DEVELOPMENT OF A SOCIAL REINFORCEMENT LEARNING BASED AGGREGATION METHOD WITH A MOBILE ROBOT SWARM

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

BY

EMRE GÜR

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

SEPTEMBER 2022

Approval of the thesis:

DEVELOPMENT OF A SOCIAL REINFORCEMENT LEARNING BASED AGGREGATION METHOD WITH A MOBILE ROBOT SWARM

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

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

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

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

Examining Committee Members:

Assoc. Prof. Dr. Ahmet Buğra Koku
Mechanical Engineering, METU

Assoc. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU

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

Assoc. Prof. Dr. Ender Yıldırım
Mechanical Engineering, METU

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

Date:

ABSTRACT

DEVELOPMENT OF A SOCIAL REINFORCEMENT LEARNING BASED AGGREGATION METHOD WITH A MOBILE ROBOT SWARM

Gür, Emre
M.S., Department of Mechanical Engineering
Supervisor: Assoc. Prof. Dr. Ali Emre Turgut

September 2022, 66 pages

In this thesis, the development of a social, reinforcement learning-based aggregation method is covered together with the development of a mobile robot swarm of Kobot-Tracked (Kobot-T) robots.

The proposed method is developed to improve efficiency in low robot density swarm environments especially when the aggregated area is difficult to find. The method is called Social Reinforcement Learning, and Landmark-Based Aggregation (SRLA) and it is based on Q learning. In this method, robots share their Q tables inside the aggregated area. The developed robot swarm uses mostly off-the-shelf, open source, additive manufactured components and is extendable, and easily maintainable by design. Kobot-T is intended to be used together with a three-part heterogeneous swarm system capable of accomplishing tasks from foraging to aggregation for long hours. To prove and report the success of SRLA several aggregation methods from the literature are tested and the result of the comparison is given.

Keywords: Swarm robotics, Multi-robot systems, Swarm communication

ÖZ

SOSYAL TAKVİYELİ ÖĞRENME BAZLI BİR TOPLANMA METODU İLE BİR MOBİL ROBOT SÜRÜSÜNÜN GELİŞTİRİLMESİ

Gür, Emre
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ali Emre Turgut

Eylül 2022 , 66 sayfa

Bu tezde, sosyal, pekiştirmeli öğrenmeye dayalı bir toplanma yönteminin geliştirilmesi, bir Kobot-Tracked (Kobot-T) robot sürüsünün geliştirilmesi ile birlikte ele alınmaktadır. Önerilen yöntem, özellikle toplanılacak alanın bulunmasının zor olduğu düşük robot yoğunluklu sürü ortamlarında verimliliği artırmak için geliştirilmiştir. Yöntem; Sosyal Takviyeli Öğrenme ve Landmark-Based Toplanma (SRLA) olarak adlandırılır ve Q öğrenmeye dayanır. Bu yöntemde robotlar, Q tablolarını toplu alan içinde paylaşır. Geliştirilen robot sürüsünde, çoğunlukla kullanıma hazır, açık kaynak, eklemeli imalat ile üretilmiş bileşenler kullanılmıştır ve genişletilebilir, tasarım gereği kolayca bakımı yapılabilirdir. Kobot-T, uzun saatler boyunca besin toplamadan toplanmaya kadar görevleri yerine getirebilecek şekilde ve üç parçalı heterojen bir sürü sistemi ile birlikte kullanılmak üzere tasarlanmıştır. SRLA'nın başarısını kanıtlamak ve raporlamak için literatürden birkaç toplama yöntemi test edilir ve karşılaştırmanın sonucu verilir.

Anahtar Kelimeler: Sürü robotiği, Çoklu Robot Sistemleri, Sürü içi iletişim

To whom who reads this :)

ACKNOWLEDGMENTS

To begin with I would like to thank Assoc. Prof. Dr. Ali E. Turgut for his support.

As I built my entire system in ROMER laboratory, I would like to thank Assoc. Prof. Dr. Erol Sahin for his allowance to use ROMER and his support for this thesis.

For their support, Arash Sadeghi Amjadi, Cem Bilaloğlu, Mehmet Şahin, Umutcan Üstüntaş, and Atakan Botasun will always have a special place in my heart.

All KOVAN staff had hard times while I made my mess during my studies. I would like to thank their patience.

I would like to thank my family for raising me.

Finally I would like to thank my brother Eren Gür, who will always be my joy in this world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
RnB	Range and Bearing Sensor Card
RPi-0	Raspberry Pi Zero W Single Board Computer
OTS	Off The Shelf
RFID	Radio Frequency Identification
MDP	Markov Decision Process
RL	Reinforcement Learning
RLA	Reinforcement Learning based Aggregation
cRLA	Communicated Reinforcement Learning based Aggregation
UAV	Unmanned Aerial Vehicles
USV	Unmanned Surface Vehicles
UGV	Unmanned Ground Vehicles
UUV	Unmanned Underwater Vehicles
OTS	Off-the-shelf
MCU	Micro Controller Unit
STM	STMicroelectronics
Kobot-T	Kobot-Tracked
LCC	Low-Level Control Card

CHAPTER 1

INTRODUCTION

Swarms of ants can construct complex nests [2], keep farms of fungi[3], and build bridges [4] using their pheromone-based mechanism for self-organization [5]. The creation of these types of structures in systems composed of few or many components is called self-organization behavior. It is a concern for researchers that principles of self-organization exists for nature to solve problems [6] just like ant swarms solving survival problems.

Besides ants, there are many swarms of animals in nature that exhibit self-organizing behavior with different mechanisms. Cockroaches and bees use a different mechanism based on their relatively higher environmental sensing capabilities to forage, aggregate, or flock for their survival [7],[8]. Considering how efficient these swarms can solve their problems in nature with their self-organizing behavior; swarm intelligence discipline has been introduced [9] to tackle large-scale, distributed, self-organized natural and artificial systems.

Further, to understand self-organization, researchers have built robotic artificial systems [10]. In the beginning, these systems have been used to understand mechanisms of nature, which resulted in the conception of the first swarm algorithms. These were the algorithms for robots to perform some of the behavior made by animals.

For example, to understand the mechanism behind aggregation, that is gathering of individuals in an environmental having a cue, the algorithm of BEECLUST has been developed [11]. In time, the realization that many problems in the field of robotics can be solved with the introduction of swarm intelligence to robotics, the study of swarm robotics was born.

The scope of this thesis includes the design of groups of robots that operate without relying on external infrastructure or centralized control. Swarm robotics can be used when a high degree of redundancy with the lack of single points of failure is desired, and when building the necessary infrastructure to control the robots is a technological challenge such as mining, search and rescue, exploration, and surveillance [12]. Recently, swarm robotics has gained popularity and importance due to the high demand for Unmanned Aerial and Ground Vehicle swarms. Therefore, there is an increasing need for robots to be used in swarm robotics to develop and test novel swarm algorithms for both understanding nature and from this inspiration from nature, developing systems for the future of humankind.

Many robot platforms and algorithms have been developed and used in swarm robotics studies. These robots are generally designed to be used for a specific task and are not capable enough to be used for the development of different swarm algorithms with their base design. Some of these robots are covered in the next chapter and some of them have extensions for different applications later designed. There are also not many tracked swarm robots capable of moving in uneven terrain. Thus, it is realized that a robot to be used in swarm robotics studies, that is able to move in uneven terrain and that is able to implement different swarm algorithms is a challenge in the swarm robotics field.

To add to these challenges, a behavior that some animals exhibit is selected to be developed as a swarm algorithm. Bees, inside their hive [13] share information about foraging areas. It is also known that bees use environmental landmarks for navigation [14]. Therefore, in this thesis, we have developed an algorithm with these two features of bees. In addition, we have designed a new mobile robot to conduct swarm robotic studies on uneven terrain called Kobot-Tracked (Kobot-T) robot. The Kobot-T robot is able to run different swarm algorithms such as foraging, flocking, or aggregation.

In this thesis, we applied the developed swarm algorithm, which is a novel social reinforcement learning-based aggregation method, to Kobot-T as well, for both testing the capabilities of Kobot-T and examining the real life application of the developed swarm algorithm. In order to test the capabilities of the Kobot-T robot, we also conducted experiments by running some other swarm algorithms from literature on

Kobot-T as well.

The thesis starts with a brief literature survey covering robots that are previously used in swarm robotics research and relevant aggregation algorithms. Later we present the development of the Kobot-T robot and sub-systems. Then, in Chapter 4, the novel social reinforcement learning-based aggregation algorithm is introduced together with other relevant aggregation methods that we run on Kobot-T. In Chapter 5, metrics and platform setups for simulation-based and real robot experiments are provided. This is followed by results of conducted experiments in Chapter 6 and in Chapter 7 we concluded the thesis.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we discussed a literature survey of robots and several aggregation methods that are used in swarm robotics research. The chapter starts with a chronological survey of robots, which were used previously in swarm robotics studies, that led to the need for Kobot-T. Then we give a comparison table between the Kobot-T robot and the other robots from this literature survey. The chapter ends with the discussion of aggregation algorithms that led to the development of the novel algorithm covered in this thesis.

2.1 Literature Survey of Swarm Applicable Robots

There exist many swarm robots with different approaches to swarm robotics, since their popularity has increased over swarm robotics applications. Here we give some of the previously studied robots in a chronology which leads to the need for the production of the Kobot-T robot.

An early example of these robots is S-bot [15]. S-bot has two treels that allow locomotion. It is extensible to allow parts to be added or subtracted. This extensibility allows swarm intelligence to be used not only in the control layer but also in the physical layer. It has a built-in gripper which leads two or more S-bots to be able to connect to form a chain of robots that allows passage through large gaps. All systems that allow regular operations during experiments run onboard this robot. Infrared proximity sensors, light and humidity sensors, accelerometers, and incremental encoders on each degree of freedom are all included in each S-bot as navigational aids. To locate and connect with other S-bots, each robot also has sensors and communication equipment, thus two robots can communicate locally without any medium, allowing robots to easily form different configurations. They feature an omnidirectional camera, color LEDs, local color detectors, sound emitters, and sound receivers. S-Bot is not fully open source and its mechanical parts are custom designed.

Following the S-bot, a wheeled robot Alice [16] is also not open source and its mechanical parts are custom designed. Its extensibility allows several enhancements to the robot, from different locomotion to control systems. All systems run onboard this robot and local communication is maintained through infrared signal (IR) communication.

ZeeRo [17] is similar to the previous two, it has wheels for handling locomotion. In this robot, systems run onboard. Different from than previous two, sources for this robot are shared as open source. In this robot communication is achieved through a Bluetooth module, however, there exists no study on robot-to-robot communication in the paper. The mechanical parts of this robot are also custom designed.

E-Puck [18], is a mobile robot dedicated for education and has many extensions that are designed for various applications. In its original form, robot-to-robot communication is not implemented. However, with an extension module, local communication is enabled later [19]. Similar to previous studies; parts and components are custom designed, locomotion is achieved through wheels, and systems are controlled onboard. Even though all of its parts and components are custom-designed complex structures, their details are shared as open source.

Kilobot [20] uses vibration motors for locomotion. All communication between robots and programming of Kilobot is handled through IR signaling. It is a custom-designed open-source robot with an onboard controller. Kilobots are simple in terms of both hardware and software. They enable many swarm features. No extensions are available for Kilobots to enable more realistic swarm algorithms such as grippers for foraging; however, foraging can be implemented through virtual food sources or virtual pheromones can be used with IR signaling [21]. They are used in different swarm tasks such as collective motion [22], collective decision-making [23] and spatial organization [24]. Kilobots had a great impact on swarm studies. However, vibration motors restricted robots to a leveled floor. In addition, depending on IR signaling restricted robots in the experiment arena.

Following Kilobot, ChIPR [25] was built driven by the popularity of additive manufacturing, which researchers to share their custom-designed parts as open source, hence anyone can quickly build and test these parts without incompatibility issues from other production techniques. ChIPR has wheels for locomotion.

Duckiebot [26] is mostly designed with off-the-shelf (OTS) components allowing robots to be built by almost everyone. Similar to ChIPR, this robot has wheels for locomotion, it is open-source, it has onboard control, and it is extensible for different studies. It has multi-robot communication capability based on LEDs.

In Crazyflie [27], flying is used for the mode of locomotion. This robot is used as a platform to carry extensions depending on the application. However, its flight capacity is a limiting factor. It is used with a motion capture system to operate while running swarm algorithms since the base robot has no sensor to track other robots. The robot communicates with the main computer by radio signals for several control systems. It has an onboard Inertial Measurement Unit (IMU) sensor for significant radio packet loss. Robot as a whole is a custom-built off-the-shelf (OTS) platform.

Finally, a recent example of such a robot is Mona [28]. In this study, an educational robot that is adaptable, easy to use, and open source was built. The main robot does not operate with Robotic Operating System (ROS) onboard to enable swarm supervision options however, later researchers developed a separate module rather than existing as embedded system for using ROS [29]. Mona consists mostly of OTS and custom-designed parts. It is also extensible with different modules, for example, a range and bearing module are introduced to Mona [30], which has only one emitter in front of the robot and is mainly covered with receivers in all directions.

In this thesis, we designed Kobot-T considering multiple swarm applications. Therefore, we fit a range and bearing system [1] that performs kin and obstacle detection. Its parts and components are easily replaceable to keep up with the technology. We designed the whole body for ease of maintenance. The color of the body is selected to be white so that it is reflective enough for the operation of the range and bearing sensors. We allowed any researcher to add or subtract their application-specific design details by making the design fully open-source. It also has grippers for holding objects and potentially other robots for tasks such as foraging and chain formation. The gripper connection, battery placement, and actuator-sensory board are in the lower part of Kobot-T so that its balance would not be affected by the added extra weight. We selected tracked locomotion to minimize the effect of experiment arena surface conditions such as uneven terrain. We designed a custom board handling low level controls, and used a 3D printed body that allows expansions to be added to the robot.

Table 2.1: Comparison of Robots that allow swarm applications.

Robot	Year	Locomotion	Open Source	Extensible	Mechanical Parts	Onboard	Local Communication
S-bot[15]	2004	Treel	No	Yes	Custom	Yes	Yes
Alice[16]	2005	Wheel	No	Yes	Custom	Yes	Yes
ZeeRo[17]	2006	Wheel	Yes	Yes	Custom	Yes	No
E-Puck[18]	2009	Wheel	Yes	Yes	Custom	Yes	No
Kilobot[20]	2012	Vibrating	Yes	No	Custom	Yes	Yes
ChIRP[25]	2014	Wheel	Yes	Yes	3-D Printed	Yes	No
Duckiebot[26]	2017	Wheel	Yes	Yes	OTS	Yes	Yes
Crazyflie[27]	2017	Flying	Yes	Yes	Custom	No	No
Mona[28]	2019	Wheel	Yes	Yes	OTS	Yes	No
Kobot-T	2022	Tracked	Yes	Yes	3-D Printed	Yes	Under development

In Kobot-T design, the robot mostly consists of OTS electronic components. Communication between the electronic boards is done via I2C. This eases the addition of new boards with newer OTS components by just connecting two wires and simple software changes. The operation time of Kobot-T is about one hour but it can be extended with the addition of an extra battery pack. We compared previously designed robots with Kobot-T in Table 2.1.

2.2 Cue-based Aggregation Algorithms Covered by Previous Swarm Robots

Cue-based aggregation is defined as the gathering of individuals around a cue in the environment. Thermotactic behavior, that is moving toward a thermal stimulus, of young honeybees [11] is an example for such behaviour. Schmickl et al. proposed cue-based aggregation method BEECLUST [11] to simulate this behaviour with robots. In the BEECLUST method, robots move randomly until they encounter another robot. They stop and wait for a predefined time based on cue intensity. When the waiting time is over, they turn in a random direction and move. Certain aspects that affect the performance of BEECLUST have been studied; such as the density of robots [31] and the number of cues. It was implemented in a real-world scenario [32], in which robots were required to clean contaminated environments where contaminated locations were regarded as the cues. These studies showed that as the robot density in an environment decreases, aggregation performance decreases considerably, since the probability of encountering another robot decreases [31].

The ODOCLUST method was proposed [33] to improve the performance of the BEECLUST. In this method, a continuously active odometry-based homing process is used. In this process, the robot records the position of impact with another robot and then starts to seek this position. Here, the cue that is used to aggregate on was another robot. This method effectively outperformed the BEECLUST method, however high aggregation performance is heavily reliant on high-performance odometry sensors with little noise. Also, dependence on a high density of robots still existed. This resulted researchers to seek for other sensing capabilities of animals for further improvements on aggregation with low robot density environments.

Considering the visual and olfactory sensing capabilities, different landmarks such as flowers [34] or scents [35] are used by the animals, such as honeybees [36]. These animals frequently use visual and olfactory landmarks during navigation [37]. The concept of landmarks was also used in robots proving better localization [38], navigation [39], and mapping [40], [41]. In swarm robotics, odometer and landmarks were both used in foraging behavior, using RFID tags as the landmarks [42]. Following this approach, a cue-based aggregation method called landmark-based aggregation (LBA) was developed to improve the performance of aggregation in low robot density settings using landmarks and odometery [43]. Since the LBA method heavily depends on the odometry data, it does not perform well whenever there is high odometry noise.

Later, by using reinforcement learning, susceptibility to noise was aimed to be decreased by a method called the reinforcement learning based aggregation (RLA) [44]. In this method, even using noisy odometer data, the performance of aggregation was high and the swarm was able to adapt to the changes in cue location.

In this thesis, the RLA method was extended using communication inside the cue area, to increase its performance in environments with small cue areas that are relatively difficult to find.

2.3 Contributions of thesis

The contributions of this thesis are:

- The Design of an open-source swarm robotic platform, Kobot-T, using only OTS components and 3D printed parts, that is capable of moving in uneven terrain.
- The Design and manufacturing of an electronic board based on STM32 Bluepill card to handle low-level actuation and sensory needs of the Kobot-T platform.
- An extension of the RLA method using communication, and testing the new method called the Social RLA (SRLA) systematically in low robot density environments.

CHAPTER 3

DEVELOPMENT OF KOBOT-T ROBOT

This chapter presents the system architecture of Kobot-T for both its hardware and ROS architectures. The details of the hardware are given together with the dynamic calculations for the robot where we calculate Kobot-T's closed loop velocity control parameters. Lastly, details of the software sub-systems are given where we present usage types, motor controller software and communication interfaces used in Kobot-T.

3.1 Architecture

Kobot-T uses a common architecture (CoRe) of the Kobot robotic system [1]. The hardware architecture of Kobot-T is depicted in Figure 3.1. At the bottom, there is the locomotion and power layer, and it is responsible for the locomotion of the robot using the velocity commands from the high level controller. It also supplies regulated power to all layers. In the middle, there is the sensing layer containing all the sensors including the novel range and bearing sensor, Raspberry-Pi (RPi) camera, floor sensors and battery sensor are present. At the top there is the high-level control layer that runs the main swarm algorithm on a RPi-0. It takes input from the sensing layer and transmits velocity commands to the locomotion layer via I2C. The Robotic Operating System (ROS) architecture is given in Figure 3.2 and discussed in detail later in this chapter.

3.1.1 Hardware Architecture

3.1.1.1 High-Level Control Layer

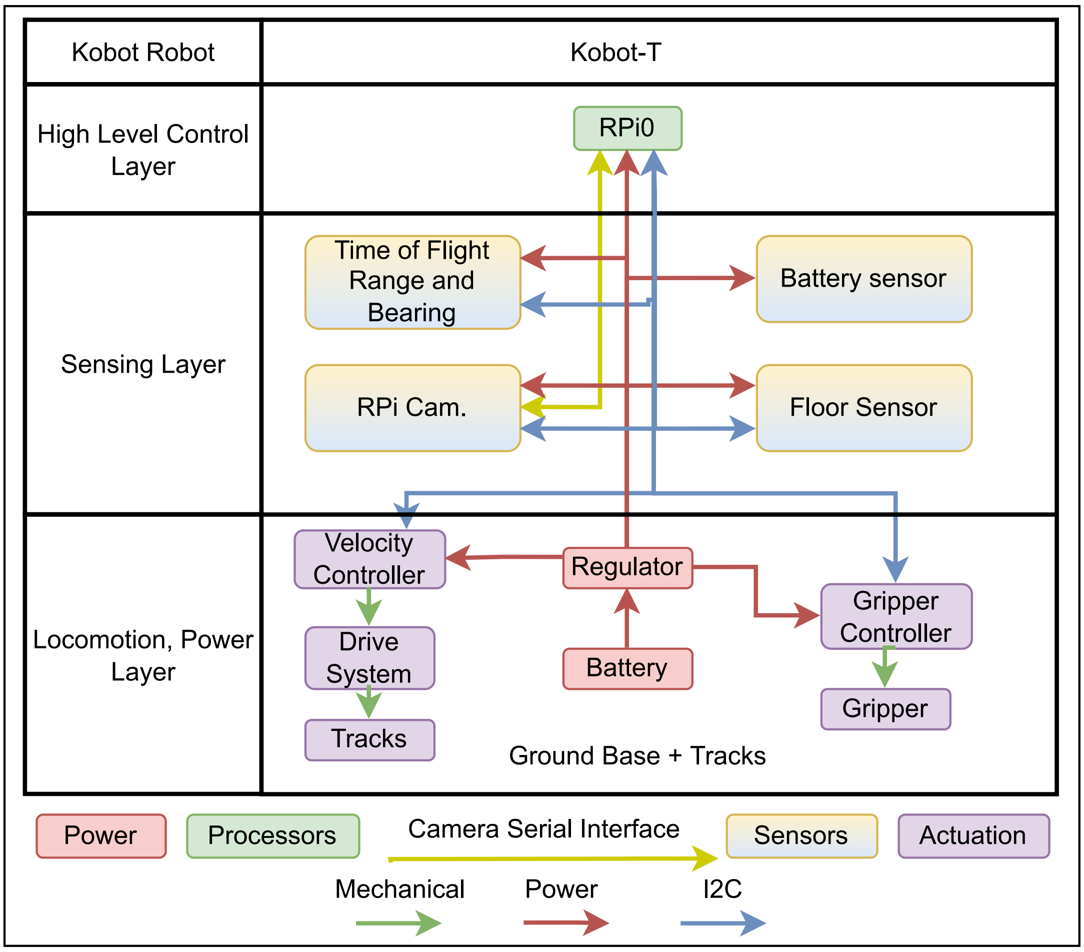

Figure 3.1: Flow diagram showing hardware architecture of Kobot-T, integrated from [1]

We used a RPi-0 single board computer with Wi-Fi connectivity in Kobot-T for High-level controller. It has a small footprint and it has low power consumption. Although the computational power of RPi-0 is not very high, it is sufficient to operate Kobot-T. In addition, some simple computer vision tasks such as detecting ArUco markers can also be executed in this board. Kobot-T is not intended for complex computational tasks such as complex image processing applications, but if needed, Kobot-T can connect to its neighboring robots or a host computer and use their computational resources.

RPi-0 communicates with the other layers via I2C bus due to its high speed and ease of use. Bus-type architecture is preferred since it is straightforward to add new peripherals without introducing extra complexity to the system. Additionally, many third-party breakout boards on the market use I2C bus as the communication interface.

3.1.1.2 Sensing Layer

We placed the sensing layer so that it consists of all the sensors present in Kobot-T robot. In its current configuration, this layer has the RPi camera, range and bearing sensor, floor sensor, and the battery sensor. All the sensors are connected to the RPi-0 via I2C bus and the data is sent upon the request of the controller. This makes it convenient to add different third party sensors such as an IMU or a magnetometer that uses I2C bus. There is also a spare UART port for legacy sensors such as ultrasonic sensors.

3.1.1.3 Locomotion and Power Layer

In this layer, all the actuators for locomotion and gripper exist. There is also the battery and regulator that supply power to the robot. In its current configuration, Kobot-T uses tracks driven by two motors equipped with optical encoders. Their speed is controlled at 50 Hz by the mentioned STM32-based card in a closed-loop fashion. High-level controller calculates and sends reference values for each motor.

3.1.2 ROS architecture of Kobot-T

In Kobot-T, Robotic Operating System (ROS) is used to facilitate the connection between low-level drivers and developed swarm algorithms [45].

In Figure 3.2, one can see that there are several hardware driver nodes, frameworks, and hardware that are driven. Hardware driver nodes accomplish tasks for the usage of hardware that are responsible of. Communication between behaviour algorithm or UI scripts, and different hardware nodes is accomplished through ROS topics. ROS topics are information packs that are carried from publisher to subscriber. Frameworks are structures that are used for specific tasks. For example, OpenCV is an image processing framework, which is used for landmark tracking in this thesis. ROS TF framework is used to track transformations and coordinate frames., The reason for ROS tf messages showed differently in Figure 3.2 is that transformations and coordinate frames are published with special information packs other than ROS topics.

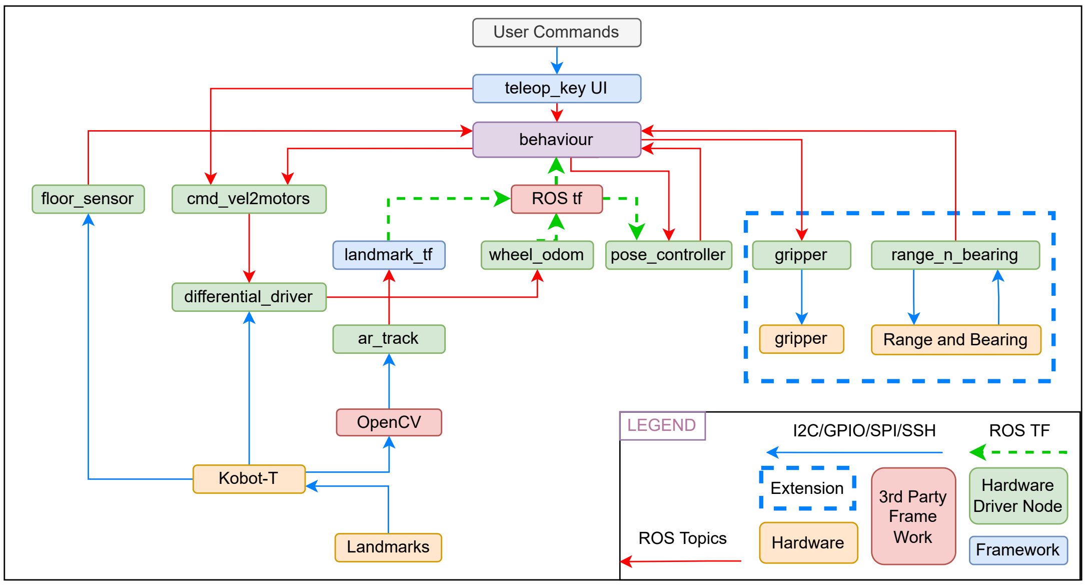

Figure 3.2: Flow diagram showing ROS architecture of Kobot-T, integrated from [1]

Some of the hardware sub-systems, such as RnB, camera, LCC, and gripper, are embedded with their software and their maintenance requires fast response and specific libraries. Nodes that use I2C library has to run onboard to access regulated addresses. For example, the floor sensors' hardware driver node is a low-level driver software and has to run onboard since it regulates I2C communication between floor sensors with Kobot-T. However, other hardware driver nodes such as "cmd_vel2motors" node, require minimal information to operate; since they only regulate speed values coming from different sources to "differential_driver" node. Therefore, it is possible to run them off-board. This procedure is tested but is not needed for algorithms covered in this thesis. Only some behavior scripts run off-board, in the base computer, so that CPU power is kept at low levels. Roscore, the prerequisite collection of nodes and programs for a ROS-based system to run, runs on the base computer for each Kobot-T for now for the same reason.

From ROS nodes for hardware drivers to hardware, low-level reference inputs or sensory outputs are listened by various ways. Details of these are covered in Subsection 3.3

3.1.2.1 Extensions

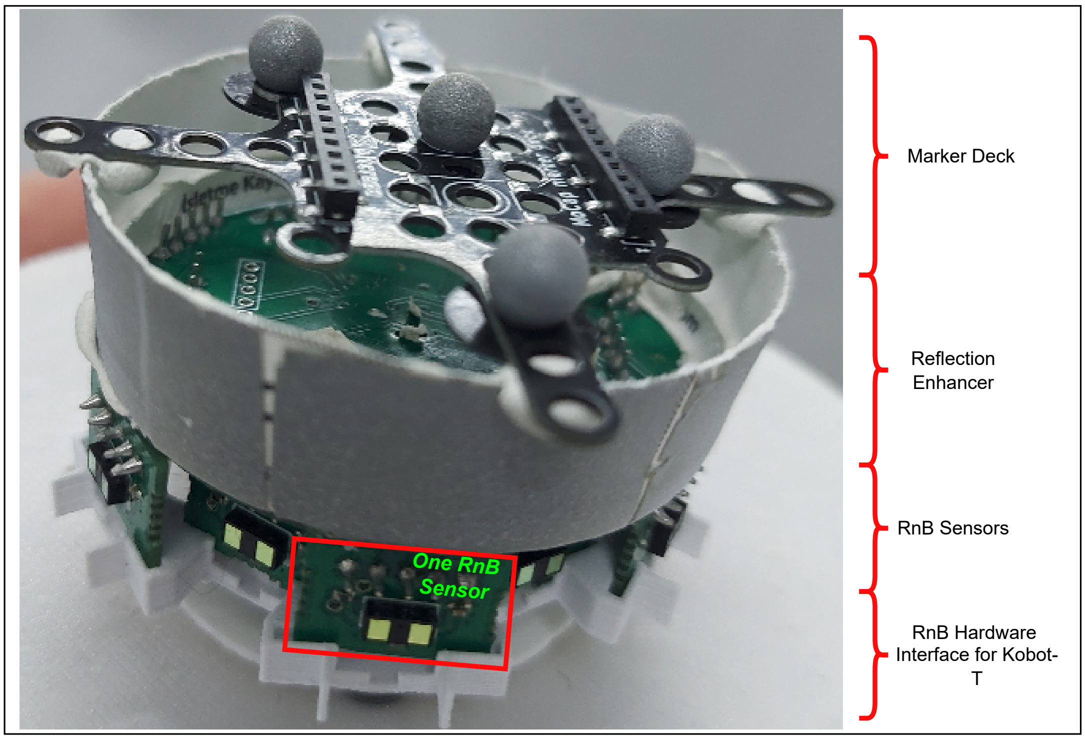

Figure 3.3: Marker deck with designed reflection enhancer on top of RnB Sensor

For different swarm algorithms, may require different sub-systems. For example, RnB sensor extension and its interface are used for experiments conducted in this thesis to supply Kobot-T with robot and kin detection.

Two interfaces for two subsystems for Kobot-T are designed. These are RnB and gripper extensions. In Figure 3.3, RnB extension complex is shown.

3.2 Details of Kobot-T Hardware

3.2.1 Off-the-shelf Systems

Being open-source hardware to let any researcher build the Kobot-T, different OTS products are utilized in the robot such as:

- An RPi Cam for image acquisition particularly for landmark detection.
- A 5V Step-Up Voltage regulator to increase the voltage of the battery to 5V and regulate the voltage at 5V.
- A Tello battery, simple and reliable high capacity battery.
- An RPi-0, high-level controller of the robot used both to implement swarm algorithms and to run simple image processing routines for landmark detection.
- A battery sensor, to keep track of the state of the battery.
- A servo controller, to add additional actuators, grippers sensors, etc, and distribute power to the actuators.

3.2.2 Low-level Control Board

In Kobot-T, a custom low-level control (LLC) board is designed for motor control, motor actuation and ground sensing. This choice is made to save space and decrease the cost of the robot. Since using four different boards each floor sensor, two motor driver boards, and a microcontroller for motor control increase the footprint and price considerably. We made the design of the LLC board open source.

The LLC board has two motor drivers, four analog IR sensors for ground reflectivity measurement, and a Bluepill controller board based on STM32 microcontroller. The LCC board also has a UART connection as shown in Figure 3.1 for connecting legacy sensors. The board is placed at the bottom of the chassis of the robot. A/D conversion of the floor sensors are performed in the LLC board and data is sent to the main controller via I2C. Reference velocity commands are received from the main controller via I2C and actual motor speeds are measured using optical encoders and fed to the LCC board via input capture port; in which, speeds of the motors are controlled on STM32.

3.2.3 Tracked Locomotion

In Kobot-T, we used high torque and low speed gearhead motors with magnetic encoders to drive tracked locomotion. These motors are controlled by LLC board at 50Hz in closed loop and speed measurements from these motors are demanded at 10Hz by High Level Control Layer. The motor parameters are given in Table 3.1. As previously mentioned, High Level Control Layer sends reference values to LLC board. Controller parameters and how LLC closed loop control is achieved is given in another section.

In order to understand position and orientation of the robot, measured encoder count per revolution values are evaluated. Since tracked locomotion is a type of differential drive locomotion, position and orientation of the robot is calculated considering the rotation of the driving wheel. Forward kinematic equation of differential drive, ζ_{robot}, is modeled at Equation 3.1 relates rotation of track-driving wheel to translation of the body [46].

$$\zeta_{robot} = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} (r_{\dot{r}}\dot{\phi}_r + r_l\dot{\phi}_l)/2 \\ 0 \\ (r_{\dot{r}}\dot{\phi}_r + r_l\dot{\phi}_l)/2d_{wb} \end{bmatrix} \tag{3.1}$$

Here $\dot{x}_r, \dot{y}_r$ and $\dot{\theta}_r$, represents forward, lateral and angular velocities that robot can follow. Kobot-T is unable to move laterally without turning so it is equal to zero at all times. The parameters of $\dot{\phi}_r$ and $\dot{\phi}_r$ are angular speed of driving wheel, while calculating the resolution, they are dependent on encoder resolution (δ_{enc}) for one wheel. It should be noted that the left and right wheel radii ($r_l = r_r = r = 18,25$ $[mm]$) are the same and d_{wb} represents the driving axle distance that is $d_{wb} = 99$ $[mm]$

Using Equation 3.1, one can calculate theoretical resolution of position and orientation. The same evaluation is used inside High Level Controller for position and velocity control of Kobot-T. $\dot{x}_r$ can be divided to an infinitesimal parameter of δx_r, together with infinitesimal angle parameter $\delta\theta$ for pure rotation resolution. Using given information one can now write a resolution equation for position control using first line of Equation 3.1.

$$\delta x_r = \frac{r(\delta_{enc} + \delta_{enc})}{2} \tag{3.2}$$

Here, magnetic encoder counts 900 pulses per one wheel revolution ($c_{wheel} = 900$). Since one rotation is 2π, encoder resolution for the wheel can be calculated as in Equation 3.3.

$$\delta_{enc} = \frac{2\pi}{c_{wheel}} = 6,981317.10^{-3}[rad] \tag{3.3}$$

Therefore, the forward resolution becomes $\delta x_r = 1,2740903525.10^{-1}[mm]$ Using the third row of Equation 3.1, pure rotation resolution can be found.

$$\delta\theta_r = \frac{r(\delta_{enc} + \delta_{enc})}{2.d_{wb}} = 1,28695995.10^{-3}[rad] \tag{3.4}$$

Another important parameter for tracked locomotion is the calculation of the maximum linear and rotational speeds of the robot. Since the maximum speed of the motor is $\omega_{max} = 100$ [RPM] $= 10,4719755$ [rad/s], the maximum linear speed can be calculated by using the first line of Equation 3.1. That being said, the calculation of $\dot{x}_{r_{max}} = r.\omega_{max}$ is equal to $0,191114$ [m/s]. The maximum rotational speed for robot to turn around itself can be calculated by using the third line of Equation 3.1.

$$\dot{\theta}_{r_{max}} = r.\omega_{max}/d_{wb} = 0,193044[rad/s] = 11,060606\ [deg/s]. \tag{3.5}$$

These maximum speed values are not possible to achieve in real life due to physical limitations of the robot and due to limitation of the maximum peak voltage supplied to the motors. Kobot-T is never able to supply motors its peak voltage value which is 6V. Instead, it can supply a maximum of 5V. Therefore, the maximum linear and rotational speeds are $0,12$ [m/s] and $6,58$ [rad/s], respectively. From the datasheet, the torque value at this maximum speed can be calculated as follows.

$$\tau = (RPM - 100)/2,6 = 8,2740887[kg.mm] = 0,0827[N.m] \tag{3.6}$$

At the maximum efficiency of the motor, the torque, τ_{max} is $0,065$ [N.m]. Taking the mass of the robot as 360 [gr], the maximum acceleration can be calculated and used to calculate the time to reach the maximum linear speed as follows.

$$f_{eff_{max}} = \tau_{max}/r = 3,33[N] \tag{3.7}$$

$$a_{eff_{max}} = 2.f_{eff_{max}}/m \tag{3.8}$$

$$\dot{x}_{r_{max}}/a_{eff_{max}} = 0,01[sec]; \tag{3.9}$$

It is important to note that many parameters that are calculated in this part are purely theoretical since non-lineer effects whereas slippage in the tracks, backlash in the gearbox, and tolerances between the parts are not considered in the model. However, robot is accurate enough to conduct required rotations and linear movements.

Table 3.1: Motor parameters

Voltage [V]	6
Min. Current [A]	0.07
Max. Current [A]	1.6
Stall torque [Nm]	0.4
Max. Velocity [RPM]	100
Ratio	1:298

3.2.4 Mechanical Design

We designed Kobot-T in three mechanical sections. The bottom section has two tracks driven independently by two DC gearhead motors. The tracked driving part is from Polulu's Zumo robot. At the very bottom of the robot, originally there was the battery case for 4 AA NiMH batteries. In Kobot-T, instead of using these batteries, we decided to use one cell LiPo battery. This freed more space and increased run time of the robot. Therefore, we removed the battery case and fit the LLC board there as shown in Figure 3.5. In the middle section, Kobot-T has its battery and this part also provides a mechanical interface to connect a gripper. We designed this section such that cables are easily accessible without the need to disassemble the top section. There are mainly the OTS parts in the top part including the voltage regulator, camera, RPi-0,

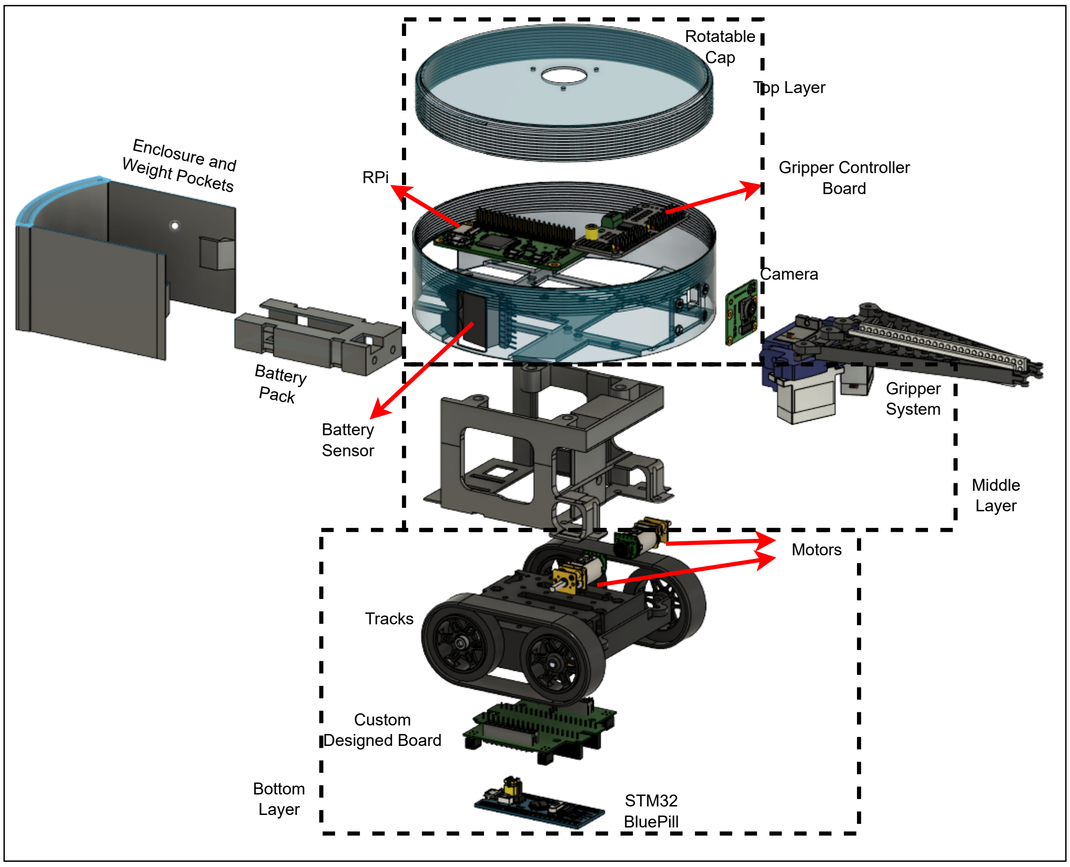

Figure 3.4: Exploded View of Kobot-T

gripper control board, battery sensor, and a rotatable hat to ease calibration of any system fixed to it. In the current setting of Kobot-T, we fixed the RnB sensor to this hat with the help of spacers. This hat also provides a fastening interface for anything to be put on top with the help of spacers and fasteners. The hat has locking teeth side with enough resistance so that it would not shift its position during the experiment. Metric nuts, spacers, and fasteners are selected in Kobot-T to ease maintenance. It is possible to see all three sections in the exploded view shown in Figure 3.4. We connected all three sections with the help of metric bolts and nuts. Anyone can change any section with ease as long as the main connector parts are provided. Also, new sections can be added on top with the help of vertical spacers. For example, the hat can be removed, and instead, a new part can be added using vertical stand-offs. To solve balancing problems, when needed extra weights can be added to the pockets to lower center of gravity. In its original configuration, Kobot-T already has a lower center of gravity with its battery, gripper holder section, and motors that are placed

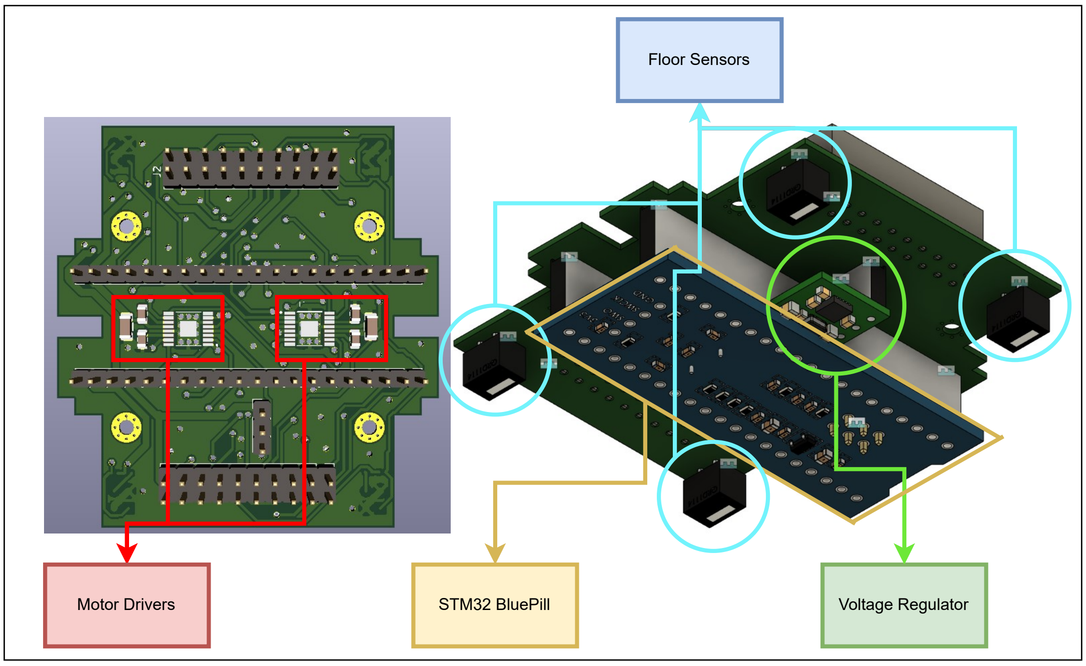

Figure 3.5: Low-level Control Board

close to the ground.

3.3 Software Details of Kobot-T

Kobot-T's software architecture consists of layers. These layers mainly work with their upper and lower peers only with input-output relation and details of low-level parts of the software are given in the sources file. Although having algorithms run on a base computer would allow further improvements in communication and run-time; it is not necessary. Kobot-T does not need a base computer to operate algorithms.

Kobot-Ts can communicate with each other through Wi-Fi. A router is needed for the preliminary application but it is possible to apply the same system with Wi-Fi direct. CoRe [1] is used in Kobot-T, allowing supervisor options and a higher level of control over swarming. In addition, algorithm performance increases significantly while using a base computer to run swarm algorithms. However, the same algorithms can be run over any robot itself, and parameter changes, thus supervisor options can be enabled without a loss. Since communication is also another issue of Kobot-T, if communication is required, a decentralized structure allows it throughout the Wi-Fi network; if all robots are connected to a router. If base computers or any single robot need to connect and communicate with the swarm, it first has to establish an SSH connection with others and publish topics using ROS. Swarm operation, observation, and parameter change can be done in this way. Both a completely decentralized multi-robot flow diagram and a base computer using an old version can be seen in the figures. Using ROS and Linux OS on RPi, ease of access to experimental and parameter data is established to further ease experiments done by the researcher.

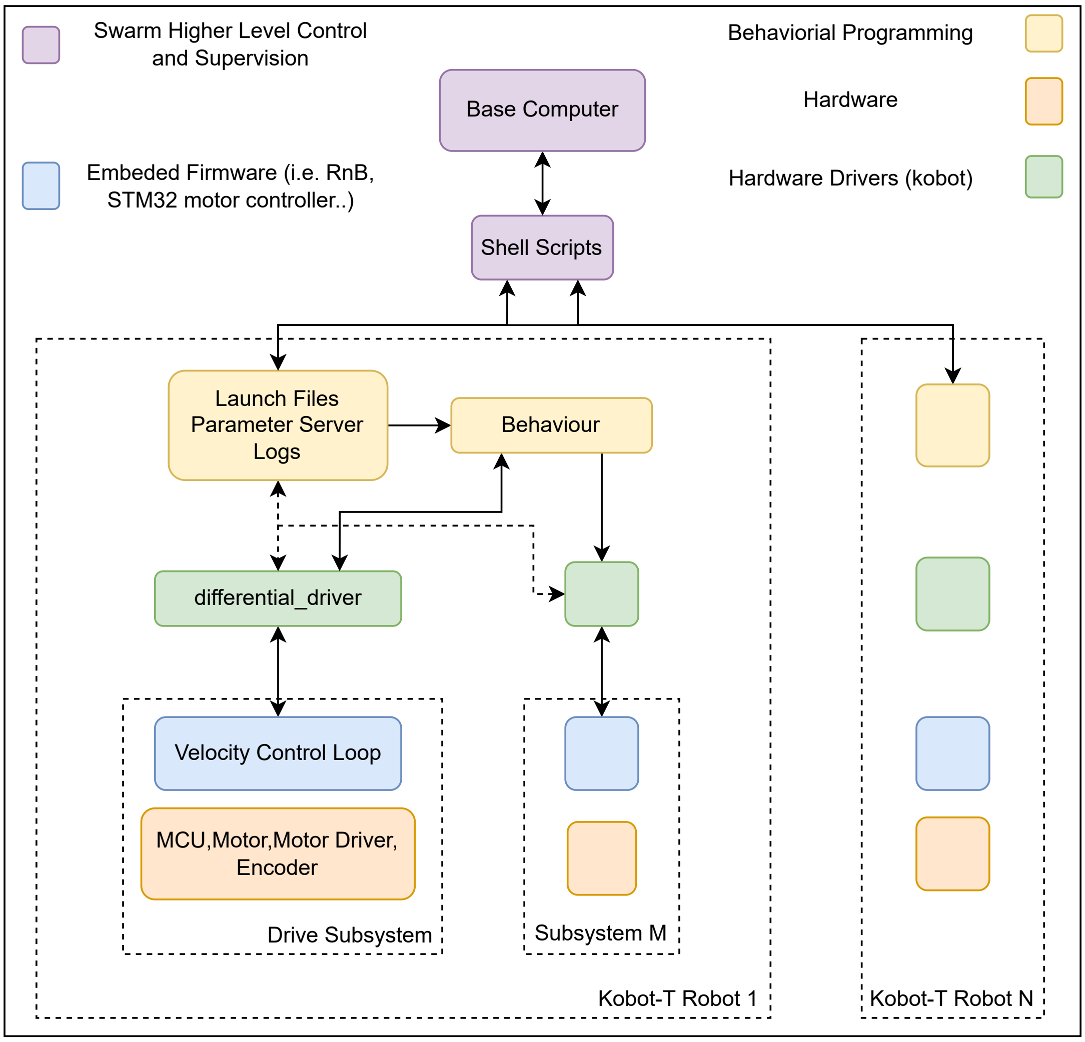

Figure 3.6: Flow diagram showing base computer dependent usage mode of Kobot-T

Since the purpose of using CoRe software is that it can distribute the workload of the algorithm that is tested, away from behavior; a single higher-level Python file is used for each tested behavior, while anything related to low-level control is handled by

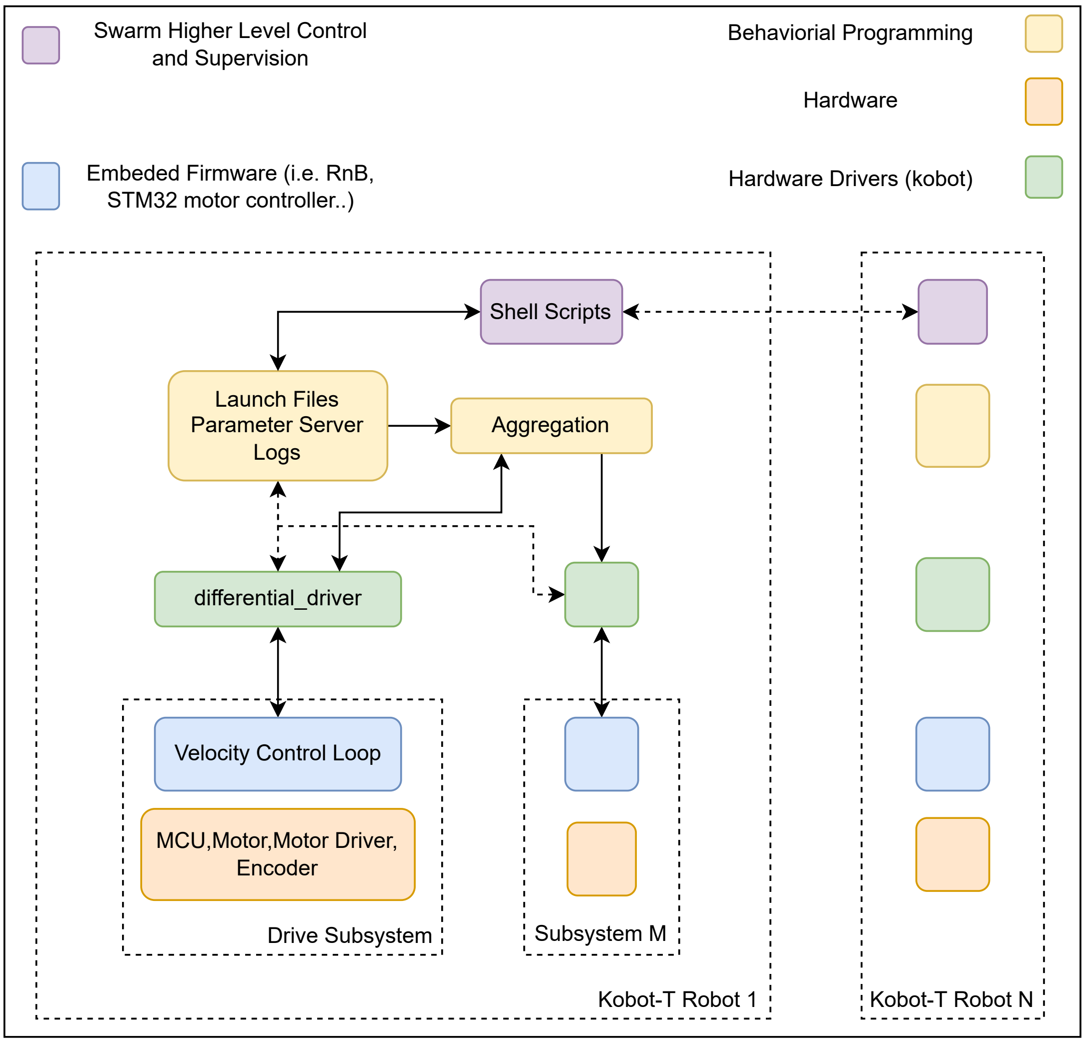

Figure 3.7: Flow diagram showing independent usage mode of Kobot-T

launch files and low-level control nodes embedded inside a robot. Calibrations and parameter changes can be done through ROS parameter change. To test an algorithm researcher can simply change these parameters or change the algorithm completely as wished.

3.3.1 Communication Interfaces of Kobot-T

Different communication schemes are used in Kobot-T as depicted in Figure 3.8. Communication between different layers (RPi-0, gripper controller, RnB sensor, and the LLC board) of Kobot-T is facilitated by I2C bus. It allows a data stream using only two wires. In Kobot CoRe, the default I2C bus speed is 100Kbit/s and it is determined by the I2C master, which is RPi-0 in Kobot-T. Bus messages are parsed into bytes if they are more than 8bit (i.e. 16bit, 32bit like float or long). Whenever I2C read is requested by the master if the slave is not responding fast enough (determined by SCL line baud rate) slave performs clock-stretching. There is also a UART port in the LLC board to connect some legacy sensors.

Communication between different Kobot-T robots is achieved using WiFi via a router allowing robots to send a messages to another robots directly. This is discussed in the SRLA method in the next chapter. WiFi is also used to transfer log files and deploy the latest software in robots remotely. An indirect communication scheme between the robots exists due to range and bearing sensor. When IR signals of two range and bearing sensors collide, each robot indirectly communicates its presence to the other.

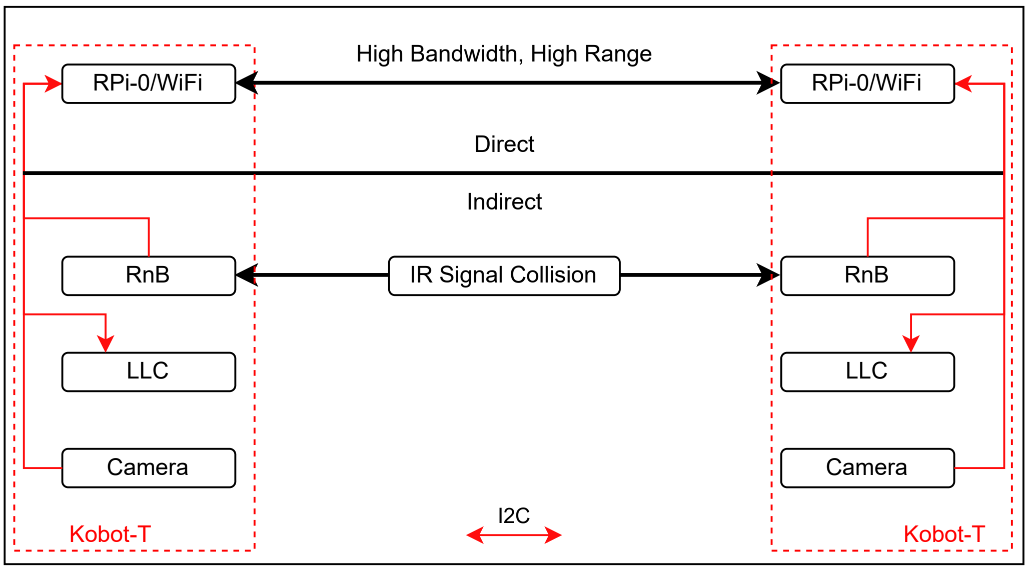

Figure 3.8: Flow diagram showing communication architecture between two Kobot-T robots

3.3.2 Controller Parameters

The tracked driving system consists of two motors with encoders. These motors are controlled by the LLC board at 50 Hz. The feedback is taken from the magnetic encoders and reference input is taken from the high-level controller that are fed to the

PID controller for speed control. PID controller is designed using the Ziegler-Nichols method [47]. The controller parameters are then fine-tuned for the robots.

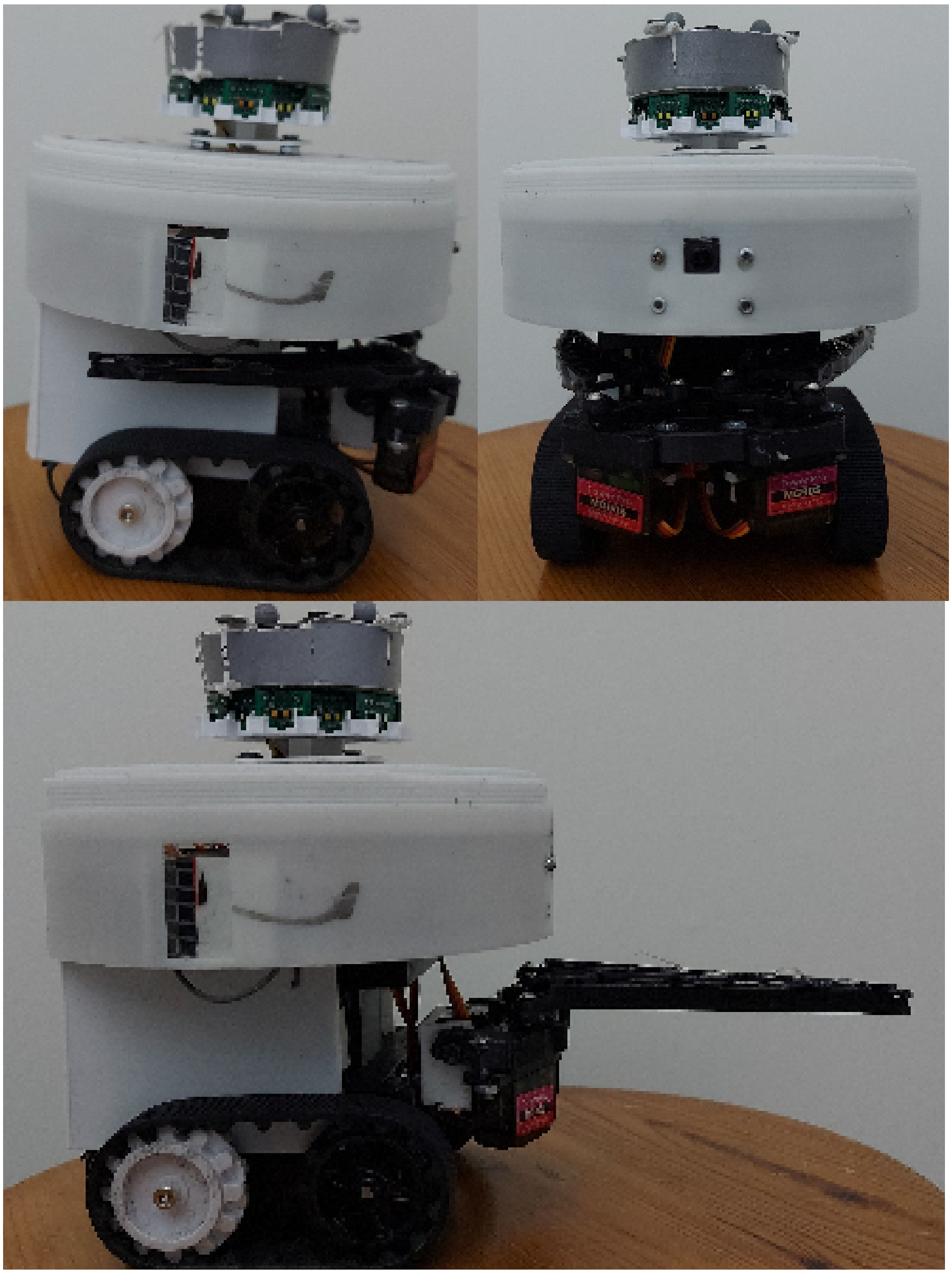

Figure 3.9: Kobot-T physical robot

CHAPTER 4

METHODOLOGY

In this chapter, details of swarm algorithms used in the thesis are discussed. First, several functions that are used during run time of swarm algorithms are explained then the swarm algorithms that are used in this thesis are introduced.

4.1 Avoidance Mechanism

For any swarm algorithms, avoidance of other objects is an important mechanism to accomplish. For the implementation of avoidance behavior, range and bearing information gathered from environment is required. Classification of the object encountered is also important since avoidance is only done from non-robot obstacles. The range and bearing (RnB) sensor used in Kobot-T can distinguish between obstacles and robots by comparing IR sensor signals. Mechanism behind this is not in the scope of this thesis. In this section robot-to-obstacle encounter and robot's reaction mechanism to this encounter is covered. Robot-to-robot encounters require different mechanism that is covered in next section.

In this mechanism, robot avoids obstacles by creating a virtual "force" that pushes it away from obstacles [48]. A desired distance value σ_{do} is set by researcher to the maximum desired range that robot should stay away from obstacle. The force is generated after a certain threshold distance is passed according to given desired distance value. The threshold value is also a parameter to be set by researcher. Calculated virtual force of f_k is applied to robot by nearby objects for each sensor that detects an obstacle. Magnitude of this force can be proportional to the square of difference from the desired distance and is given by Equation 4.1.

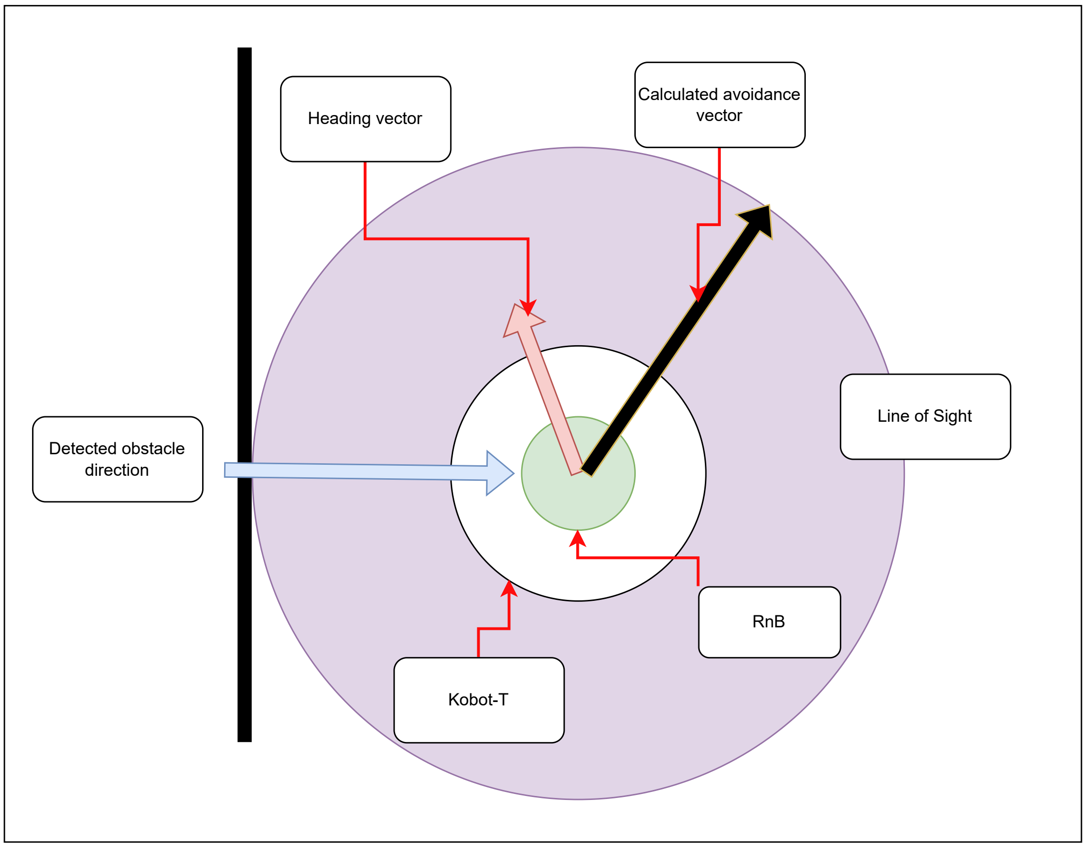

Figure 4.1: Free Body Diagram of obstacle encounter

$$f_k = \frac{-(\sigma - \sigma_{do})^2}{C} \tag{4.1}$$

Where, σ is the distance of object, σ_{do} is desired distance from object and C is the scaling constant for normalization of f_k between 1 and -1.

It is also possible to calculate and use another force magnitude without the square, and use only difference from the desired distance, if used sensor has higher resolution and more linear response. In this case magnitude of this force would be as

$$f_k = \frac{(\sigma - \sigma_{des})}{C} \tag{4.2}$$

Desired distances from obstacles are set 800 [mm], no desired distance values are used for avoiding robots. Since there are 'N' many sensors for gathering distances, and they are placed in a circular path with $\phi_k = 2\pi/N$ angular distances; resultant force of $\vec{p}$ to be calculated as Equation 4.3

$$\vec{p} = 1/N.\sum_{n=1}^{N} f_k.e^{i\phi_k} \tag{4.3}$$

```
Data: Ranges: obstacle range and robot range values
while true do
    /*Resultant force vector*/
    $\vec{p} = get\_vf(ranges)$
    /*Resultant heading vector (this is always [1,0] for now to always go
      forward when no object is visible, since Kobot-T does not follow any
      heading information yet)*/
    $\vec{h} = get\_vh(headings)$
    /*Desired heading vector*/
    $\vec{a}_{des} = \frac{\vec{h}+\beta\vec{p_o}}{|(\vec{h}+\beta\vec{p_o})|}$
    $e_h = atan2(\vec{a_c}) - atan2(\vec{a}_{des})$
    $\omega = K_p.e_h$
    if $abs(e_h < \pi/2$ then
        $u = (\vec{a}_{des}.\vec{a}_c)^{\gamma} u_{max}$
    else
        $u = 0$
    end
    go(u,$\omega$)
end
```

Algorithm 1: Avoidance Algorithm

In Figure 4.1 this mechanism is explained briefly. Here the robot sees an obstacle and avoids it by the created virtual force vector. Blue colored arrow is the resultant virtual force actiong on robot. Red colored arrow is the first velocity vector. Black colored arrow is the resultant velocity vector. Force vector to be generated when obstacle is inside of the line of sight area that is detectable by robot.

Further more, in Algorithm 1, avoidance mechanism is explained in detail. Here, desired heading vector calculation is given. It should be noted that this algorithm can be converted to a flocking algorithm if there exists a heading vector feed from other robots. However, there exists no such communication in Kobot-T yet, thus heading vector is now taken only to correspond forward velocity vector.

4.2 Robot-to-robot Encounter Mechanism

When a robot encounters another robot, it performs a maneuver by randomly selecting an angle value in which robot can escape from. Boundaries of this selection are predetermined. Experiments in this thesis are conducted by determining these values in between $[\pi, 3\pi/4]$ for robot to easily avoid other robot. After robot selects the random angle, the desired rotation is achieved by turning $\phi_o + random(\pi, 3\pi/4)$.

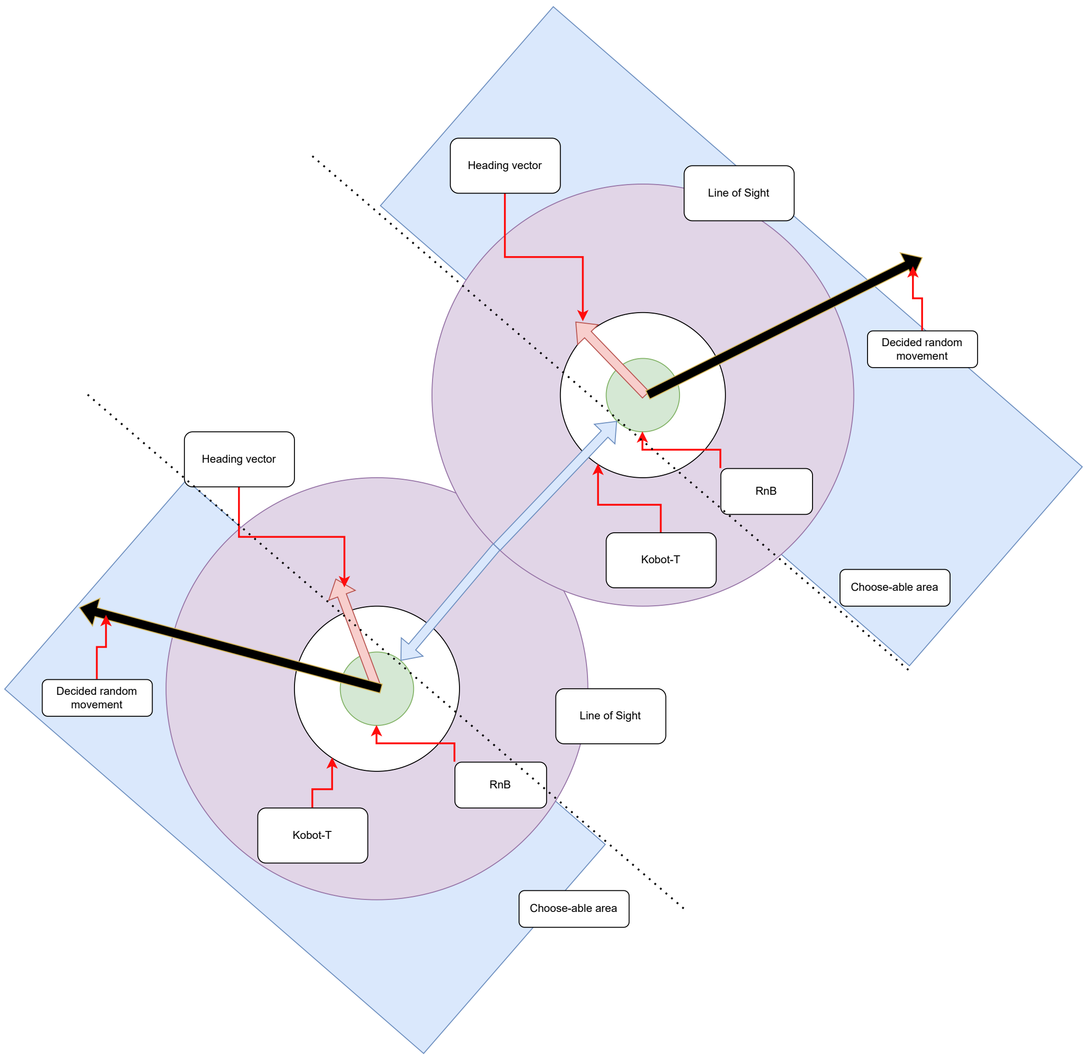

Figure 4.2: Movement after of two robot encounter

Here ϕ_o is the obtacle detection angle. This maneuver starts after cue waiting period, if robots encountered inside the cue area. Waiting period is calculated by checking the illumination of cue area and mapping it between zero to maximum waiting time set by user.

In Figure 4.2, two robots detect each other by blue arrows when one enters other's line of sight (the purple area). Perpendicular blue detection arrow, blue area shows $\phi_o + random(\pi, 3\pi/4)$. The obtacle detection angle is the blue vector's angle with respect to the red shown forward heading vector. This heading vector is considered to be the 0^{th} angle.

4.3 Random action vector selection and reward table calculation

RLA makes random motions and avoids obstacles just like BEECLUST. However,the random action vector selection process differs from the previous only robot-to-robot based random angle generation. In RLA, the randomness for exploration is given also from landmarks. The moment a robot sees a landmark, it collects its ID and position for the landmark and stores it for further use. After facing a landmark, robot randomly selects an action vector from predetermined list of action vectors with a probability function. This process using a probability function allows robot to learn cue area in time.

Markov Decision Process (MDP) is a mathematical framework where outcomes are partly random and partly controlled. State space for this framework can be $S = \{landmark_1, landmark_2, ..., landmark_m\}$ for every landmark in the testing ground where Kobot-T chooses an action vector depending on these landmarks. For finite MDP problems including unknown reward function and unknown transition function, like experiment arena; it is possible to use the Q-learning method [49]. It is an off-policy reinforcement learning method that can be used to derive optimal behavior for an agent [50]. Using this method, a definition of action space for each landmark is first defined including discrete action vectors for translation motion from one landmark to its n number of movable points inside the environment. These are defined as $A = \{\vec{a}_1, \vec{a}_2, ..., \vec{a}_n\}$ and their start point is always center of landmark. Because of this, there has to be an addition of vectors to compensate for the position vector between landmark position and action vector. For instance, an $\vec{R}$ position of landmark center relative to the robot, it compensates for this divergence by following the sum of $\vec{R}+\vec{a}_i$. It is possible to have infinitely many action vectors inside a movable area; however, in this study, in order not to overwhelm the system, discretized and relatively sparsely populated action endpoints are used. These points are referred to with their polar coordinates so that odometer data can be used to move the robot from its landmark sight position to the action end point position.

Since Q learning requires a reward table, its calculation depends on the ground sensor to sense the cue intensity value of I_c resulting the reward to be between 0 to 255. The reward is what is seen by the ground sensors, and its value is updated into the Q table by recursive Equation 4.4. Similar to BEECLUST, robot-robot encounter results in random turning without reward; however, obstacle encounter results in a reward, or punishment, of $r_t = -1$

$$Q^{new}(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha[r_t] \tag{4.4}$$

RL techniques, while in application, have two phases. One is training and the other is testing. The trained RL algorithm can perform better with its updated rewards in the test and sometimes these two phases are separate. However, in RLA these are not separate since many features of the test area are mystery for robot such as cue texture, land conditions, size of the area, starting location, etc.

In the recursive equation of Equation 4.4 it can be seen that Q includes two parameters. Q is dependent on s_t, the state vector of the robot representing the landmarks in close vicinity of it at time t, and a_t, the action vector of the robot representing the action capabilities of the robot for state transitions at time t, in correspondence to the detected landmarks in s_t. While calculating Q table, α is an important parameter for the Q table recursive update rule. It is the parameter to determine the learning rate. This rate is between 0 and 1, and if it is close to one, the robot's learning speed becomes superior but the response becomes very oscillatory similar to the overshooting problem in classical control systems yet the other end keeps the robot back from learning at all [50]. In the original form of Equation 4.4, a parameter of $\gamma \max_a\{Q(s_{t+1}, a)\}$ exists where $Q(s_{t+1}, a)$ refers to future predictions. Since

Kobot-T needs to move freely after completing an action vector; it is not possible to predict any future reward, resulting a non-continuous state transition. Therefore, we set gamma to zero using this equation for calculating Q table.

In detecting a landmark, a robot requires a policy function to choose an action. That being said, the policy function's work is to map state space to an action space like in Equation 4.5. The action can be chosen randomly or by adding stability with the usage of the value function (explore or exploit). The previously mentioned explore or exploit dilemma can be seen here as well while choosing an experience-based action or exploring the environment's possibilities. For example, reward value can get stuck in a local minimum or maximum rather than having a global maximum value by exploring. In RLA this problem is solved by the $\varepsilon - greedy$ method [49]. In this method, a parameter $0 \leq \varepsilon \leq 1$ is defined for the randomness of taken action where 1-ε defines greedy action. Here for $\varepsilon = 1$ actions are randomized always and for the other end, where $\varepsilon = 0$, the actions are always the learned actions.

$$\text{Policy function: } \pi_t(s_t) : s_t \rightarrow a_t \tag{4.5}$$

In other words, for a state (landmark) s_t, the agent chooses the action according to a probabilistic predetermined rate at which its rate determines the exploration and exploitation difference, with the increase in rat, increasing the chance of random vector selection, and decreasing rate, increasing the chance of previously rewarded action selection.

$$a_t = \begin{cases} \text{random selected } a_r \in \text{action space}, & \text{with probability } \varepsilon \\ \max\limits_a Q(s_t, a), & \text{with probability} 1 - \varepsilon \end{cases} \tag{4.6}$$

4.4 Reward table share and comparison

If two robots coincide with each other inside the cue area and start to sleep, they will broadcast their previously gained Q table rewards. It is logical to think that some landmarks and actions taken would not have any rewards as they enter the cue area and, they will be left empty. Comparison for each s_t and a_t will be done by their reward value comparison. If one landmark/action space combination has a higher reward value than another, that will be the one dominating the other and the other reward will be deleted.

4.5 BEECLUST

The BEECLUST algorithm is implemented on Kobot-T. For aggregation algorithm; floor sensor, sense cue area intensity to determine the waiting time; odometer, to understand the orientation of the robot; RnB, to detect range, bearing and robot sensing systems are used with velocity motion control. For the BEECLUST algorithm that is studied, two types of motions are crucial. One is a turn in the case of object encounter, to explore; and the other is sleep depending on cue intensity, to result in aggregation. The algorithm of BEECLUST can be summed up in *Algorithm 2*

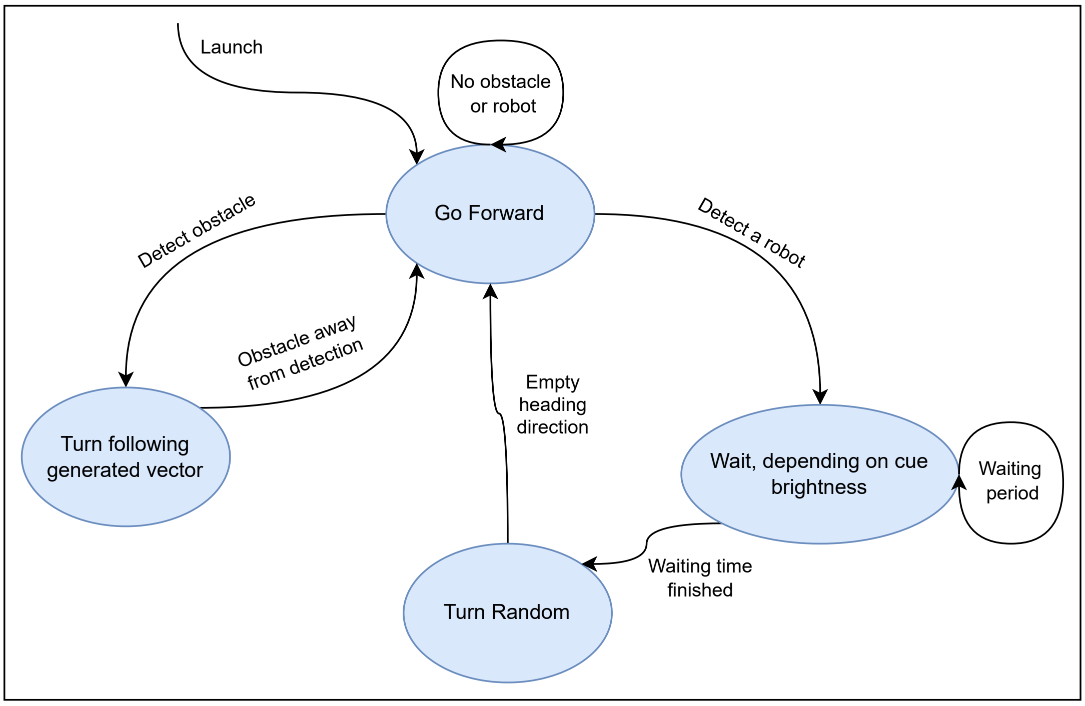

Figure 4.3: Abstract Algorithm of BEECLUST

4.6 RLA

RLA algorithm is a landmark and reinforcement learning based aggregation algorithm from literature, is tested with Kobot-T. The floor sensor and RnB are used similarly to BEECLUST. However, odometry data is used to understand the orientation and action vector follower of the robot; a camera is used to detect landmarks. For the RLA algorithm to be studied, two functions are crucial. One is random action vector selection, to explore, and the other is reward table calculation. The algorithm of RLA can be summed up in *Algorithm 3*

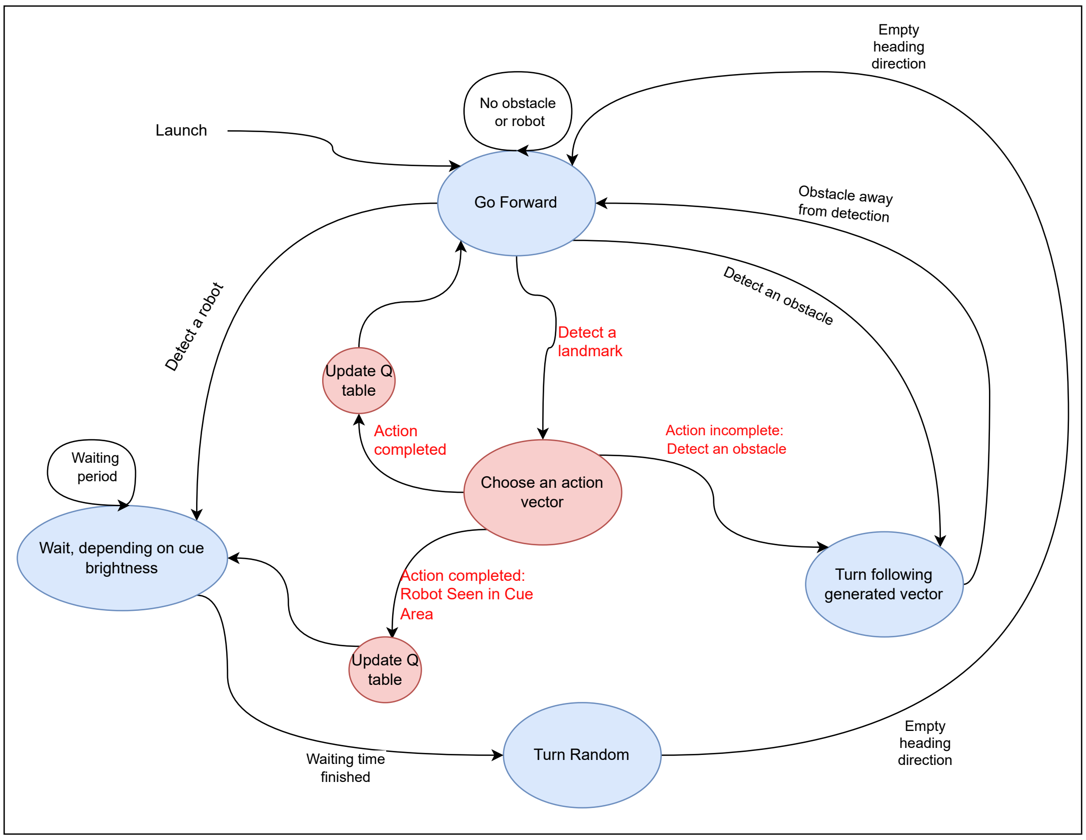

Figure 4.4: Abstract Algorithm of RLA

4.7 SRLA

We proposed novel method as the SRLA algorithm; which is a decentralized landmark and reinforcement learning-based aggregation algorithm where Q tables are

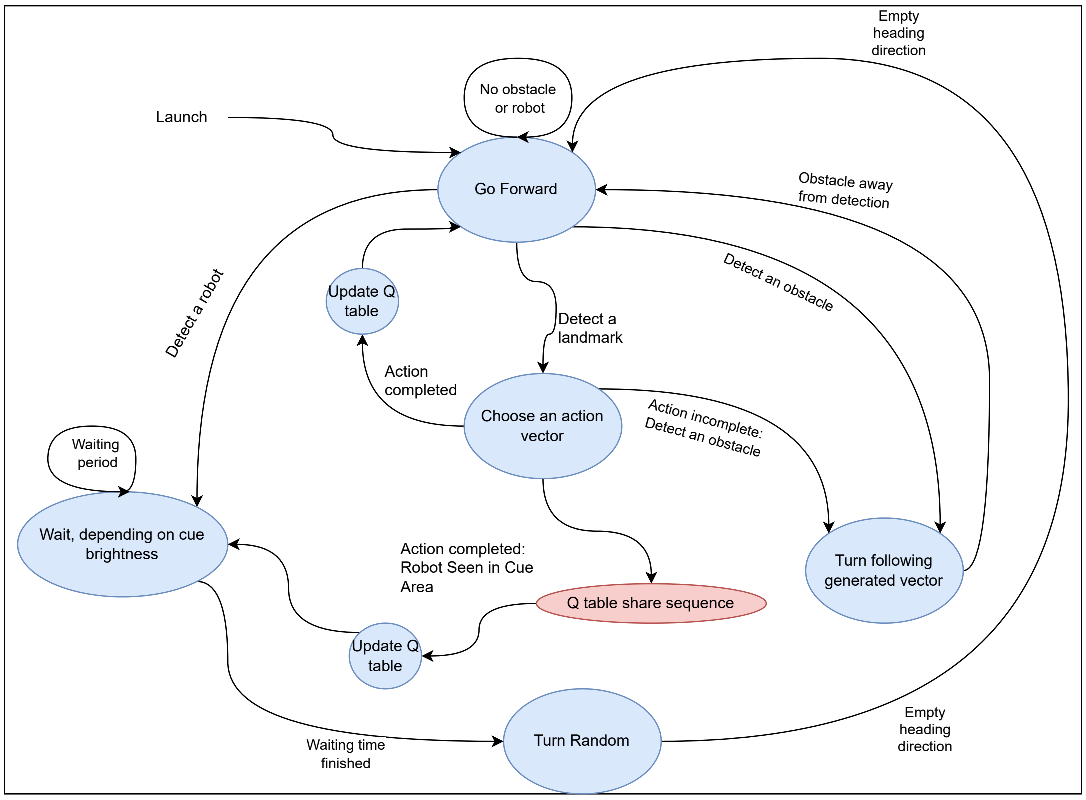

Figure 4.5: Abstract Algorithm of SRLA

shared between robots when they are in the cue area. The floor sensor and RnB are used similarly to BEECLUST. However, the odometer is used to understand the orientation and action vector follower of the robot; the camera is used to detect landmarks. For the SRLA algorithm to be studied, one function is crucial; the sharing of reward table calculation and comparison. Algorithm of SRLA can be summed up in *Algorithm 4*

```
Data: Intensity (I) from ground sensors, RnB and Odometry
while true do
    $forward\_motion(u_{max})$;
    if $object\_detected$ then
        $object\_angle = \phi_o$;
        if $not\_robot$ then
            $generate(\vec{p_o})$ #Force vector generation according to object's
              angular position, details of this generation of vector is covered in
              avoidance mechanism section
            $\vec{h} = [1, 0]$ #Heading generation always forward
            $\vec{a}_{des} = \frac{\vec{h}+\beta\vec{p}_o}{|(\vec{h}+\beta\vec{p_o})|}$ #Desired heading generation;
            $\theta = atan2(\vec{h}) - atan2(\vec{a}_{des})$;
            if $abs(\theta) < \pi/2$ then
                $forward\_motion(u_{max}.(\vec{h}.\vec{a}_{des}))$;
            else
                $forward\_motion(0)$
            end
            $turn(\theta)$;
        end
        if $robot$ then
            if $I > 0$ then
                # Waiting time is calculated for intensity value measured:
                 $t = \frac{I}{I+I_{calibrated}}.t_{max}$;
                $forward\_motion(0)$;
                $wait(t)$;
            end
            $\theta = \phi_o + random(\pi/2, \pi + \pi/2)$;
            $turn(\theta)$
        end
    end
end
```

Algorithm 2: BEECLUST Algorithm

```
Data: Intensity (I) from ground sensors, RnB, ID of Aruco marker, Reward
      table and Odometer
while true do
    $forward\_motion(u_{max})$;
    if $landmark\_detected$ then
        $choose(\vec{a_i})$ # Choose action vector $\vec{a_i}$;
        $go\_through(\vec{a_i})$;
        if $action\_completed$ then
            #Update $Q(s_t, a_t)$ with $r_t = I$;
            $break$;
        else
            if $robot\_detected$ then
                $wait(t)$;
                $turn(\theta_{random})$ # Selected random angle mechanism is given in
                  Random Vector Decision Mechanism and BEECLUST
            else
                $turn(\theta_{reflection})$ # Reflection angle found by calculating force
                  vector previously covered in avoidance mechanism and
                  BEECLUST
                #Update $Q(s_t, a_t)$ with $r_t = -1$
            end
        end
    else
        if $robot\_detected$ then
            $wait(t)$;
            $turn(\theta_{random})$;
        else
            $turn(\theta_{reflection})$;
        end
    end
end
```

Algorithm 3: RLA Algorithm

```
Data: Intensity (I) from ground sensors, RnB, ID of Aruco marker, Reward
      table, Shared reward table and Odometer
while true do
    $forward\_motion(u_{max})$;
    if $landmark\_detected$ then
        $choose(\vec{a_i})$#Choose action vector $\vec{a_i}$;
        $go\_through(\vec{a_i})$;
        if $action\_completed$ then
            #Update $Q(s_t, a_t)$ with $r_t = I$;
            $break$;
        else
            if $robot\_detected$ then
                $wait(t)$;
                $Take\_Q\_shared(Q(s_{pt}, a_{pt}))$;
                $Compare\_Q\_tables(Q(s_{pt}, a_{pt}), Q(s_t, a_t))$;
                $share\_Q(Q(s_t, a_t))$;
                $turn(\theta_{random})$;
            else
                $turn(\theta_{reflection})$;
                #Update $Q(s_t, a_t)$ with $r_t = -1$
            end
        end
    else
        if $robot\_detected$ then
            $wait(t)$;
            $turn(\theta_{random})$;
        else
            $turn(\theta_{reflection})$;
        end
    end
end
```

Algorithm 4: SRLA Algorithm

CHAPTER 5

EXPERIMENTAL ANALYSIS

In this chapter, we examine metric, setup, and simulation platform details of conducted three swarm behavior experiments to prove the designed Kobot-T. While proving the capabilities of Kobot-T we proposed and tested a novel cue-based aggregation method based on reinforcement learning and cue communication, SRLA. We conducted experiments on two platforms. First, we performed kinematic simulations on the algorithms so that they provide what is to be expected from real robot experiments. Also, we tested the novel SRLA method in kinematic simulation experiments with different settings to clarify its distinction from the RLA algorithm. Next, Kobot-T's capability of realizing these algorithms is shown with a real setup. There, real and simulated values are compared. The chapter stars with metric parameters used in simulators to measure performances. This is followed by the details of experimental setup and simulation platform details.

5.1 Metric

Normalized Aggregation Size (NAS) data is a commonly used metric for measuring the success rate of aggregation in swarm behavior experiments. Mean NAS data after reaching a steady state is a common parameter to understand which algorithm is better than the other. NAS values while robots are reaching steady state from transience, and their mean NAS values after reaching steady state are evaluated to compare overall performance of each algorithm. This procedure is followed for different settings. For kinematic simulation experiments, a steady state is declared if the mean value of the moving averaged metric parameter is within the range of 5% for the last 500 time-steps and waited enough to prove steadiness further. Calculation of NAS data is given in Equation 5.1. Both data collected in software simulations and real robot experiments are calculated and plotted.

$$\text{NAS} = \frac{n_{cue}}{n_{total}} \tag{5.1}$$

All experiments are repeated for a given number of Monte-Carlo trials to deny unwanted behaviors caused by initial conditions and noise. Experiments conducted in both kinematic simulation experiments and real robot experiments have the same specifications for Kobot-T. Both experiments have the same arena specification of consisting 6 landmarks, allowing the RLA action cluster to include cue and empty action points, for each arena size. Three of these landmarks are located at equal distances from each other on one side of the experiment arena and the other three are located on the opposite side. Both experiments have a cue modeled as a grey-scale gradient at which its distribution of the intensity along radial direction was considered to be 2D Gaussian, centered at the cue.

Table 5.1: Swarm scenario and Kobot-T subsystem tested by.

Subsystem tested	BEECLUST	RLA	cRLA
Motor Position Control	-	+	+
Motor Velocity Control	+	+	+
Odometry	+	+	+
Floor Sensors	+	+	+
Camera	-	+	+
Communication node	-	-	+
Range and Bearing Sensor Integration	+	+	+
On-Board High Level Control	+	-	-

Since landmarks and action vectors represent states in the Q-learning algorithm, their number is proportional to learning time. As the learning time increases, the time to reach a steady state for an algorithm increases. Thus landmark sizes are selected considering this trade-off.

From different swarm scenarios, different subsystems of Kobot-T are tested. These subsystems and their corresponding tested scenarios are given in Table 5.1.

Finally, backward motion was disabled in all experiments for Kobot-T in order not to confuse the robot during encountering operation.

5.2 Simulation Platform

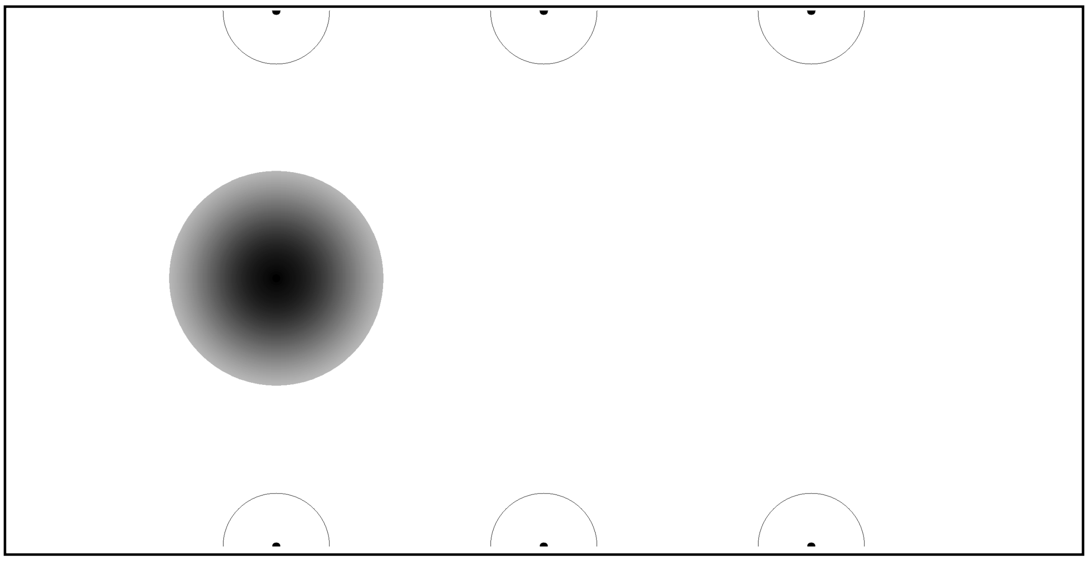
Figure 5.1: Kinematic simulator test arena, cue in the middle, aruco markers in side walls

In kinematic simulations, the physical properties of robots are ignored and robots are modeled only as disk-shaped objects without dynamics. The gradient of cue area was detected by center illumination of the robot's position and robots calculate the waiting time and Q table rewards according to their detection of this gradient. The mechanism of this procedure is given in previous sections. The action vector size is selected as 44 for each landmark, having four lengths and eleven degree values. This number of action vectors is selected because of the previously mentioned trade-off. Since kinematic experiments are conducted on a computer, there is no physical limitation; thus, the action vector size is kept larger than in real experiments. However, since it limits the computational time, a larger value is not selected.

Python programming language is used in kinematic simulator experiments. The number of robots used in kinematic simulation experiments was kept low for each arena size. In these experiments, robots' maximum waiting time inside the cue after encountering another robot is 90 [secs]. Kinematic simulation experiments are conducted with different arena and cue sizes to show performance change corresponding to different settings. In addition, these experiments laid the path for understanding the novel method of SRLA covered in this thesis and its performance difference from other methods.

Each experiment is shown with the 1st and 3rd quartile between each iteration. In every plot, the center line means the 2nd quartile (or median) of all iterations of that experiment. These experiments are conducted for 5.10^5 seconds and the initial positions of robots are selected as random. For 5.10^5 seconds, 5.10^4 data points are taken. To show a smoother result graph, a moving average procedure with a window number of 1000 is appointed. In every plot, dashed lines show the performance of RLA and the same colored continuous lines show the performance of SRLA.

In every time step t of $0.512[sec]$, the location of n^{th} robot was updated as

$$\begin{bmatrix} \mathbf{x}_{t+1}^n \\ \mathbf{y}_{t+1}^n \\ \theta_{t+1}^n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & u_t^n cos\theta_t^n \\ 0 & 1 & 0 & u_t^n sin\theta_t^n \\ 0 & 0 & 1 & \omega_t^n \end{bmatrix} \begin{bmatrix} \mathbf{x}_t^n \\ \mathbf{y}_t^n \\ \theta_t^n \\ \Delta t \end{bmatrix} \tag{5.2}$$

Where general parameters used in the experiments are given in Table 5.2. Note that each length for the action vector is calculated with a percent of the diagonal length of experiment area, and given "t" value represents each time step.

5.3 Experimental Setup

Since the number of robots is few to conduct high and medium-density swarm algorithms, only low-density swarm experiments are conducted. Parameters for experiments are as shown in Table 5.3. Three different algorithms are used in the experiments as mentioned before. To compare these values, kinematic simulator experiments with the same settings are placed in the results section. Each experiment using a different algorithm is repeated 3 times and the results are depicted as 1st and 3rd

Table 5.2: Kinematic Simulation Experiment Parameters

Parameter name	Parameter magnitude
Arena widths	{2.5,3.54,5} [m]
Arena heights	{5,7.07,10}[m]
Diagonal lengths	{5.59, 7.91, 11.18} [m]
Cue radii	{0.35,0.5,0.69,0.7,0.98, 1.09,1.38,1.55,2.19} [m]
Radius of a robot	0.06 [m]
Range for robots and obstacles	0.06[m]
Cue intensity	[0 255]
Maximum waiting time at the cue	90 [sec]
Total length of each experiments	500000 [sec]
Maximum linear speed	0.12 [m/s]
Number of robots	{5,10,20}
Number of landmarks	6
Discount factor of RLA	0
Learning rate of RLA	0.1
Length of action vectors	Diagonal length * {0.25,0.50,0.75,1}[m]
Angle of action vectors	{0,18,36,54,72,90,108, 126,144,162,180}[deg]
Time step	0.512[sec]

quartiles between each iteration. In every plot, the center line means the 2nd quartile (or median) of all iterations of that experiment. These experiments are conducted for 5400 seconds and the initial positions of robots are selected as random. For 5400 seconds, 648000 NAS data points are taken from Vicon. From these, re-sampling is conducted every 10 seconds. Thus, only 540 data points remained. To show a smoother graph, a moving average procedure with a window of 10 data points is appointed. In every plot, dashed lines show the performance of RLA and the same colored continuous lines show the performance of SRLA.

To test algorithms on actual robots, we put landmarks for the RLA and SRLA meth-

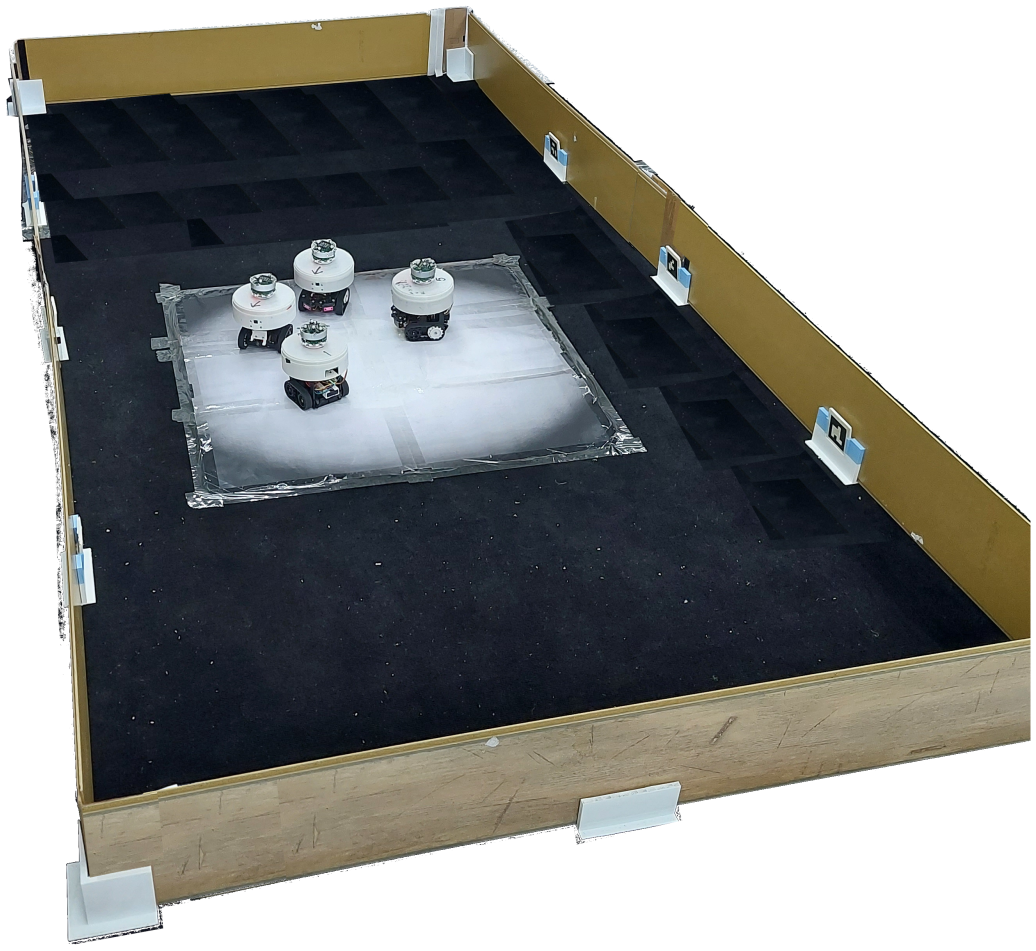

Figure 5.2: Test arena, cue in the middle, aruco markers in side walls

ods, a tracking system for experimental data collection, four walls to fixate test arena size, and a gradually changing cue area was needed. In addition, robots need to distinguish obstacles and neighboring robots. The RnB system in [1] is used for detection of robots and obstacles. It uses the reflected signal for discrete range values. It is capable of high-resolution RnB detection thanks to ToF (time-of-flight) sensors. With this sensor, robots are capable of detecting obstacles and robots while distinguishing robots from obstacles.

AruCo markers are used for environmental landmarks [51] since they are easy to implement and use. For them to be held steadily, holder parts are produced. AruCo markers are held in an inclined position so that their orientation is easily understood by RPi cam.

For the tracker system to be applied successfully, it should be able to collect time and pose data together accurately so that it is used for the resulting data for experimental data collection. Considering this, one of the best tracking systems is used consisting of 12 IR-capable, Vicon Vantage v16 [52] cameras. From Vicon GUI, users can define any specific object and track its orientation change with its time stamps. For Vicon to understand objects and distinguish one object from another; it requires retro deflective ball markers. These markers reflect any oncoming light array directly to its source so that it is distinguished by IR Cameras. Since robots' movements sometimes occlude some of the markers to be seen and this situation interferes with the data collected; a separate ball marker deck is added on top of the Range and Bearing system; together with a retro-reflective sight enhancer for robots to see each other better. The ball marker deck is an OTS product that is originally designed for CrazyFlie drones

For every robot, objects are created from Vicon GUI, and time-stamped location data is collected with 120 Hz. The collection process was simple; the origin of the Vicon arena is set to be in the middle of the cue arena and robots' position data was collected by collecting log data of created ROS node for Vicon. In order not to interfere robot's location information and prove that Vicon does not position control robots, a separate computer was used to collect data. Collected data is then stored as an array and time corrections, and further calculations over data are done throughout this separate computer.

A cue arena for robots to understand was created as a grey-scale gradient as previously mentioned. This gradient was detected by floor sensors and robots calculated the waiting time and Q table rewards according to their detection of this gradient. Before experiments, floor sensors are calibrated to reject any noise and to sense inside the desired cue radius. To understand aggregation, the collected pose data that coincides with the desired cue radius was selected from the overall collection of the pose data and counted (n_{cue}). Since Vicon sees every robot inside the experiment area, the number of robots (n_{robot}) used in the experiment can be counted by it. Before the experiment, since it is arranged that cue is the origin of the Vicon map, and robots entering and exiting the cue are automatically recorded with this selection. This recorded data is then calculated as normalized aggregation size data (NAS). Parameter sets used during experiments are given in Table 5.3

Table 5.3: Experiment parameters

Parameter name	**Parameter magnitude**
Maximum linear speed	0.12 [m/s]
Maximum angular speed	0.8[rad/s]
Number of landmarks	6
Area lengths	1400 x 2800 [mm]
Number of robots	4
Radius of a robot	0.06 [m]
Experiment time	5400 [sec]
Cue Radius	0.3 [m]
Cue intensity	[0 255]
Maximum waiting time at the cue	90 [sec]
Number of landmarks	6
Discount factor of RLA	0
Learning rate of RLA	0.1
Length of action vectors	1,1.4[m]
Angle of action vectors	36,90,144[deg]
Discount factor of RLA	0
Learning rate of RLA	0.1
Data collection speed (from Vicon)	120[Hz]

CHAPTER 6

RESULTS AND DISCUSSION

In this chapter, experiment results of both kinematic simulator and real robot experiments are reported. The chapter describes kinematic simulator experiments followed by robot experiments. Lastly, all the experimental results are discussed in detail.

6.1 Kinematic Simulator Results

To understand the differences in performance between RLA and SRLA, different cue radii and cue to total area rates settings were applied.

At first, SRLA and RLA are compared with the same cue radius size and different total areas. Here the effect of different arena sizes is used to see the change in performance of SRLA compared to RLA. In this experiment; five robots in the arena of 2.5x5 [m] stabilized around 0.61 NAS value using SRLA and 0.57 for RLA, ten robots in the arena of 3.54x7.07 [m] stabilized around 0.47 NAS value using SRLA and 0.36 for RLA,twenty robots in the arena of 5.0x10.0 [m] stabilized around 0.20 NAS value using SRLA and 0.11 for RLA. Results of the first experiment with the same cue radius of 0.7 [m] are given in Experiment 6.1.

In the second experiment, SRLA and RLA are compared at the same cue to total arena rates with different cues and total arenas. Here the effect of different arena sizes is used to see the change in performance of SRLA compared to RLA; while keeping the cue area proportional to the total area. In this experiment; five robots in the arena of 2.5x5 [m] stabilized around 0.46 NAS value using SRLA and 0.28 for RLA, ten robots in the arena of 3.54x7.07 [m] stabilized around 0.29 NAS value using SRLA and 0.17 for RLA,twenty robots in the arena of 5.0x10.0 [m] stabilized around 0.20 NAS value using SRLA and 0.11 for RLA. Results of the second experiment with the same cue to the total area rate of 0.03 [m] are given in Experiment 6.2. To ease performance comparison, in both experiments, green showed 20 robots, and the 5x10 [m] total area used part is repeated.

In the third and fourth experiments, SRLA and RLA are compared at the same cue to total arena rates with different cues and total arenas but with larger rates than the second experiment. Here the effect of the increase in this rate is used to see the change in performance of SRLA compared to RLA; while keeping the cue area proportional to the total area for each experiment.

For the third experiment, cue to a total area rate of 0.12 is used. Here, five robots in the arena of 2.5x5 [m] stabilized around 0.64 NAS value using SRLA and 0.56 for RLA, ten robots in the arena of 3.54x7.07 [m] stabilized around 0.57 NAS value using SRLA and 0.45 for RLA,twenty robots in the arena of 5.0x10.0 [m] stabilized around 0.42 NAS value using SRLA and 0.36 for RLA. Results of the third experiment with the same cue to total area rate of 0.12 [m] are given in Experiment 6.3.

For the fourth experiment, cue to a total area rate of 0.3 is used. Here, five robots in the arena of 2.5x5 [m] stabilized around 0.67 NAS value using SRLA and 0.64 for RLA, ten robots in the arena of 3.54x7.07 [m] stabilized around 0.64 NAS value using SRLA and 0.58 for RLA,twenty robots in the arena of 5.0x10.0 [m] stabilized around 0.54 NAS value using SRLA and 0.50 for RLA. Results of the third experiment with the same cue to total area rate of 0.3 [m] are given in Experiment 6.4.

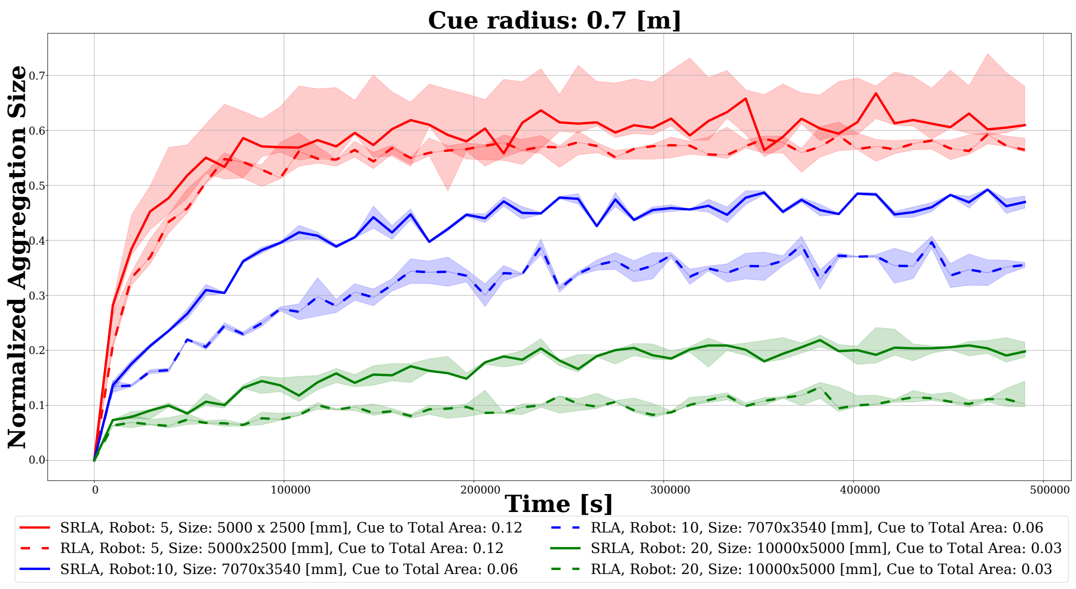

Figure 6.1: Comparison of RLA and SRLA in same cue radius of 0.7 [m], varying experiment area sizes. Dashed line shows results from RLA experiments having 5 iterations and single line shows results from SRLA experiments having 5 iterations. Experiment arena sizes for red lines are 2500x5000 [mm], for blue lines are 3540x7070 [mm], for green lines are 5000x10000 [mm].

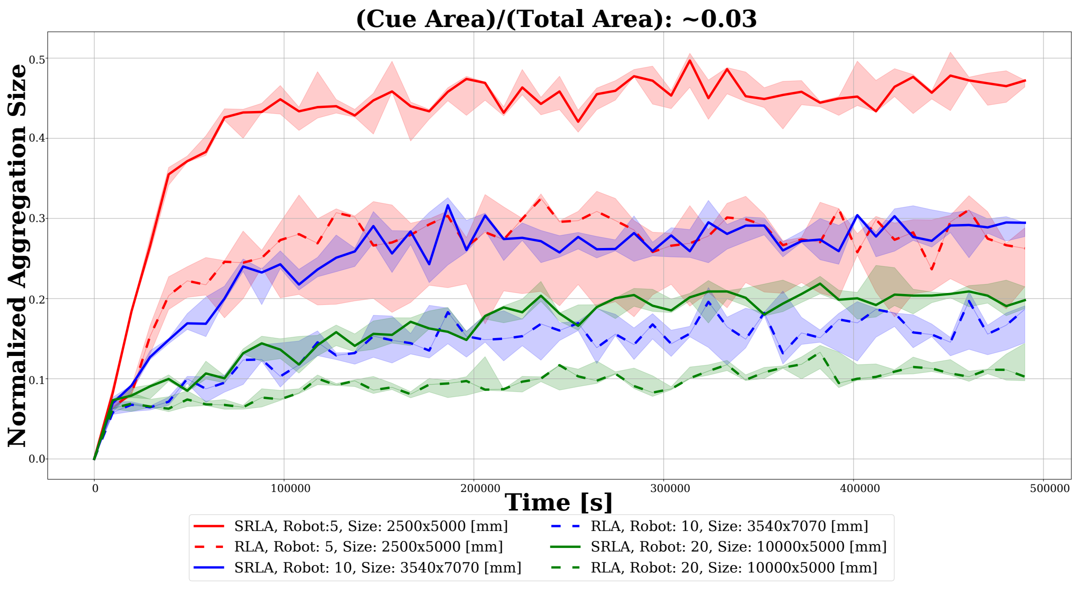

Figure 6.2: Comparison of RLA and SRLA in same cue to total area rate of 0.03, varying cue radii. Arena sizes changes same as Figure 6.1 together with cue area radius. For red lines it is 0.35 [m], for blue lines it is 0.50 [m], and for green lines it is 0.7 [m].

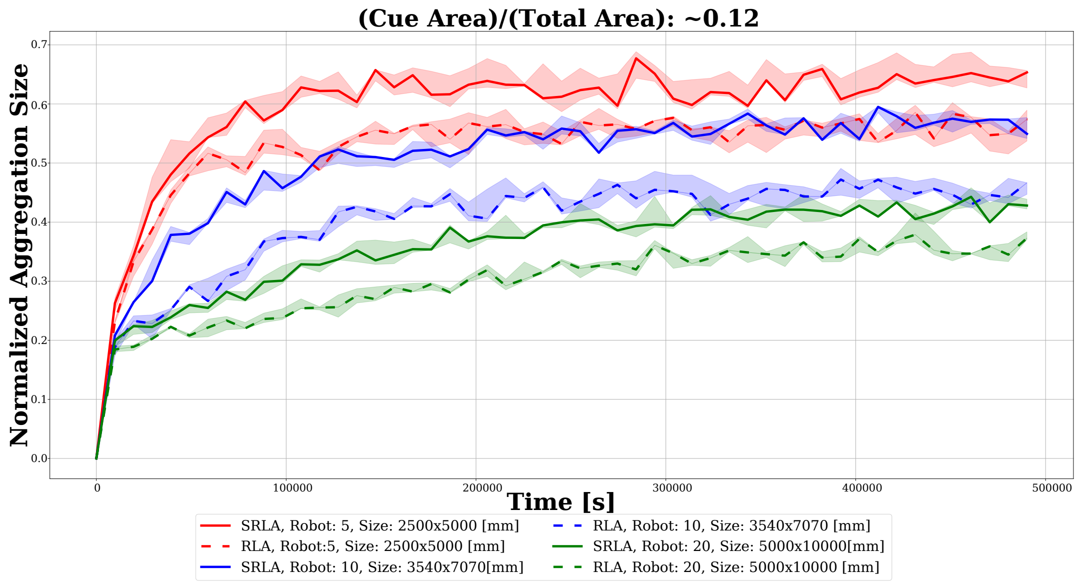

Figure 6.3: Comparison of RLA and SRLA in same cue to total area rate of 0.12, varying cue radii. Arena sizes changes same as Figure 6.1 together with cue area radius. For red lines it is 0.69 [m], for blue lines it is 0.98[m], and for green lines it is 1.38 [m].

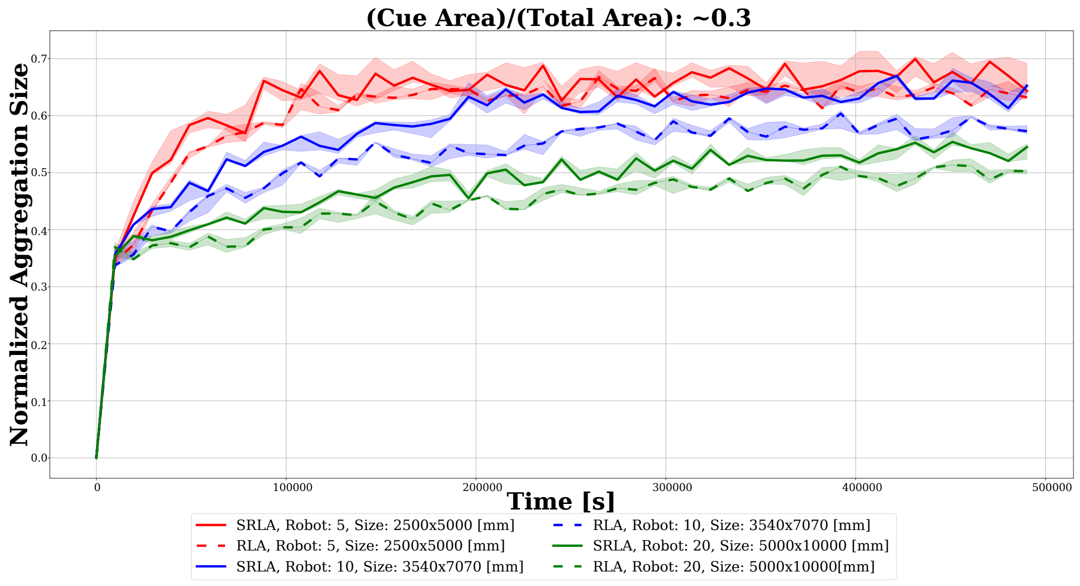

Figure 6.4: Comparison of RLA and SRLA in same cue to total area rate of 0.3, varying cue radii. Arena sizes changes same as Figure 6.1 together with cue area radius. For red lines it is 1.09 [m], for blue lines it is 1.55 [m], and for green lines it is 2.19 [m].

6.2 Real Robot Results

To understand if Kobot-T accomplishes each algorithm, kinematic simulator experiments with the same settings as real robot experiments are run.

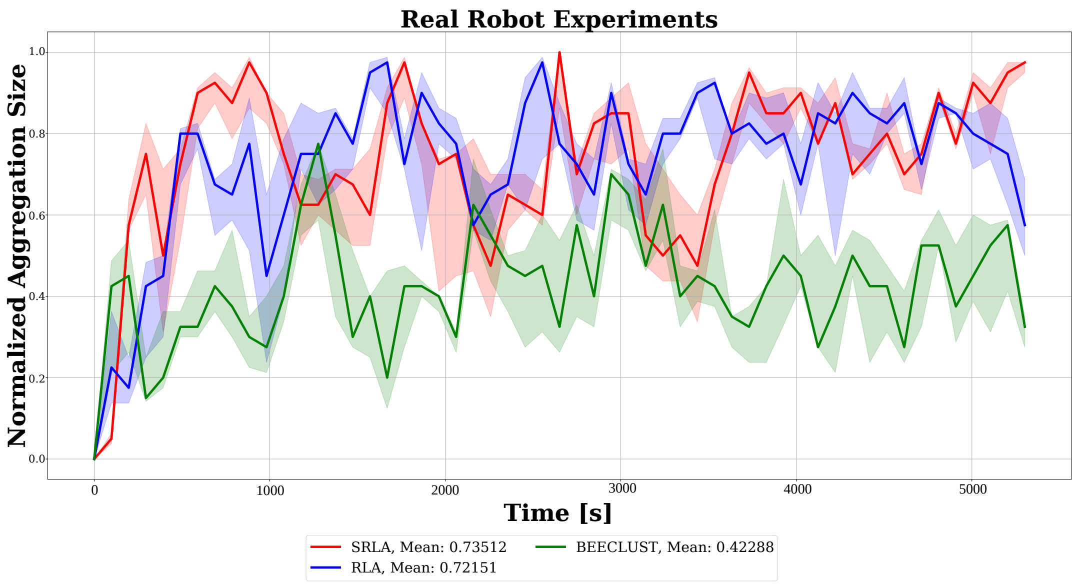

Figure 6.5: Comparison of RLA, SRLA and BEECLUST in real robot experiment. 1.4x2.8[m] arena with the cue radius of 0.3 [m] is used. Each algorithm used for three iterations. 1st and 3rd quartiles from the data gathered from three iterations are shown as faded areas and 2nd on (mean) is shown as center line. Red line is the result from RLA algorithm. Green line is the result from SRLA algorithm. Blue line is the result from BEECLUST algorithm.

In Figure 6.6 BEECLUST, SRLA and RLA results are compared. These values are from the kinematic simulator. Here the same cue to total arena size with real experiment area and same cue radius is used; that is 1.4x2.8 [m] total area with 0.3 [m] cue radius. In this experiment; four robots in the arena stabilized around 0.71 NAS value using SRLA, 0.68 for RLA, and 0.23 for BEECLUST. The same declaration of stabilization method is used for kinematic simulator experiments in this experiment as well.

In Figure 6.5, BEECLUST, SRLA and RLA results are compared. These values are from real robot experiments. Here 1.4x2.8 [m] total area size with 0.3 [m] cue radius is used as experiment arena. In this experiment; four robots in the arena stabilized around 0.74 NAS value using SRLA, 0.72 for RLA, and 0.42 for BEECLUST.

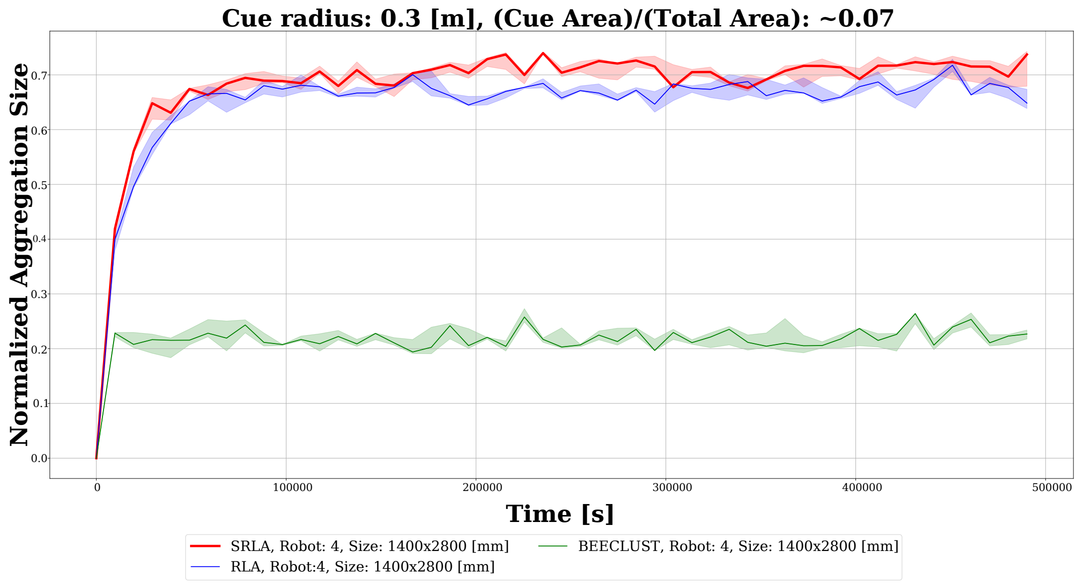

Figure 6.6: Comparison of RLA, SRLA and BEECLUST in kinematic simulator. Same sizes are used. Each algorithm tested used for five iterations. 1st and 3rd quartiles from the data gathered from three iterations are shown as faded areas and 2nd on (mean) is shown as center line. Red continuous line is the result from SRLA algorithm. Red dashed line is the result from RLA algorithm. Blue line is the result from BEECLUST algorithm.

6.3 Discussion

SRLA is advantageous in low-density environments similar to RLA. Between RLA and SRLA, as the cue size and cue-to-total area rate decrease, the distinction between RLA and SRLA gets clear. As the possibility of robot-to-robot encounters decreases, even in the same robot density; SRLA shows better performance.

To understand better, box plots of previously given experimental results for different cues and cue-to-total area size rates are given. These box plots only include raw data results after observing stabilization in previous plots. Since previous plots only included moving averaged data, and even if plots are stabilized in these; raw data comparison can give a reliable comparison. Thus after reaching stabilization and waiting sufficiently enough, raw data is taken from previously used line plots and box plots are created from them. The last 50000 seconds from 500000 seconds of experiments are taken for this action. In these plots, the orange line shows stabilized median NAS data. Boxes show upper and lower quartiles of the data gathered and lines show minimum and maximum whiskers.

In Figures 6.7 and 6.8 one can observe that after stabilizing interquartile ranges of SRLA always remain superior than RLA values.

However, as cue-to-total-area rate increases and the possibility of robot-to-robot encounters increases; these ranges start to converge into similar values. This is clearly observed in Figures 6.9 and 6.10. There are several possible reasons for this to occur. First, since the cue size increases more action vectors end up inside the cue arena; resulting in the privilege gained from communication fading. However, this only explains the result of cue size change. Second, considering the total area size decreases with the same cue-to-total-area but waiting time over the cue arena does not change, the number of robots that can encounter while one is waiting inside the cue arena decreases, since the speed of robots also does not change in this setup.

Real robot experiments show better results for the same experiment specifications. Several reasons for this can be given. Robots' physical and dynamic models are not covered fully in the kinematic experiments. For example, while escaping the cue area after waiting time, robots can see each other and end "cue escape movement"; resulting in the robot sleeping again. Also "cue escape movement" checks nearby obstacles within a certain radius and then starts the movement. The line of sight on the front side of the robot is programmed to be more sensitive and detects robots from further distances. However, sides are less sensitive thus controlled areas for obstacles are fewer. If there is an obstacle outside of the controlled circle; and the robot starts moving as if there is nothing in that area, sleeping is initiated again after encountering a robot further from the controlled side area. Another example of this divergence is that filtering of RnB sensors sometimes is not enough to filter somehow amplified signal data from the environment or multiple robots; thus, generates noise data as obstacle data and sees robots or obstacles earlier than it should. It is also possible that signals that Vicon uses to interfere with the RnB sensor.

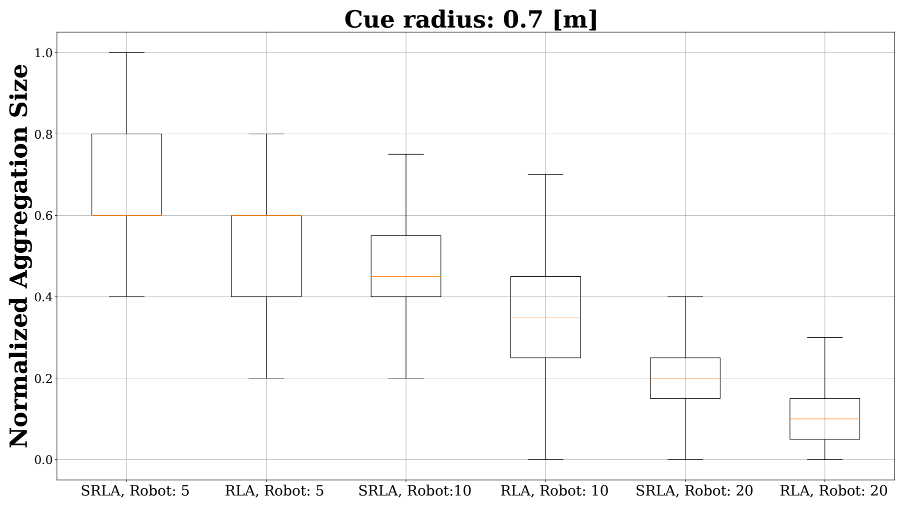

Figure 6.7: Comparison of RLA and SRLA in same cue radius of 0.7 [m], varying experiment area sizes. From left to right results from experiment arena sizes for 2500x5000 [mm], 3540x7070 [mm], and 5000x10000 [mm] can be seen with pairs. As the number of robot and arena size increases, stabilized NAS value decreases.

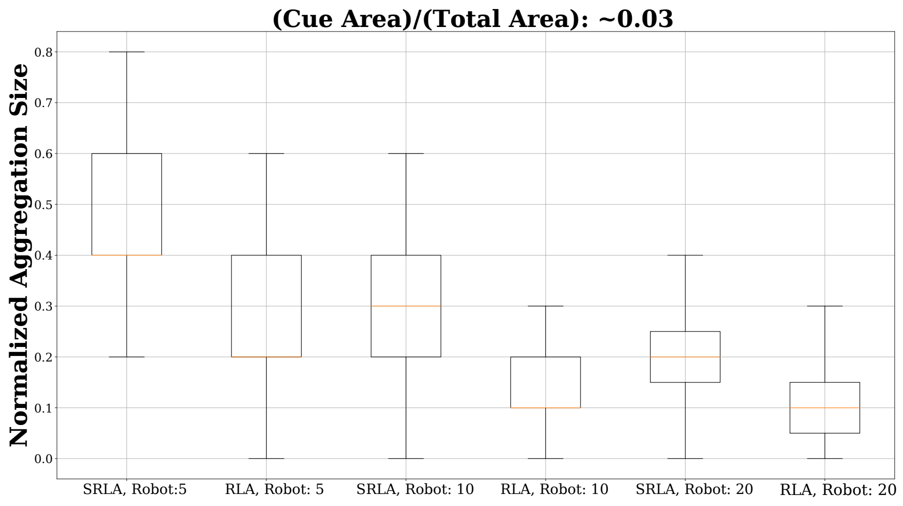

Figure 6.8: Comparison of RLA and SRLA in same cue-to-total-area rate of 0.03, varying cue radii. From left to right results from experiment arena sizes are the same as 6.7. As the number of robot and arena size increases, stabilized NAS value decreases.

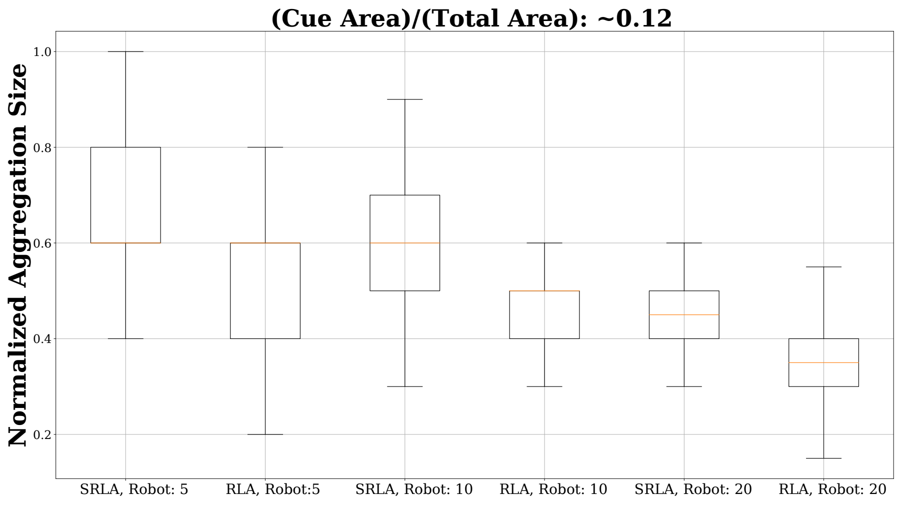

Figure 6.9: Comparison of RLA and SRLA in same cue-to-total-area rate of 0.12, varying cue radii. From left to right results from experiment arena sizes are the same as 6.7.

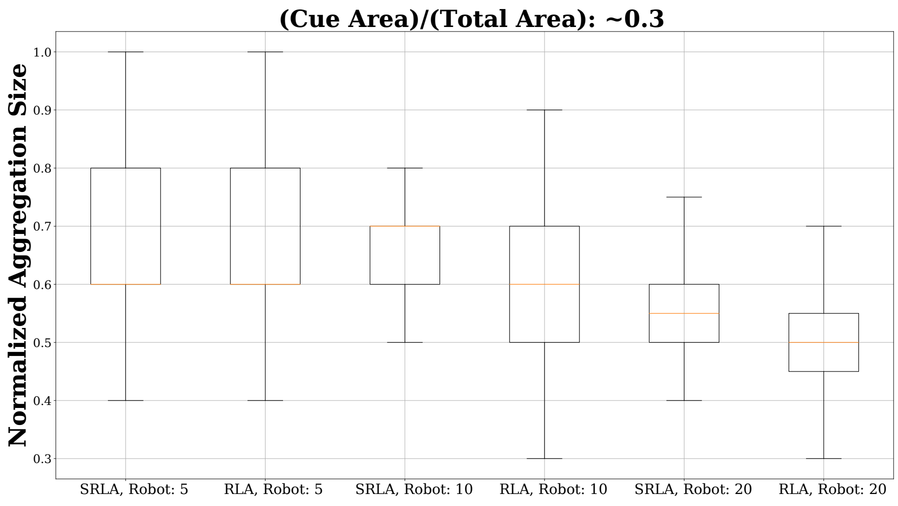

Figure 6.10: Comparison of RLA and SRLA in same cue-to-total-area rate of 0.3, varying cue radii. From left to right results from experiment arena sizes are the same as 6.7.

CHAPTER 7

CONCLUSION

In this thesis, development of social reinforcement learning-based aggregation method is covered together with the production of Kobot-T, a tracked robot to be used in swarm experiments. Kobot-T consists of several onboard systems. Kobot-T uses parts of Kobot CoRe architecture. To this architecture, Kobot-T introduces communication and in Kobot-T several subsystems are simplified by the designed board.

We developed Kobot-T to be used in foraging and aggregation experiments with simple subsystem addition and subtraction. Interfaces for these subsystems are developed and some of them, such as the RnB subsystem, are tested in cue-based aggregation experiments. Results gathered from cue-based aggregation experiments show that the interface of RnB works well.

We tested different aggregation methods, and the communication capability added to Kobot-CoRe together with novel social reinforcement learning-based aggregation method. To understand its effects, at first, previously used kinematic simulation in [44] was used. Here, important aspects and powerful sides of this method from the previously developed non-communicating version of this method, RLA, were determined. Then, robotic applications of all of the methods are done. Robotic application of SRLA could not show SRLA and RLA distinction well enough. Simulation results having a larger total area of 2.8x3.5 [m] with the same cue and robot parameters shows more distinct results. It is possible to repeat real robot experiments with larger experiment areas to show the importance of SRLA in real robots as well.

The need for prior knowledge about the experiment arena, that is previously needed for RLA method, still exists in the proposed method. However, the communication among robots is currently made through Wi-Fi connections. Thus Its possible that this thesis might develop in four directions. First, local communication can be added to terminate Wi-Fi-based local communication. Second, different communication strategies for changing cue areas can be adapted. Third, the developed gripper interface can be used in foraging swarm applications. Fourth, with different search and mapping algorithms, dependence on prior knowledge in the experiment arena can be avoided.

REFERENCES

[1] C. Bilaloğlu, “Development of an extensible heterogeneous swarm robot platform,” Master’s thesis, Middle East Technical University, 2022.

[2] I. d. C. Guimaraes, M. C. Pereira, N. R. Batista, C. A. P. Rodrigues, and W. F. Antonialli, “The complex nest architecture of the ponerinae ant odontomachus chelifer,” *PLoS One*, vol. 13, no. 1, p. e0189896, 2018.

[3] R. M. Adams, U. G. Mueller, T. R. Schultz, and B. Norden, “Agro-predation: usurpation of attine fungus gardens by megalomyrmex ants,” *Naturwissenschaften*, vol. 87, no. 12, pp. 549–554, 2000.

[4] C. Wen, J. Chen, W.-Q. Qin, X. Chen, J.-C. Cai, J.-B. Wen, X.-J. Wen, and C. Wang, “Red imported fire ants (hymenoptera: Formicidae) cover inaccessible surfaces with particles to facilitate food search and transportation,” *Insect Science*, vol. 28, no. 6, pp. 1816–1828, 2021.

[5] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraula, and E. Bonabeau, “Self-organization in biological systems,” in *Self-Organization in Biological Systems*, Princeton university press, May 2020.

[6] H. Haken, “Self-organization,” *Scholarpedia*, vol. 3, no. 8, p. 1401, 2008. revision #139276.

[7] A. Gupta, Y. Das, and B. Rao, “Anatomy of the cockroach,” *Cockroaches as Models for Neurobiology: Applications in Biomedical Research*, vol. 1, pp. 33–39, 1990.

[8] R. E. Snodgrass, *Anatomy of the honey bee*. Cornell University Press, 1956.

[9] M. Dorigo and M. Birattari, “Swarm intelligence,” *Scholarpedia*, vol. 2, no. 9, p. 1462, 2007. revision #138640.

[10] G. Beni, “From swarm intelligence to swarm robotics,” in *International Workshop on Swarm Robotics*, pp. 1–9, Springer, 2004.

[11] T. Schmickl and H. Hamann, “Beeclust: A swarm algorithm derived from honeybees,” *Bio-inspired computing and communication networks*, pp. 95–137, 2011.

[12] M. Dorigo, M. Birattari, and M. Brambilla, “Swarm robotics,” *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014. revision #138643.

[13] J. R. Riley, U. Greggers, A. D. Smith, D. R. Reynolds, and R. Menzel, “The flight paths of honeybees recruited by the waggle dance,” *Nature*, vol. 435, no. 7039, pp. 205–207, 2005.

[14] F. C. Dyer and J. L. Could, “Honey bee navigation: The honey bee’s ability to find its way depends on a hierarchy of sophisticated orientation mechanisms,” *American Scientist*, vol. 71, no. 6, pp. 587–597, 1983.

[15] F. Mondada, G. C. Pettinaro, A. Guignard, I. W. Kwee, D. Floreano, J.-L. Deneubourg, S. Nolfi, L. M. Gambardella, and M. Dorigo, “Swarm-bot: A new distributed robotic concept,” *Autonomous robots*, vol. 17, no. 2, pp. 193–221, 2004.

[16] G. Caprari and R. Siegwart, “Mobile micro-robots ready to use: Alice,” in *2005 IEEE/RSJ international conference on intelligent robots and systems*, pp. 3295–3300, IEEE, 2005.

[17] R. B. Rusu, R. Robotin, G. Lazea, and C. Marcu, “Towards open architectures for mobile robots: Zeero,” in *2006 IEEE International Conference on Automation, Quality and Testing, Robotics*, vol. 2, pp. 260–265, IEEE, 2006.

[18] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, “The e-puck, a robot designed for education in engineering,” in *Proceedings of the 9th conference on autonomous robot systems and competitions*, pp. 59–65, IPCB: Instituto Politécnico de Castelo Branco, 2009.

[19] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli, “Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics,” in *International Workshop on Swarm Robotics*, pp. 103–115, Springer, 2006.

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

[21] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. Marshall, and A. Reina, "Sophisticated collective foraging with minimalist agents: a swarm robotics test," *Swarm Intelligence*, vol. 14, no. 1, pp. 25–56, 2020.

[22] C. Dimidov, G. Oriolo, and V. Trianni, "Random walks in swarm robotics: an experiment with kilobots," in *International conference on swarm intelligence*, pp. 185–196, Springer, 2016.

[23] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 kilobots: Speed versus accuracy in binary discrimination problems," *Autonomous agents and multi-agent systems*, vol. 30, no. 3, pp. 553–580, 2016.

[24] M. Agrawal and S. C. Glotzer, "Scale-free, programmable design of morphable chain loops of kilobots and colloidal motors," *Proceedings of the National Academy of Sciences*, vol. 117, no. 16, pp. 8700–8710, 2020.

[25] C. Skjetne, P. C. Haddow, A. Rye, H. Schei, and J.-M. Montanier, "The chirp robot: A versatile swarm robot platform," in *Robot Intelligence Technology and Applications 2*, pp. 71–82, Springer, 2014.

[26] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, *et al.*, "Duckietown: an open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, IEEE, 2017.

[27] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3299–3304, IEEE, 2017.

[28] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox, "Mona: an affordable open-source mobile robot for education and research," *Journal of Intelligent & Robotic Systems*, vol. 94, no. 3, pp. 761–775, 2019.

[29] A. West, F. Arvin, H. Martin, S. Watson, and B. Lennox, "Ros integration for miniature mobile robots," 07 2018.

[30] Z. Liu, C. West, B. Lennox, and F. Arvin, "Local bearing estimation for a swarm of low-cost miniature robots," *Sensors*, vol. 20, no. 11, p. 3308, 2020.

[31] F. Arvin, A. E. Turgut, T. Krajník, and S. Yue, "Investigation of cue-based aggregation in static and dynamic environments with a mobile robot swarm," *Adaptive Behavior*, vol. 24, no. 2, pp. 102–118, 2016.

[32] A. S. Amjadi, M. Raoufi, A. E. Turgut, G. Broughton, T. Krajník, and F. Arvin, "Cooperative pollution source exploration and cleanup with a bio-inspired swarm robot aggregation," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 469–481, Springer, 2020.

[33] A. Vardy, "Aggregation in robot swarms using odometry," *Artificial Life and Robotics*, vol. 21, no. 4, pp. 443–450, 2016.

[34] J. Reinhard, M. V. Srinivasan, D. Guez, and S. W. Zhang, "Floral scents induce recall of navigational and visual memories in honeybees," *Journal of Experimental Biology*, vol. 207, no. 25, pp. 4371–4381, 2004.

[35] J. Reinhard, M. V. Srinivasan, and S. Zhang, "Scent-triggered navigation in honeybees," *Nature*, vol. 427, no. 6973, pp. 411–411, 2004.

[36] M. V. Srinivasan, "Honey bees as a model for vision, perception, and cognition," *Annual Review of Entomology*, vol. 55, pp. 267–284, 2010.

[37] T. Collett, "Insect navigation en route to the goal: multiple strategies for the use of landmarks," *The Journal of Experimental Biology*, vol. 199, no. 1, pp. 227–235, 1996.

[38] S. A. Kumar, B. Sharma, J. Vanualailai, and A. Prasad, "Stable switched controllers for a swarm of ugvs for hierarchal landmark navigation," *Swarm and Evolutionary Computation*, vol. 65, p. 100926, 2021.

[39] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, vol. 27, no. 5, pp. 511–533, 2010.

[40] M. S. Teymouri, *Landmark-based Distributed Topological Mapping and Navigation in GPS-denied Urban Environments Using Teams of Low-cost Robots*. PhD thesis, Lehigh University, 2021.

[41] L. Halodová, E. Dvořáková, F. Majer, T. Vintr, O. M. Mozos, F. Dayoub, and T. Krajník, "Predictive and adaptive maps for long-term visual navigation in changing environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7033–7039, IEEE, 2019.

[42] N. Lemmens and K. Tuyls, "Stigmergic landmark foraging," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 497–504, 2009.

[43] A. S. Amjadi, M. Raoufi, and A. E. Turgut, "A self-adaptive landmark-based aggregation method for robot swarms," *Adaptive Behavior*, vol. 30, no. 3, pp. 223–236, 2022.

[44] A. S. Amjadi, "Landmark-based aggregation method for robot swarms," Master's thesis, Middle East Technical University, 2021.

[45] "Ros official website." `https://www.ros.org/`.

[46] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[47] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, pp. 220–222, jun 1993.

[48] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[49] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[51] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[52] "Vicon brochure." `https://www.vicon.com/cms/wp-content/uploads/2019/05/Vantage_brochure.pdf`.