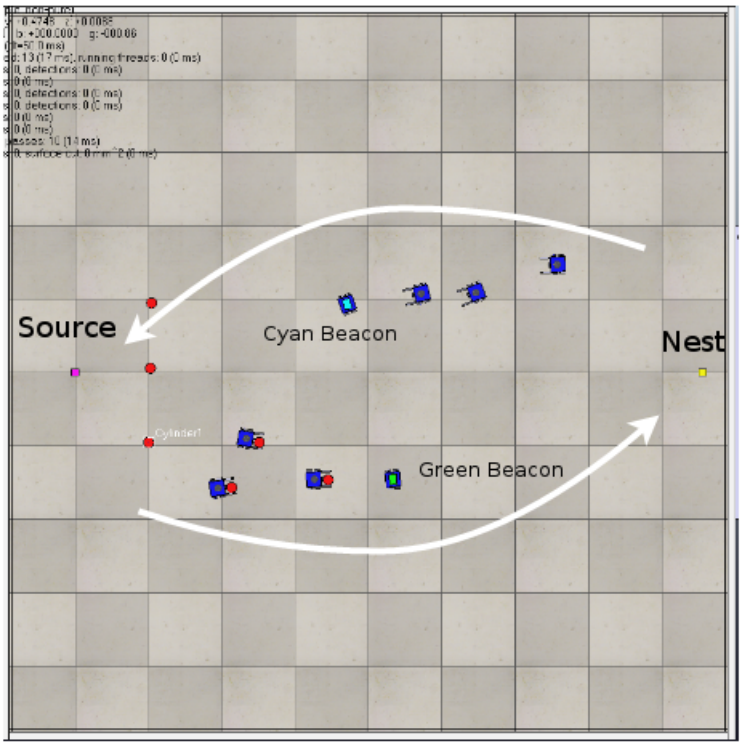
Figure 5.4: Priority-Based HSI Behaviour Flowchart

Algorithm 5.4 Priority-Based HSI Foraging Algorithm.

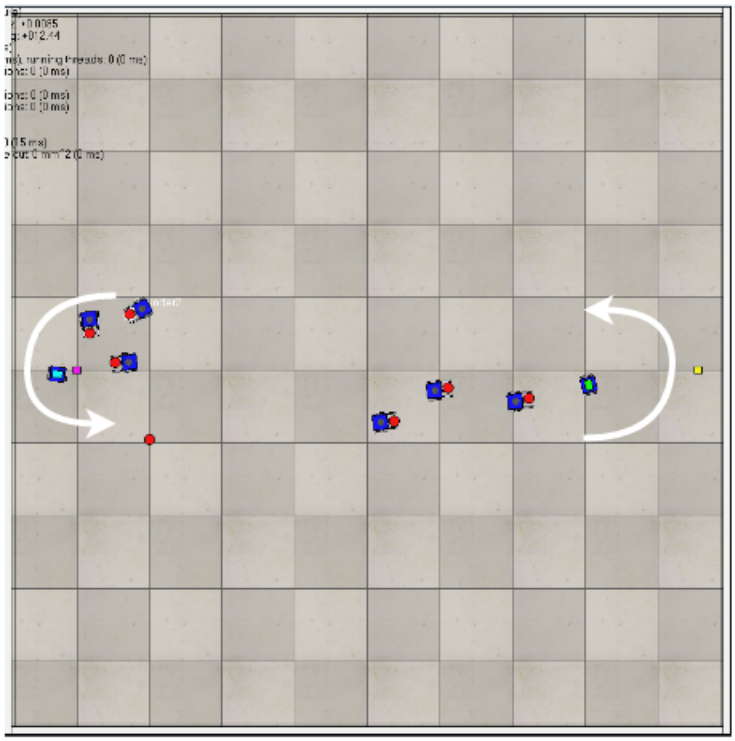
```
1:  procedure FORAGE(destination coordinates)
2:      if has a puck and at the nest then
3:          deposit the puck
4:          return
5:      end if
6:      if has a puck and can see the nest then
7:          go to the nest
8:          return
9:      end if
10:     if at a source and does not have a puck and can see a puck then
11:         pick up the nearest puck
12:         return
13:     end if
14:     if can see a source and does not have a puck then
15:         go to the source
16:         return
17:     end if
18:     if can see beacon(s) then
19:         if at detected beacon with highest priority then
20:             wander
21:             return
22:         else
23:             go to the detected beacon with highest priority
24:             return
25:         end if
26:     end if
27:     if (does not have a puck and cannot see a source) or (has a puck and cannot see the
        nest) then
28:         wander
29:         return
30:     end if
31:     if at a source and does not have a puck and cannot see a puck then
32:         wander
33:         return
34:     end if
35: end procedure
```

The strategy used by the operator controlling the beacons is to split the swarm up into subswarms, each subswarm following a beacon. Then, by keeping a beacon within view of its subswarm, the operator uses the beacon to attract them to a destination. Ideally, a cycle

similar to the one shown in Figure 5.5 will be established, where both subswarms move to their destinations along one path, maintaining the separation between them, then upon reaching their destination, i.e. the source or the nest, they take the second path back.



(a) Beacon Control Paths Between the Nest and the Source.



(b) Switching Paths

Figure 5.5: Beacon Control Path Cycle. Screenshots are taken from Simulations.

5.3 Experimental Setup

Two different experiments were carried out, one in simulation and one with the Bupimo robots.

5.3.1 Simulations

V-REP [7] was used as the simulation platform. It is a robot simulator based on a distributed control architecture, that uses an integrated development environment. It allows the simulation of realistic physics through a selection of physics engines. The one used in our experiments is the Bullet open source physics engine that simulates dynamics, with the dynamics settings set to "Very Fast" to speed up the simulations.

In the simulations of the approach without HSI, six of the SimBots described in Section 3.2 are used, with two of the BeaconBots added to them in the simulations of the approach with HSI.

5.3.1.1 Environment

Figures 5.6 and 5.7 show the environment in V-REP. It spans a $5 \times 5\,m^2$ area. The SimBots are to the right surrounding the yellow marker that marks the nest. The nest is placed at a distance of approximately 25 cm to the wall directly behind it. The environment is walled by 10 cm high walls, lined on the inside with a 5 cm high, blue bar which the robots detect for the purpose of avoiding the walls. The blue bodies of the robots are 10 cm long, 10 cm wide and 3 cm high. Their grippers are 7 cm long, 1 cm wide and 3 cm high. At the side opposite to the nest, is a purple marker representing a source. The source is assigned 30 pucks. After the 30 pucks have been picked up, the source marker is removed. The source marker is placed at a distance of approximately 50 cm to the wall directly behind it. The source is no farther than 75 cm from three pucks at a time, except when there are no

more pucks to collect from the source, i.e. 28 or more pucks have been picked up from its vicinity. Limiting the number of pucks readily available at a source is primarily to prevent clutter; having all 30 pucks in play per source presents a hectic, messy situation. Whenever a puck is removed by a robot, another simply takes its place.

The nest and source markers are both $5cm \times 5cm$, and lying flat on the floor to reduce their detection range. This leads to an implicit detection range of 100-150 cm for both of them. The goal is to emulate the Bupimo's real-world vision, where markers are only visible up to a certain range. Having markers visible everywhere in the environment gives the SimBots an unfair advantage over the Bupimos.

The robots start in the positions shown in figures 5.6 and 5.7.

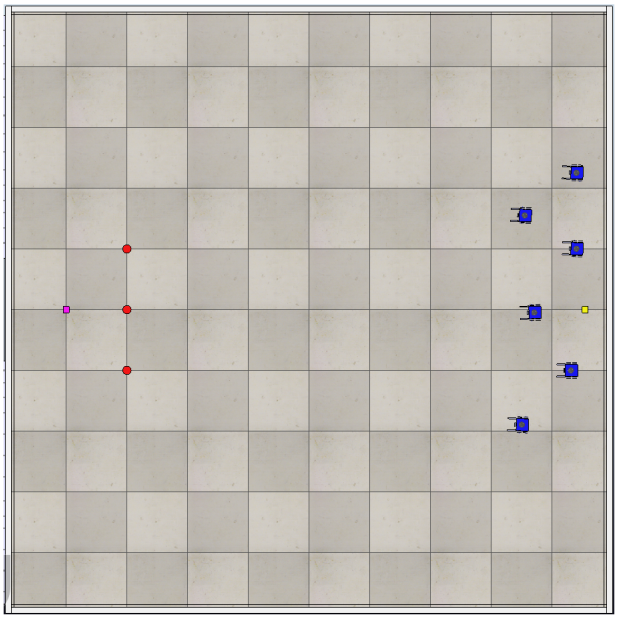


Figure 5.6: Environment Top View - First Experiment. The yellow marker represents the nest and the purple marker represents the source. Red pucks are placed close to the source and robots begin the experiment at the nest.

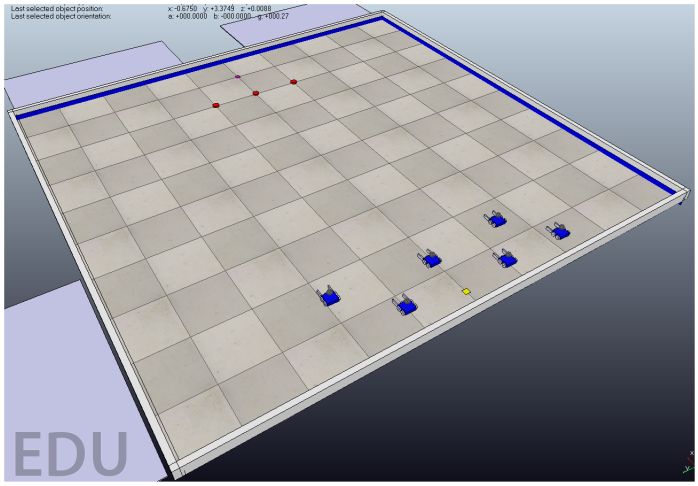


Figure 5.7: Environment Normal View - First Experiment

Figure 5.8 shows the second experiment with the BeaconsBots. The two beacons' markers are of dimensions 5 cm $\times$ 7 cm $\times$ 1 cm lying flat on the SimBot's body. They allow for a 150 to 175 cm detection range depending on the orientation of the detecting robot. The BeaconsBots' bodies are blue to trigger the collision avoidance reaction of nearby robots when they are close enough. This is just an added fail-safe to avoid robot-beacon collisions in case the robots wander too close to the beacons.

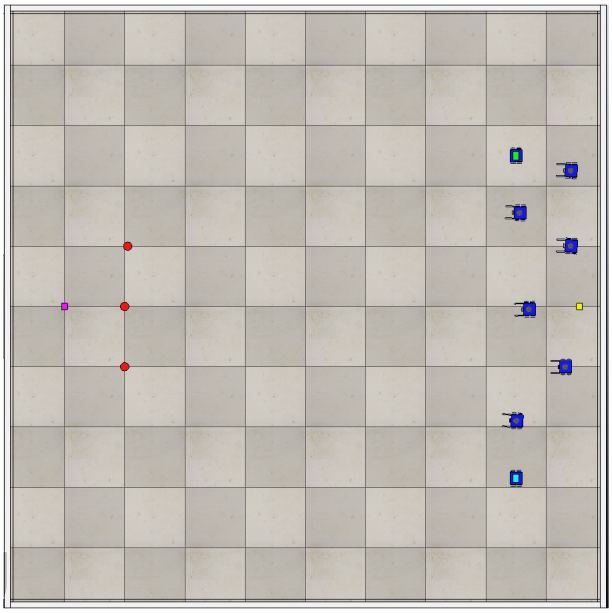


Figure 5.8: Environment Top View - Second Experiment. The beacons carry a green and cyan marker.

5.3.1.2 Beacon Control

In order to control the beacons, a “keyPublisher” ROS node that is separate from the simulation is started. It publishes pressed keystrokes to a “keys” topic. The beacons listen to that topic and, depending on the key pressed, they move. For example, “w”, “a”, “s”, “d”, “q” correspond to the movement commands “forwards”, “left”, “backwards”, “right”, and “stop” respectively for one beacon , while “i”, “j”, “k”, “l”, and “o” correspond to the same commands respectively for the other beacon. By inputting those letters in the terminal that launched the “keyPublisher” ROS node, the beacons can be controlled independently.

In this experiment, the beacon control strategy discussed in subsection 5.2.2 is used to move half the swarm to the source while the other half lingers at the nest. Picking up the pucks can take time given the robots’ tendency to avoid each other in close quarters. The process shown in Figure 5.5 is used to move the subswarms freely without entangling them. This process is repeated until all the pucks have been successfully retrieved.

5.3.2 Bupimos

In the experiments of the first, HSI-less approach, four of the Bupimos described in Section 3.1 are used, with two beacons added to them in the experiments of the HSI approach.

5.3.2.1 Environment

Figure 5.9 shows the environment that the experiments take place in, in our lab. It consists of a $188 \times 188\, cm^2$ walled square-shaped arena. The walls are 15.5 cm high, and black so that they trigger the robots' collision avoidance behaviour. They are contained within a 32 cm high white frame. The Bupimos' dimensions are $17 \times 10 \times 19$ cm. Attached to two opposite ends of the arena are visual markers representing the nest and the source as shown in Figure 5.10. The Bupimos can detect both at a distance of approximately 50 cm. Twenty green pucks represent the resource being foraged for and are deposited in the arena so that there are always two pucks available at the source at a given time. When all the pucks have been deposited at the nest, the experiment is complete.

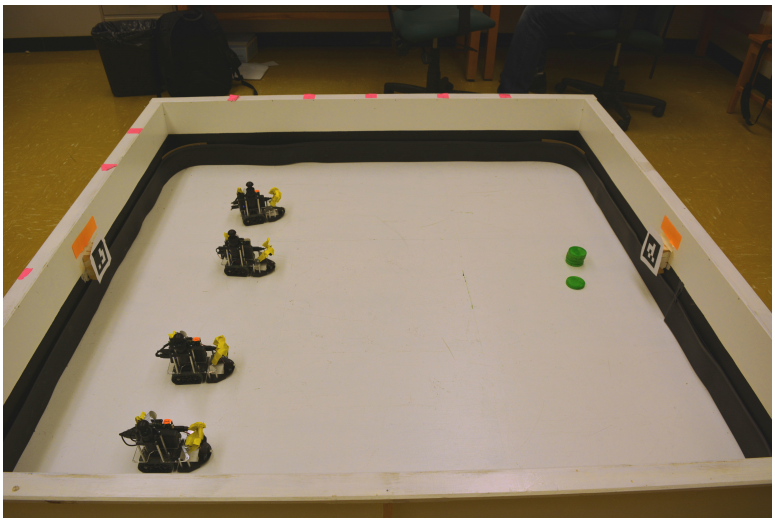
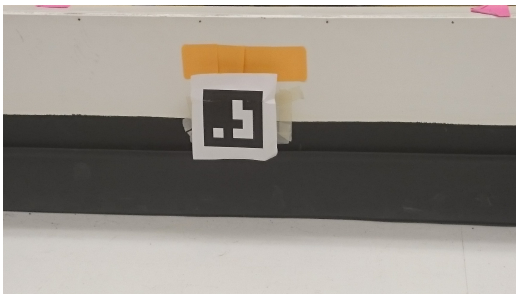
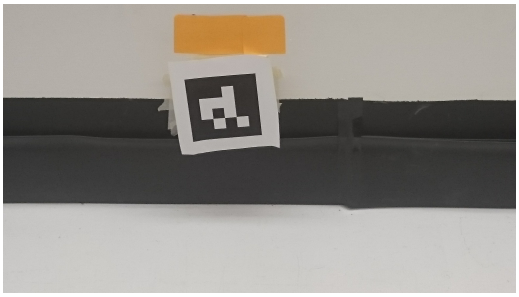


Figure 5.9: Bupimo Environment. The pucks used in the experiments are stacked in front of the source marker.



(a) The Nest Marker

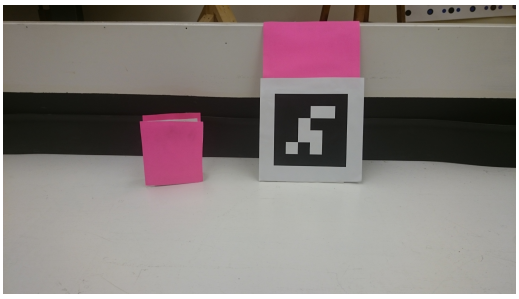


(b) The Source Marker

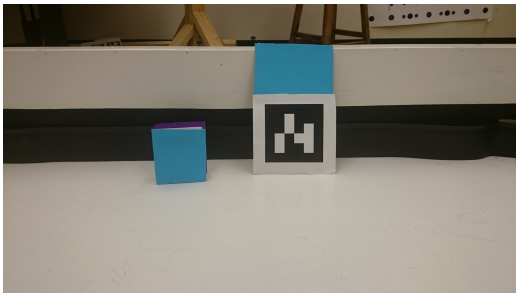
Figure 5.10: The Nest and Source Markers

5.3.2.2 Beacons

The beacons in this case are visual markers as shown in Figure 5.11, held and moved physically by the operator. For the first half of the experiments the visual markers using both a colour and an April Tag pattern were used, however, they could not stand upright on their own and needed to rest against the wall for support. They were replaced in the second half of the experiments by the colour-only visual markers that were attached to blocks of woods, allowing them to be set anywhere in the environment. The same strategy shown in Figure 5.5 was used by the operator to lead the robots.



(a) Beacons Used to Control the First Subswarm



(b) Beacons Used to Control the Second Subswarm

Figure 5.11: Beacons of the Bupimo Experiments

Performance and Results

In this chapter, the metrics of evaluating the approaches described in Chapter 5 are introduced and the means of obtaining them are explained. The results themselves are analyzed and discussed, and their qualitative implications are explored.

6.1 Overview

The effectiveness of both approaches, with and without HSI, is measured by five main metrics. First is the total amount of time it takes for the robots to complete an experiment, i.e. for all the pucks at the source to be retrieved. Solutions that solve the problem faster are, within the scope of this work, more efficient compared to solutions that take more time. Second is the distribution of deposits across the robots. The number of deposits across all robots should ideally be uniform. Third is the average time taken by a robot to make a deposit. Minimizing the time between deposits decreases the overall time of the experiment and thus increases the efficiency of the solution. Fourth is the amount of time spent idly wandering by the robots; the less time spent wandering, the better a solution's performance. Fifth is the amount of time spent in an "active state" that we refer to as "active time". A robot is considered actively foraging if it is following a beacon, approaching a nest or source, or picking up or depositing a puck. Spending more time in an active state should increase the deposit rate and thus decrease the overall time. Thus, the total time of the

experiment is the amount of time spent idly wandering added to the amount of time spent in an active state.

In general, the better solution is expected to be faster from start to finish, its robots must use their time well with less aimless wandering and more active foraging. The robots should contribute as equal a share of deposited pucks as possible, with less time between deposits.

This chapter presents and compares the performances of both approaches described in Chapter 5. To obtain the results, 15 runs of the simulations were performed per approach, and similarly, 15 runs of the real world experiments were performed per approach for a total of 60 runs. Thus, the data can be thought of as the following four groups of 15 runs each:

- Simulations with HSI.
- Simulations without HSI.
- Physical robot experiments with HSI.
- Physical robot experiments without HSI.

The data presented in this chapter's charts and tables represent the mean, and standard deviation of values extracted from each of these groups.

6.2 Data Extraction

The results were obtained by keeping a log file of important actions taken by the robots, both real and simulated. For each robot, when some triggering event is performed, for example when they are about to start a certain action, an entry is made in the log with a timestamp of when it was initiated. These events and actions are:

- The experiment starts.

- Wandering with a puck, while looking for the nest.
- Wandering without a puck, while looking for a source.
- Depositing a puck at the nest.
- Picking up a puck from the source.
- Approaching the nest.
- Approaching the source.
- Following a beacon (in the case of the HSI approach).
- The experiment terminates.

The log entries are then processed by a parser that extracts the required data pertaining to the robots' performance. The total simulation time is calculated by subtracting the experiment's starting time from the experiment's termination time. The amount of time spent idly wandering per robot is measured by subtracting the timestamp of the action that follows wandering from the timestamp at which wandering started. The wandering time combines wandering with a puck looking for a nest and wandering without a puck looking for a source. The active time is calculated similarly. The number of times a puck is deposited is calculated by counting the occurrence of the log entries stating that a puck was deposited. The same log entries' timestamps are used to calculate the average time taken in an experiment to deposit a puck per robot.

6.3 Simulation Results

6.3.1 Total Time

Table 6.1 shows the mean and standard deviation of the total time, in minutes, taken by the SimBots to finish an experiment with and without HSI. Figure 6.1 shows a box plot of that time across all simulations.

Approach	Mean	Standard Deviation
With HSI	30.024	3.539
Without HSI	101.785	10.672

Table 6.1: Simulations Total Time Comparison.

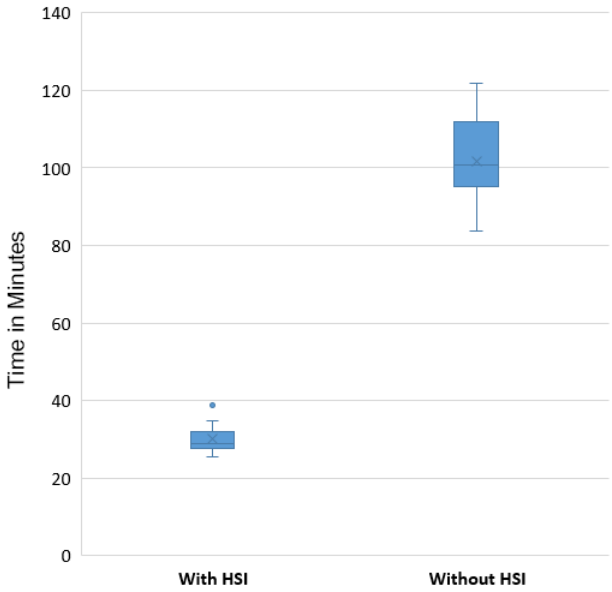


Figure 6.1: Simulations Total Time Comparison. In the box plot a box's top boundary represents the data's third quartile, the bottom boundary represents the first quartile, the line dividing a box represents the median or second quartile value, the upper whisker represents the highest value, the lower whisker represents the lowest value and the free points above or below the whiskers represent outliers.

6.3.2 Deposits

Table 6.2 shows the mean and standard deviation of the number of deposits made per SimBot for all simulated experiments with and without HSI. Figure 6.2 shows a box plot of the number of deposits.

Approach	Mean	Standard Deviation
With HSI	5	0.212
Without HSI	5	1.07

Table 6.2: SimBots' Average Number of Deposits per Robot Comparison

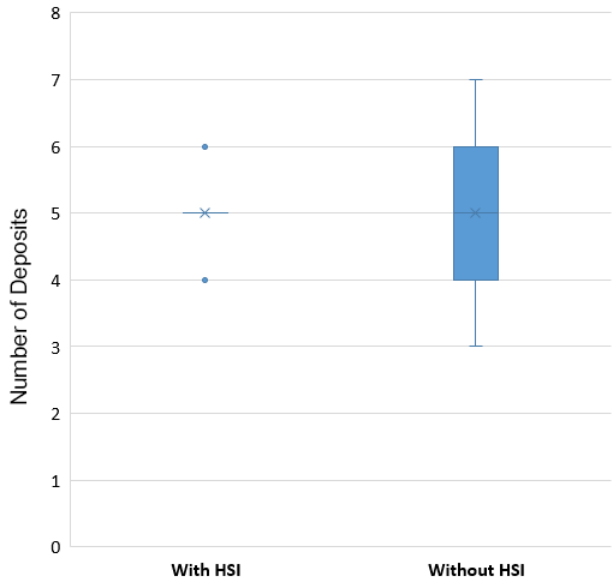


Figure 6.2: SimBots' Average Number of Deposits per Robot Comparison

6.3.3 Average Time to Make a Deposit

Table 6.3 shows the mean and standard deviation across all simulations, with and without HSI, of the average time taken to make a deposit, in minutes, for all SimBots. Figure 6.3 shows a box plot of that time across all simulations.

Approach	Mean	Standard Deviation
With HSI	5.66	0.733
Without HSI	18.875	4.942

Table 6.3: SimBots' Average Time to Deposit Comparison

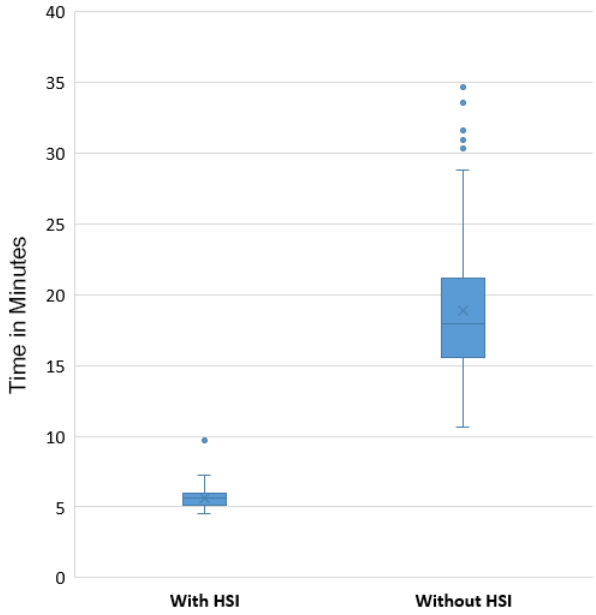


Figure 6.3: SimBots' Average Time to Deposit Comparison

6.3.4 Wandering Time

Table 6.4 shows the mean and standard deviation across all simulations, with and without HSI, of the time spent wandering, in minutes, for all SimBots. Figure 6.4 shows a box plot of that time across all simulations.

Approach	Mean	Standard Deviation
With HSI	5.292	1.454
Without HSI	99.328	10.136

Table 6.4: SimBots' Wandering Time Comparison

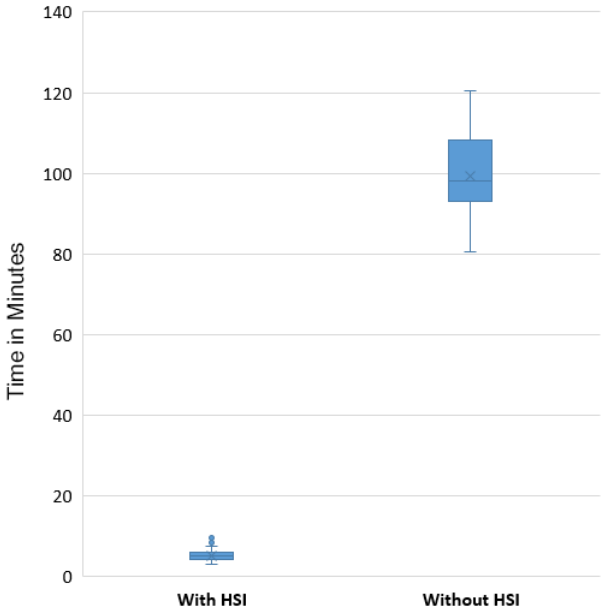


Figure 6.4: SimBots' Wandering Time Comparison

6.3.5 Active Time

Table 6.5 shows the mean and standard deviation across all simulations, with and without HSI, of the time spent actively foraging, in minutes, for all SimBots. Figure 6.5 shows a box plot of that time across all simulations.

Approach	Mean	Standard Deviation
With HSI	24.392	3.595
Without HSI	2.456	1.087

Table 6.5: SimBots' Active Time Comparison

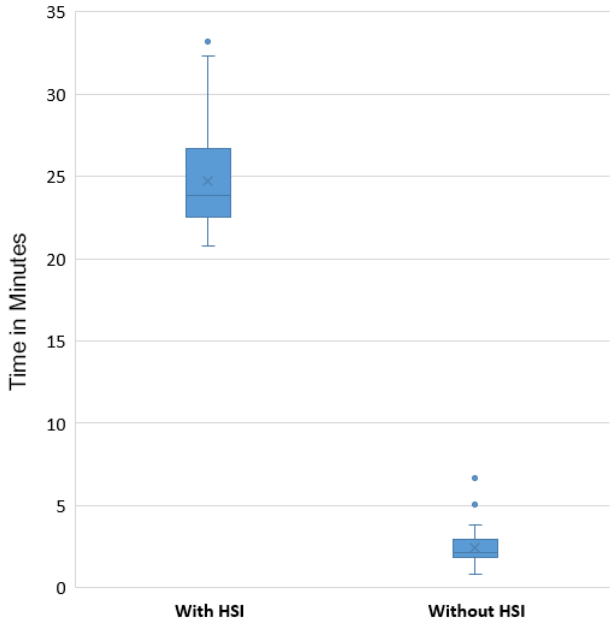


Figure 6.5: SimBots' Active Time Comparison

6.4 Results of Bupimo Experiments

6.4.1 Total Time

Table 6.6 shows the mean and standard deviation of the total time, in minutes, taken by the Bupimos to finish an experiment with and without HSI. Figure 6.6 shows a box plot of that time across all experiments.

Approach	Mean	Standard Deviation
Without HSI	52.28	6.985
With HSI	23.376	5.683

Table 6.6: Bupimo Experiments' Total Time Comparison

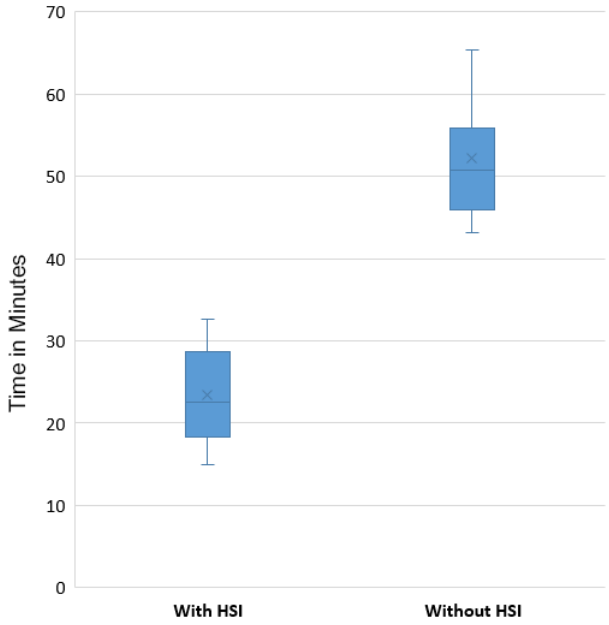


Figure 6.6: Bupimo Experiments' Total Time Comparison

6.4.2 Deposits

Table 6.7 shows the mean and standard deviation of the number of deposits made per Bupimo for all experiments with and without HSI. Figure 6.7 shows a box plot of the number of deposits.

Approach	Mean	Standard Deviation
With HSI	5	0.451
Without HSI	5	1.518

Table 6.7: Bupimo Average Number of Deposits per Robot Comparison

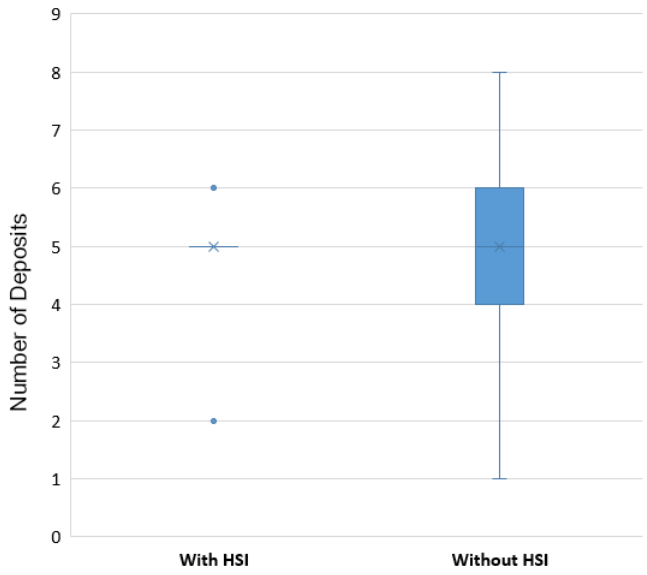


Figure 6.7: Bupimo Average Number of Deposits per Robot Comparison

6.4.3 Average Time to Make a Deposit

Table 6.8 shows the mean and standard deviation across all experiments, with and without HSI, of the average time taken to make a deposit, in minutes, for all Bupimos. Figure 6.8 shows a box plot of that time across all experiments.

Approach	Mean	Standard Deviation
With HSI	4.208	0.975
Without HSI	10.079	3.198

Table 6.8: Bupimo Average Time to Deposit Comparison

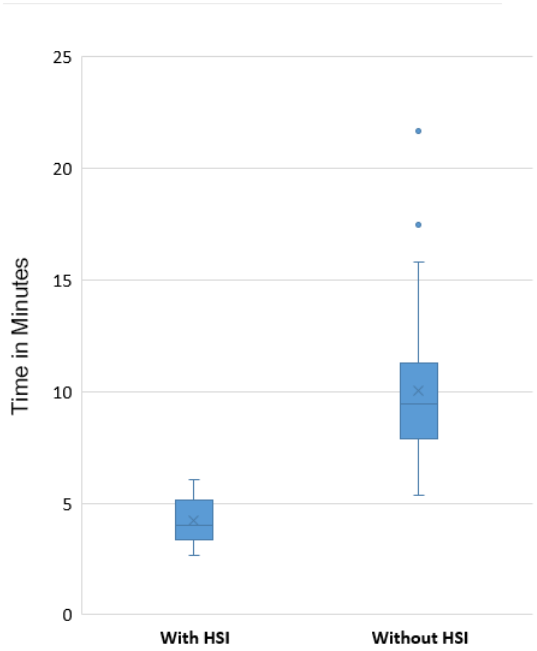


Figure 6.8: Bupimo Average Time to Deposit Comparison

6.4.4 Wandering Time

Table 6.9 shows the mean and standard deviation across all experiments, with and without HSI, of the time spent wandering, in minutes, for all Bupimos. Figure 6.9 shows a box plot of that time across all experiments.

Approach	Mean	Standard Deviation
With HSI	7.397	7.205
Without HSI	48.684	9.564

Table 6.9: Bupimo Wandering Time Comparison

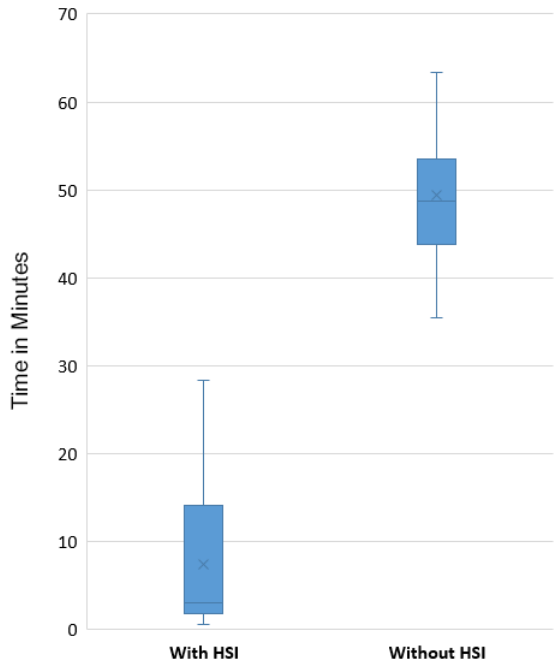


Figure 6.9: Bupimo Wandering Times Comparison

6.4.5 Active Time

Table 6.10 shows the mean and standard deviation across all experiments, with and without HSI, of the time spent actively foraging, in minutes, for all Bupimos. Figure 6.10 shows a box plot of that time across all experiments.

Approach	Mean	Standard Deviation
With HSI	15.71	3.933
Without HSI	2.639	1.77

Table 6.10: Bupimo Active Time Comparison

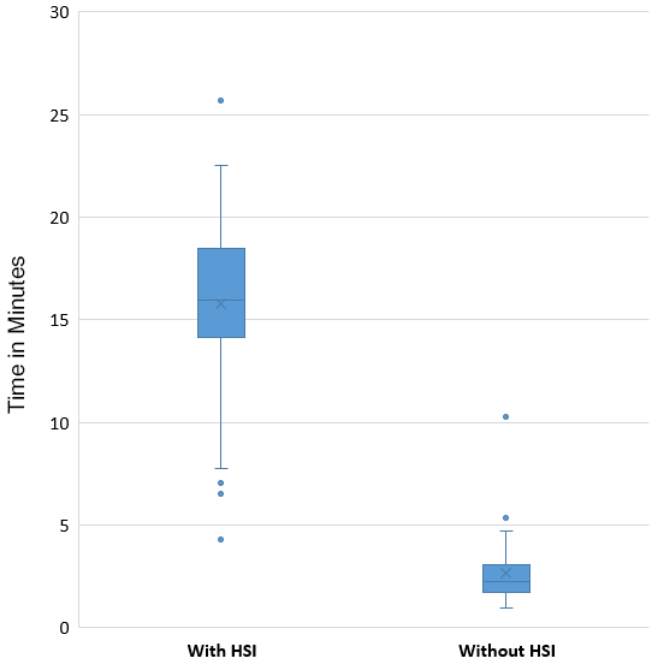


Figure 6.10: Bupimo Active Time Comparison

6.5 Discussion

Both the approach that utilizes HSI and the one that did not accomplished the desired task. The HSI approach's average total time is more than 70% faster than the approach without HSI. The reduced time required for a full run of an experiment meant that three times the number of runs of an experiment could be accomplished per battery charge using the Bupimos with HSI than without it. The HSI experiments, both real and simulated, required the constant presence of an operator to control the beacons. Leaving the beacons unattended would result in the beacon's assigned subswarm's wandering locally around it. Therefore, the HSI approach is much faster, but its efficiency is completely dependent on an operator's presence and full awareness of all the robots and beacons and that operator's ability to maximize their productivity. Ultimately, it is a question of whether the task is time-sensitive

enough to warrant a high level of human involvement or lax enough to allow leaving some of the robots unattended while the rest are being supervised by the operator.

Similarly, the rate of deposits is much quicker and more consistent with HSI than without it. The HSI approach is more suited to tasks that need to be done with a fixed frequency. The distribution of deposits per robot was more predictable and consistent with HSI, however, the differences between both approaches in this regard are not as drastic as was previously anticipated. Still, if the task is to find a solution with optimized load balancing amongst the robots, then the HSI approach would be better suited.

The SimBots and Bupimos spend less time wandering with the HSI approach than without it, however, the results are not as consistent when using HSI as they are without it. This is due to the fact that, in practice, an operator would focus on using one beacon to interact with the subswarm that prioritizes it, leaving the other beacon stationary. Half the swarm follows the beacon that the operator is using, and the other half approaches the idle beacon until it is reached and then the wander behaviour is triggered for agents close enough to the beacon. The resulting wander time is arbitrary, dependent on how the operator decides to divide their time amongst the beacons and how fast they are at switching between subswarms. This HSI solution is thus more efficient than the solution that does not utilize HSI, having achieved a much reduced wander time, however it is not optimal if the goal is to devise a solution that minimizes the time robots spend wandering while foraging with an informed operator. The nature of the wandering behaviour itself also differs between approaches. While using HSI, robots tend to wander within the neighborhood of an idle beacon, whereas without HSI they tend to wander all over the environment. In this experimental setup it was assumed that the operator knows where the source is and could use that information to guide the subswarms, however, if that assumption was to be dropped, then having the robots freely wander the environment autonomously in exploration would be a better alternative to moving around several beacons that serve as anchors and are manually

controlled.

The active time is a more complex metric to analyze. In general, a higher active time is considered better, however, experiments have shown that too much of it could be counter-productive. Crowding over the source or nest while getting tangled up in collision avoidance loops that resolve over an extended period of time adds to the overall active time without contributing to the foraging goal. It is observed that the active time for the HSI approach is much higher than that of the HSI-less approach which is not always desirable, given that the active time adds to the total time of an experiment. Similar to how the wander time can increase when the operator neglects one subswarm in favor of another the active time increases when the operator decides to break up crowds by leading a subswarm away, which also does not contribute to the foraging goal.

6.6 Qualitative Results and Observations

Using the priority-centric HSI approach proposed in this thesis, and with the use of two beacons, the robots were successfully split up into two groups/subswarms in all the HSI experiments. The operator did not need to know which robots prioritized which beacons; introducing two stationary and distinct beacons to the environment allowed the subswarms to separate automatically. The starting configuration of the robots did not change how the experiments were carried out.

Experiments have shown that due to visual occlusion, not all robots that prioritize a beacon can see it when it is within visual range due to the other robots getting in the way as they crowd around the beacon. The robot may need to avoid another robot or wall and steers too far away from the beacon to see it. Therefore, it is possible to pause the ongoing cycle-creation strategy discussed in subsection 5.2.2 should the operator feel the need to have a beacon veer off from the desired trajectory so as to become visible to more robots. Given

the limited number of robots per subswarm in both simulated and real world experiments, if one or more robots cannot see their beacon, the operator makes the beacon circle back and "herd" the robots by moving into their visual fields. This can lead to some wasted time and unintended overlap between the subswarms. The cycle is simple to establish in simulations, given that both beacons are controlled with the keyboard and can be moved easily into place. It was not, however, easy to recreate using the Bupimos and required the operator to be on the move all the time to get each subswarm on the right track, a process that can become exhausting after a while.

Conclusion

Based on the experiments performed and the results obtained it can be concluded that the priority-based HSI approach presented in this thesis improves the performance of beacon-controlled robots that use simple SI in a foraging task. By utilizing an operator's knowledge of the environment, foraging is faster and more efficient than an unguided approach. There are some disadvantages to using the proposed approach that prevent it from being a primary and optimal solution for all HSI problems. This chapter summarizes the research done and presents some suggestions for future work that can be done to further the gains of using this approach and remove some of its drawbacks.

7.1 Summary

Relevant concepts were introduced in the first two chapters as well as similar work that inspired this thesis. The robots used both in simulation, the SimBots, and in real-world experiments, the Bupimos, were introduced and their capabilities and limitations were discussed. The vision systems employed by both robots was described in detail, along with some experimentation that aimed to produce a visual marker that enables the robots to identify objects in their environment. The results of these experiments were presented and three types of markers were selected for use in our experiments. A simple method for obstacle detection was developed and discussed.

The two approaches to be compared in this thesis, foraging with and without HSI, were explained in detail, along with the experimental setup that they would be implemented in and the algorithms that controlled the robots. The priority-based HSI approach was successful in splitting up the swarm into multiple subswarms and enabling an operator to control them by using beacon control. The results of the experiments were analyzed, and according to our evaluation metrics, the HSI approach was much more efficient, though not optimal, compared to the approach that did not utilize HSI.

7.2 Future Work

This work provides the base on which further HSI research can be conducted. More complex SI solutions that are more effective at solving the problem will no doubt be an upgrade to the type of HSI proposed in this thesis. These solutions would typically utilize the concepts of stigmergy and rely on a more direct type of agent cooperation and alterations to the environment. The priority-based approach can be applied to other problems and its effectiveness can be measured. The extent of human control is also a viable topic of research; too much human control has presented some shortcomings and too little is not as effective, hence a threshold of how much human control is too much needs to be established. The ways in which human control can be accomplished in large swarms and its limitations has already been touched upon in various degrees of detail in [43, 49, 20, 47]. In our approach, an operator having to divert their attention and interact with different subswarms had its disadvantages, and with a larger number of subswarms we could find ourselves with the same problem of controlling a multitude of agents, but in a different context; the question becomes how many beacons can a human control, given that beacons are less complex than robots, rather than how many robots. Allowing the beacons to be more autonomous while the operator is busy with other tasks would certainly increase efficiency. Other methods

with which the operator can control the beacons without mental or physical exhaustion would greatly improve the approach presented in this thesis.

Establishing hierarchies of beacons, to enable more nuanced and complex control can also be explored after deciding on the maximum number of active beacons an operator can keep track of. The idea is to have higher-tier beacons controlling lower-level beacons that in turn control the robots of the swarm. Another angle to consider, is to determine how many subswarms are needed by a given application to maximize productivity, i.e. produce an optimal solution. We also need to determine what the relationship between the number of swarm agents and the number of subswarms is, in order to maintain optimality.

The vision system can be further improved. Future work can look into integrating an orientation estimation system for accurate estimation of a perceived robot's orientation and heading, to enhance the performance of collision detection and path planning.

Bibliography

[1] Bio-Inspired Robotics Lab. http://bots.cs.mun.ca/.

[2] Insignia Portable Charger. http://www.insigniaproducts.com/products/computer-speakers-accessories/NS-MB7800.html.

[3] OpenCV. http://opencv.org/, June 2000.

[4] Pixy Camera. http://charmedlabs.com/default/pixy-cmucam5/, 2001.

[5] Zumo. https://www.pololu.com/category/170/zumo-32u4-robot, 2001.

[6] E-Puck. http://www.e-puck.org/, 2004.

[7] V-REP. http://www.coppeliarobotics.com/, 2010.

[8] Bubblescope. http://store.bubblepix.com/, 2012.

[9] Raspberry Pi. https://www.raspberrypi.org/, 2012.

[10] ZBar bar code reader. http://sourceforge.net/projects/zbar/, 2013.

[11] H. Ahmed and J. Glasgow. Swarm intelligence: concepts, models and applications. *School Of Computing, Queens University Technical Report*, 2012.

[12] S. Alers, B. Ranjbar-Sahraei, S. May, K. Tuyls, and G. Weiss. An experimental framework for exploiting vision in swarm robotics. *ADAPTIVE 2013*, page 83, 2013.

[13] S. Alers, K. Tuyls, B. Ranjbar-Sahraei, D. Claes, and G. Weiss. Insect-inspired robot coordination: foraging and coverage. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 761–768, 2014.

[14] J. Alonso-Mora, R. Siegwart, and P. Beardsley. Human - robot swarm interaction for entertainment: From animation display to gesture based control. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 98–98, New York, NY, USA, 2014. ACM.

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

[16] A. Becker, G. Habibi, J. Werfel, M. Rubenstein, and J. McLurkin. Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 520–527, Nov 2013.

[17] W. J. Bell. Searching behavior patterns in insects. *Annual review of entomology*, 35(1):447–467, 1990.

[18] G. Beni and J. Wang. *Swarm Intelligence in Cellular Robotic Systems*, pages 703–712. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[19] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[20] D.S. Brown, S.C. Kerman, and M.A. Goodrich. Human-swarm interactions based on managing attractors. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 90–97, New York, NY, USA, 2014. ACM.

[21] A. Brutschy, L. Garattoni, M. Brambilla, G. Francesca, G. Pini, M. Dorigo, and M. Birattari. The tam: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*, 9(1):1–22, 2015.

[22] A. Campo and M. Dorigo. Efficient multi-foraging in swarm robotics. In *Advances in Artificial Life*, pages 696–705. Springer, 2007.

[23] A. Campo, Á. Gutiérrez, S. Nouyan, C. Pinciroli, V. Longchamp, S. Garnier, and M. Dorigo. Artificial pheromone for path selection by a foraging swarm of robots. *Biological cybernetics*, 103(5):339–352, 2010.

[24] K. Cavallin and P. Svensson. Semi-autonomous, teleoperated search and rescue robot. 2009.

[25] A. Chatty, I. Kallel, P. Gaussier, and A.M. Alimi. Emergent complex behaviors for swarm robotic systems by local rules. In *Robotic Intelligence In Informationally Structured Space (RiiSS), 2011 IEEE Workshop on*, pages 69–76, April 2011.

[26] C. Chen and C. Chu. Low complexity iris recognition based on wavelet probabilistic neural networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1930–1935. IEEE, 2005.

[27] D.W. Corne, A.P. Reynolds, and E. Bonabeau. Swarm intelligence. In Grzegorz Rozenberg, Thomas BÃ€ck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 1599–1622. Springer, 2012.

[28] F. Ducatelle, G.A. Di Caro, and L.M. Gambardella. Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. *Swarm Intelligence Journal*, 4(3):173–198, 2010.

[29] A. Elio and R. Ventocilla. Swarm-based Area Exploration and Coverage based on Pheromones and Bird Flocks. Master's thesis, Uppsala Universitet, Uppsala, Sweden, 2013.

[30] M. Fiala. Artoolkit applied to panoramic vision for robot navigation. In *Proc. of Vision Interface*, pages 119–127, 2003.

[31] M. Fiala. Linear markers for robot navigation with panoramic vision. In *Computer and Robot Vision, 2004. Proceedings. First Canadian Conference on*, pages 145–154, May 2004.

[32] S. Garnier, F. Tache, M. Combe, A. Grimal, and G. Theraulaz. Alice in pheromone land: An experimental setup for the study of ant-like robots. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 37–44, April 2007.

[33] M. Goodrich, S. Kerman, and S. Jun. On leadership and influence in human-swarm interaction, 2012.

[34] D. Goulson. Foraging strategies of insects for gathering nectar and pollen, and implications for plant ecology and evolution. *Perspectives in plant ecology, evolution and systematics*, 2(2):185–209, 1999.

[35] E.H. Gustafson, C.T. Lollini, B.E. Bishop, and C.E. Wick. Swarm technology for search and rescue through multi-sensor multi-viewpoint target identification. In *System Theory, 2005. SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium on*, pages 352–356, March 2005.

[36] H. Hamann and H. Wörn. An analytical and spatial model of foraging in a swarm of robots. In *Swarm Robotics*, pages 43–55. Springer, 2006.

[37] H. Hamann and H. WÃ¶rn. A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2-4):209–239, 2008.

[38] M. P. Hassell and T. R. E. Southwood. Foraging strategies of insects. *Annual Review of Ecology and Systematics*, 9(1):75–98, 1978.

[39] S.T. Hayes and J.A. Adams. Human-swarm interaction: Sources of uncertainty. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 170–171, New York, NY, USA, 2014. ACM.

[40] N. R. Hoff III. *Multi–Robot Foraging for Swarms of Simple Robots*. PhD thesis, Citeseer, 2011.

[41] N. R. Hoff III, A. Sagoff, R. J. Wood, and R. Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 123–130. IEEE, 2010.

[42] L. S. Junior and Nadia Nedjah. Efficient strategy for collective navigation control in swarm robotics. *Procedia Computer Science*, 80:814 – 823, 2016. International Conference on Computational Science 2016, {ICCS} 2016, 6-8 June 2016, San Diego, California, {USA}.

[43] A. Kolling, K. Sycara, S. Nunnally, and M. Lewis. Human swarm interaction: An experimental study of two types of interaction with foraging swarms. *Journal of Human-Robot Interaction*, 2(2):103–128, June 2013.

[44] M. Lewis, M. Goodrich, K. Sycara, and M. Steinberg. Human factors issues for interaction with bio-inspired swarms. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 56(1):61–64, 2012.

[45] Y. Liu, J. Yang, and M. Liu. Recognition of qr code with mobile phones. In *Control and Decision Conference, 2008. CCDC 2008. Chinese*, pages 203–206. IEEE, 2008.

[46] M. J. Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:51–80, 1995.

[47] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 72–75. AAAI, 2006.

[48] F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J. L. Deneuborg, and M. Dorigo. The cooperation of swarm-bots: physical interactions in collective robotics. *IEEE Robotics Automation Magazine*, 12(2):21–28, June 2005.

[49] J. Nagi, A. Giusti, L.M. Gambardella, and G.A. Di Caro. Human-swarm interaction using spatial gestures. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3834–3841, Sept 2014.

[50] E. Olson. Apriltag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3400–3407, May 2011.

[51] A.H. Purnamadjaja and R.A. Russell. Pheromone communication in a robot swarm: necrophoric bee behaviour and its replication. *Robotica*, 23:731–742, 11 2005.

[52] A.H. Purnamadjaja and R.A. Russell. Guiding robots' behaviors using pheromone communication. *Auton. Robots*, 23(2):113–130, August 2007.

[53] M. Quigley, K. Conley, B. 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*, volume 3, page 5. Kobe, Japan, 2009.

[54] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 25–34, New York, NY, USA, 1987. ACM.

[55] M. Rubenstein, A. Cornejo, and R. Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[56] V.K. Tzanov. *Distributed Area Search with a Team of Robots*. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2006.

[57] A. Vardy. BuPiGo. `http://bots.cs.mun.ca/robots/bupigo/`, 2015. [Online; accessed 16-November-2015].

[58] P. Walker, S. Nunnally, M. Lewis, A. Kolling, N. Chakraborty, and K. Sycara. Neglect benevolence in human-swarm interaction with communication latency. In *Swarm, Evolutionary, and Memetic Computing*, pages 662–669. Springer, 2012.

[59] S. Wright, A.A. Arroyo, I.A. Arroyo, J.W. Simpkins, B. Wright, and L.M. Zamstein. Emergent collective behavior in autonomous synergistic swarm robots. 2011.

[60] Y. Zhang, P. Agarwal, V. Bhatnagar, S. Balochian, and J. Yan. Swarm intelligence and its applications. *The Scientific World Journal*, 2013, 2013.