DEVELOPMENT OF AN EXTENSIBLE HETEROGENEOUS SWARM ROBOT PLATFORM

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

BY

CEM BİLAŁOĞLU

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

JANUARY 2022
DEVELOPMENT OF AN EXTENSIBLE HETEROGENEOUS SWARM ROBOT PLATFORM

submitted by CEM BİLALOĞLU 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 Kalpçilar
Dean, Graduate School of Natural and Applied Sciences

Prof. Dr. M. A. Sahir Arıkan
Head of Department, Mechanical Engineering

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

Assoc. Prof. Dr. Erol Şahin
Co-supervisor, Computer Engineering, METU

Examining Committee Members:

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

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

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

Assoc. Prof. Dr. Mehmet Bülent Özer
Mechanical Engineering, METU

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

Date: 13.01.2022
I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname:  Cem Bilaloğlu

Signature   :

iv
ABSTRACT

DEVELOPMENT OF AN EXTENSIBLE HETEROGENEOUS SWARM ROBOT PLATFORM

Bilaloğlu, Cem
M.S., Department of Mechanical Engineering
Supervisor: Assist. Prof. Dr. Ali Emre Turgut
Co-Supervisor: Assoc. Prof. Dr. Erol Şahin

January 2022, 86 pages

This thesis introduces Kobot – an extensible heterogeneous swarm robot platform. Kobot platform uses a common hardware and software architecture based on off-the-shelf components, 3-D printing, and open-source software that evolves with state of the art. Robots built using this common architecture range from wheeled to flying robots and formed a heterogeneous swarm. The common architecture enabled developing and testing systems for the lightweight flying robots on resourceful ground robots. As a result, Kobot platform proved its significance for the future of swarm robotics research with multiple novel swarm behaviors implemented for the first time in real robots.

Keywords: Swarm Robotics, Multi Robot Systems, Heterogeneous Swarm, Robot Perception, Decentralized Control
ÖZ

GENİŞLETİLEBİLİR BİR HETEROJEN SÜRÜ ROBOT PLATFORMUNUN GELİŞTİRİLMESİ

Bilaloğlu, Cem
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ali Emre Turgut
Ortak Tez Yöneticisi: Doç. Dr. Erol Şahin

Ocak 2022 , 86 sayfa


Anahtar Kelimeler: Sürü Robotiği, Çoklu Robot Sistemleri, Heterojen Sürü Robotu, Robot Algısı, Dağıtık Kontrol
to Ceren Kocaman, she has always been there for me
ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor Assist. Prof. Dr. Ali E. Turgut for introducing me to swarm robotics. He has given me his utmost support in every aspect possible and has always encouraged me.

I would like to thank my co-supervisor, Assoc. Prof. Dr. Erol Sahin for motivating me to extend the Kobot CoRe to flying robots and teaching me the delicacy of academic writing and presentation.

I owe special thanks to The Scientific and Technological Research Council of Turkey for the financial support they provided throughout my graduate studies.

I am grateful to Assoc. Prof. Dr. A. Bugra Koku for helping me to find my passion in robotics, develop my perspective, and get me acquainted with the open-source community.

Many thanks go to Assoc. Prof. Dr. Farshad Arvin, for motivating and guiding me whenever I needed it.

I would like to acknowledge my friend, Mehmet Şahin, who helped me in aerial robot experiments and saved me when I was stuck deep.

Furthermore, I would like to thank my family for their endless love, support, and encouragement.

Lastly, I would like to thank my sister Gizem, who is my inspiration and guide.
# TABLE OF CONTENTS

**ABSTRACT** ................................................................. v

**ÖZ** ................................................................. vi

**ACKNOWLEDGMENTS** ....................................................... viii

**TABLE OF CONTENTS** .................................................... ix

**LIST OF TABLES** ........................................................ xii

**LIST OF FIGURES** ........................................................ xiv

**LIST OF ALGORITHMS** .................................................... xvii

**LIST OF ABBREVIATIONS** .................................................. xviii

**CHAPTERS**

1 **INTRODUCTION** ..................................................... 1

2 **REVIEW OF AVAILABLE SWARM ROBOTS** ......................... 3

   2.1 Swarm Robot Design Trends ........................................... 3

   2.2 Heterogeneous Swarm Robots ......................................... 4

   2.3 Comparison of Swarm Robots ......................................... 4

      2.3.1 Contribution of Kobot Platform ................................. 6

3 **KOBOT PLATFORM HARDWARE** ........................................ 9

   3.1 Kobot CoRe ........................................................... 9
# LIST OF TABLES

## TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Comparison of available swarm robots</td>
<td>5</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Motor parameters</td>
<td>23</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>Available peripherals of Kobot-W for extensions</td>
<td>28</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Component costs for Kobot-W. Unit costs given for minimum of 5 Kobots</td>
<td>29</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Systems used and tested on a swarm setting by the self-organized flocking algorithm</td>
<td>36</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Systems used and tested on a swarm setting by the BEECLUST algorithm</td>
<td>39</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>Systems used and tested on a swarm setting by the LBA algorithm</td>
<td>40</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>Register map of the kobotMotor class. R/W denotes read and write operations by the I2C master</td>
<td>46</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Kinematic parameters that are used for odometry calculation (values are in mm)</td>
<td>57</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Parameter set used in aggregation experiments</td>
<td>64</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Parameter set used in flocking experiments</td>
<td>66</td>
</tr>
</tbody>
</table>
Table 6.1 Comparison of the selected algorithms in terms of used hardware and software architecture
LIST OF FIGURES

FIGURES

Figure 1.1 Kobot-I robot .................................................. 2

Figure 3.1 Kobot-W (Wheeled) and Kobot-F (Flying) robots built using Kobot CoRe .................................................. 10

Figure 3.2 Flow diagram showing hardware architecture of Kobot-W and Kobot-F .................................................. 10

Figure 3.3 Kobot-I’s IR range and bearing system .................................................. 12

Figure 3.4 a) Printed circuit board (PCB) layout b) assembled ToF R&B system of the CoRe .................................................. 13

Figure 3.5 ToF sensor daughterboards. a) 4x3 Panel used for the production, daughter boards are held together by breakable mousebite features. b) Assembled daughterboard of low-cost ToF sensor VL53L0. b) Assembled daughterboard of high-range ToF sensor VL53L1 .................................................. 14

Figure 3.6 a) PCB layout b) assembly of floor sensing system .................................................. 15

Figure 3.7 Local communication module daughterboards. a) 2x3 Panel used for the production, daughter boards are held together by breakable mousebite features. b) Assembled daughterboard, where direct cross-talk between emitter and receiver is eliminated by a female header soldered in between .................................................. 17

Figure 3.8 a) CAD model b) assembly of 2-DoF compliant gripper system .................................................. 18
Figure 3.9  a) Exploded view of printed in place gripper arm b) Numerical calculation of gripper dimensions by applying geometrical constraints on a CAD program ................................. 19

Figure 3.10 Indirect inter-robot communication interfaces of the CoRe . . . . . . 22

Figure 3.11 Exploded views of Kobot-W and Kobot-F ............................... 26

Figure 3.12 Pictures of a Kobot-W used for development of the CoRe systems 28

Figure 3.13 Disassembled Kobot-F ............................................................... 30

Figure 3.14 Prototype level Kobot robots with alternative locomotion systems 30

Figure 4.1 Flow diagram showing the software CoRe ................................. 32

Figure 4.2 Flow diagram showing the software CoRe modularity. Dashed boxes lists alternative mobile bases and extensions for different robots using the CoRe. The symbol ‘/’ separates alternative modules. For mobile base the first alternative is Kobot-W, together with ‘diff_driver’, ‘kobotMotorController’ and the second alternative is Kobot-F with the respective modules. For extensions the first alternative is ‘Floor Sensor’ using ‘smbus2’ and the other alternatives are gripper, e-ink display and led ring using corresponding modules ................................. 34

Figure 4.4 Kobot-F tracking Kobot-W robot by performing pose estimation using its onboard camera and ArUco marker on Kobot-W ................................. 45

Figure 4.5 Ranging cycle of a ToF sensor ...................................................... 48

Figure 4.6 Ambient reading of a ToF sensor facing another ToF sensor . . . 49

Figure 4.7 Robot detection of two ToF sensors. Symbol [ ] denotes no robot detection and symbol $r_{i,j}$ denotes that robot $i$ detected robot $j$ ................................. 50

Figure 5.1 The experimental setup used for the swarm scenarios ................................. 53
Figure 5.2  a) Glass tea cup filled with bolts weighing 580 g b) Gripper holding glass tea cup ................................. 55

Figure 5.3  Odometry performance of Kobot-W using kinematic parameters from design and estimation ................................. 56

Figure 5.4  Shaded line plots showing mean, minimum, and maximum errors of pose estimation vs. distance to marker, with 480p camera resolution ................................. 58

Figure 5.5  Boxplot showing the discrete sensor readings of the IR R&B vs. actual inter-robot distance ................................. 59

Figure 5.6  Scatter plots showing the ranging performance of the ToF R&B ................................. 60

Figure 5.7  The robot detection performance of ToF R&B. Upper plots show the IQR value of the ambient buffer. Lower plots show the robot detection values by using an IQR threshold of 5[kcps] ................................. 61

Figure 5.8  Line plots showing raw ambient readings and IQR value of a buffer of the ToF sensor with exposure to the sunlight ................................. 62

Figure 5.9  A snapshot from the aggregation experiments ................................. 63

Figure 5.10  Shaded line plots showing mean, minimum, and maximum NAS values for BEECLUST and LBA (Landmark-Based Aggregation) experiments of Kobot-W ................................. 64

Figure 5.11  A snapshot from the flocking experiments. The ceiling camera and ArUco markers on Kobots are used for pose estimation ................................. 65

Figure 5.12  Performance metrics of Kobot-W flocking experiments ................................. 67

Figure 5.13  Snapshots from Kobot-F flocking experiments. Yellow arrows show heading direction. Red LEDs and arrows show nearby obstacle direction. Blue LEDs and arrows show neighbor robot direction ................................. 68
LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1 Self-Organized Flocking [1] ......................... 35
Algorithm 2 BEECLUST ........................................... 39
Algorithm 3 Landmark Based Aggregation [2] ..................... 40
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-D</td>
<td>2 Dimensional</td>
</tr>
<tr>
<td>3-D</td>
<td>3 Dimensional</td>
</tr>
<tr>
<td>BT</td>
<td>Bluetooth</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>DoF</td>
<td>Degree of Freedom</td>
</tr>
<tr>
<td>I2C</td>
<td>Inter Integrated Circuit</td>
</tr>
<tr>
<td>IQR</td>
<td>Inter Quartile Range</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>LBA</td>
<td>Landmark-based Aggregation</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>MAV</td>
<td>Micro Aerial Vehicle</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro Controller Unit</td>
</tr>
<tr>
<td>OTS</td>
<td>Off-the-shelf</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>R&amp;B</td>
<td>Range and Bearing</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>RPi</td>
<td>Raspberry Pi</td>
</tr>
<tr>
<td>SDEV</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>ToF</td>
<td>Time of Flight</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>WiFi</td>
<td>Wireless Fidelity</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Local interactions of social animals result in global outcomes beyond the capabilities of individuals. Such collective behaviours have been investigated in various populations with all kinds of distinctness, ranging from swarms of amoeba [3], ants [4], and honeybees [5] to herds of land animals, schools of fish and flocks of birds [6]. The simplicity and scalability of biological populations’ local interactions for collective behaviors inspired researchers from various fields, including robotics. Swarm robotics which is an active research area, advantages this bio-inspiration [7] and aims to coordinate robot populations.

Swarm robotics research is mainly conducted using computer-based simulations. Using a simulator is a low-cost solution and speeds up testing algorithms. However, it lacks in testing the robustness and scalability of a swarm in real-world conditions. Accordingly, researchers need robots to collect data, model the hardware realistically for simulation, and perform real-life experiments for validation. However, the design of a swarm robot is essentially different from an autonomous robot designed for individual operation. A solitary robot relies on the performance of its onboard resources for a successful operation, whereas swarm robots grow stronger in numbers. Simpler as an individual, swarm robots are inherently difficult to design and maintain in a scalable fashion. Nevertheless, available robots are specialized, not modular, and not re-usable in various tasks [8]. This problem is more profound with the increased popularity of the flying swarm robots [9, 10, 11, 12, 13], which require custom lightweight systems that are not already available on ground robots.

Admitting that flying robots’ additional degree of freedom (DoF) creates vast opportunities, they are inadequate for development since they carry their burden. Although
computer simulations solve this problem to some extent, we believe testing flying robot software and hardware on resourceful ground robots with a common architecture is the missing link. After developing the system and performing tests on the ground, researchers can use the novel systems on flying robots for the actual application. In this thesis, we repeatedly demonstrate the relevance of this claim for swarming hardware and software systems.

Kobot-I [14], the predecessor of the platform presented in this thesis, is one of the first robots developed for swarm robotics and was only used for self-organized flocking experiments. Like all the contemporary robots, Kobot-I had no other choice but to use custom parts and interfaces that limited extending and upgrading. Learning from that experience, we designed Kobot, not a fixed architecture robot but an open, heterogeneous swarm robotics platform that can adapt to novel tasks, extend to different robots and upgrade with state of the art. Kobot uses off-the-shelf (OTS) components open-source software and relies on standard interfaces to be a platform that endures long years of research, updates with the advancements, and customizes by the researcher’s needs.

Figure 1.1: Kobot-I robot
CHAPTER 2

REVIEW OF AVAILABLE SWARM ROBOTS

Over the years, fixed architecture robots for various environments have been developed, including swarms of underwater vehicles, flying robots, and frequently, ground robots. We reviewed different robots to build an inclusive platform for swarm robotics and determine its must-have features. The review first summarizes the critical trends and design decisions observed in swarm robots chronologically, then compares existing robots and deducts necessary attributes for Kobot platform.

2.1 Swarm Robot Design Trends

Pioneering swarm robots, Swarm-bot [15], Alice [16], ZeeRo [17] used bio-inspiration on a scale that investigates high-level swarm interactions rather than the physiology of animals. This practical abstraction resulted in imitating animal behaviors with the available traditional hardware such as wheels or tracks. Wheeled or tracked systems are not miniature despite the ease of maneuvering and control. Hence experiments with differential drive robots have a population size of tens. For that reason, robots with alternate actuation systems such as Kilobot [18] and Droplet [19] emerged, enabling experiments with unprecedented populations like hundreds. As a result, different institutions around the globe use Kilobots for various swarm experiments ranging from collective motion [20] and decision making [21] to spatial organization [22].

Most recently, following the popularity of Kilobots in the media, influence of swarm robots from research to the general community has accelerated. Interesting aspects of this expansion are seen on robot platforms specialized on human-swarm interactions (Zooids [23], Cellulo [24], Reactile [25]), swarm capable educational robots
(Aeris [26], R-one [27], Andruino R2 [28], Mona [29], Duckiebot [30]) and commercial robots (Tello [31], Sphero [32], Thymio [33]).

2.2 Heterogeneous Swarm Robots

Swarm robots do not always extend the capability by extensions but with other robots working in the same setup. When the individual robots composing the population are of the same type, having the same abilities, the swarm is classified as homogeneous. On the other hand, heterogeneous swarms leverage different types of interacting robots to enhance the skills that swarm possesses. The Swarmonoid Project [34] used a heterogeneous swarm to form a meta-humanoid showing the advantages of heterogeneity by succeeding at tasks that a homogeneous setup could not perform. This meta-humanoid consists of Foot-bots, successors of the S-bot [15] from the Swarm-bots Project, hand-bots (manipulator robots), and eye-bots (flying robots). Eye-bots integrated infrared (IR) range and bearing (R&B) sensor [35, 36] to the flying robots for the first time and enabled the first autonomous swarm operation with indoor drones. Although the design of the R&B is similar in concept to the other IR R&B systems [37, 38, 39, 40] eye-bot’s systems are unique by ‘tilt immunity’ compensating for the misalignment of drones flying at different altitudes.

2.3 Comparison of Swarm Robots

Different locomotion strategies of the available swarm robots resulted in different levels of extensibility and onboard capabilities as depicted in Table 2.1. Dominated by the differential drive ground robots, the table also lists alternative locomotion strategies such as vibration [18, 19], legged [54] and flying [46, 9, 10]. Investigating the table, it is evident that most of the wheeled or tracked robots are fully onboard and alternative locomotion systems usually need external resources. Moreover, the platforms which use alternative locomotion are not extensible by being miniature or lightweight to fly.
Table 2.1: Comparison of available swarm robots

<table>
<thead>
<tr>
<th>Robot</th>
<th>Year</th>
<th>Locomotion</th>
<th>Open Source</th>
<th>Extensible</th>
<th>Mechanical Parts</th>
<th>Onboard</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-bot[15]</td>
<td>2004</td>
<td>Treel</td>
<td>No</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>Alice[16]</td>
<td>2005</td>
<td>Wheel</td>
<td>No</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>ZeeRo[17]</td>
<td>2006</td>
<td>Wheel</td>
<td>Yes</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>Kobot-I[1]</td>
<td>2008</td>
<td>Wheel</td>
<td>No</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>MindART[41]</td>
<td>2008</td>
<td>Wheel</td>
<td>No</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>AMiR[42]</td>
<td>2009</td>
<td>Wheel</td>
<td>No</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>E-Puck[43]</td>
<td>2009</td>
<td>Wheel</td>
<td>Yes</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>Jasmine[44]</td>
<td>2009</td>
<td>Wheel</td>
<td>Yes</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>marXbot[45]</td>
<td>2010</td>
<td>Treel</td>
<td>No</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>sFly[46]</td>
<td>2012</td>
<td>Flying</td>
<td>No</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>Eyebots[34]</td>
<td>2012</td>
<td>Flying</td>
<td>No</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>Kilobot[18]</td>
<td>2012</td>
<td>Vibrating</td>
<td>Yes</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>R-One[27]</td>
<td>2013</td>
<td>Wheel</td>
<td>No</td>
<td>No</td>
<td>OTS</td>
<td>Yes</td>
</tr>
<tr>
<td>Droplet[19]</td>
<td>2014</td>
<td>Vibrating</td>
<td>Yes</td>
<td>No</td>
<td>Custom</td>
<td>No</td>
</tr>
<tr>
<td>ChIRP[47]</td>
<td>2014</td>
<td>Wheel</td>
<td>Yes</td>
<td>Yes</td>
<td>3-D Printed</td>
<td>Yes</td>
</tr>
<tr>
<td>Colias[48]</td>
<td>2014</td>
<td>Wheel</td>
<td>Yes</td>
<td>No</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>Elisa 3[49]</td>
<td>2014</td>
<td>Wheel</td>
<td>No</td>
<td>No</td>
<td>3-D Printed</td>
<td>Yes</td>
</tr>
<tr>
<td>GRITSBot[50]</td>
<td>2015</td>
<td>Wheel</td>
<td>Yes</td>
<td>No</td>
<td>OTS</td>
<td>Yes</td>
</tr>
<tr>
<td>Zooids[23]</td>
<td>2016</td>
<td>Wheel</td>
<td>Yes</td>
<td>No</td>
<td>3-D Printed</td>
<td>No</td>
</tr>
<tr>
<td>Aeris[26]</td>
<td>2017</td>
<td>Wheel</td>
<td>No</td>
<td>No</td>
<td>3-D Printed</td>
<td>No</td>
</tr>
<tr>
<td>Cellulo[24]</td>
<td>2017</td>
<td>Omniwheel</td>
<td>No</td>
<td>No</td>
<td>3-D Printed</td>
<td>Yes</td>
</tr>
<tr>
<td>Duckiebot[30]</td>
<td>2017</td>
<td>Wheel</td>
<td>Yes</td>
<td>Yes</td>
<td>OTS</td>
<td>Yes</td>
</tr>
<tr>
<td>HeRo[51]</td>
<td>2017</td>
<td>Wheel</td>
<td>Yes</td>
<td>No</td>
<td>3-D Printed</td>
<td>No</td>
</tr>
<tr>
<td>Khepera IV[52]</td>
<td>2017</td>
<td>Wheel</td>
<td>No</td>
<td>Yes</td>
<td>Custom</td>
<td>Yes</td>
</tr>
<tr>
<td>microMVP[53]</td>
<td>2017</td>
<td>Wheel</td>
<td>Yes</td>
<td>No</td>
<td>3-D Printed</td>
<td>No</td>
</tr>
<tr>
<td>Crazyswarm[6]</td>
<td>2017</td>
<td>Flying</td>
<td>Yes</td>
<td>Yes</td>
<td>Custom</td>
<td>No</td>
</tr>
<tr>
<td>Spiderino[54]</td>
<td>2017</td>
<td>Legged</td>
<td>No</td>
<td>No</td>
<td>3-D Printed</td>
<td>No</td>
</tr>
<tr>
<td>VioSwarm[10]</td>
<td>2018</td>
<td>Flying</td>
<td>No</td>
<td>No</td>
<td>Custom</td>
<td>No</td>
</tr>
<tr>
<td>Mona[29]</td>
<td>2019</td>
<td>Wheel</td>
<td>Yes</td>
<td>Yes</td>
<td>OTS</td>
<td>Yes</td>
</tr>
<tr>
<td>Proteus[55]</td>
<td>2019</td>
<td>Wheel</td>
<td>No</td>
<td>No</td>
<td>3-D Printed</td>
<td>Yes</td>
</tr>
<tr>
<td>Kobot</td>
<td>2021</td>
<td>Wheel, Flying</td>
<td>Yes</td>
<td>Yes</td>
<td>3-D Printed</td>
<td>Yes</td>
</tr>
</tbody>
</table>
2.3.1 Contribution of Kobot Platform

Kobot-I [14] is a CD-sized (i.e., 12 cm) robot used for swarm robotics research and is the predecessor of the hardware presented in this work. Kobot-I has a range and bearing system based on modulated IR signal, and it has been verified in a flocking scenario. Other noteworthy specifications of Kobot-I include parallel programming, a virtual heading sensor based on a compass, an RF communication module, and 10 hours of operation. As a result, for us, Kobot-I represents what needs to be conserved to be a successful swarm robot platform and what pioneer robots lack today.

The pace of advancement in software and hardware show that it is meaningless to claim that a robot can survive for more than a few years with a fixed architecture. If we extend this period to a decade, robots will become almost obsolete. We observed this problem firsthand in Kobot-I and most other robots in the literature. This issue stems from old robots relying on unsupported software, hardware, and custom parts. At best, early robots try to survive only by adding new extensions to the old base. Unfortunately, this endeavor does not resolve the inadequacy of the legacy robots to today’s standards. To solve this problem, we designed the Kobot platform from the ground up as a standard platform that keeps improving and getting updated with off-the-shelf components.

Developing extensible and open-source robots were previously intended, yet the robots proposed in the literature lacked crucial elements. First of all, open-source does not ensure accessibility alone. The accessibility problem was more significant early on since the DIY (do it yourself) community and 3-D Printing were underdeveloped. Correspondingly, robots were produced by third parties and sold at remarkably high prices. However, extended Kobot robots use solely 3-D printed parts and OTS components, hence accessible for everyone to produce. Secondly, previous platforms were designed using proprietary software, and the provided sources were generic files (such as PDF, Gerber, STEP) that are not directly modifiable. On the other hand, we designed Kobot platform using either open source or educationally licensed programs and directly provided the design files.
Furthermore, Kobot platform is not just another robot but is an extensible heterogeneous swarm robot platform. Therefore, researchers can extend desired robots from the platform like we do multiple times in this work, advantaging common architecture, open-source tools, 3-D Printing, and OTS components. Moreover, unlike any other robot, computation is not fixed. Instead, the Kobot platform uses lightweight, low-power, or high-power, high-performance controllers depending on the researchers’ needs and improves the computational power with the updated OTS components.

Extended robots abstracted on the common architecture ranging from aerial robots to ground robots contribute to an easy to develop and maintain heterogeneous swarm. As a result, aerial robots of the platform can leverage the systems developed on resourceful ground robots such as the first time of flight (ToF) R&B. This system enabled the first truly decentralized and onboard swarm behavior with indoor aerial robots. On the other hand, the ground robots performed the first real robot experiments of a novel swarm behavior.

To sum up, Kobot platform:

- use open-source or educationally licensed programs
- is based on, and upgrades with OTS components, 3-D Printing, and open-source software
- extends to fully onboard ground and aerial swarm robots with a common architecture
- enables implementation of multiple algorithms for the first time on real robots
Kobot platform introduces a common architecture (CoRe) that can extend to different robots ranging from ground to aerial robots according to the needs of the researchers. The CoRe hardware is designed around the principles of upgradability and expandability. Thus, the CoRe is based upon off-the-shelf, mature hardware, which will not change any time soon. Microcontrollers (MCUs), which can be programmed by the Arduino framework and Raspberry Pi (RPi) boards, are selected for this purpose. Many third-party extensions depend on their physical dimensions and interfaces. More importantly, their design iterations showed improved performance with preserved interfaces. Since OTS components’ interfaces will not change abruptly in consecutive versions, the CoRe aims to endure long years of hardware updates with minor modifications.

In this chapter, hardware design of the CoRe and extended ground, aerial robots Kobot-W and Kobot-F (see Fig. 3.1) will be introduced, presenting preliminary design calculations.

3.1 Kobot CoRe

Figure 3.2 outlines abstracted hardware layers of Kobot-W and Kobot-F robots. The bottom layer is the locomotion and power layer, and it is responsible for the robot’s locomotion by external velocity commands and supplying regulated power to the robot. The middle layer comprises swarming sub-systems for interacting with the environment and neighbor robots. Lastly, the top layer is the high-level controller, responsible for the operation by controlling the below layers.
Figure 3.1: Kobot-W (Wheeled) and Kobot-F (Flying) robots built using Kobot CoRe

<table>
<thead>
<tr>
<th>Kobot Robot</th>
<th>Kobot-W (Wheeled)</th>
<th>Kobot-F (Flying)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-level Controller Layer</td>
<td>RPi 3/4</td>
<td>RPi Zero</td>
</tr>
<tr>
<td>Swarm Sensory-Motor,</td>
<td>Time of Flight Range&amp;Bearing</td>
<td>Time of Flight Range&amp;Bearing</td>
</tr>
<tr>
<td>Communication Layer</td>
<td>IMU</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Floor Sensing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Infrared Range&amp;Bearing</td>
<td></td>
</tr>
<tr>
<td>Locomotion, Power Layer</td>
<td>Drive System</td>
<td>Tello Drone</td>
</tr>
<tr>
<td></td>
<td>Wheels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ground Base + Wheels</td>
<td>Battery</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Battery</td>
</tr>
</tbody>
</table>

Figure 3.2: Flow diagram showing hardware architecture of Kobot-W and Kobot-F
3.1.1 High-level Controller Layer

The high-level controller layer consists of a single OTS component RPi – the main computational unit. Different Kobot robots can use various RPi boards to trade computational power to weight and power consumption. For instance, Kobot-F uses lightweight and low-power RPi Zero, whereas resourceful Kobot-W can use RPi 3 or 4 for computationally intensive tasks.

The high-level controller layer is the simplest in hardware but most complex in software and wiring. All the lower layers should be connected to RPi for communication. The communication between the RPi and the peripherals is either by universal asynchronous receiver transmitter (UART), the inter-integrated circuit (I2C) bus, or serial peripheral interface (SPI). Bus-type architectures are preferred since it is straightforward to add new peripherals without introducing extra complexity to the wiring scheme. Additionally, many third-party breakout boards on the market use above mentioned digital communication protocols as the communication interface.

3.1.2 Swarm Sensory-Motor, Communication Layer

This subsection will introduce modular sensory-motor and communication systems included in the CoRe. Modular systems extend the capabilities of robots to a variety of applications by interacting with kin robots and the environment. The CoRe systems are stand-alone in hardware and software and solely use OTS components. Accordingly, the systems are accessible for other robot platforms to use.

IR Range and Bearing System

Kobot robots have two different R&B systems. The first one, IR R&B, is designed for the Kobot-I robot comprises a central MCU that controls the operation of eight MCUs that operate IR emitter-receiver pairs. The system detects robots and uses the reflected signal’s power for discrete range values. The system was proven earlier by the self-organized flocking behavior performed by the Kobot-I swarm. Besides, Kobot-W can still use this system for tasks not requiring a high resolution ranging.
**ToF Range and Bearing System**

The second R&B system (see Fig. 3.3) is a drop-in replacement for the IR R&B but capable of higher resolution and maximum range thanks to ToF (time-of-flight) sensors. However, the ToF sensors cannot detect robots without modification and report a random distribution of ranges when used with small or irregularly shaped robots like micro aerial vehicles (MAVs). Additionally, ToF sensors are not suitable for a swarm setup since neighboring robots equipped with the same sensors create a noisy environment.

To overcome the problems mentioned above, we modified the robots’ hardware to increase the signal to noise ratio and filtering sensor readings on the MCU controlling the sensors’ operation. The goal was to build the first robot detection capable ToF R&B to the best of our knowledge.

On its own, reflected signals from small objects and robots are not enough to be detected by the sensor and as an object with consistent range reading. The first approach we took was to make the robots’ cross-sections larger and fill the gaps of truss-like structures. However, in controlled experiments, it is found that increasing
cross-sectional size has side effects on the flight performance of the MAVs by increasing the air drag. More importantly, increasing the cross-sectional size have negligible positive effects on the reflected signal strength. Next, we increased the reflectivity of the robot by retroreflective coating, instead of increasing the cross-sectional area.

The ToF R&B can work with two different sensors from the same product line that are drop-in replacements for each other in hardware but incompatible in software. The lower-cost sensor with a lower range is suitable for Kobot-W, whereas the high-range, expensive sensor, is preferred for Kobot-F. The sensors interfaces to the R&B motherboard by detachable daughterboards comprising sensor and nearby bypass capacitors to decrease supply voltage ripple. The overall system comprises twelve daughter boards placed equally spaced on the circumference of the R&B motherboard. Therefore, to reduce the assembly and production costs, PCBs are manufactured and assembled as a $4 \times 3$ grid of daughter boards (see Fig. 3.5). For soldering to the motherboard, they are further clipped to individual daughterboards.

For ground robots, height is constant, and the vertical field of view (FoV) of sensors is not an issue. However, aerial robots should fly at the same altitude with the allowable
Figure 3.5: ToF sensor daughterboards. a) 4x3 Panel used for the production, daughter boards are held together by breakable mousebite features. b) Assembled daughterboard of low-cost ToF sensor VL53L0. b) Assembled daughterboard of high-range ToF sensor VL53L1

maximum height difference determined by the vertical FoV of the sensor.

For the maximum range of the sensor in the short ranging mode, $d_{max} = 1300[mm]$ allowable maximum height difference $y_{max}$ between any two robots is

$$y_{max} = \pm x \times \tan(\Phi_{half})$$

$$= \pm 312.1[mm]$$

where the half FoV of the sensor from the datasheet is $\Phi_{half} = 13.5[deg]$. Allowable height difference value $y_{max}$ decreases linearly according to the Eqn. (3.2) when the inter robot distance is smaller.

Secondly, one must decide how many sensors are needed on the circumference to ensure full 2-D coverage. Considering full FoV of the sensor $2 \times \Phi_{half}$, the number of sensors required is $360/(2 \times \Phi_{half}) = 13.3[deg]$ which means we need to use at least 14 sensors for full coverage, that would result in overlapping regions of sensor FoVs. In contrast to using a configuration with overlapping FoVs, we took a different approach considering that the robots will be moving and blind zones will move with the robots. Hence, we restricted the blind zones to be smaller than the diagonal projection of the robot onto the sensor plane. Using twelve sensors will result in such a configuration,
and the maximum blind-zone width in the sensor plane is:

\[ y = 2 * x * (\tan(\phi_{sep}/2) - \tan(\Phi_{half})) + y_{sep} \tag{3.3} \]

\[ = 89.26[mm] \tag{3.4} \]

where distance between consecutive sensors distributed equally on the circumference is \( y_{sep} = 16.8[mm] \) and \( \phi_{sep} \) corresponds to the separation angle between the sensors \( 360/12 = 30[deg] \).

**Floor Sensing System**

The CoRe’s floor sensing system (see Fig. 3.6) detects cues of the aggregation experiments and senses the quality of a particular point inside the cue. Higher intensity readings than the free space denote aggregation cues abstracting robots’ attraction to a specific zone. For that purpose, reflective object sensors are used in the floor sensing system to detect the intensity of the reflected IR signal.

In the designed system, sensors report analog values. First, an analog to digital con-
verter integrated circuit converts analog readings to digital. Then, the high-level controller requests readings from the system. The floor sensing system uses four sensors for filtering sensor noise, compensating for the tilt of the robot base, and intensity gradient calculation.

The potential use cases for the reflective object sensors are two folds. Firstly, these sensors can be used for very short-range proximity readings when the floor pattern is uniform. Thus, they can detect cliffs for ground robots. Secondly, in aggregation experiments, sensors can report a signal proportional to the IR reflectivity of the floor pattern, provided that the distance to the floor is fixed.

**IR Local Communication System**

An IR local communication daughterboard is designed for scalable local communication without carrier congestion. The module houses an IR light emitting diode (LED) emitting 940 [nm] signal and an IR receiver, which demodulates 940 [nm] signals at 40 [kHz]. A female header soldered between the receiver emitter pairs to eliminate the cross-talk. The emitter-receiver pair has a high field of view, and a configuration of three modules placed radially and one module vertically guarantees full coverage. Notably, IR emitter-receiver pairs are high power and suitable even when the receiver and emitter are not in a direct line of sight. This is the case because the reflected signals from the environment have enough power for the receiver to detect. Nevertheless, we prioritize the locality of communication and filter out reflected signals. This is performed by using a current limiting resistor to reduce the reflected signals. The resistor package is selected to be hand-soldered to be replaced when higher or lower communication ranges are desired.

One can calculate the required resistor value and corresponding range using IR emitter’s irradiance given in [mW/Sr] and minimum irradiance detected by the receiver 0.7[mW/m²]. Although this units are incompatible, the relation between [Sr] and [m²] can be constructed since the surface area of a 1[Sr] (Steradian) for a sphere with radius r is $r^2$[m²]. From datasheet of the selected IR emitter maximum radiant intensity is $I_{E_{\text{min}}} = 1.0$[mW/Sr] when the $I_f = 20[mA]$ at $V_{f_{\text{min}}} = 1.2[V]$. The irradiance values have a large spread and calculations should be based on the worst-
Figure 3.7: Local communication module daughterboards. a) 2x3 Panel used for the production, daughter boards are held together by breakable mousebite features. b) Assembled daughterboard, where direct cross-talk between emitter and receiver is eliminated by a female header soldered in between case-scenario which corresponds to the minimum irradiance. Considering the supply voltage of the module is $V_{sup} = 3.3[V]$, the required resistor value for $I_f = 20[mA]$ can be found:

$$R = \frac{V_{sup} - V_{f_{max}}}{I_f}$$

$$R = 105[\text{ohm}] \quad (3.5)$$

Selecting $R = 100[\text{ohm}]$ is a conservative choice and will result in $I_f > 20[mA]$ and $I_{E_{min}} > 1.0[mW/Sr]$. Next, considering the minimum irradiance required by the receiver is $0.7[mW/m^2]$, the maximum range can be calculated as:

$$1/r_{max}^2 [mW/m^2] = 0.7[mW/m^2]$$

$$r_{max} = 1.195[m] \quad (3.7)$$

which corresponds to the maximum range $r_{max}$ in the default configuration of the module.

For a higher range with the same emitter $I_f = 100[mA]$ at $V_{f_{min}} = 1.4[V]$ and $I_{E_{min}} = 8.0[mW/Sr]$ can be used. By Eqn. 3.6 the resistor needs to be $R = 19[\text{ohm}]$ and the corresponding maximum range can be calculated by Eqn. 3.8 as $r_{max} = 3.38[m]$. 

17
2-DoF Compliant Gripper System

A 2 degree of freedom (DoF) compliant gripper system (see Fig. 3.8) is designed for performing manipulation tasks with Kobot robots. The system can elevate, grip, and fully retract inside the robot’s body. This way, the collision geometry of the robot is kept simple. The gripper arm is printed in place and does not require any assembly, and the mechanism’s compliance results from a fin-ray-shaped arm with embedded revolute joints. Compliance is preferred to use the same gripping action to hold a variety of shapes.

![Figure 3.8: a) CAD model b) assembly of 2-DoF compliant gripper system](image)

The design specifications for the gripper arm are as follows:

- Gripper system should be placed as low as possible, not to raise the robot’s center of mass and not to tip-over when carrying a heavy object
- Maximum extend of the gripper arms limited by the length of the gripper, which can be fully embedded inside the robot body when arms are in the retracted configuration
- Gripper should be able to elevate objects from the ground to move freely without the risk of dropping the object due to the friction from the ground.
- Gripper arms should be compliant, 3-D printed, and should not require any complex assembly procedures
The gripper arm is a perpendicular triangle whose dimensions are calculated by applying the geometric constraints using a CAD program. For the gripper to be embedded inside the robot body, as can be seen from Figure 3.9, the vertex of the perpendicular triangle should be coincident with the external diameter of the robot. The gripper arm’s rotational axis, which corresponds to the midpoint of the short perpendicular edge of the triangle, is also at a fixed location. This point ensures that the motors are inside the outer shell of the robot body and the gripper arms do not collide when rotating. Next, the short perpendicular edge should be tangent to the outer shell, and the triangle’s hypotenuse should be tangent to the standoff. This standoff has a known location and is a structural component for the integrity between the robot layers. Finally, the gripper’s action plane is placed as low as possible not to raise the height of the center of mass and to have a stable robot base. Applying all these constraints one can compute the largest triangle allowed to be \( l_{\text{short}} = 20.8[\text{mm}] \) \( l_{\text{long}} = 104.9[\text{mm}] \).

Note that Figure 3.9 shows the angle between hypotenuse and long perpendicular edge for readability.

Figure 3.9: a) Exploded view of printed in place gripper arm b) Numerical calculation of gripper dimensions by applying geometrical constraints on a CAD program
After determining the size of the gripper arm and selecting servo motors for the minimum size, we can calculate the maximum weight which the system can elevate. The offset of the gripper arms from the axis of rotation for the elevation is $r_{\text{grip}} = 13.9 \rightarrow 14[\text{mm}]$ and total mass of the two gripper arm assemblies with two low-cost servo motors is $m_{\text{grip}} = 48.66 \rightarrow 50[\text{g}]$. Maximum torque of elevation motor which is identical to grip motors is $T_{\text{max}} = 0.176[\text{Nm}]$. Lastly the length of total grip surface is $l_{\text{grip}} = 85[\text{mm}]$ and its closest point to the elevation axis has an offset $r_{\text{off}} = 26[\text{mm}]$. Then, one can calculate the maximum mass that can be elevated using:

$$T_{\text{max}} = r_{0}m_{0}g + r_{\text{obj}}m_{\text{obj}}g$$  \hspace{1cm} (3.9) \\
$$T_{\text{eff}} = T_{\text{max}} - r_{0}m_{0}g$$  \hspace{1cm} (3.10) \\
$$m_{\text{obj}} = 251[\text{g}]$$  \hspace{1cm} (3.11)

This value corresponds to a theoretical maximum and is used only as an initial design calculation. Considering Kobot-W’s mass is $550[\text{g}]$, the maximum $251[\text{g}]$ is satisfactory. We should also verify that the robot will not tip-over when the gripped object is heavy. During a tip-over, the front caster wheel will be the fulcrum point, and one has to perform torque equilibrium for the caster’s contact point. The lumped masses and their relative positions from the front caster wheel contact point are obtained from the CAD model. Mass of the elevation assembly is $m_{\text{el}} = 32[\text{g}]$ and the center of mass offset is $r_{\text{el}} = -3[\text{mm}]$ (- by direction convention). Mass of the robot’s body excluding the gripper assembly is $m_{\text{robot}} = 550[\text{g}]$, and its offset is $r_{\text{robot}} = -45[\text{mm}]$. Next, the torque at the contact point of the caster wheel is:

$$T_{\text{caster}} = (-r_{\text{el}}m_{\text{el}} - r_{\text{robot}}m_{\text{robot}} + r_{\text{grip}}m_{\text{grip}} + r_{\text{obj}}m_{\text{obj max}})g$$  \hspace{1cm} (3.12) \\
$$= -0.085[\text{Nm}]$$  \hspace{1cm} (3.13)

By our definition of positive torque direction, $T_{\text{caster}} < 0$ shows that robot will not tip-over and will have contact with the ground at the drive wheels. Further when the object is elevated the offset distance $r_{\text{obj}}$ will become smaller and traction at drive wheels will increase.
Off-the-shelf Systems

The CoRe extends functionality by advantaging various stand-alone OTS products:

- A 9-axis inertial measurement unit (IMU) consisting 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer to calculate the heading of the robots with respect to a common orientation reference (magnetic north). IMU communicates with the RPi using the I2C bus.

- A digital camera for onboard image processing applications. To minimize central processing unit (CPU) load, 640x480 resolution, 5 frames per second (480p5) is preferred even the hardware can stream 1080p30 or 480p90. The camera module communicates using the custom camera port of RPi.

- Flexible LED ring containing addressable red, green, blue (RGB) LEDs to debug sensor readings and events related to the swarm experiments. It is interfaced using a pulse width modulation (PWM) pin of the RPi.

- Xbee radio frequency module for direct communication between robots. Xbee enables the deployment of various low-power wireless communication protocols. The network topology of Xbees is software reconfigurable, and they interface with RPi via UART.

- An E-ink module dynamically displays fiducial markers with embedded pose and id information. The module can be integrated on Kobot robots in tangential and vertical configurations. Prior configuration is for experiments where robots operate on the same plane and communicate with e-ink displays through the camera. The latter configuration is for aerial robots to perform pose estimation from ground robots or vice versa. RPi interfaces with the module using SPI.

Communication Interfaces

The CoRe is prominent in communication capabilities (see Fig. 3.10), which is essential for emerging collective behavior from decentralized control of autonomous individuals. In addition to high-level communication interfaces WiFi and bluetooth (BT)
that RPi provides, other independent direct and indirect communication media are available in the CoRe, as shown in Figure 3.10. Unlike other robotics fields, swarm robotics embrace low bandwidth, low range communication, which is bio-inspired and solves the problem of carrier congestion without any computational overhead. For that reason, despite having much more capable WiFi and BT, the CoRe advantages an IR local communication system and Xbee module.

Figure 3.10: Indirect inter-robot communication interfaces of the CoRe

Xbee module is the primary direct communication medium between robots of Kobot platform since it is the middle-ground between WiFi and IR communication. IR local communication module, on the other hand, is interfaced to ToF based R&B system when testing the algorithms developed for the absolute minimum range and bandwidth. Lastly, WiFi is only used to transfer log files and deploy the latest software remotely.

As indirect, local communication, Kobot exploits environmental landmarks by either detecting cues by its floor sensors or using computer vision to detect landmarks available in the environment abstracted as ArUco [56, 57] markers. Over and above that, Kobot exploits signal collision from IR, ToF sensors, or detection of landmarks placed on other robots to communicate R&B information indirectly. In addition, onboard camera and E-ink displays (see Figure 1.1) open up many possibilities for indirect
communication between robots by sending information embedded in markers.

### 3.1.3 Locomotion & Power Layer

#### Wheeled Locomotion

Kobot-W uses two motors equipped with magnetic encoders (see Table 3.1), and their speed is controlled at 100 Hz by a dedicated MCU in a closed loop fashion. RPi sends reference values and requests measured values at 25 Hz. Selected motors are relatively high power to increase expandability and controllability and minimize undesired dynamics. It is also possible to electronically control drive systems with up to four brushed direct current (DC) motors to extend the ground robot-base for alternative locomotion strategies such as omni-wheels, meccanum-wheels, or 4x4 Ackermann’s steering.

<table>
<thead>
<tr>
<th>Voltage (V)</th>
<th>Min. Current (A)</th>
<th>Max. Current (A)</th>
<th>Stall torque (Nm)</th>
<th>Max. Velocity (RPM)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.1</td>
<td>1.6</td>
<td>0.166</td>
<td>310</td>
<td>1:100</td>
</tr>
</tbody>
</table>

Motor parameters obtained from the datasheet of the motor are in Table 3.1. Magnetic encoder coupled to the motor shaft reads 12 counts per revolution $c_{enc} = 12$. Using the gear ratio $n = 100$ one can calculate encoder counts per revolution for the wheel as $c_{wheels} = nc_{enc}$ then using one full rotation corresponds to $2\pi$, encoder’s resolution for the wheel is found as $\delta_{enc} = 2\pi/c_{wheels} = 523.6 \times 10^{-3}[rad]$. However, translational and rotational resolution of the wheel odometry are more useful for a differential drive robot. One can use differential drive kinematic equations to relate rotation of the wheel to rotation and translation of the robot body by using the forward kinematic equation [58]:

$$
\zeta_{robot} = \begin{bmatrix}
\dot{x}_r \\
\dot{y}_r \\
\dot{\theta}_r
\end{bmatrix} = \begin{bmatrix}
\frac{r_r \dot{\theta}_r + r_l \dot{\theta}_l}{2} \\
0 \\
\frac{r_r \dot{\theta}_r - r_l \dot{\theta}_l}{2d_{wb}}
\end{bmatrix}
$$

(3.14)

\footnote{Retrieved from page 31 of https://www.pololu.com/file/0J1487/pololu-micro-metal-gearmotors-rev-4-2.pdf}
By the CAD model of the drive system, radii of the wheels are \( r = r_r = r_l = 19\,[mm] \) and wheelbase is \( d_{wb} = 105\,[mm] \). If one can match the frequency of the encoder pulse with interrupt pins, then for an encoder tick, pure translation of the robot (when the wheels are turning in the same direction) will be:

\[
\delta x_r = \frac{r(\delta_{enc} + \delta_{enc})}{2} = 0.099\,[mm] \tag{3.15}
\]

and its pure rotation (when the wheels are rotating in opposite directions) will be:

\[
\delta \theta_r = \frac{r(\delta_{enc} - (-\delta_{enc}))}{2d_{wb}} = 947.5 \times 10^{-6}\,[rad] \tag{3.17}
\]

\[
= 0.054\,[deg] \tag{3.19}
\]

We should note that these values are theoretical maximum and not possible to attain in reality due to imperfections like backlash in gearing and clearances of the drive elements.

Using once again Eqn. 3.14 and substituting maximum velocity of the motors by converting [RPM] to [rad/s], maximum translational \( \dot{x}_{r_{max}} \) and rotational \( \dot{\theta}_{r_{max}} \) velocities are calculated as:

\[
\dot{x}_{r_{max}} = r \omega_{max} = 616.8\,[mm/s] \tag{3.20}
\]

\[
\dot{\theta}_{r_{max}} = \frac{r \omega_{max}}{d_{wb}} = 5.8\,[rad/s] \tag{3.22}
\]

\[
= 336.6\,[deg/s] \tag{3.24}
\]

\( \dot{x}_{r_{max}} \) and \( \dot{\theta}_{r_{max}} \) would not be possible to obtain in reality since there will be external forces acting on the robot and imperfections, frictions of the drive system. Fortunately, these velocities are much higher than required. Thus, we will limit the velocity of the robots to \( \dot{x}_{r_{max}} = 300\,[mm/s] \) which corresponds to velocity of the motor at maximum power point \( \omega_{power_{max}} = 150.8\,[RPM] \). From datasheet, it is known that torque at max efficiency point is \( T_{eff_{max}} = 0.028\,[Nm] \) and one can calculate the maximum acceleration of the robot using this value for efficient operation of the
motors:

\[ F_{\text{wheel}} = T_{\text{max}}/r \]  \hspace{1cm} (3.25)

\[ a_{\text{max}} = 2F_{\text{wheel}}/m \] \hspace{1cm} (3.26)

\[ a_{\text{eff max}} = 5.35[m/s^2] \] \hspace{1cm} (3.27)

Torque at the maximum efficiency of the motor would take 55\([ms]\) to reach the maximum velocity of the robot. In other words, even with acceleration limited by efficiency, motors are powerful enough to consider acceleration as instantaneous. Accordingly, there is no need to model the robot’s dynamics for practical purposes.

**Flying Locomotion**

In Kobot-F, a commercial quadrotor Ryze Tello \[31\] is used as the flying base. Tello interfaces only with the RPi to receive the reference velocity command and feedback internal IMU readings. For the height control, one can use the drone’s embedded altitude sensor or one of the ToF sensors of the R&B system, placed vertically.

### 3.2 Mechanical Design

Fused deposition modeling 3-D printing technology is the most commonly used method for accessible 3-D printing and fabricates the only custom mechanical parts of Kobot. These parts (see Fig. \[3.11\]) enclose and provide fastening interfaces for the hardware and sometimes be functional parts such as gripper arm, coupler for the wheel rim, and idler wheels. As the fastening interfaces, we preferred embedding metric nuts in printed parts instead of bolts to open up threads or modeling threads directly to the printed parts to provide more durable interfaces that endure numerous assembly cycles. For the structural integrity and load-carrying capacity of the Kobot robots, vertical standoffs and horizontally placed planarly stiff PCBs are used. 3-D Printed parts, on the other hand, are lightweight, compliant, and easy to print. All 3-D printed parts are tested by manufacturing on three different printers in various price ranges. By doing so, we ensure that the design is easy to print even with modest printers.

As can be seen in Figure\[3.11\] all the printed parts are modular and connected with the
Figure 3.11: Exploded views of Kobot-W and Kobot-F
standard interfaces. As a case study, if a research team wants to use a different motor for their operation, they should only change the motor holder part, and they do not have to re-design or re-print the base part. The new design is easy to integrate by using the fastening holes provided in the base. On top of that, all parts are stacked vertically in layers. This way, new layers or unnecessary layers for a particular scenario can be integrated or removed using a pair of brass standoffs. To compensate for instability resulting from adding weight on top with additional layers, we lowered the center of gravity by placing motors and the battery as low as possible.

3.3 Extended Robots

3.3.1 Kobot-W

Kobot-W is a differential drive mobile swarm robot with various resources in cognition and perception. Moreover, it is more straightforward to operate and provides a longer operation when compared to Kobot-F. Due to these attributes, it acts as the testbed of Kobot platform. All the developed systems are initially tested on Kobot-W (see Fig. 3.12) both individually and in a swarm setting. After fine-tuning the performance, systems are commonly used across all the robots of the platform. We built ten Kobot-W robots but used six of them for swarm experiments. The rest are used for testing and debugging purposes.

The base design of Kobot-W uses a minor portion of the peripherals available in RPi and Arduino. Accordingly, unused peripherals are left for researchers to use depending on their specific needs. These peripherals use de-facto connectors and are listed in Table 3.2.

Lastly, the manufacturing and component cost of the proposed architecture has to be taken into account. The OTS component cost of Kobot-W is > 95% of the overall cost, and all of the high-cost components used are generic and easily detachable. Detachable components enable using the components interchangeably for different purposes. Hence, component purchases can be interpreted as inventory investments for the laboratory. Table 3.3 lists manufacturing and component costs.
Figure 3.12: Pictures of a Kobot-W used for development of the CoRe systems

Table 3.2: Available peripherals of Kobot-W for extensions

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Raspberry Pi (3.3 V)</th>
<th>MCU (5 V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>I2C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SPI</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Analog Inputs</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>Digital Input/Outputs</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Maximum Output Current</td>
<td>500 [mA]</td>
<td>1000 [mA]</td>
</tr>
</tbody>
</table>

3.3.2 Kobot-F

Kobot-F, the flying robot of the Kobot platform, is the first of its kind and uses only onboard resources. Kobot-F relies solely upon the software and hardware systems that were previously tested and proved to be fully functional in a swarm scenario.
Table 3.3: Component costs for Kobot-W. Unit costs given for minimum of 5 Kobots

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage Regulator</td>
<td>8 $</td>
</tr>
<tr>
<td>Li-Po Battery</td>
<td>12 $</td>
</tr>
<tr>
<td>Raspberry Pi 3B+</td>
<td>35 $</td>
</tr>
<tr>
<td>Raspberry Pi Camera v2.1</td>
<td>25 $</td>
</tr>
<tr>
<td>SD Card 16 GB</td>
<td>6.6$</td>
</tr>
<tr>
<td>Arduino Nano x2</td>
<td>6 $</td>
</tr>
<tr>
<td>Xbee S3</td>
<td>18 $</td>
</tr>
<tr>
<td>Motors &amp; Encoders x2</td>
<td>36 $</td>
</tr>
<tr>
<td>PCB Manufacturing &amp; Assembly</td>
<td>50 $</td>
</tr>
<tr>
<td>Electronic Components</td>
<td>35 $</td>
</tr>
<tr>
<td>Mechanical Components</td>
<td>5 $</td>
</tr>
<tr>
<td>3-D Printed Parts</td>
<td>3.5$</td>
</tr>
<tr>
<td><strong>Total Cost</strong></td>
<td><strong>200 $</strong></td>
</tr>
</tbody>
</table>

with Kobot-W. This way, the development phase is accelerated and does not result in collisions which can damage the aircraft. Moreover, Kobot-F can move at speeds an order of magnitude higher than the Kobot-W and has the unique ability to observe a scene from the top at a wider angle using its camera to guide the ground robots for a given task.

Kobot-F has a highly modular structure and does not use any fasteners. Instead, the body’s integrity relies on snap-fit 3-D printed parts. As a result, Kobot-F can be completely assembled and disassembled as shown in Figure 3.13 in a few minutes without any extra tools.

### 3.3.3 Alternative Locomotion Prototypes

The CoRe abstracts locomotion from the swarm operation and enables the same implementation to work with different locomotion strategies of Kobot-F and Kobot-W. Similar progress is possible with alternative locomotion strategies such as holonomic
ground robots and quadruped swarm robots. Figure 3.14 shows two such prototypes using systems of the CoRe.

(a) Kobot-Q (Quadruped)  (b) Kobot-O (Omni-wheel)

Figure 3.14: Prototype level Kobot robots with alternative locomotion systems
The Kobot platform software abstracts the systems in layers as presented in Figure 4.1. Layers of the architecture are not directly connected to the other layers and can only interface with the one above and the one below. This way, developers can work independently of the hardware and be concerned only by the input-output relationships of the systems requiring modification. Therefore, rather than describing how the software tools work, which are documented in the source files, the subsections corresponding to the layers in Figure 4.1 explain how we leveraged the open-source software tools to solve challenges related to swarm robots.

4.1 Swarm Supervision

Operating a swarm task is challenging for the researchers. We addressed this issue in the CoRe by using a unicast network between robots and the ‘base computer’. Then, custom shell scripts automate the repetitive tasks for all swarm members. Unicast network is exclusively for supervising the overall operation of the swarm. This network is independent of local inter-robot communication and is not a part of the swarm scenario.

Swarm operation control commands include selecting the scenario to be performed, starting, ending the experiment, and changing the values of dynamic parameters. When the base computer needs to communicate with the swarm, it connects to the first robot using a secure shell and publishes to a topic or performs write or read operations to the parameter server. After finishing the procedure, the shell script moves to the next robot, defined by a file including the robot hosts. Moreover, this network
Figure 4.1: Flow diagram showing the software CoRe can transfer logged data from robots to the base computer or parameter files from the base computer to the robots.

The second type of high-level network the CoRe use is a multi-master network. A particular node can run on any computer or robot available on the network, and selected topics or parameters can be synchronized. In this configuration, the base computer visualizes data in real-time or synchronizes the local data of the robots with the ground truth available from the tracking system of the experimental setup. It is up to the researcher to use this type of network as an alternative inter-robot communication medium and a component of the swarm scenario.

Since the CoRe use a single board computer RPi running a full-size operating system, it has access to a file system and can log local experiment data. This data can be used later to reconstruct the experiment. Log files allow working offline and simplify debugging the errors or modeling the hardware. Furthermore, experiments can
continue indefinitely by storing the final state of the robot in the non-volatile memory and resuming the experiment from this state even after a power-off. Ultimately, this software-based solution addresses the issues related to the battery capacity that arise when the required experiment time is more than a single battery would allow.

4.2 Behavioral Programming

The CoRe software’s goal is to distribute the inherent complexity of an application away from the behavior. The rationale is that swarm researchers are only interested in developing and testing the behavior, and everything else is just a building block. Due to this, the CoRe uses a single high-level node to program the behavior. This node solely controls the behavior, and it is not directly linked to the hardware. Instead, standard ROS messages and hardware driver nodes abstract the hardware. High-level nodes subscribe to topics published by the hardware nodes, perform the computation and control the behavior by publishing to topics subscribed once again by the hardware nodes. Figure 4.2 outlines the modular CoRe.

Before running a behavior node, the CoRe uses launch files to prepare the robot for a particular scenario. A launch file starts all the required hardware and third-party nodes and loads hardware calibrations and parameters to the parameter server. Subsequently, if a hardware node requires calibration, it checks the parameter server and starts its operation after parameter loading. On the other hand, researchers can fine-tune dynamic parameters by observing the result from the actual process.

4.2.1 Self-organized Flocking

A conventional decentralized algorithm, self-organized flocking (see Algorithm 1), is the first swarm behavior implemented in the CoRe.

For a flocking behavior, robots need range and bearing information from nearby objects, classified as robots or obstacles by the R&B system. If the object is classified as an obstacle, then the robot should avoid it by acting as if a virtual force pushes it

\[ \text{euc}(\vec{v}) \text{: Euclidian norm of the } \vec{v} \]
Figure 4.2: Flow diagram showing the software CoRe modularity. Dashed boxes lists alternative mobile bases and extensions for different robots using the CoRe. The symbol ‘/’ separates alternative modules. For mobile base the first alternative is Kobot-W, together with ‘diff_driver’, ‘kobotMotorController’ and the second alternative is Kobot-F with the respective modules. For extensions the first alternative is ‘Floor Sensor’ using ‘smbus2’ and the other alternatives are gripper, e-ink display and led ring using corresponding modules.
Algorithm 1: Self-Organized Flocking [1]

Data:

- ranges: obstacle range and robot range values from the R&B sensor
- headings: headings of the nearby robots from local communication

while true do

  /* Resultant virtual force vector */
  \( \vec{p} = \text{get\_vf}(\text{ranges}); \)

  /* Resultant virtual heading vector */
  \( \vec{h} = \text{get\_vh}(\text{headings}); \)

  /* Desired heading vector */
  \( \vec{a} = \vec{h} + \beta \vec{p}; \)
  \( e_h = \tan^{-1}(\vec{h}c) - \tan^{-1}(\vec{a}c); \)
  \( \omega = K_p * e_h; \)

  if abs\( (e_h < \pi/2) \) then
    \[ u = (\vec{a} \cdot \vec{a}c)^\gamma u_{max}; \]
  else
    \[ u = 0; \]
  go\( (u, \omega); \)

away from the obstacle. The algorithm handles this by setting a desired distance variable for obstacles \( \sigma_{des,obs} \) equal to the maximum range of the sensor \( \sigma_{des,obs} = r_{max} \). If the detected object is classified as a robot by the R&B system, then the robot should behave as if it is connected to the sensed robot by a virtual spring with an equilibrium length of \( \sigma_{des,rob} \in (0, r_{max}) \). In other words, the virtual force between two robots should ensure collision avoidance and inter-robot coherence at the same time.

Self-organized flocking implementation of the CoRe is first developed using Kobot-W robots and IR R&B system. For the IR R&B system, calculated virtual force \( f_{kr} \) exerted to the robot by nearby objects is proportional to the square of the deviation from the desired distance. Because the IR R&B system is noisy, low resolution, and reports non-linear distance readings. The virtual force is calculated by the formula
Table 4.1: Systems used and tested on a swarm setting by the self-organized flocking algorithm

<table>
<thead>
<tr>
<th></th>
<th>Motion Control</th>
<th>R&amp;B</th>
<th>Comm.</th>
<th>IMU</th>
<th>Odometry</th>
<th>Floor Sensing</th>
<th>Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

given below:

\[
f_{ki} = \frac{\text{sgn}(\sigma, \sigma_{\text{des}})(\sigma - \sigma_{\text{des}})^2}{C}
\] (4.1)

where the \( \sigma \) is the distance read by the sensor and \( \sigma_{\text{des}} \) is the desired distance to the object and \( C \) is the scaling constant used for normalizing the virtual force \( f_k \) to \([-1, 1]\). Note that, \( \sigma_{\text{des}} \) have different values for obstacles and robots. As a result \( C \) will have different values for obstacles and robots to ensure the \( f_k \) is in \([-1, 1]\) for different \( \sigma_{\text{des}} \) values. Sign function \( \text{sgn}(\sigma, \sigma_{\text{des}}) \) determines whether the virtual force pulls or pushes the robot and calculated as:

\[
\text{sgn}(\sigma, \sigma_{\text{des}}) = \begin{cases} 
-1 & \sigma \leq \sigma_{\text{des}} \\
+1 & \sigma > \sigma_{\text{des}} 
\end{cases}
\] (4.2)

Note that, for all sensor readings classified as obstacles \( \sigma \leq \sigma_{\text{des,obs}} = r_{\text{max}} \) and the sign of the \( f_k \) is \(-\) which corresponds to a push.

Next, we expanded this implementation to the Kobot-W robots using the ToF R&B system. ToF R&B has a higher resolution with a more linear response. Furthermore, since the algorithm dictates that \( f_k \in [-1, 1] \), we used a linear force generation, which results in a more responsive control action around the equilibrium point. Thus, the virtual force for the ToF R&B system is calculated by:

\[
f_{k_{\text{tof}}} = \frac{(\sigma - \sigma_{\text{des}})}{C}
\] (4.3)

which enabled successful flocking behavior with ToF R&B and Kobot-W swarm and does not require an explicit sign function. However \( C \) and \( \sigma_{\text{des}} \) has different values for the obstacles and robots again. These values are also different from the ones used with IR R&B. By the ToF sensor’s maximum range, the desired distance for obstacles is \( \sigma_{\text{des,obs}} = r_{\text{max}} = 1300[mm] \) and we set the desired inter-robot distance to \( \sigma_{\text{des,rob}} = r_{\text{max}}/2 = 650[mm] \). Then, \( C_{\text{obs}} \) needs to map \( f_k \) by Eqn. 4.3 to \([-1, 1]\).
and calculated by substituting $f_k = -1$ and $\sigma = 0$:

$$C_{\text{obs}} = \sigma_{\text{des}} - \sigma \quad (4.4)$$

$$C_{\text{obs}} = 1300. \text{ Using the same formula for } f_k = 1 \text{ and } \sigma = 1300, \text{ one can find } C_{\text{rob}} = 650. \text{ Note that, if } \sigma_{\text{des}} > r_{\text{max}}/2, \text{ one must use } \sigma = 0 \text{ and } f_k = -1 \text{ for calculating } C_{\text{rob}}.$$

To use the same implementation with Kobot-Fs, which can move in 3-D space, we implemented a closed-loop height control to restrict the motion of the drones in 2-D. Additionally, we used different parameters for the virtual force calculation since MAVs create an airflow that disturbs nearby MAVs. Hence, the Kobot-F flock uses a higher inter-robot distance than the ground robots. The virtual forces are calculated by the Eqn. 4.1 and 4.3 for the ranges of the IR and ToF R&B systems are presented in the Figure 4.3

![Figure 4.3: Virtual forces of obstacles and robots for the IR and ToF R&B systems](image)

Finally, resultant force $\vec{p}$ exerted on robot can be calculated by:

$$\vec{p} = \frac{1}{N} \left| \sum_{k=1}^{N} f_k e^{j\phi_k} \right|$$

(4.6)
where $\phi_k = 2\pi/N$ and $N = 8$ for IR R&B, whereas $N = 12$ for ToF R&B system.

In addition to virtual force, robots need to sense their heading and communicate it with neighbors. We use IMU to compute the heading of the robots and rely on the magnetometer for all headings to have the same reference (magnetic-north).

Additional implementation preferences of the flocking algorithm are as follows:

- The positive direction for the rotation is counter-clockwise and a robot’s heading always hold the condition $\vec{h}_i \in [-\pi, \pi)$ using the relation $\vec{h}_i = \vec{h}_i \pm 2\pi$.

- The interval $[-\pi, \pi)$ contains a discontinuity and result in problems for the virtual heading $\vec{h}$ calculations. In order to eliminate this problem each robot first convert heading of neighbor robots holding the condition:

$$\text{abs}(\angle \vec{h}_i - \angle \vec{h}_j) \leq \pi$$

by once again using the relation $\vec{h}_i = \vec{h}_i \pm 2\pi$. Then, one can calculate $\vec{h}$ ensuring the robots turn coherently as a flock.

### 4.2.2 Cue-based Aggregation

Two different cue-based aggregation algorithms are implemented in Kobot-W. These algorithms exploit different hardware and software capabilities of the CoRe that the flocking experiments could not test.

**BEECLUST**

BEECLUST algorithm is implemented (see Algorithm 2) as the first aggregation behavior and uses the CoRe systems listed in Table 4.2.

Additional implementation preferences of BEECLUST are as follows:

- Angle $\theta$ guarantees to be obstacle free on the sensing distance of the R&B system,

$$\theta = \angle \vec{p}$$

(4.8)
Algorithm 2: BEECLUST

**Data:**

$I_c$: Intensity from floor sensors

**while** true **do**

    $go\_straight(u_{max});$

    **if** obstacle_detected **then**

        $\theta = \phi_{obs} + \text{random}(\pi/2, \pi + \pi/2);$  

        $\text{turn}(\theta, \omega_{max});$

    **if** robot_detected **then**

        **if** $I > 0$ **then**

            $w = \frac{I}{I + I_c}w_{max};$

            $\text{wait}(w);$

---

Table 4.2: Systems used and tested on a swarm setting by the BEECLUST algorithm

<table>
<thead>
<tr>
<th>Motion Control</th>
<th>R&amp;B</th>
<th>Comm.</th>
<th>IMU</th>
<th>Odometry</th>
<th>Floor Sensing</th>
<th>Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

where $\vec{p}$ can be calculated by the Eqn. 4.6 or if not implemented one can use

$$\theta = \angle \min(f_i, \ldots, f_n) \tag{4.9}$$

which corresponds to the angle of the sensor which detected the nearest object. Note that the interval $[\pi, \pi + \pi/2]$ is used to emphasize $0$ is not in the interval

- when the blocking $\text{wait}(w)$ terminates, the robot performs an exit maneuver by turning randomly and for a predefined time neglects robot detection by all the sensors, excluding the sensor aligned with the translation axis of the robot. The front sensor is neglected for not colliding with other robots, and the exit maneuver is performed to avoid getting trapped inside the cue.
Landmark Based Aggregation

A novel algorithm landmark-based aggregation (LBA) is implemented (see Algorithm 3) in the CoRe. This scenario extends the BEECLUST and aims to converge to a stable aggregation even with a smaller population. Notably, this is the first real robot implementation of the algorithm. Table 4.3 lists the systems tested by LBA on a swarm setting.

Algorithm 3: Landmark Based Aggregation [2]

Data:

$I_c$: Intensity from floor sensors
$L_i$: Detected landmark id

while true do

...;
if robot_detected then

if $I > 0$ then

...;
if $L_i \neq \{\}$ then

store_landmark2cue($L_i$);

if landmark_detected then

if $L_i \in \hat{L}$ then

go2cue($L_i$);
else

record_odometry($L_i$);

Table 4.3: Systems used and tested on a swarm setting by the LBA algorithm

<table>
<thead>
<tr>
<th>Motion Control</th>
<th>R&amp;B</th>
<th>Comm.</th>
<th>IMU</th>
<th>Odometry</th>
<th>Floor Sensing</th>
<th>Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos., Vel.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Kobot-W uses its onboard camera and computation for landmark detection and the pose estimation for a successful LBA experiment. ArUco markers denote the landmarks.
Additional implementation preferences of the LBA algorithm are as follows:

- 2-D vectors from landmark to the cue are stored in a dictionary and indexed with the landmark id corresponding to the ArUco marker id:

\[ D[i] = \{ 1 : L_1, \ldots, n : L_n \} \]  

where

\[ L_i = [r_{ic}, \theta_{ic}, \tau_i] \]

- \( \tau_i \) is number of remaining trials and used for adapting to dynamic environments. When robot reach cue from a certain landmark, it stores the landmark in \( D[i] \) as \( \tau_i = \epsilon \). Whenever robot uses \( L_i \) and becomes unsuccessful in reaching the cue \( \tau_i \rightarrow (\tau_i - 1) \) and \( L_i \) is removed from \( D[i] \) when \( \tau_i = 0 \)

- If a robot encounters another robot when headed but outside of the cue from \( L_i \), \( \tau_i \) remains unchanged

- If robot encounters another robot when headed and inside the cue from \( L_i \), \( \tau_i = \epsilon \) and \( L_i \) is updated with the new \( [r_{ic}, \theta_{ic}, \tau_i] \)

### 4.3 Hardware Drivers

In the Kobot ROS package, all hardware has its specific node, which acts as a driver layer between the hardware and RPi for a modular code-base. Driver nodes handle communication between the hardware and RPi using digital communication protocols and publish standardized messages at a fixed rate to the corresponding topics. However, some hardware needs calibration, filtering, or sensor fusion. In general, ROS handles these tasks, visualization, and simulation by relying on standard messages. Unfortunately, some frequently used hardware in swarm robotics does not have pre-defined standard message types in ROS. As a result, we created our set of standard messages.

All peripheral hardware of the CoRe run embedded code, and they are connected to the RPi by a dedicated communication interface. The RPi is the master in com-
munication with the peripherals and dictates their operation by changing parameters instead of re-programming the embedded firmware.

An additional task of the hardware nodes is to perform the required hardware calibration procedures. The calibration procedure follows a parameter file generation, loaded to the parameter server at the runtime. The version control system ignores these calibration files since they are robot-specific and stored locally for each robot.

The CoRe systems can calibrate their parameters using global information as well. As a case study, kinematic calibration is performed for fine-tuning the wheel radii and wheelbase to reduce the odometry error. This process requires ground truth information and computation by the base computer and will be discussed further in Chapter 5.

4.3.1 Floor Sensing

The floor sensor system needs calibration. The calibration is used to measure the sensors’ bias values to neglect intensity levels below a certain threshold. Any surface with reflectivity below the threshold will be assumed to have a zero reading and act as free space. Conversely, a surface with higher reflectivity than the threshold would be considered an aggregation cue. The corresponding reflectivity denotes the quality of that point. First, the bias is applied, then the active region of the sensor is mapped to 8-bit values. The below formula is used to map 12-bit readings to 8-bit where $I_{k_{min}}, I_{k_{max}}$ are 12-bit values obtained from the calibration procedure:

$$I_k = \frac{I_{k_{12}} - I_{k_{min}}}{I_{k_{max}} - I_{k_{min}}} 2^8$$

(4.12)

Aggregation algorithms that implemented in the CoRe expects a single intensity value. For that reason one can feed the maximum intensity reading, or average of the readings to the algorithm.

Moreover, having multiple sensors enables calculating the intensity gradient. One can use below formula to calculate gradient vector of the obtained floor intensity.

$$\vec{I}_g = \frac{1}{N} \sum_{k=1}^{N} (I_k) e^{i\phi_k}$$

(4.13)

where $\phi_k = (2\pi/N + \phi_{off})$, $N = 4$ and $\phi_{off} = \pi/4$ by the CAD model.

42
4.3.2 Localization

Inertial Measurement Unit

A third-party python package communicates with IMU using the I2C bus and publishes readings as a ROS message. First of all, accelerometer, gyroscope, and magnetometer calibrated for gravity bias, gyroscope drift, and hard, soft iron effects, respectively. Then, Madgwick filter [59] implementation in ROS is used for filtering IMU messages and combining readings of the sensors to a transformation from a fixed frame to the IMU frame. Then it is possible to visualize the robot’s attitude using visualization tools of ROS (RViz) or to feed this pose estimation to a subsequent node for sensor fusion such as [60].

Wheel Odometry

Kobot-W calculate wheel odometry using the measured and filtered velocity at 100 Hz and collected by RPi at 25 Hz. The input to the node computing the wheel odometry is individual wheel velocities from the digital velocity controller. Then the twist of the robot \([\dot{x}_r, \dot{y}_r, \dot{\theta}_r]\) can be computed by forward kinematic Eqn. 3.14. However, the twist of the robot needs to be integrated to compute the robot’s pose \([x, y, \theta]\) with respect to a fixed reference frame. Note that components of robot twist are in the body-fixed frame, whereas the components of the robot’s pose are resolved in a fixed frame. Using orthogonal rotation matrix which is dependent on the pose of the robot \(R(\theta)\):

\[
R(\theta) = \begin{bmatrix}
\cos(\theta) & \sin(\theta) & 0 \\
-\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (4.14)

Twist resolved on the robot frame can be transformed to the fixed frame by using:

\[
\dot{\zeta}_{ff} = R(\theta)^{-1} \dot{\zeta}_{bf}
\]  \hspace{1cm} (4.15)
\[
\begin{bmatrix}
\dot{x}_{\text{robot}_{ff}} \\
\dot{y}_{\text{robot}_{ff}} \\
\dot{\theta}_{\text{robot}_{ff}}
\end{bmatrix} = 
\begin{bmatrix}
cos(\theta) & -sin(\theta) & 0 \\
sin(\theta) & cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\dot{x}_{\text{robot}_{ff}} \\
\dot{y}_{\text{robot}_{ff}} \\
\dot{\theta}_{\text{robot}_{ff}}
\end{bmatrix}
\]

(4.16)

Next, one can integrate twist resolved in fixed frame to obtain pose of the robot body resolved in the fixed frame using the below equations:

\[
x_{\text{robot}_{ff}} = x_{0_{ff}} + \dot{x}_{\text{robot}_{ff}} \ast dt
\]

(4.17)

\[
y_{\text{robot}_{ff}} = y_{0_{ff}} + \dot{y}_{\text{robot}_{ff}} \ast dt
\]

(4.18)

\[
\theta_{\text{robot}_{ff}} = \theta_{0_{ff}} + \dot{\theta}_{\text{robot}_{ff}} \ast dt
\]

(4.19)

where \(dt = 1/25 = 40ms\).

**Marker Based Visual Localization**

Kobot robots perform visual localization using OpenCV, onboard camera, and fiducial markers. From a general point of view, all three are used together to solve the Perspective-n-Point (PnP) problem. Fiducial markers are objects having 3-D points with a known configuration. Then, if one can identify these 3-D points in the 2-D image of the camera, the rotation, and translation, minimizing the re-projection error, corresponding to the pose of the marker resolved in the camera frame. QR codes, colored markers, and ArUco[57] markers can all be used for solving the PnP problem. However, we preferred ArUco markers since they are computationally efficient to detect using the available OpenCV library2 and embed ID information.

The output of the marker detection and pose estimation are the ID of the ArUco marker, \(tvec\) corresponding to x,y,z distances of the marker resolved in the camera frame and minimal representation of 3-D rotation denoted as \(rvec\). Using \(tvec\) and \(rvec\) Kobot robots can estimate their pose with respect to a marker available in the environment. Alternatively, Kobot-F can estimate its pose with respect to a Kobot-W on the ground equipped with a marker as depicted in Figure 4.4.

---

2https://docs.opencv.org/4.x/d9/d6a/group__aruco.html
Figure 4.4: Kobot-F tracking Kobot-W robot by performing pose estimation using its onboard camera and ArUco marker on Kobot-W

4.4 Embedded Firmware

The CoRe’s embedded firmware is programmed using C++ and Arduino framework.

4.4.1 Differential Driver

Differential driver firmware of the CoRe leverages an object-oriented programming (OOP) paradigm and defines a motor class. For instance, right, and left motors of a differential drive robot or additional motors for alternative actuation systems can be represented as objects from this class.

Functionality of *kobotMotor* class is as follows:

- Timer based 100 Hz control loop
- Read/Write operations to the controller registers (see Tab. 4.4) using I2C
- Open Loop (OL) velocity control using a digital proportional-integral-derivative (PID) controller
- Closed Loop (CL) velocity control using a digital PID
- Closed Loop (CL) position control using a digital PID
Table 4.4: Register map of the *kobotMotor* class. R/W denotes read and write operations by the I2C master

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Register</th>
<th>Type</th>
<th>Default Value</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Mode</td>
<td>255</td>
<td>uint8</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>$K_p$</td>
<td>255</td>
<td>Float32</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>$K_i$</td>
<td>254</td>
<td>Float32</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>$K_d$</td>
<td>253</td>
<td>Float32</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>$f$</td>
<td>252</td>
<td>uint8</td>
<td>100</td>
<td>R/W</td>
</tr>
<tr>
<td>Reference value</td>
<td>251</td>
<td>int16</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>Measured value</td>
<td>250</td>
<td>int16</td>
<td>-</td>
<td>R</td>
</tr>
<tr>
<td>Battery Voltage</td>
<td>249</td>
<td>uint8</td>
<td>-</td>
<td>R</td>
</tr>
</tbody>
</table>

I2C master RPi can perform read and write operations to the registers given in Table 4.4 to change the controller parameters.

PID controller of the *kobotMotor* use the constant coefficient difference equation (CCDE) for a digital PID controller having the form:

$$u[k] = u[k-1] + b_0 e[k] + b_1 e[k-1] + b_2 e[k-2]$$  \hspace{1cm} (4.21)

At each sampling of the controller, error $e[k]$ and control effort $u[k]$ are shifted backward and become $e[k-1]$, $u[k-1]$ in the next control loop. $b_0$, $b_1$ and $b_2$ constants can be computed using PID gains and sampling frequency of the controller $f$

$$b_0 = k_p + k_d * f + k_i/2f$$ \hspace{1cm} (4.22)
$$b_1 = -k_p - 2 * k_d * f + k_i/2f$$ \hspace{1cm} (4.23)
$$b_2 = k_d * f$$ \hspace{1cm} (4.24)

4.4.2 ToF R&B

The ToF R&B implementation uses an object-oriented programming paradigm, and each ToF sensor available on the board is an object of the *tof* class. *tof* class en-
capsulates VL53L1X application programming interface (API\textsuperscript{3}) which is a platform-agnostic collection of C functions to interface with the sensors. R&B system is a collection of \textit{tof} objects populated to any number of sensors lower than twelve, limited by the hardware interfaces.

We designed the ToF R&B system with its own MCU, which communicates with ToF sensors through its I2C interface as the master. MCU has a second I2C interface independent of the first one, and it is used for communicating with the RPi.

ToF R&B is a stand alone system with its own embedded firmware responsible for:

- configuring sensor parameters
- collecting range readings
- filtering the raw data
- perform robot detection

The system reports its processed data when requested by the I2C master and can work with any other system having $3.3V$ supply and I2C lines.

\section*{Robot Detection}

To the best of our knowledge, the ToF R&B system of the CoRe capable of robot detection is the first of its kind. The robot detection is closely related to the ranging sequence of the sensor. For that reason, first, the ranging sequence of the sensor will be presented.

A single ranging sequence of the sensor is composed of three distinct measurements as shown in Figure \ref{fig:range_sequence}, which are always consecutive and can not be called independently due to the restrictions of the API. The first measurement of the sensor is reported as ‘ambient’, and the receiver end of the sensor collects $940[\text{nm}]$ wavelength IR signals available in the environment. During this measurement emitter of the sensor is inactive. The second measurement sequence is ‘emit 1’, during which

\textsuperscript{3}VL53L1X ULD API is available at \url{https://www.st.com/en/embedded-software/stsw-img009.html}
the emitter of the sensor emits a 940\,[\text{nm}] signal with a specific pattern, and the receiver of the sensor collects the reflected signal. Next, the sensor performs the ‘emit 2’ sequence independent of the first and collects the reflected signal. The ambient reading from the first sequence is used for correcting emit 1 and emit 2 signals and calculating signal to noise (ambient) ratio to decide whether the measured range is valid. Additionally, two independent emit sequences using different signal patterns cope with a sampling problem frequently encountered in radar systems. This problem arises when echoes located beyond the maximum unambiguous range of the system are received as if they were within the valid range. Sensor reports readings from ‘emit 1’ and ‘emit 2’ sequences as not separate readings but as a single ‘signal’ and internally computed range of the object in FoV. Lastly, we integrated an external random delay to the end of a ranging sequence. This random delay takes a fixed period of $t_4$ and ideally has a probability of $1/2$ to occur since it is based on a randomly seeded binary variable. From now on, for conciseness, the ambient will be used for the signal reading when the emitter is inactive, and signal will denote the signal reading when the emitter is active. These readings have the same unit kilo-count per second [kcps].

As performed in Kobot-I’s IR R&B system [14] and many other communication networks using a common carrier, the ambient reading of the ToF sensor act as a ‘carrier sense’. For the ToF sensors, the carrier is 940\,[\text{nm}] IR light, and ‘sense’ means ambient measurement without emitting a signal. Suppose one can ensure that ‘carrier
access’ is only by the emitters of the other robots, which is the case, indoors and outdoors without sunlight. In that case, one can argue signal collision sensed in the carrier is another robot in the sensor’s FoV.

A setup comprising two ToF sensors was configured without a random delay and aligned with intersecting FoVs to test that hypothesis. Next, ambient readings collected from one of the sensors are plotted in Figure 4.6. As it can be seen, the ranging cycle of one sensor is a periodic wave on the other and can be used for robot detection. In addition, simplified distinct ranging sequences shown in Figure 4.5 are also clearly visible in the plot. However, when the ambient sequences of two sensors are in sync, ambient reading drops to zero, resulting in a false-negative robot detection and needs to be addressed.

Figure 4.6: Ambient reading of a ToF sensor facing another ToF sensor

Figure 4.7 shows the purpose of externally introduced random delay and how we ensure the robot detection. The sequences explained in Figure 4.5 are shown in a continuous manner and random delay takes place in ranging cycles of robot\(_1\) and robot\(_2\). Initially, the worst-case scenario is assumed when both sensors run in sync. As can be seen, white regions corresponding to ambient reading are simultaneous, which means neither robot\(_1\) nor robot\(_2\) will detect the emitted signal by the other robot. As a result, no robot detection will be reported. In the second ranging cycle however, due to the random delay of robot\(_1\), emit 1 sequence of the robot\(_2\) is simultaneous with the am-
bient reading of the robot\textsubscript{1}. Thus, robot\textsubscript{1} will read a non-zero ambient and will report this reading as a robot which is denoted by $r_{1,2}$. However, robot\textsubscript{2} will not report robot detection since, during its ambient reading phase, robot\textsubscript{1} was in random delay and was not emitting any signal. Lastly, in the third ranging cycle, for both robot\textsubscript{1} and robot\textsubscript{2}, ambient reading phases coincided with other robot’s emit phases and will report robot detection.

![Figure 4.7: Robot detection of two ToF sensors. Symbol {} denotes no robot detection and symbol $r_{i,j}$ denotes that robot\textsubscript{i} detected robot\textsubscript{j}](image)

In other words, random delay’s purpose is to ensure that any two robots of the swarm can sense one another without synchronization or a time division scheme. Let us assume the worst-case for a basic calculation, in which two robots are perfectly in sync, and their ambient reading phases always collide. If we have no random delay, these two robots will not detect each other under any circumstances. However, when the random delay is present, for a probability of $1/2$, in the next cycle, they will be out of sync and report each other as robots. This $1/2$ probability corresponds to only one of the robots performing a delay. For the remaining $1/2$ probability, both robots will delay or not in the next cycle and will be in sync once again. So, if one samples with a high enough frequency and collects robot readings on a buffer with a large enough size, the robot detection is guaranteed for all practical purposes.

It should be noted that when the sensor is configured for a high sampling frequency, the accuracy and repeatability of the range readings decrease. However, even in the highest sampling rate configuration, which corresponds to $66\,\text{Hz}$, accuracy and re-

\textsuperscript{4}$r_{1,2}$ stands for robot\textsubscript{1} detected robot\textsubscript{2}
peatability should be acceptable for swarm algorithms. Considering these facts and limited by the sensor’s maximum sampling rate of $66\,\text{Hz}$, we constructed a buffer holding the last five readings. Our robots do not move very fast $< 0.5\,\text{m/s}$ and they can not get out of the FoV of the sensor in $66/5 = 13.2\,\text{Hz} \rightarrow 76\,\text{ms}$. For that reason, even if one of the last five readings is a robot, the next reading is assumed as a robot.

**Embedded Filtering**

Configuring sensors for a high sampling rate result in reduced accuracy and repeatability in readings. For that reason, objects of `tof` class holds buffers for ‘ambient’, ‘signal’, ‘range’ and ‘robot detection’ variables. These buffers are used in filtering reported values to the RPi. The length of these buffers is configurable during run-time by the I2C master RPi. The buffers are ‘circular’ with a first-in, first-out (FIFO) functionality. After each ranging sequence of the ToF sensors first element (oldest reading) in the buffer is removed, and the last element (current reading) is added to the buffer. Standard deviation (SDEV) and interquartile range (IQR) are calculated as statistical dispersion metrics for each buffer.

ToF sensors always report a range value after a ranging cycle even if there is no object in the sensor’s maximum range along the FoV. One has to filter those values as erroneous readings. We filter those readings by using range and signal buffers. The basic filtering procedure that we perform to detect such readings are as follows:

- Filter if the range is higher than the maximum range of the sensor’s current configuration
- Filter if the median of the signal buffer is lower than the threshold
- Filter if IQR of range buffer is higher than the threshold

where filtered range values are set to the sensor’s maximum range in the current configuration.

Moreover, using ToF sensors in a swarm setup leads to signal collision during the ranging sequence and erroneous results. However, when the signal quality is high
enough with the retroreflective coating, the sensor still reports valid measurements even if the noise is also high. Still, we report the median of the range buffer as the distance corresponding to the object and not the actual measurement reported by the sensor.

An ambient buffer length of five by default is used to develop an invariant metric to the ambient bias available in the environment but increasing with high-frequency patterns such as those emitted during ranging sequences of the sensors. After experimenting with different metrics, we selected the IQR value of the buffer. Accordingly, if the IQR is higher than the threshold, the object is reported as a robot.

Lastly, we hold a robot detection buffer of length five by default. We assume the detected object is still a robot. Even only a single value in the buffer corresponds to a robot detection. In other words, for an object to be reported as an obstacle, it requires five consecutive, not robot (obstacle) reading from the sensors.
CHAPTER 5

EXPERIMENTS

Initial calculations for the design decisions were performed in previous Chapters 3 and 4. However, it is not feasible to verify all the CoRe systems without an experimental setup providing ground truth pose information. The system-level experiments presented in this Chapter aim to fill that gap. Moreover, Kobot platform is designed for swarm robotics research. For that reason, in addition to testing the CoRe systems with ground truth pose information, Kobot robots need to prove swarm performance in realistic scenarios.

5.1 Experimental Setup

The tracking system collects time-stamped pose information of robots, which is later used as the ground truth for the performance evaluation. The tracking system has
twelve IR pose cameras (Vicon Vantage v16\textsuperscript{1}) placed outside the arena and obtains the pose information from a pre-defined configuration of four passive retroreflective markers put non-planar on Kobot robots. A second and more accessible tracking system is also used for the same purpose, but only for Kobot-W swarm. This system is composed of ArUco markers placed on top of Kobot-Ws and a webcam set at the ceiling. These two tracking systems are entirely interchangeable since both report pose data.

5.2 System Validation

Verifying the critical systems of the proposed design is the first goal of experimentation with Kobot robots. First, data is collected from the CoRe systems and tracking system. Then, the systems’ performance is evaluated by comparing the robot’s local information to the ground truth. We used differential drive Kobot-W robots extended from the Kobot platform whenever possible in system-level experiments. Kobot-W robots are more accessible to collect data due to having a higher experiment duration with a single charge than flying robots Kobot-Fs.

5.2.1 2-DoF Compliant Gripper

We tested the gripper system using a glass cup with a cylindrical bottom diameter of 60.2[mm] and a conical top with a maximum diameter of 74.5[mm]. During tests, the system gripped and tried to elevate the glass cup. The total weight of the object was set by adding bolts inside the cup until the gripper system failed. During these experiments, the maximum current at 5V supply voltage was 1.48A.

Figure\textsuperscript{5,2} shows the test setup comprising the gripper, a glass cup, bolts, and a scale. According to our measurements, gripper arms can hold up to 200g glass cylinder by the friction of the grip pads. When the cup is gripped from its conical top 580g is the maximum mass that gripper arms can withstand. Lastly, 120g is the maximum mass that the gripper can successfully elevate without stalling the elevation motor.

\textsuperscript{1}https://www.vicon.com/cms/wp-content/uploads/2019/05/Vantage_brochure.pdf
5.2.2 Wheel Odometry

A straightforward wall following behavior is implemented for guiding the robot in odometry experiments. Trajectory resulting from this behavior is the contour of the arena and is composed of rotation and translation. We aimed a realistic motion for a swarm robot, including intermittent effects by preferring a wall-following behavior instead of a fixed trajectory. First, the controller collected wheel velocities and computed 2-D pose using the kinematic parameters of the drive system of Kobot-W. Then, we performed a crude parameter estimation for wheel radii and wheelbase by synchronizing ground truth with the robot’s odometry estimation. Total odometry error of the trajectory is minimized by estimating $r_l$, $r_r$ and $d_{wb}$ variables where initial guesses for these parameters are taken from the CAD model.

Figure 5.3 shows the results of odometry experiments. The values for kinematic parameters $r_r$ (right wheel radius), $r_l$ (left wheel radius), and $d_{wb}$ (wheelbase) from CAD and after estimation which minimizes total odometry error are given in Table 5.1.
Figure 5.3: Odometry performance of Kobot-W using kinematic parameters from design and estimation
Table 5.1: Kinematic parameters that are used for odometry calculation (values are in mm)

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$r_l$</th>
<th>$r_r$</th>
<th>$d_{wb}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>19</td>
<td>19</td>
<td>105</td>
</tr>
<tr>
<td>Estimated</td>
<td>20.3</td>
<td>20.4</td>
<td>104.7</td>
</tr>
</tbody>
</table>

5.2.3 Marker-Based Visual Localization

The CoRe uses computer vision for acquiring pose and ID information from the landmarks available in the environment. This pose estimation is used for the localization of the robot. We used two metrics to assess the localization performance. The first is the maximum distance and view angle between the Kobot camera and the detected marker, where the second is the accuracy and repeatability of the pose estimation for different view angles and distances.

Kobot-W can detect markers up to 2000 [mm] and angles up to 80°, with 480p resolution and 50[mm] ArUco marker edge length. However, pose estimation repeatability and accuracy decrease with increasing distance. Thus, we limited the active region of pose estimation to 1000[mm] and ±70° for the experiments. In the experiments, it is not feasible to place Kobot in all possible locations bounded by this region. Therefore pose estimation was performed in three different view angles 0, ±30°, and ±45°. At each angle, Kobot-W moved away from the marker continuously in the interval [250, 1000] [mm]. The experiments were repeated five times. The view angle of 0° corresponds to directly facing the marker and is the symmetry axis of the active region. Second and third view angles ±30°, ±45° correspond to acute angles between the surface normal of the marker and the normal of the camera plane. To compare the results of the local pose estimation with the ground truth, we synchronized the data and presented the results in Figure 5.4a and 5.4b.
Figure 5.4: Shaded line plots showing mean, minimum, and maximum errors of pose estimation vs. distance to marker, with 480p camera resolution
5.2.4 Range & Bearing

IR R&B

Only Kobot-W robots can use the IR R&B system. As a result, this experiment was conducted with two Kobot-W robots. The first robot was stationary while the second robot moved closer and further away from the first robot. Then, local R&B data are synchronized with the ground truth from the tracking system by directing both messages to the base computer and time-stamping them. The results are presented in Figure 5.5.

![Boxplot showing the discrete sensor readings of the IR R&B vs. actual inter-robot distance](image)

Figure 5.5: Boxplot showing the discrete sensor readings of the IR R&B vs. actual inter-robot distance

ToF R&B

Both Kobot-F and Kobot-W can use ToF R&B, but the reflected signal strength is lower for Kobot-F having a smaller cross-sectional area when compared to Kobot-W. Nevertheless, we used Kobot-F robots to prove ToF R&B performance with MAVs.

For the first experiment, two Kobot-F robots are moved closer and apart from each other, and the range information reported by the R&B plotted vs. actual inter-robot distance from the tracking system. The same experiment is repeated with three dif-
different configurations. In the first configuration, there are no hardware and software modifications to observe the stand-alone performance of the sensor. For the second configuration, a retroreflective coating was applied to the outer shells of the motors to increase signal strength. In the third configuration, MCU performed the proposed filtering presented in Chapter 4 and MAVs are coated with retroreflective sheets. Figure 5.6 shows the results of these experiments.

![Graphs showing ranging performance](image)

(a) Without modification  
(b) Retro-reflective coating  
(c) Coating and filtering

Figure 5.6: Scatter plots showing the ranging performance of the ToF R&B

The ambient readings and robot detection performance of the ToF R&B system are tested with two different configurations. In the first, without any modification on the ranging sequence, IQR of the ambient buffer is used for robot detection. In the latter
configuration, we introduced random delay as explained in Section 4.4.2 and once again used IQR of the ambient buffer for robot detection. Results are given in Figure 5.7.

![Figure 5.7: The robot detection performance of ToF R&B. Upper plots show the IQR value of the ambient buffer. Lower plots show the robot detection values by using an IQR threshold of 5[kcps]](image)

Furthermore, we tested ToF R&B under direct sunlight to interpret its performance.
outdoors. For the first experiment, we placed Kobot-F directly facing the sunlight. Next, we obstructed the sensor’s FoV. Doing so enabled observing the habituation performance of the sensor and IQR value of the ambient buffer at the transition. Later, Kobot-F continuously rotated in a room where two sides get direct sunlight, and the other two are walls without any window. Both results are given in Figure 5.8

Figure 5.8: Line plots showing raw ambient readings and IQR value of a buffer of the ToF sensor with exposure to the sunlight
5.3 Swarm Scenario Validation

Swarm scenarios for testing Kobot are selected to complement each other in testing the systems and proving them in real swarm scenarios. Kobots used only local sensory information and onboard computational capabilities and base computer calculated metrics online with the data from the tracking system.

5.3.1 Cue-Based Aggregation

The aggregation cue is a printed grayscale gradient detected by floor sensors. Intensity reading from the floor sensors dictates the waiting duration of the robot inside the cue. Floor sensors are calibrated for the readings from the carpet, which is considered free space by the robots. ArUco markers placed on the arena act as the landmarks for LBA experiments. Experimental arena and the desired aggregation behavior is presented in Figure 5.9.

Figure 5.9: A snapshot from the aggregation experiments

NAS (normalized aggregation size) is the primary metric for evaluating the performance of aggregation experiments and defined as:

$$\text{NAS} = \frac{n_{\text{cue}}}{n_{\text{total}}}$$  \hspace{1cm} (5.1)

where $n_{\text{cue}}$ is the number of robots inside the cue and $n_{\text{total}}$ is the population of the swarm.
The parameter set for the experiments is given in Table 5.2.

Table 5.2: Parameter set used in aggregation experiments

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{max}$</td>
<td>Maximum translational speed</td>
<td>200 [mm/s]</td>
</tr>
<tr>
<td>$\omega_{max}$</td>
<td>Maximum rotational speed</td>
<td>1.5 [rad/s]</td>
</tr>
<tr>
<td>$w_{max}$</td>
<td>Maximum cue wait duration</td>
<td>120 [s]</td>
</tr>
<tr>
<td>$I_c$</td>
<td>Intensity constant</td>
<td>25000</td>
</tr>
</tbody>
</table>

To verify the validity of the implementation, we compared the results of LBA to the results of the well-known BEECLUST [61] algorithm performed with the same parameters.

Experiments were conducted with a swarm of six Kobot-W robots. Each experiment was repeated five times, and corresponding performances are in Figure 5.10.

Figure 5.10: Shaded line plots showing mean, minimum, and maximum NAS values for BEECLUST and LBA (Landmark-Based Aggregation) experiments of Kobot-W
5.3.2 Self-Organized Flocking with Ground Robots

Figure 5.11 shows the desired flocking behaviour from the experiments. The self-organized flocking algorithm has well-defined performance metrics and parameters. We selected angular order and the number of equivalence classes as the performance metrics and performed flocking experiments for three different $\beta$ values.

![Flocking Experiments](image)

Figure 5.11: A snapshot from the flocking experiments. The ceiling camera and ArUco markers on Kobots are used for pose estimation.

The first metric used for evaluating the performance of the flocking experiment is angular order $\Psi(t)$, and it reflects the heading alignment performance of the flock. Angular order defined as follows:

$$
\Psi(t) = \frac{1}{N} \left| \sum_{k=1}^{N} e^{i\theta_k} \right|
$$

(5.2)

where $\theta_k$ indicates the heading of the $k^{th}$ Kobot robot.

For a flock comprising of perfectly aligned robots $\Psi(t) \to 1$, conversely for a flock of randomly aligned robots $\Psi(t) \to 0$.

The number of equivalence classes is the second metric used in the flocking experiments that determine how many sub-groups formed in the flock in terms of distance between robots. $[a]$ denotes equivalence class of $a$ where $\{x, a\} \in S$ by using the equivalence relation $\sim$.
\[ [a] = \{ x \in S | x \sim a \} \] (5.3)

For a set of 2-D points,
\[ P = \{ p_1, p_2, \ldots, p_n \} \] (5.4)

Used equivalence relation \( \sim \) maps the robots to the same equivalence class, if the Euclidian distance between two robots holds the condition \( d(p_1, p_2) < d_{thresh} \) then,
\[ [p_1] = [p_2] \] (5.5)

where \( d \) is the Euclidian distance between 2-D points \( p_1, p_2 \):
\[ d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \] (5.6)

To sum up, the number of the equivalence classes in time \( t \) represents how many subgroups formed in the flock, and its percentage reflects the formation frequency of sub-groups. Parameter set for the flocking experiments is given in Table 5.3.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>Control loop frequency</td>
<td>10 [Hz]</td>
</tr>
<tr>
<td>( u_{max} )</td>
<td>Maximum translational speed</td>
<td>200 [mm/s]</td>
</tr>
<tr>
<td>( \omega_{max} )</td>
<td>Maximum rotational speed</td>
<td>1.5 [rad/s]</td>
</tr>
<tr>
<td>( K_p )</td>
<td>Heading proportional gain</td>
<td>2.5</td>
</tr>
<tr>
<td>( \beta )</td>
<td>Proximal control weight</td>
<td>1.3, 1.5, 1.7, 2.0</td>
</tr>
</tbody>
</table>

Table 5.3: Parameter set used in flocking experiments

Self-organized Flocking with Aerial Robots

We used Kobot-F robots to perform fully-decentralized flocking for the first time using only onboard resources with indoor aerial robots. The same algorithm is used
(a) Boxplot showing angular order vs. \( \beta \)

(b) Sized and color mapped scatter plot showing the percentage of number of equivalence classes vs. \( \beta \)

Figure 5.12: Performance metrics of Kobot-W flocking experiments

with Kobot-F robots as Kobot-W robots. The resulting behavior is shown in a series of snapshots from experiments in Figure 5.13.
Figure 5.13: Snapshots from Kobot-F flocking experiments. Yellow arrows show heading direction. Red LEDs and arrows show nearby obstacle direction. Blue LEDs and arrows show neighbor robot direction.
CHAPTER 6

DISCUSSION AND CONCLUSION

6.1 Discussion

6.1.1 The CoRe Systems

2-DoF Compliant Gripper

The design calculation we performed in Section 3.1.2 and the result of the experiment in Section 5.2.1 show a \( \frac{m_{\text{calc}} - m_{\text{exp}}}{m_{\text{calc}}} = 130/251 = 0.52 \) times loss in the maximum elevation torque. We believe the most prominent cause of this loss is the flexibility of the components. The 3-D printed gripper arm, although not desired to be flexible in the longitudinal axis, acts as a torsional spring and limits the torque that the motor can transfer to the gripped object. It should also be noted that, just like the flexibility, static friction of the drive elements oppose the motion and might lead to such a result. Also, the fact that the system can hold (not elevate) 580[g], which is \( 580/250 = 2.32 \) times more than the calculated maximum torque, strengthens those assumptions. Another cause for that loss might be a considerable difference between the stall (holding) and dynamic maximum torque (moving output).

Considering 120[g] is still acceptable for the desired operation, such as moving landmarks (< 10[g]) and transporting Kobot-F < 105[g], we can argue that a compliant gripper system works as desired. For all practical purposes, a 100[g] cylinder with a > 60[mm] diameter can be safely transported by the compliant gripper of the CoRe without any problem.

Furthermore, objects the system grips range from various vegetables to glass and
plastic cups. The system’s significant advantage is compliance, enabling the grip of various objects with the same action. More importantly, the onboard camera can see gripped objects and perform pose estimation using markers to calculate the approach trajectory.

**Marker Based Localization**

In the development phase (see Fig. 4.4), we verified using marker-based localization for acquiring range and bearing information from other robots in the environment. This use case is valuable for getting range and bearing information from robots working in different planes, such as Kobot-W and Kobot-F.

For the marker-based localization experiments, we see that distance error (see Fig. 5.4a) has a similar bias for different angles. Thus, a straightforward calibration is possible to increase the performance. On the other hand, angle error rises with the skewness of the view angle, and it is challenging to compensate. Notably, Figure 5.4b shows that directly facing the marker at 0° in distant regions is prone to angle ambiguity and a considerable variation in minimum and maximum angle errors. ArUco library documentation emphasizes this pitfall as well. The ambiguity grows larger when these four co-planar points almost lie on the camera plane (i.e., 0°). We overcome this issue by tilting the markers vertically with the marker holder parts. However, an even better solution would be to use multiple markers placed on a rigid body at skew angles if landmark size is not a constraint.

**Odometry**

According to the results of Figure 5.3a, the design values for wheel radii and wheel-base obtained from the CAD model are acceptable as a first approximation. However, actual values result in different trajectories. This variation is due to using flexible rubber tires and 3-D printing for manufacturing the drive system. As can be seen from Figure 5.3a, odometry results with the calibrated parameters are closer to the ground truth. Although values do not differ much, even slight variations lead to significant accumulated errors in trajectory over time. Additionally, from Figure 5.3b it
is observed that displacement error for odometry with design parameters is ≈ $\%4$ and for odometry with estimated parameters the displacement error is ≈ $1\%$ when total displacement is $10^4[mm]$.

An important note regarding the odometry results is, our error is increasing periodically (see Fig. 5.3b) where the period corresponds to two consecutive rotations. One can see from Figure 5.3a raw odometry has a minimal error in heading direction but a considerable error in the total displacement. Error in displacement shows that the difference between wheel radii $r_l$ and $r_r$ is small, but both radii are larger than the CAD model. This is proven by the estimation procedure (see Tab 5.1). There is a $1[mm]$ difference between the design and estimated radii, but only $0.1[mm]$ difference between the left and right radius.

This parameter estimation procedure might act as a calibration for the drive system in which the estimated parameters are stored and employed whenever Kobot-W needs higher resolution odometry. Similarly, more advanced calibration procedures such as [62, 63, 64] can be implemented if required.

LBA experiments revealed that the calibrated odometry performance of Kobot is satisfactory as a swarm. When used with the landmarks available in the environment, it is possible to perform precise localization by resetting accumulated odometry errors. Combined with the visual odometry accuracy of $< 60[mm], < 5^\circ$, Kobot-W’s localization error will be minimal even in vast areas with sparsely placed landmarks. When the density of landmarks is increased, this localization error will decrease up to $20[mm]$ and $< 2^\circ$.

**ToF R&B**

ToF R&B is the most significant part of the CoRe enabling decentralized swarm operation. It provides essential information to the host robots about their environments and neighbors. For that reason, its performance is investigated in multiple experiments, showing how our modifications made the system usable with a swarm composed of MAVs. Furthermore, when MAVs use retroreflective sheets coated to the cylindrical outer shell of the motors, the IR tracking systems perceive these as passive markers,
removing the need for additional markers.

From Figure 5.6a it is evident that ToF sensors can not be used without any modification on a MAV. This inadequacy is because MAVs do not reflect enough signal, and other robots’ sensors act as a noise source, decreasing the signal-to-noise ratio. As Figure 5.6b shows, retroreflective markers make a considerable difference enabling valid range readings by increasing the signal-to-noise ratio. Later, embedded filtering removes the outlier data and adds a bias to report always closer than the actual value, as Figure 5.6c depicts. In other words, as a whole, Figure 5.6 shows increased ranging performance by the modifications presented in this thesis. The performance is promising and results in a best fit line $y = 1.00x - 38.5$ and a reliable fit of $R^2 = 0.98$. As a result, the system detects robots as small as MAVs that are 100 [mm] in diagonal, in a range up to 1.3 [m]. Additionally, the system is less than 10 [g], which can be used with MAVs under 80 [g].

Further, we investigated the robot detection performance of the proposed system. Figure 5.7a shows ambient readings of a ToF sensor with unmodified ranging sequence and robot detection values corresponding to those readings. Periodic ambient reading is visible, and when the two sensors are in sync, the robot detection becomes a false-negative. In other words, periodic ambient reading results in frequent and periodic false-negative robot readings. In Figure 5.7b however, the periodicity of the ambient reading is removed with the introduced random delay and prevented false-negative robot detection.

We also tested the ambient reading performance of ToF sensors for the first glance at their potential outdoor use. Figure 5.8a is used to exhibit the habituation performance of the sensors. We noticed no decrease in the ambient readings when the sensor was exposed to sunlight for a prolonged duration which will help determine an ambient bias and remove it. The figure also shows IQR of the ambient is below the threshold during full exposure to sunlight and result in a peak only in drastic changes in the ambient level, such as obstructing FoV. Figure 5.8b depicts a more realistic case where ambient reading from sunlight is continuous by rotating in a room where two sides are prone to direct sunlight and the other two only reflect. It showcases a lower SDEV and IQR when compared to the emit sequences of the sensor. For that reason, for
robot detection, instead of using the ambient value directly, we have used the IQR of the ambient buffer of length five, which results in false-positive robot detection only when there is a remarkable difference in the ambient reading in consecutive readings. As a result, one might conclude that the system is promising for outdoor use.

6.1.2 Swarm Scenarios

The CoRe’s behavior implementations tested the performance of different systems in self-organized flocking and cue-based aggregation experiments as shown in Table 6.1.

Table 6.1: Comparison of the selected algorithms in terms of used hardware and software architecture

<table>
<thead>
<tr>
<th>Systems</th>
<th>BEECLUST</th>
<th>LBA</th>
<th>Flocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Event driven</td>
<td>Event driven</td>
<td>Clock Driven</td>
</tr>
<tr>
<td>Motion Control</td>
<td>Vel.</td>
<td>Pos. &amp; Vel.</td>
<td>Vel.</td>
</tr>
<tr>
<td>Range and Bearing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Local Comm.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IMU</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Odometry</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Floor Sensors</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Camera</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

In swarm scenario experiments, we observed that the event-driven algorithms (i.e., aggregation) are more susceptible to sensor noise since triggering a false event would result in fraudulent behavior that has long-term effects on the algorithm’s performance. On the other hand, in periodically sampled algorithms (i.e., flocking), an inaccurate measurement by the sensor is compensated in the next measurement cycle. Therefore, it does not affect the algorithm’s long-term performance. Accordingly, we want to emphasize that this transparency makes event-driven algorithms preferable in testing the performance of a swarm robot since it does not filter out sensor noises. Furthermore, flocking and aggregation is two very different yet common swarm robot behaviors. Therefore, their implementation prepares the required software basis and
utilities needed for various algorithms. For that reason, aggregation and flocking behaviors would be suitable candidates to define a minimum set of behaviors that standard swarm robots should perform as discussed in [8].

**Cue-Based Aggregation**

Figure 5.10 shows higher performance for LBA, and results have the same trend with the simulation results published by [2]. Higher performance is expected, considering the abundant information from the landmarks and odometry. However, the significance of LBA for the CoRe is the use of multiple novel systems together for a successful swarm behavior. These systems are floor sensing, marker-based localization, and odometry. Isolated tests of these systems provided insight into the performance. However, this is not always enough for swarm robots since unpredicted interactions between robots during the behavior are present. Thus the successful LBA experiment proves the capability of the mentioned CoRe systems for swarm robotics.

**Self-Organized Flocking**

In self organized flocking experiments, for all $\beta$ values tested, angular order is close to 1.0 (see Figure 5.12a). Also, angular alignment value decrease, and its spread increase for larger $\beta$ since increasing weight of proximal control decrease the alignment effect of nearby robots.

As $\beta$ increases, more sub-groups and an increased percentage of sub-groups are observed from Figure 5.12b since increasing the weight of proximal control prioritizes the robot’s local readings, decreasing the flock’s coherence. Although one might conclude that lower $\beta$ values result in higher flocking performance, $\beta$ is low bounded by the increased inter-robot collisions.

The optimal value for the weight of proximal control is $\beta = 1.3$ by our experiments, and the results are in agreement with the results published by [1].

For flocking experiments performed with Kobot-F, we used parameters tuned in Kobot-W. Then, we tested $\beta$ values in the vicinity and found out once again the optimal
value $\beta = 1.3$. This finding proves the relevance of developing and testing systems on resourceful ground robots and using them for application in alternative robots like Kobot-F.

Lastly, the self-organized flocking algorithm introduces additional challenges for Kobot-F, which an alternative flocking algorithm enabling holonomic motion can solve. Such a novel behavior would increase flock performance since robots might reach higher velocities due to not requiring rotation to align their heading. Nevertheless, it is a valid argument that non-holonomic behavior is more bio-inspired and in theory and can be performed with minimal sensing.

6.2 Conclusion

This thesis presents Kobot, an extensible heterogeneous swarm robot platform. The platform consists of ground and aerial swarm robots Kobot-W and Kobot-F that use only onboard resources and the common hardware-software architecture called Kobot CoRe. The CoRe includes novel sensory-motor and communication systems such as the first robot detection capable ToF R&B.

In this study, the CoRe systems are first developed on resourceful Kobot-W, then used with Kobot-F for decentralized swarm behaviors with indoor drones. The performance of the proposed systems were evaluated in sub-system experiments and with two existing swarm robotics behaviors. From the results of the experiments, we found out that the performance is satisfactory both at the sub-system and swarm level. Furthermore, Kobot-W performed the first real robot experiments of the novel LBA algorithm, and Kobot-F performed the first decentralized flocking with MAVs.

We believe this study might develop in several directions. First, as preliminary experiments show, the ToF R&B usage can be extended to outdoors, where the main challenge would be robot detection under direct sunlight. Secondly, one can design novel swarm robots with alternative locomotion systems built using the CoRe, such as the Kobot-O and Kobot-Q prototypes. Finally, implementing novel decentralized behaviors on Kobot-F or developing heterogeneous swarm behaviors using Kobot-W and Kobot-F for a common goal might be potential future works.
REFERENCES


78


APPENDIX

All the open materials of this thesis can be accessed by using the below links:

- https://sites.google.com/view/cembilaloglu/kobot-platform
- https://romer.metu.edu.tr/en/kobot-platform
- https://romer2.metu.edu.tr/kobot-platform/

7.1 Experiment Videos

- Gripper
- Kobot-O, Kobot-Q
- Kobot-W mapping
- LBA Kobot-W
- Kobot-W self organized flocking
- Kobot-F self organized flocking

7.2 Design calculations

Jupyter Notebook
7.3 CAD Models

- Kobot-W
- Kobot-F
- 2-DoF Compliant Gripper System
- ToF R&B System

7.4 PCB Designs

- Floor Sensing System
- ToF R&B Motherboard
- ToF Daughterboards
- IR Local Communication Daughterboards

7.5 Git Repositories

- kobot_ros
- kobot_base
- kobotMotor
- kobotRangeNBearing

7.6 SD Card Images

- RPi Zero
- RPi 3B+