

Formation Control of Mobile Robot Swarm in Search Operation

Mr. Chanun Asavasirikulkij



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Cyber-Physical System
Department of Mechanical Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2022
Copyright of Chulalongkorn University

การควบคุมการเคลื่อนที่แบบกลุ่มของฝูงหุ่นยนต์ในปฏิบัติการค้นหา



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาระบบกายภาพที่เชื่อมประสานด้วยเครือข่ายไซเบอร์ ภาควิชาวิศวกรรมเครื่องกล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2565

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

Thesis Title Formation Control of Mobile Robot Swarm in Search
Operation
By Mr. Chanun Asavasirikulkij
Field of Study Cyber-Physical System
Thesis Advisor Associate Professor RATCHATIN CHANCHAROEN

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University
in Partial Fulfillment of the Requirement for the Master of Engineering

..... Dean of the FACULTY OF
ENGINEERING
(Professor SUPOT TEACHAVORASINSKUN)

THESIS COMMITTEE

..... Chairman
(Associate Professor Gridsada Phanomchoeng)
..... Thesis Advisor
(Associate Professor RATCHATIN CHANCHAROEN)
..... Examiner
(Assistant Professor WIDHYAKORN ASDORNWISED)
..... External Examiner
(Associate Professor Phaderm Nangsue)



จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

ชานันท์ อัสวศิริกุลกิจ : การควบคุมการเคลื่อนที่แบบกลุ่มของฝูงหุ่นยนต์ใน
ปฏิบัติการค้นหา. (Formation Control of Mobile Robot
Swarm in Search Operation) อ.ที่ปรึกษาหลัก : รศ. ดร.รัชทิน จันทร
เจริญ

การใช้ฝูงหุ่นยนต์เป็นวิธีที่คาดว่าจะเพิ่มประสิทธิภาพในการกิจการค้นหา วิทยานิพนธ์
นี้มุ่งเน้นเสนอแนวทางการออกแบบและการผลิตที่มีประสิทธิภาพสำหรับสร้างฝูงหุ่นยนต์บน
ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System (ROS)) และนำเสนอ
สถาปัตยกรรมเครือข่ายสำหรับฝูงหุ่นยนต์ที่มีประสิทธิภาพในการสื่อสารระหว่างหุ่นยนต์ในทีม
เป้าหมายหลักของงานวิจัยนี้คือการศึกษาการควบคุมรูปแบบการเคลื่อนที่ของหุ่นยนต์แบบกลุ่ม
และศึกษาความโดดเด่นที่เกิดขึ้นกับภารกิจการค้นหา การศึกษาครอบคลุมถึงการดำเนินการ
ทดลองและการวิเคราะห์ผลเชิงปริมาณ ผลลัพธ์จากการศึกษาเชิงทดลองแสดงให้เห็นถึง
ประสิทธิผลของภารกิจที่ดีขึ้นอย่างมีนัยสำคัญ โดยสามารถเพิ่มความเร็วในการสำรวจและค้นหา
เมื่อใช้เทคนิคการควบคุมฝูงหุ่นยนต์ที่ได้นำเสนอ งานวิทยานิพนธ์นี้นำเสนอข้อมูลที่มี
ความสำคัญเกี่ยวกับจุดเด่นและข้อจำกัดของการควบคุมรูปแบบการเคลื่อนที่ของหุ่นยนต์แบบ
กลุ่มเพื่อเพิ่มประสิทธิภาพในการกิจการค้นหา

จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

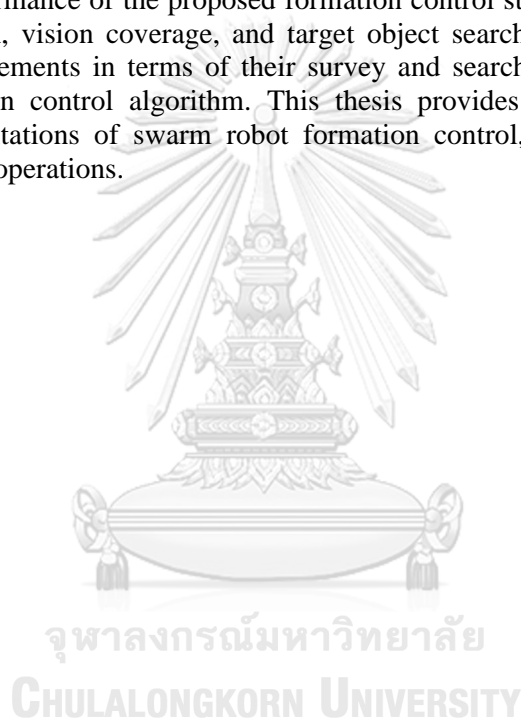
สาขาวิชา	ระบบกายภาพที่เชื่อมประสาน	ลายมือชื่อนิติ
	ด้วยเครือข่ายไซเบอร์
ปีการศึกษา	2565	ลายมือชื่อ อ.ที่ปรึกษาหลัก
	

6470019021 : MAJOR CYBER-PHYSICAL SYSTEM

KEYWORD: swarm robotics, mobile robot, formation control, search operations,
network architecture

Chanun Asavasirikulkij : Formation Control of Mobile Robot Swarm in Search
Operation. Advisor: Assoc. Prof. RATCHATIN CHANCHAROEN

The use of swarm mobile robots has emerged as a promising approach to enhance search efficiency in various applications. This thesis focuses on proposing an effective design and fabrication methodology for a swarm mobile robot system based on the Robot Operating System (ROS) framework. Additionally, a novel network architecture is presented to improve communication efficiency among the robots. The primary objective of this research is to investigate the formation control of swarm mobile robots and their superior ability in search operations. Through extensive experiments and quantitative analysis, the performance of the proposed formation control strategy is evaluated in terms of survey duration, vision coverage, and target object search time. The results indicate significant improvements in terms of their survey and search time when employing the proposed formation control algorithm. This thesis provides valuable insights into the strengths and limitations of swarm robot formation control, offering a foundation for optimizing search operations.



Field of Study: Cyber-Physical System
Academic Year: 2022

Student's Signature
Advisor's Signature

ACKNOWLEDGEMENTS

I am grateful to my advisor, Assoc. Prof. Ratchatin Chanchaen, for his guidance and insights throughout the entire research process. His expertise and continuous support have been important in shaping the direction of my thesis and improving its quality.

I would like to extend my sincere appreciation to the members of my thesis committee, Assoc. Prof. Gridsada Phanomchoeng, Asst. Prof. Widhyakorn Asdornwised, and Assoc. Prof. Phadern Nangsue. Their valuable feedback and constructive criticism have significantly contributed to my thesis improvement.

I would also like to show appreciation to my family and friends for their support, encouragement, and understanding throughout my thesis journey. Additionally, I would like to acknowledge my colleagues who have offered their assistance, shared valuable insights, and engaged in discussions related to my research.

I am thankful to Chulalongkorn University for providing me with the necessary resources, facilities, and opportunities to pursue my research. The Scholarship from the Graduate School, Chulalongkorn University to commemorate the 72nd anniversary of his Majesty King Bhumibol Adulyadej is gratefully acknowledged.

TABLE OF CONTENTS

	Page
ABSTRACT (THAI)	iii
ABSTRACT (ENGLISH).....	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS	vi
Chapter 1: Introduction	1
1.1. Background and Significance	1
1.2. Research Challenges.....	2
1.3. Research Objectives.....	2
1.4. Expected Benefits	2
Chapter 2: Literature Review.....	3
Chapter 3: Single Robot Methodology	7
3.1. Hardware Design/Sensor Selection	7
3.2. Software Design.....	9
3.2.1. Low-level Control	10
3.2.2. High-level Control.....	11
3.2.3. Edge-level Control	12
3.3. Robot Workspace Mapping	12
3.4. Robot Localization.....	13
3.4.1 Robot Self-localization for Autonomous Operation.....	14
3.5. Robot Navigation.....	15
3.6. Robot Vision.....	17
Chapter 4: Multiple Robot System Methodology	19
4.1. Multiple Robot Agent Overview	19
4.2. Edge Computing Computer Overview	21
4.3. Edge Teleoperation Computer Overview	23

4.4. Network Architecture	23
Chapter 5: Formation Control of Robot Swarm	25
5.1. Swarm Edge Computing	26
5.2. Robot Edge Computing	27
Chapter 6: Experimental Scenarios.....	28
6.1. Formation Control of Multiple Robot Experiment.....	30
6.1.1. Experiment #1	31
6.1.2. Experiment #2	33
6.1.3. Experiment #3	35
6.2. Single Robot Experiment.....	37
6.2.1. Experiment #4	37
6.2.2. Experiment #5	38
Chapter 7: Experimental Results	40
7.1. Robot Survey: Path.....	40
7.2. Robot Survey: Vision Coverage and Survey Time.....	41
7.3. Robot Object Search: Position of Detection	42
7.4. Robot Object Search: Orientation of Detection and Survey Time	43
7.5. Robot Swarm Search and Surround Target Object.....	43
Chapter 8: Conclusion.....	45
REFERENCES	47
VITA	50

Chapter 1: Introduction

1.1. Background and Significance

As we move towards the fourth industrial revolution, robots must be adaptable and flexible. A key aspect of the next generation is semi-autonomous mobile robots. Wireless network performance is a crucial technology driving this revolution, enabling mobile robots to operate in much larger spaces without the constraints of wired connections. The robots should be able to work as autonomous stand-alone robots or collaborate with other machines. The same type of robot or completely different type of robot could collaborate to effectively complete a given task. One main obstacle in robot research is the cost of robots, as it requires a high amount of resources. Another big concern is the requirement for advanced sensors. Robots should be cost-efficient but have enough capabilities and intelligence to have decentralized decision-making. Currently, most robots come with high processing power through both on-board computing and on-edge computing (edge computing). This contributes to the overall efficiency of the robot to autonomously operate in their working environment.

Many applications require several standalone robots that can work together as a robot swarm. The primary objective is to integrate multiple agents, often simple robots, in a decentralized manner. Typically, there is no central control governing the actions of the robots. Instead, robots rely on local interactions with their neighboring robots to accomplish their objectives. Swarm robots' collaboration takes advantage of wireless network performance to communicate among themselves. This self-organization and local coordination enable the swarm to exhibit complex behaviors and adapt to changing environmental conditions.

With the challenge to control the swarm efficiently, researchers aim to uncover several swarm control principles and algorithms that can be used to design robust and flexible collective behavior. One such algorithm is designed to govern the spatial arrangement and coordination of multiple robots within a swarm. It involves controlling the positions and orientations of individual robots in a way that it collectively achieves a desired formation, also known as "formation control".

Swarm robots have the potential to be used in several real-world applications in various domains. First, in search and rescue operations, the formation can be dynamically adjusted to adapt to changes in the environment or to focus on specific regions of interest. Second, in surveillance and monitoring, the swarm can adopt formations optimized for efficient coverage and exploration, ensuring thorough search of the affected area while maintaining communication and coordination within the team.

1.2. Research Challenges

- Deploying the system with several robots, keeping each robot alive, and dealing with physical interruption from the outside environment.
- The robot should have high mobility and endurance, being able to move in any direction without hardware constraints.
- The system should be able to handle communication constraints between each member in the swarm and deal with communication disturbance from other devices in the environment.
- The system should be able to scale to a higher number of robots.
- The robustness and adaptability of the system should be significantly high.

1.3. Research Objectives

- Design and fabrication of mobile robot swarm (3 robots), based on Robot Operating System (ROS) framework.
- Develop formation control techniques used in search operation, along with a proposed method to evaluate the effectiveness of formation control.

1.4. Expected Benefits

- Evaluate the effectiveness of the proposed network architecture, sensor selection, hardware and software design, and formation control algorithm.
- Provide a framework for swarm robotics projects.

Chapter 2: Literature Review

Swarm robotics is moving towards real-world applications. The paper [1] describes some promising applications for swarm robotics, such as search and rescue missions, environmental monitoring, and precision agriculture. However, according to [2], there are several challenges that deter the transition to real-world use case. First, the difficulties in deploying high-density swarm, keeping each robot alive and dealing with physical interruptions from the outside environment. Second, communication constraints between robots and other devices working on the same mission. Third, the limitation of mobility and endurance of robots. Therefore, further research is still needed to solve these challenges, evaluate performance, and enhance reliability. The robots should be equipped with enough sensors and actuators required for the experiment, while being small and low-cost.

For the hardware aspect, sensor selection is one of the most important as it would determine the efficiency of the robot. According to [3], there are two main types of sensors. First, the internal sensors are integrated within the robot's body and are used to obtain information about its internal state, such as position sensor, velocity sensor, acceleration sensor, etc. Second, the external sensors assist the robot in gathering information about the environment and enable it to sense external objects. Examples of external sensors include cameras, range sensors, force sensors, etc.

The sensor selection is usually done to fulfill the robot requirements, allowing it to perform mapping and localization as standalone. A considerable amount of research has been conducted to investigate and explore mapping and localization methodologies for mobile robots. Over the years, a wide range of techniques, such as odometry [3], range scanners [4], RFID [5], and machine vision [6, 7] have been explored. However, every method has its own limitations. Therefore, combining several methods offers significant advantages.

A widely implemented method for mobile robot localization involves a combination of odometry and LiDAR. Odometry estimates the robot's position based on the rotational position of its wheels, which is sensed by encoders. Although odometry helps determine the robot's position and orientation, the resulting estimation introduces incremental errors over the robot's motion [8].

LiDAR, short for "Light Detection and Ranging," is a sensor that utilizes laser beams to measure the distance to objects. By employing laser scanning, LiDAR enables the perception of the surrounding environment, producing a two-dimensional image that can be easily processed to determine the position of the robot. Unlike odometry, which offers a relative position, LiDAR provides an absolute position. This overcame the challenges associated with odometry, increasing the overall accuracy and reliability of the robot's localization.

Another localization method involves deploying sensors into the environment, such as using motion capture [9]. This method is typically utilized with large swarms since the sensors can be shared throughout the entire swarm. However, in this approach, robots are considered blind and rely solely on sensors from the environment for navigation. This method is most suitable for indoor and fixed workspace. However, motion capture could be used to track object or even human operators within in the workspace with high accuracy [10].

To ensure efficient functioning of the sensor with the robot, the selection of microcontrollers is crucial. There is a variety of microcontrollers available for use in modern robots. For example, in [11], the "STM32F4DISCOVERY" board is used as the main controller for motor control and odometry calculation. The "UP SQUARED" board is utilized to send velocity commands and publish camera information to the edge computer. The edge computer receives the information from the joystick and transforms it into velocity command.

The Robot Operating System (ROS) framework is widely regarded as the most preferable programming framework for building robotics applications and managing packages. According to [12], the Robotic Operating System is peer-to-peer, tools-based, multi-lingual, lightweight, free, and open-source. First, the term "peer-to-peer" signifies that devices can communicate without relying on a central server. This approach prevents high traffic congestion in a single location, ensuring smooth communication. Second, ROS supports four languages: C++, Python, Octave, and LISP. This allows different coding languages to be employed simultaneously when running the entire system. Third, when executing the system, ROS attempts to reuse drivers or algorithm code from existing packages, promoting code efficiency, and

reducing redundancy. Fourth, ROS can be used for system development for both commercial and non-commercial projects. Fifth, ROS is open source, allowing everyone in the community to contribute by adding add-ons or debugging the existing packages.

One standalone robot may sometimes not achieve a complex task. Therefore, the use of multi-robot systems (MRSs) can be an option. According to [13], multi-robot systems (MRSs) are capable of tackling demanding tasks in complex scenarios. The author utilizes a hierarchical framework that enables multiple robots to navigate, forming optimized formations in unfamiliar environments with both static and dynamic obstacles. The framework allows each robot to independently navigate towards a global target using its local perception, even with limited communication, while maintaining an optimized formation throughout their progress.

However, MRSs may lack some of the benefits of swarm robotics stated in [14]. Swarm robotics, a subset of MRSs, offers main advantage in terms of flexibility, scalability, and robustness. Swarm flexibility refers to the ability of a multi-robot system to adapt and respond to changes in the environment or task requirements. It implies that the system can dynamically adjust its behavior, coordination, and formation to handle different scenarios effectively. Swarm scalability refers to the ability of a multi-robot system to handle an increasing number of robots without compromising performance or efficiency. It involves designing algorithms and control mechanisms that can accommodate a growing number of robots in a coordinated manner. Swarm robustness refers to the ability of a multi-robot system to maintain its functionality and achieve the desired objectives even in the presence of failures, malfunctions, or adverse conditions. It involves designing resilient algorithms and control strategies that can handle individual robot failures, communication losses, or environmental disturbances.

The features of swarm robotics have been summarized in [15]. Swarm robots are commonly characterized by their small size and cost-effectiveness, enabling them to operate efficiently. These robots possess the ability to perceive and navigate their surroundings autonomously. Ideally, the robots within a swarm should share similar attributes, promoting coordination among them. Simplicity is a fundamental aspect of

swarm robots, as they rely on collaboration rather than individual capabilities to accomplish tasks. The swarm operates in a decentralized, self-organized, and distributed manner, resembling the natural collaborative behavior observed in various organisms. The behavioral norms guiding swarm agents are typically straightforward and executed independently.

Robots used for swarm robot experiments are designed and built in different forms as summarized in [15]. However, this research focuses solely on a three-wheel omni-directional robot since it meets the mobility requirement for the experiment. The design and fabrication of three-wheel mobile robot has been extensively researched [11] [16] [17].

To control the swarm, several methods can be used. However, in this research, we focus mainly on formation control. The velocity-based formation control [18], focus on controlling the velocities or speeds of individual robots to achieve a desired formation. Each robot is assigned a specific velocity vector that determines its direction and speed. The advantage of velocity-based formation control is that it allows for flexible and smooth movement of robots within the formation. Some example of velocity-based formation control is shown in [19].

However, in this research, we will focus on position-based formation control [20]. The emphasis is on controlling the positions of individual robots to achieve the desired formation. Each robot is assigned a specific position or coordinate within the formation, and the control algorithm ensures that each robot moves to and maintains its designated position relative to other robots.

Chapter 3: Single Robot Methodology

This chapter considered a selection of devices (sensors and actuators) to be used with the robot. Several devices are managed together, transitioning from “device layer” into “machine layer”.

3.1. Hardware Design/Sensor Selection

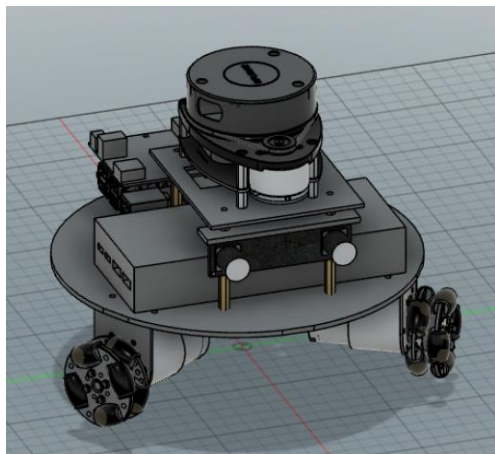


Fig. 1. Mobile robot design with Fusion360

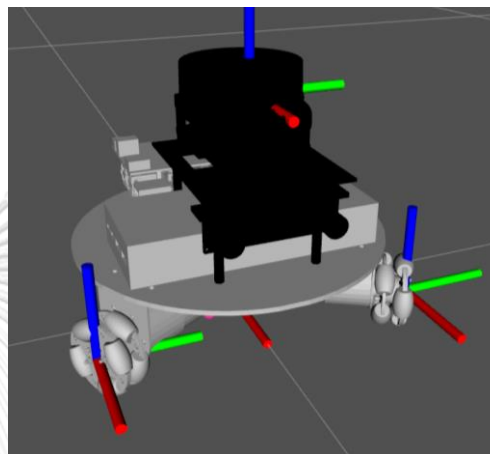


Fig. 2. URDF of the robot model in ROS

The mobile robot in this study is designed as an omnidirectional-wheel robot as shown in Fig. 1. Each of the mobile robots is equipped with both internal and external sensors. The internal sensor includes the encoder attached to the motor of each wheel, while the external sensor includes the stereo camera and LiDAR. The main microcontrollers used are the Raspberry Pi 4 with the attached Raspberry Pi HAT and STM32. It is worth noting that the robot can operate independently without relying on any sensors in the surrounding workspace. From Fig. 2, the URDF consists of position and orientation detail for each robot’s link and joint. In ROS, with URDF file, each frame will be attached to a reference frame or the map.

The power source is a significant concern in mobile robots as it needs to be compact, lightweight, and rechargeable. The current robot’s design, both 5 V and 12 V voltage supply are required. Therefore, power-delivery (PD) power bank was selected because of its portability, high power rate, and variety of adjustable voltage sources. The PD interface allowed the device to draw high rate of energy. However, the power delivery protocol needs the right communication due to the safety. As a solution, decoy modules were deployed to inform the power bank to generate 12 V

source with Type-C port, which power the motor through the STM32. The 5 V USB port was utilized to provide power to the sensors connected to a Raspberry Pi 4.

The designed robot is then fabricated into a physical form, shown in Fig. 3. The mobile robot features three motorized omni-directional wheels, evenly spaced at 120 degrees from each other, as shown in Fig. 4. This configuration allows the robot to exhibit holonomic motion, providing three degrees of freedom (DOF). The robot can move freely in the x-axis, y-axis, and θ (orientation) without any constraints.

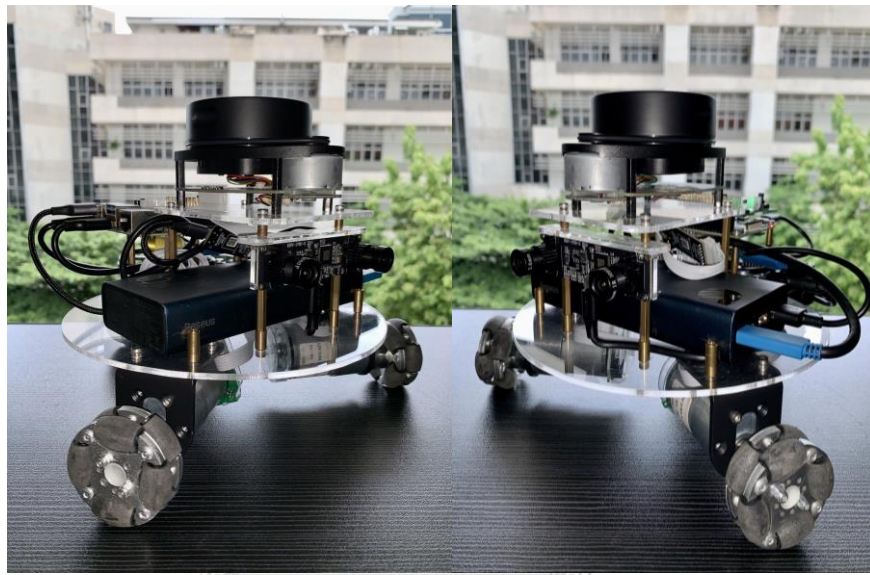


Fig. 3. Left and right views of the robot

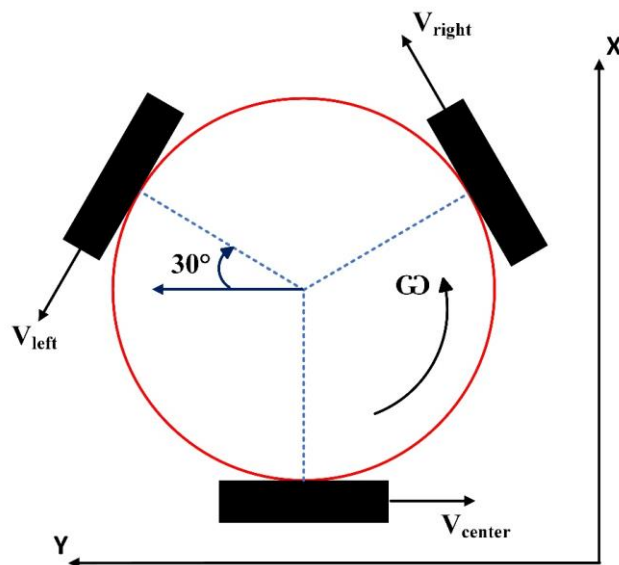


Fig. 4. Robot kinematics

3.2. Software Design

From Fig. 5, it can be observed that the three omnidirectional wheeled mobile robot could be divided into three control levels. These levels consist of the low-level control, high-level control, and edge-level control. The high-level and low-level were powered by a 65W-20000mAh power bank. The decoy modules are implemented to establish communication with the power bank to generate 12 volts to power the STM32 connected to the motors. The standard 5-volt port is primarily used to power the Raspberry Pi 4 and other devices (LiDAR sensor and stereo camera) connected to the board. The high-level control and low-level control are interconnected via a serial port. On the other hand, the edge-level control communicates wirelessly with high-level control wirelessly through Wi-Fi.

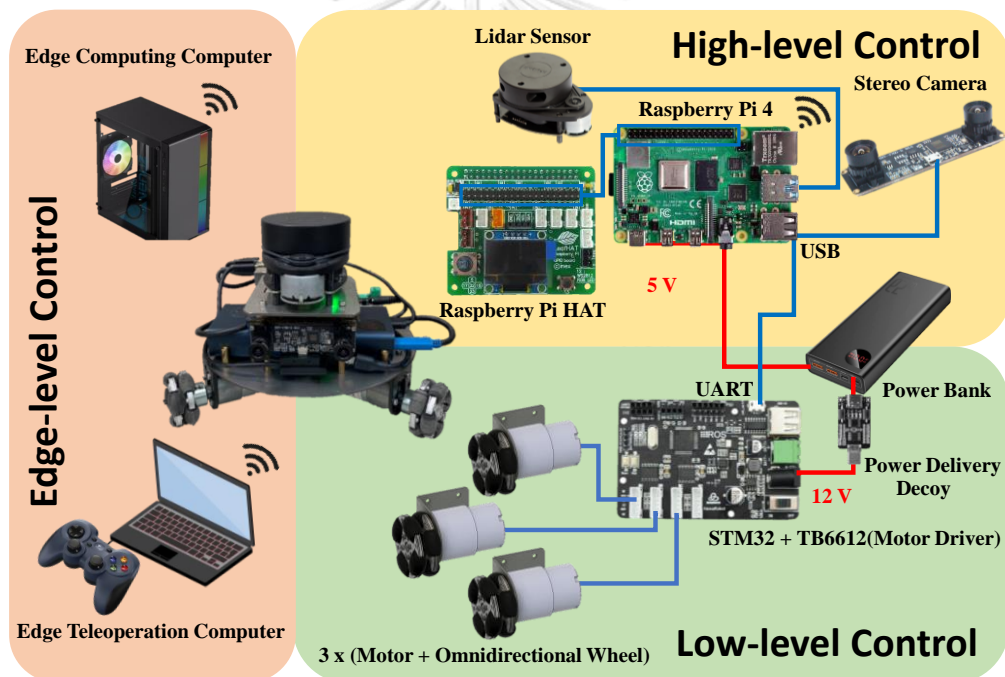


Fig. 5. System diagram of the mobile robot used in this experiment.

In this proposed robot framework, the robot is capable of handling simple tasks on-board at the high-level control, while sending large data to be processed at the edge-level control. The raw data of LiDAR scans and compressed image from the stereo camera are transmitted from the high-level control to process at the edge-level control. The mobile robots will be lighter, as they only need to collect dynamic environment databases on-board, while the analysis is offloaded to edge-level control.

3.2.1. Low-level Control

The low-level control is tasked with controlling the robot's actuator and receiving low-level encoders. STM32 was used as a micro processing unit (MPU) to control three DC motors and receive three encoder signals. The benefit of STM32 is its timer module which enables software motor commands and hardware encoder reading at the same time. Each omni-directional wheel requires precise velocity control, achieved through closed-loop control of motor based on encoder feedback.

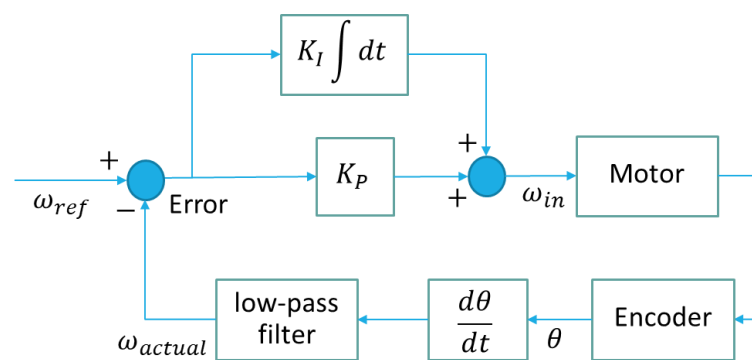


Fig. 6. Proportional and integral of velocity control

From Fig. 6, the reference value for the system output represents the desired operating value of the output. The angular velocity of the wheels is derived by low-pass filtering the derivative of the encoder position. The PI feedback control is calculated after the low-pass filter angular velocity was determined, sending the command to each motor. The data sampling rate was fixed to stabilize the derivative with STM32's timer interrupt. The PI feedback control is calculated after the velocity was determined and then the value was sent to control PWM generated by the STM32 timer. The PWM signal was brought to TB6612 motor controller chip.

In this case, the STM32 implements three control modes: power, position, and velocity. Serial command options are used to receive commands from the high-level control, and the specific command options are listed on the following page.

- M0 → Power Control = voltage control= open loop control
- M1 → Position Control = PI control = close loop control
- M2 → Velocity Control = PI control = close loop control
- R → reset encoder reading.
- S1 → start encoder reading.
- S2 → stop encoder reading.
- Ax → left motor followed by its value.
- Bx → right motor followed by its value.
- Cx → center motor followed by its value.

An example in Fig. 7 shows position control “M1A100B700C1000”, from the graph each robot wheel could rotate to the correct position quickly.



Fig. 7. Graph of the motor reaction after position control input

3.2.2. High-level Control

The Ubuntu 20.04 is the operating system installed into the Raspberry Pi 4, used to run ROS Noetic for robot’s high-level control. ROS (Robot Operating System) is a framework for building robotics applications and managing packages. The benefit of using ROS is that working module could be divided into “node” to handle specific task. Each node communicates with each other by published and subscribed target topic.

The high-level control receives encoder data from STM32 and publishes each motor velocity command to STM32. Moreover, the high-level control gathers all the external (LiDAR, stereo camera) and internal (encoder) sensor data to be computed at the edge computing computer. The Raspberry Pi Hardware Attached on Top (HAT) is

used to identify their identification through OLED screen and LED shown in Fig. 8. When the robot is powered on, it will connect directly to the preassigned network. After connection, it will automatically display its dynamic IP address on the OLED screen. Each robot will be assigned to a LED color which would help human operators identify the robot from a far distance or through a surveillance camera.

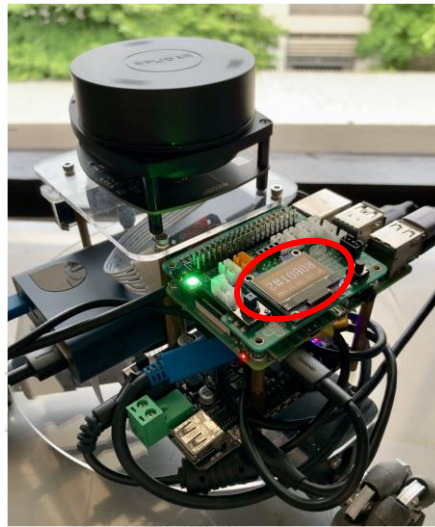


Fig. 8. LED and OLED screen output

3.2.3. Edge-level Control

The edge-level control consists of two main machines, “Edge Computing Computer” and “Edge Teleoperation Computer”. The “Edge Computing Computer” will receive all the sensor data from the high-level control for localization, path-planning, obstacle avoidance, target object search, etc. Human operators could visualize the visualization tool “rviz” which shows what the robot thinks is happening based on the given command/sensor data. Human operators use “Edge Teleoperation Computer” to work with the robot both operating manually through a control interface or commanding autonomous operation.

3.3. Robot Workspace Mapping

For the robot to navigate around the workspace, the robot requires a map. G-mapping is a SLAM approach that doesn’t require odometry but relies on only data from the LiDAR system. The robot is controlled with a joystick around the workspace. In order to acquire a good map, the robot needs to move and rotate slowly.

The data from the LiDAR that are collected at high frequency are superimposed to create a large environment map. After acquiring the map, “GIMP” program was used to manually clear map noise. The resulting map before and after clearing map noise is shown in Fig. 9. Equation (1) is used for human operator to input value for both linear velocity (V_x and V_y) and angular velocity (ω) from a control interface, i.e., joystick.

$$\begin{bmatrix} V_{right} \\ V_{left} \\ V_{center} \end{bmatrix} = \begin{pmatrix} \cos(30^\circ) & \sin(30^\circ) & R \\ -\cos(30^\circ) & \sin(30^\circ) & R \\ 0 & 1 & R \end{pmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \quad (1)$$

$V_{right}, V_{left}, V_{center}$ represent the velocity of each wheel.

V_x, V_y, ω represent the robot’s desired velocity relative to the world axis.

R represents the robot base radius

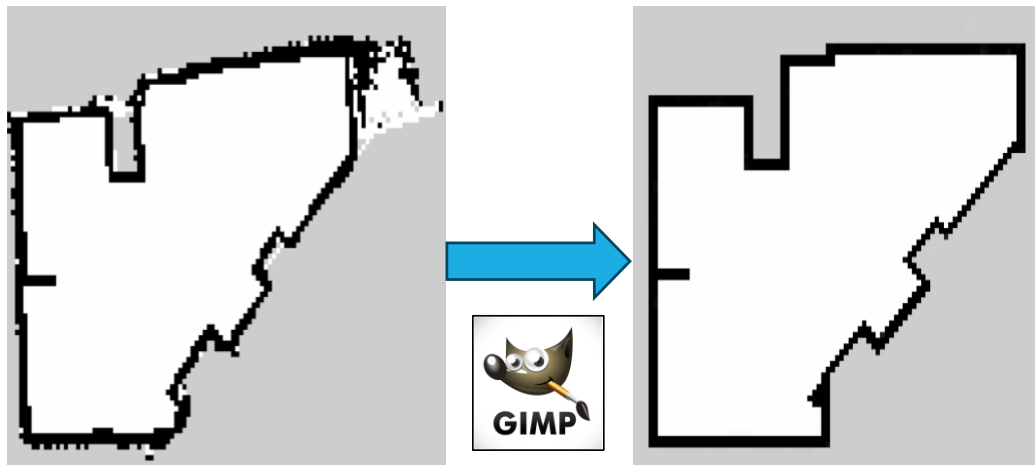


Fig. 9. Map created with G-Mapping package.

3.4. Robot Localization

During autonomous operation, localization is very important and needs to be done very accurately. The “amcl” (adaptive Monte Carlo localization) package is mainly used to localize the robot pose. The odometry calculated from the encoders and the point cloud data from the LiDAR are calculated to match the environmental map. The odometry is calculated with equation (2), the encoder data from each wheel is transformed into the distance each wheel has traveled. In this case, the motor used is GB37-En 448P/R Ratio 1:18 (37mm) with 16-bit encoder ($-2^{15} \rightarrow 2^{15}$). Noted that the encoder is collected very frequently, the change in orientation between each timestep can be assumed very small difference. Therefore, it can be assumed that

$d\theta = ds/dt$. Noted that the encoder is collected very frequently, the distance that the robot moves can be assumed to have a very small difference. The conversion between the change in distance traveled for each wheel is converted to a change in distance traveled in the x - y - θ coordinate is done with equation (2). Then the updated position of the robot is then calculated with equation (3).

$$\begin{bmatrix} d_x \\ d_y \\ d_\theta \end{bmatrix} = 2/3 \begin{pmatrix} \cos(30^\circ) & -\cos(30^\circ) & 0 \\ \sin(30^\circ) & \sin(30^\circ) & -1 \\ \frac{1}{L_0} & \frac{1}{L_0} & \frac{1}{L_0} \end{pmatrix} \begin{bmatrix} d_{right} \\ d_{left} \\ d_{center} \end{bmatrix} \quad (2)$$

d_x, d_y, d_θ represents the distance that the robot traveled relative to the world axis.

$d_{right}, d_{left}, d_{center}$ represents the distance traveled for each robot wheel.

L_0 represent the robot base width.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} + \begin{pmatrix} \cos(\theta') & -\sin(\theta') & 0 \\ \sin(\theta') & \cos(\theta') & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} d_x \\ d_y \\ d_\theta \end{bmatrix} \quad (3)$$

d_x, d_y, d_θ represents the distance that the robot traveled relative to the world axis.

x', y', θ' represents the robot position in the past timestep relative to the world axis.

x, y, θ represents the current robot position relative to the world axis.

3.4.1 Robot Self-localization for Autonomous Operation

There are several ways to initialize localization poses. For example, pre-assigned robot position and places the robot in that assigned position. However, for this autonomous operation, robots' initial location should be able to identify autonomously using a global localization service from the "amcl" package as shown in Fig. 10. Instead of robot pose possibility being placed in a small area on the map as specified by human operator, robot pose possibility is spread out across the map. As soon as the program starts, the robot will rotate about itself for 10 seconds. The robot will be able to identify its position close to its actual position.

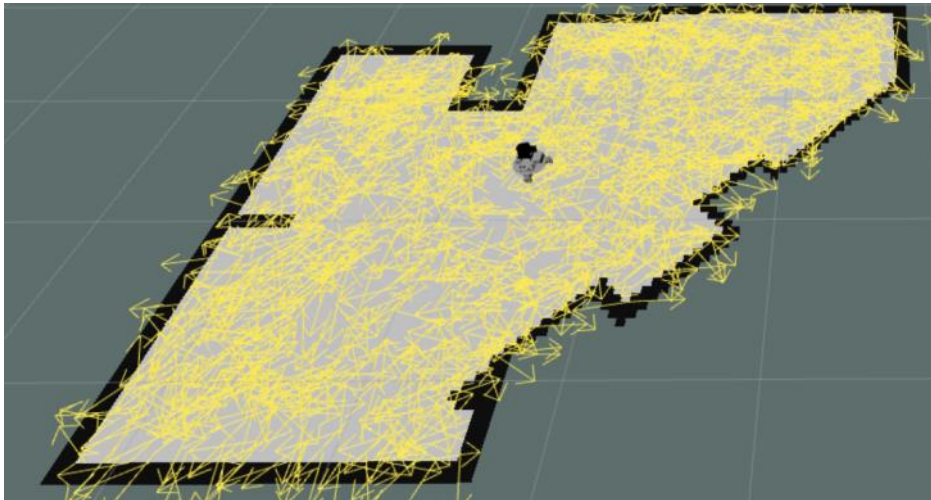


Fig. 10. Initialize robot position through a global localization service.

3.5. Robot Navigation

The robot could navigate autonomously with the use of the “move_base” package in ROS. Fig. 11 illustrates the node communication from a software development tool called “rqt”. The nodes are shown in circles while the topic is shown in rectangles. Firstly, the “/STM_Sub” node receives the encoder values from STM32 then publishes the of value into the “/rwheel”, “/cwheel”, “/lwheel” topic. The “/Omni_tf” node is responsible for calculating robot’s odometry for “/amcl” node. “/move_base” node was used to handle the path planning and obstacle avoidance using “dwalocalplanner”. The “dwalocalplanner” gathers data from “/scan” topic from “rplidarNode” node and static map from “/map_server” node to create a “localcostmap” for the robot to avoid obstacle during movement around the map. The command velocity was send with a “Twist” message type which consists of linear velocity (V_x, V_y, V_z) and angular velocity (roll, pitch, yaw). The three omni-directional wheel robot kinematics equation (1) is used to convert twist format command velocity relative to the world axis to velocity of each robot wheel. The high-level control then sends this wheel velocity to the STM32 by “STM_Pub” node.

3.6. Robot Vision

The stereo camera used in this robot could detect objects and measure distance between itself and the object of interest. First, compressed images are sent from the mobile robot. The compressed image is converted to OpenCV format. Then it was applied to a HSV filter (mask) converting image into a binary “black and white” in image according to the color of interest. The color range would be converted to white while the rest would be black. Then, based on the output of HSV filter, shape recognition will be used to create a bound base on the shape of interest. A stereo camera (like human eye) is used to find the distance between the camera and the object. The theory used is called triangulation. There are 3 main cases when categorizing image.

- 1) **“NOT FOUND”** = No color range of interest therefore shape recognition is not able to detect (shown in Fig. 13).
- 2) **“DETECTED: NOT TRACKING”** = There is a color range of interest and the shape recognition was able to detect the shape. However, when applying triangulation, the object diameter was used in parallel with the distance to verify that the object distance was too far or too near the camera. In this case, the object depth or the object diameter is out of range (shown in Fig. 14).
- 3) **“FOUND”** = Object color, the object distance, and the object diameter are in range (shown in Fig.15).



Fig. 13. Case 1: No object detected (“NO FOUND”)

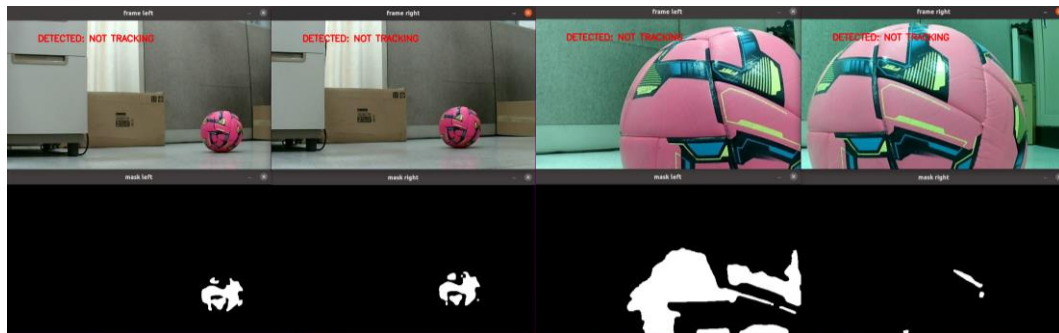


Fig. 14. Case 2: Object detected but too close or too far from the robot (“DETECTED: NOT TRACKING”).

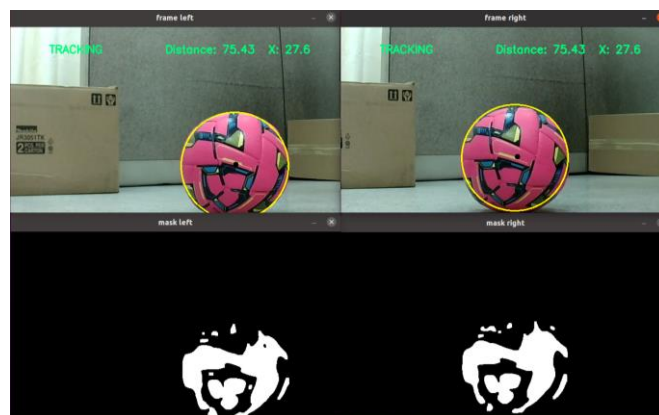


Fig. 15. Case 3: Object detected, with size and distance within range (“FOUND”).

Chapter 4: Multiple Robot System Methodology

This chapter considered a system of machines (identical standalone robots). Several robots are managed together, transitioning from “machine layer” into “cell layer”.

4.1. Multiple Robot Agent Overview

The robot is cloned from a single robot into three identical robots shown in Fig. 16. Each robot has their own on-board microcontroller. The robotic agents possess individual intelligence and decision-making capabilities. Robot agents in a swarm communicate and coordinate their actions through direct or indirect means, such as wireless communication or local sensing. For each robot to be capable of wireless communication, each robot would have its own identification (IP address) as shown in Fig.17. The system is designed so that the robots work together in the up-to-date reference map generated by a robot. As seen from Fig. 18, all the robot transformation are rooted from the same map topic. When initializing, each robot would independently initialize itself on the same map as seen from Fig.19.



Fig. 16. Fabrication of three swarm robots.

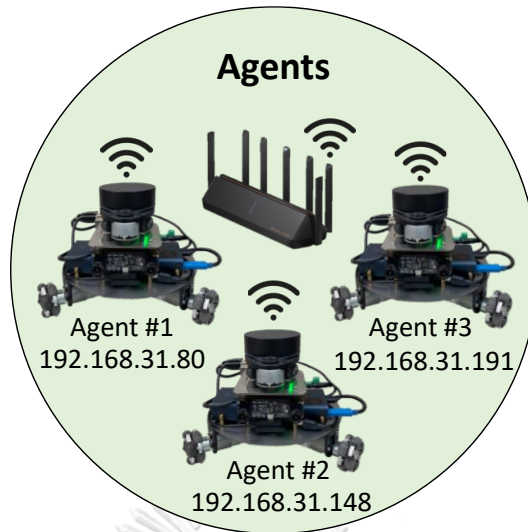


Fig. 17. Agents in the system with their IP addresses.

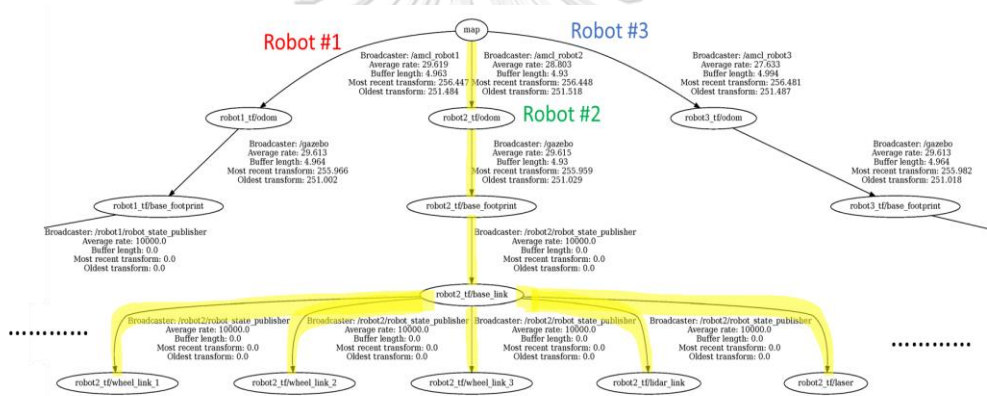


Fig. 18. “Transformation Tree” of multiple robots acquired from the URDF.

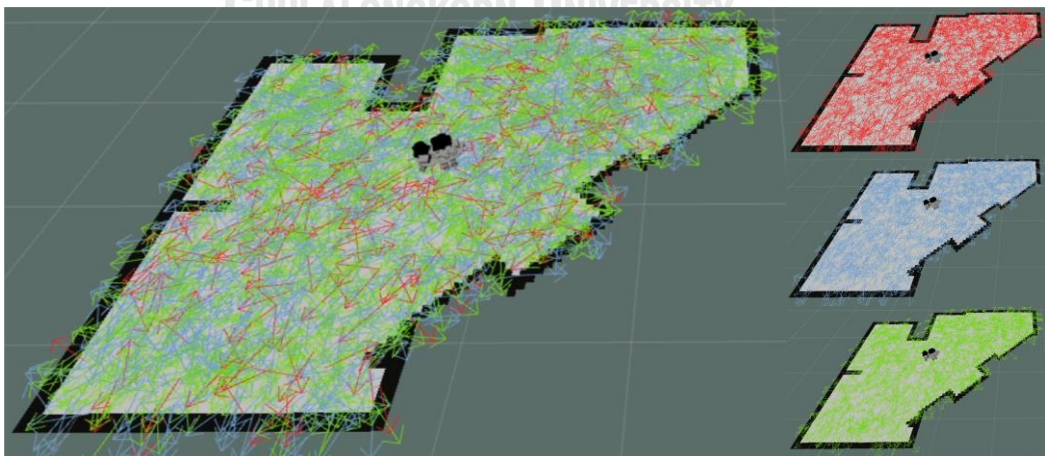


Fig. 19. Multiple robots initializing on the same map.

4.2. Edge Computing Computer Overview

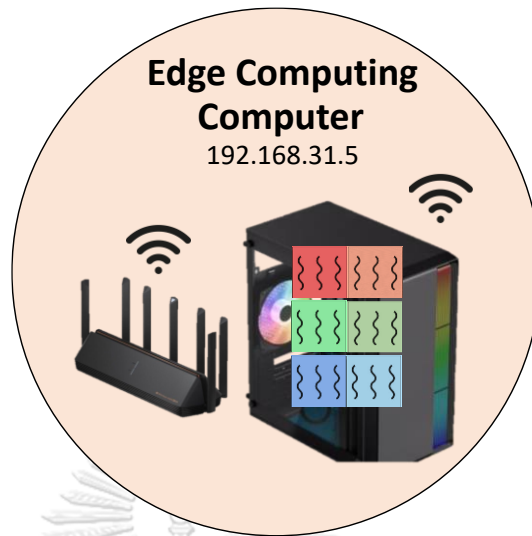


Fig. 20. Edge Computing Computer

Edge computing computer (Intel® Core™ i7-9700KF CPU @ 3.60GHz × 4) have high computation capability which could accept high amount of data from different sources. The edge computing computer is running on multiple processes of multiple threads as seen in Fig. 20. Each robot agent is connected to an allocated edge computing portion. Each portion can be divided into 2 sub-portions: “Swarm Edge Computing (SEC)” and “Robot Edge Computing (REC)”. The details of the sub-portions will be stated in the next chapter.

To operate the several robots at the same time, the edge computing computer is designed so it can launch the robot directly from itself through SSH connection as seen from Fig. 21. After the edge computing computer establish SSH connection with each robot. Each robot would run a similar node as seen from Fig. 22 consisting of

- 1) **STM_Pub** = Publish velocity command to STM32.
- 2) **STM_Sub** = Subscribe encoder data from STM32.
- 3) **hardware, controller_starter** = Acquire image from the camera hardware and publish the image into edge computing computer.
- 4) **rplidarNode** = Acquire laser data from LiDAR and publish it to edge computing computer.


```

started roslaunch server http://192.168.31.5:33351/
remote[192.168.31.80-0] starting roslaunch
remote[192.168.31.80-0]: creating ssh connection to 192.168.31.80:22, user[ubuntu]
launching remote roslaunch child with command: [env ROS_MASTER_URI=http://192.168.31.5:11311
/home/ubuntu/catkin_ws/devel/env.sh roslaunch -c 192.168.31.80-0 -u http://192.168.31.5:33351
/ --run_id b2c52772-0bed-11ee-82d0-59fb37d57179 --sigint-timeout 15.0 --sigterm-timeout 2.0]
remote[192.168.31.80-0]: ssh connection created
remote[192.168.31.148-1] starting roslaunch
remote[192.168.31.148-1]: creating ssh connection to 192.168.31.148:22, user[ubuntu]
launching remote roslaunch child with command: [env ROS_MASTER_URI=http://192.168.31.5:11311
/home/ubuntu/catkin_ws/devel/env.sh roslaunch -c 192.168.31.148-1 -u http://192.168.31.5:33351
/ --run_id b2c52772-0bed-11ee-82d0-59fb37d57179 --sigint-timeout 15.0 --sigterm-timeout 2.0]
remote[192.168.31.148-1]: ssh connection created
remote[192.168.31.191-2] starting roslaunch
remote[192.168.31.191-2]: creating ssh connection to 192.168.31.191:22, user[ubuntu]
launching remote roslaunch child with command: [env ROS_MASTER_URI=http://192.168.31.5:11311
/home/ubuntu/catkin_ws/devel/env.sh roslaunch -c 192.168.31.191-2 -u http://192.168.31.5:33351
/ --run_id b2c52772-0bed-11ee-82d0-59fb37d57179 --sigint-timeout 15.0 --sigterm-timeout 2.0]
remote[192.168.31.191-2]: ssh connection created

```

Fig. 21. Initializing SSH connection with multiple robots in the terminal.

```

MACHINES
* robot1
* robot2
* robot3

NODES
/robot1/
STM_Pub (arobota2/publish_STM_omni.py)
STM_Sub (arobota2/recieve_STM_omni.py)
controller_starter (controller_manager/controller_manager)
hardware (nodelet/nodelet)
rplidarNode (rplidar_ros/rplidarNode)
/robot2/
STM_Pub (arobota2/publish_STM_omni.py)
STM_Sub (arobota2/recieve_STM_omni.py)
controller_starter (controller_manager/controller_manager)
hardware (nodelet/nodelet)
rplidarNode (rplidar_ros/rplidarNode)
/robot3/
STM_Pub (arobota2/publish_STM_omni.py)
STM_Sub (arobota2/recieve_STM_omni.py)
controller_starter (controller_manager/controller_manager)
hardware (nodelet/nodelet)
rplidarNode (rplidar_ros/rplidarNode)

```

Fig. 22. Launching all on-board program for all robots in the terminal.

4.3. Edge Teleoperation Computer Overview

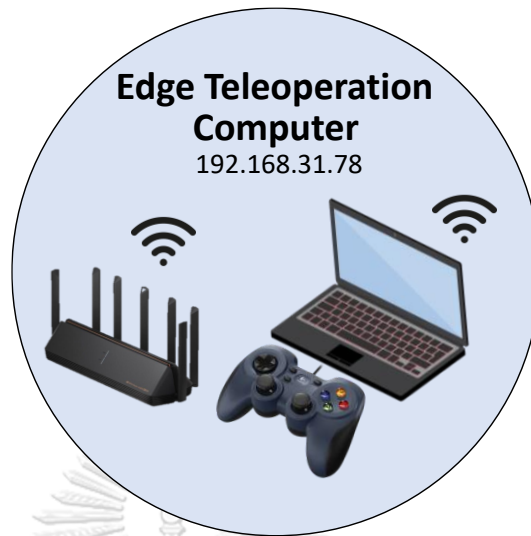


Fig. 23. Edge Teleoperation Computer

In real applications, human operators and edge computing computers may not be working in the same location. However, human operators should have control of the edge computing computer. In this research, VNC (Virtual Network Computing), a graphical desktop-sharing system was used. Another benefit of having a computer for teleoperation is giving human operators an option to manually control robots through different command interfaces (joystick, VR controller, etc.) seen in Fig. 23. For example, using a joystick to control the robot while creating a map.

4.4. Network Architecture

For the system to achieve maximum performance, every device should be connected to 5GHz band rather than 2GHz band since it provides higher speed data transfer and higher bandwidth resulting in lower latency. However, one problem that deters mobile robots from moving in large workspace is the 5GHz Wi-Fi network supporting only about 15 meters from the router. Therefore, the system was integrated with mesh Wi-Fi system. With these requirements, the experiment was conducted using Xiaomi WiFi6 mesh router. It supports IEEE 802.11ax protocol with a maximum bandwidth of 160MHz (bit/sec) and maximum speed of 4804 Mbps on 5GHz band. Moreover, with the 4K QAM technology high-speed transmission, increase data transmission by 20% due to data compression. The network should be able to perform handovers between several routers as robots move from place to

place. As shown in Fig. 24, the network architecture is designed so that the human operators, edge computers, and robot agents don't need to be close to each other. For example, human operators could teleoperate the swarm in the control room and the edge computing computer can be in the server room. While the robot agents could be working anywhere around the workspace. As shown from Fig. 24, the general workflow of the system is robot agents in the system send sensor data to the edge computing computer. Edge computing computer has high computation capability to accept high amount of data from different robot agents at once. Instead of robot computing on-board, it would be more efficient to compute externally and send the result back. The robot agent needs to send the compress camera data to the edge computing to perform computer vision analyze task. LiDAR and wheel encoder data are sent from each robot to perform localization on the edge computing computer. The architecture relies on high data exchange between the robot and the edge computing computer rather than relying on the robot computing power on-board.

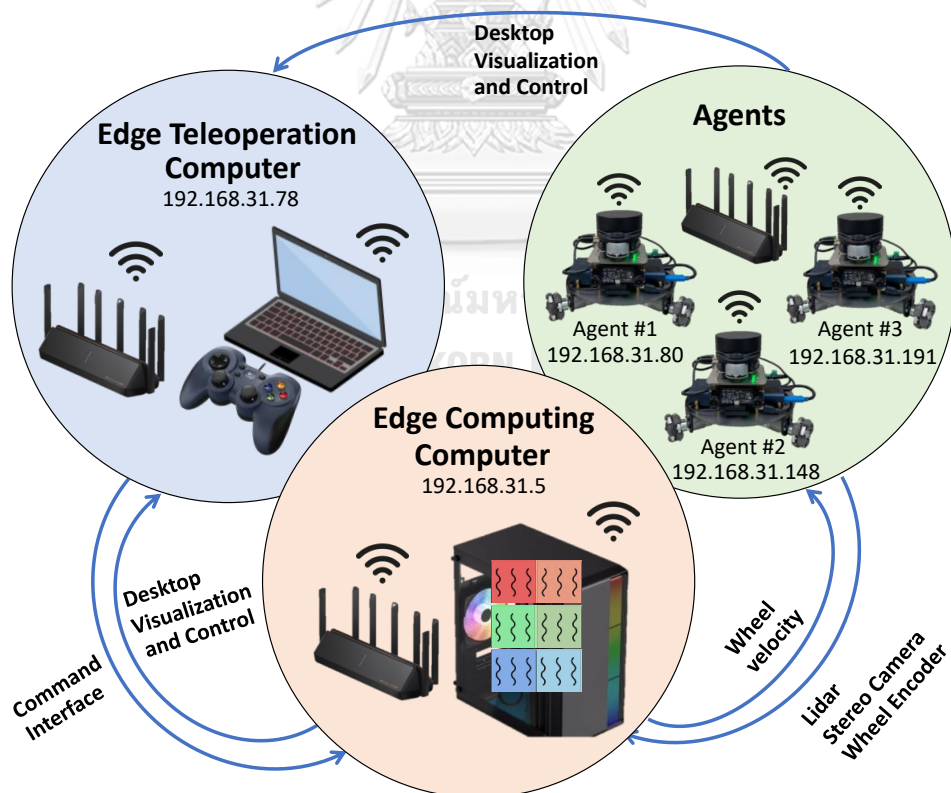


Fig. 24. Network Architecture

Chapter 5: Formation Control of Robot Swarm

In the previous chapter, it was established that multiple robot systems can communicate with each other through a proposed network architecture, with the support of the edge computing computer. In this chapter, the formation control algorithm will be proposed to create a swarm collective behavior and have interactions between individual robots.

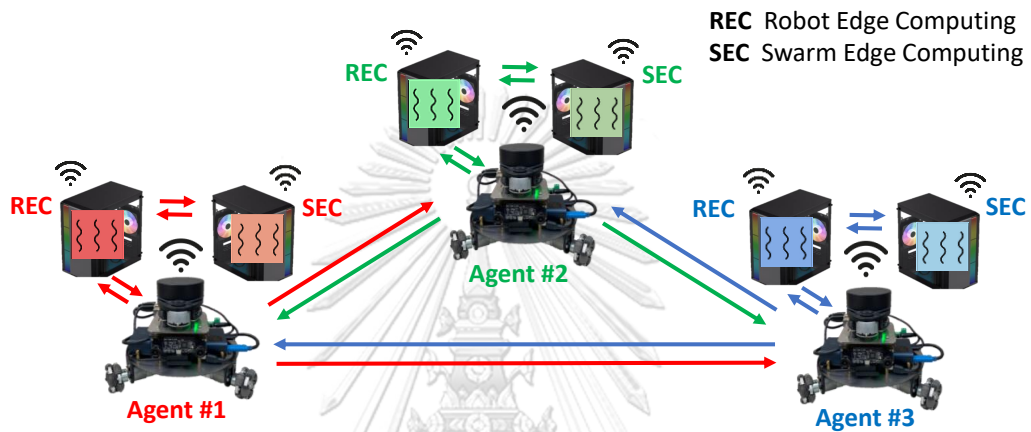


Fig. 26. Sub-portions of edge computing for each robot in the swarm.

As stated in the previous chapter, the edge computing for each robot consists of two sub-portions as seen in Fig. 26. First, the “Swarm Edge Computing (SEC)” can be compared to the innate instinct of an animal. The innate sense, known as “Swarm Edge Computing”, enables each robot to know the general routes of the swarm. This is represented by the reference_goal_point (\vec{q}) shown in an orange point. Second, “Robot Edge Computing (REC)”, which can be compared to a learned navigational skill of an animal. The brain-like function of the “Robot Edge Computing” plays a crucial role in processing sensory information, guiding navigation, and coordinating various tasks. When robots move together, each robot's pose in the formation is flexible and can change based on individual behavior due to environmental changes.

As shown in Fig. 27, the innate instinct of the swarm route will be pre-assigned as waypoint shown in orange point. Each “Robot Edge Computing”, with the given formation parameters (d_i, α_i, β_i) would control their robot to achieve the goal_pose ($\vec{p}_1, \vec{p}_2, \vec{p}_3$) shown in red, green, and blue respectively.

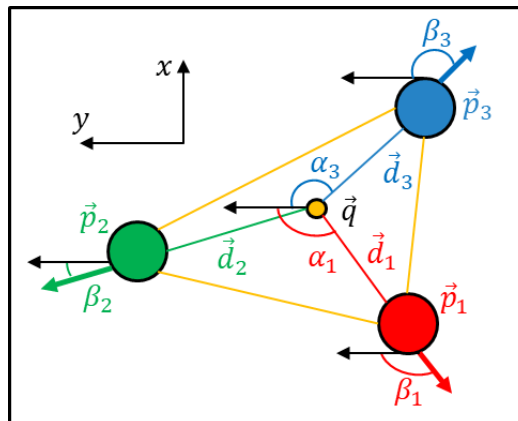


Fig. 27. Formation Control Parameters

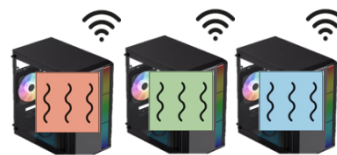
5.1. Swarm Edge Computing

Algorithm: Multiple Robot with Formation Control Survey
(Swarm Edge Computing)

```

Subscribe: initialize_state, robot_statusi, custom_waypoint
Publish : reference_goal_point ( $\vec{q}$ ), swarm_status
1 if finished_initialize_state then
2   read custom_waypoint;
3   for waypoint in custom_waypoint do
4     set swarm_status to FALSE;
5     return swarm_status ;
6     return reference_goal_point ( $\vec{q}$ ) ;
7     while waiting_for_each_robot_statusi do
8       if all_roboti_reached_goal then
9         set swarm_status to TRUE ;
10        return swarm_status ;
11        break ;
12      end
13    end
14  end

```



A human operator preassigns a custom_waypoint by specifying any x-y-coordinate on the map. The swarm's reference_goal_point passes through this preassigned waypoint. Once the entire swarm is initialized and the pose of each robot in the swarm pose is known, each "Swarm Edge Computing" begins by sending the first reference_goal_point to each corresponding "Robot Edge Computing". Each robot then exchanges its status with other robots, indicating whether it has successfully reached its goal. When every robot in the swarm reaches their goal, each "Swarm Edge Computing" sends the next reference_goal_point to its respective "Robot Edge Computing".

5.2. Robot Edge Computing

Algorithm: Multiple Robot with Formation Control Survey
(Robot Edge Computing)

```

Subscribe: swarm_status, reference_goal_point ( $\vec{q}$ ) ,
             custom_formation( $d_i, \alpha_i, \beta_i$ )
Publish : robot_status_i, goal_pose ( $\vec{p}_i, \theta_i$ )
1 while TRUE do
2   set robot_status_i to FALSE ;
3   return robot_status_i ;
4   read reference_goal_point ;
5    $\vec{d}_i = [ d_i \times \cos(\theta_i), d_i \times \sin(\theta_i), 0 ]$ ;
6    $\vec{p}_i = \vec{q}_i + \vec{d}_i$  ;
7    $\theta_i = \beta_i$  ;
8   return goal_pose ;
9   while not swarm_status do
10    if robot_i reached goal then
11      set robot_status_i to TRUE ;
12      return robot_status_i ;
13    end
14 end

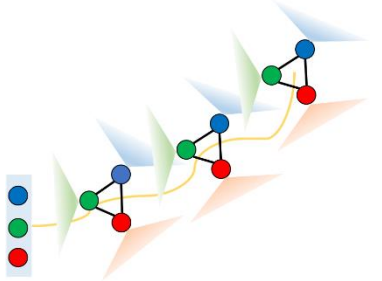
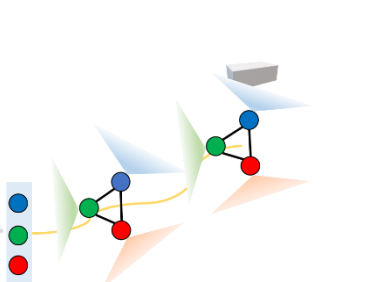
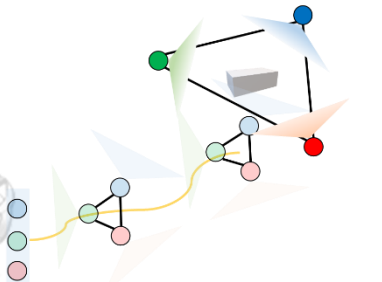
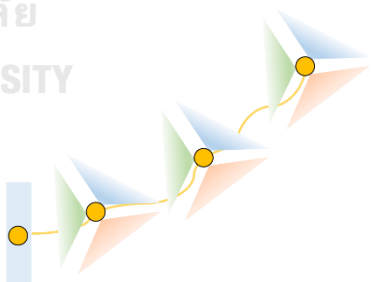
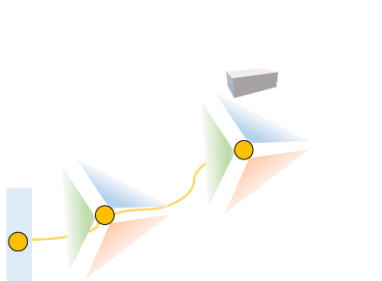
```



A human operator specifies the formation, including the distance (d_i) and angle (α_i) from the reference_goal_point to each robot, as seen in Fig. 23. Additionally, the human operator would specify the robot orientation (β_i). When the robots receive the reference_goal_point, the reference_goal_point and formation parameters will undergo some mathematical calculations. After the calculations, each robot could determine its own goal_pose. As specified in Chapter 3, the “amcl” and “movebase” packages are used in combination. The “amcl” updates the robot’s current pose and the “movebase” takes the updated pose and the goal_pose as input, generating target velocity in x-axis, target velocity in y-axis, and angular velocity as output. At the high-level control, according to equation (1), these velocities are converted into velocity command for each wheel (right wheel, left wheel, and center wheel) and sent to the low-level control to control real robot hardware.

When each robot reaches its goal, it provides feedback on its robot_status to its respective “Swarm Edge Computing”. Each robot waits for the swarm_status, ensuring that every robot has reached its goal. Then, each robot is ready to accept a new waypoint from the “Swarm Edge Computing”. The algorithm repeats indefinitely.

Chapter 6: Experimental Scenarios

	Experiment Name	Graphic Representation
Experiment #1	Multiple Robot with Formation Control <u>Survey</u>	
Experiment #2	Multiple Robot with Formation Control <u>Object Search</u>	
Experiment #3	Multiple Robot with Formation Control <u>Object Search</u> and <u>Surround Target Object</u>	
Experiment #4	Single Robot <u>Survey</u>	
Experiment #5	Single Robot <u>Object Search</u>	

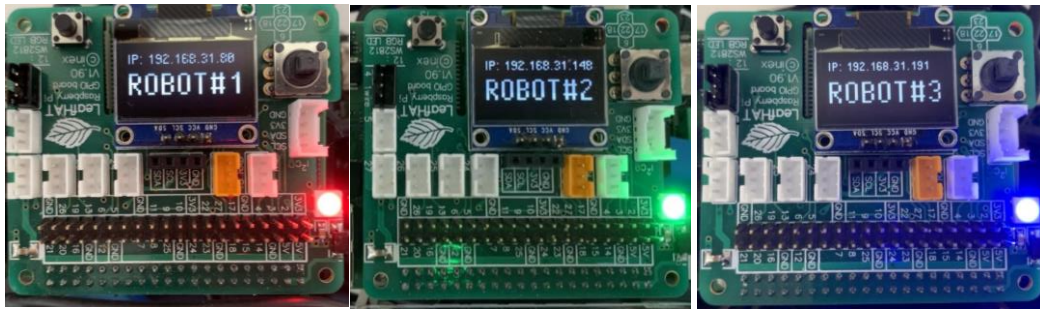


Fig. 28 OLED screen and LED output for multiple robot experiment

For multiple robot operation, robot name and LED color are preassigned as shown in Fig. 28 and summarized in the table below.

Robot Number	LED color	IP address
Robot #1	RED	192.168.31.80
Robot #2	GREEN	192.168.31.148
Robot #3	BLUE	192.168.31.191



Fig. 29. OLED screen and LED output for single robot experiment

For a single robot operation, robot name and LED color are preassigned as shown in Fig. 29 and summarized in the table below.

Robot Number	LED color	IP address
Robot #1	YELLOW	192.168.31.80

The camera algorithm has been introduced in Chapter 3. In all these experiments, robots are required to search for a “pink ball” in the workspace. For HSV_filter, the lower bound value (100, 100, 100) and the higher bound value (180, 255, 225) are used to filter only pink color to pass through as seen in the left image of Fig. 30. Since the ball is our target object, the shape_recognition is used to find the

minimum enclosing circle (shown in yellow) as seen from the right image of Fig. 30. However, there may be cases where the color or the shape pass through the HSV_filter and shape_recognition but is not our target object. Therefore, there are several conditions used to identify whether the detected object is our target object which can be seen from the algorithm below.

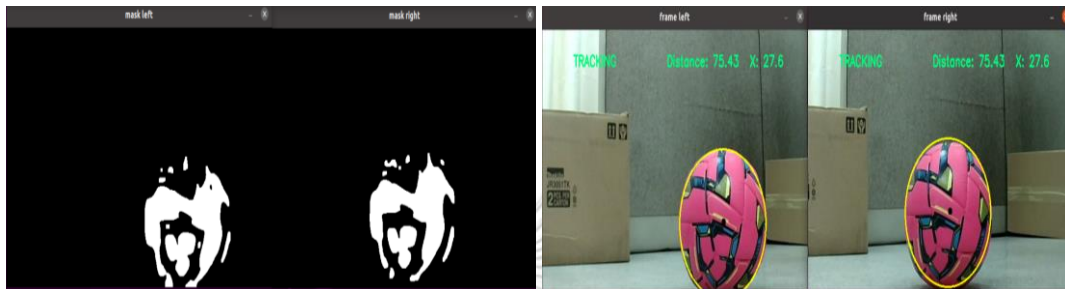


Fig. 30 HSV_filter and shape_recognition used to find the target object.

Algorithm: Object Detection (OpenCV)

```

Subscribe: compressed_image
Publish : object_depth, camera_interrupt, camera_status
1 while TRUE do
2   read compressed_image ;
3   convert compressed_image to OpenCV format ;
4   apply HSV_filter ;
5   apply shape_recognition ;
6   apply triangulation ;
7   if object not detected then
8     set camera_status to "NOT_FOUND" ;
9   else
10    return object_depth ;
11    if  $40 < object\_depth < 80$  and  $110 < object\_diameter < 200$  then
12      set camera_status to "TRACKING";
13      set camera_interrupt to TRUE;
14      return camera_interrupt ;
15    else
16      set camera_status to "DETECTED: NOT TRACKING"
17    end
18  end
19  return camera_status ;
20  convert from OpenCV format to compressed_image ;
21 end

```

6.1. Formation Control of Multiple Robot Experiment

For multiple robot operations (experiment #1, #2, #3). The formation is design to be a fixed equilateral triangle formation with $\alpha_1 = 120^\circ, \alpha_2 = 0^\circ, \alpha_3 = -120^\circ$ relative to centroid. The distance between the robot and the centroid is $d_1 = d_2 = d_3 = 0.25$ meter. The orientation of each robot swarm is faced out of the

centroid position ($\beta_1 = 120^\circ, \beta_2 = 0^\circ, \beta_3 = -120^\circ$) as seen in Fig. 31. Each robot will take charge of 72-degree vision. With this formation, the robot swarm system can visualize almost the whole workspace while moving around.

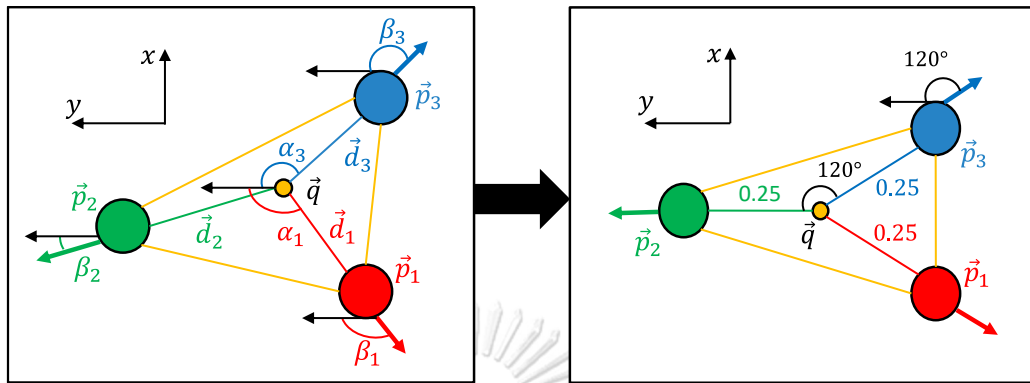


Fig. 31. Formation control configuration for multiple robots in this experiment

6.1.1. Experiment #1

A total of 12 reference waypoints are set throughout the workspace. At the start, the robot swarms will self-initialize in the initialize zone. After each robot knows their initial position and orientation. The “Swarm Edge Computing” sends its first reference goal to the “Robot Edge Computing”. Since each robot knows its identification, it was able to calculate their first goal position and orientation shown in Fig. 32. For each reference_goal_point, if all robot reached their goal_pose, each robot would receive a new reference goal. This loop will continue forever until the 12 waypoints are achieved. Fig. 33 shows a snapshot toward the end of the survey operation, the formation still is kept. The algorithm details can be found on the following page.

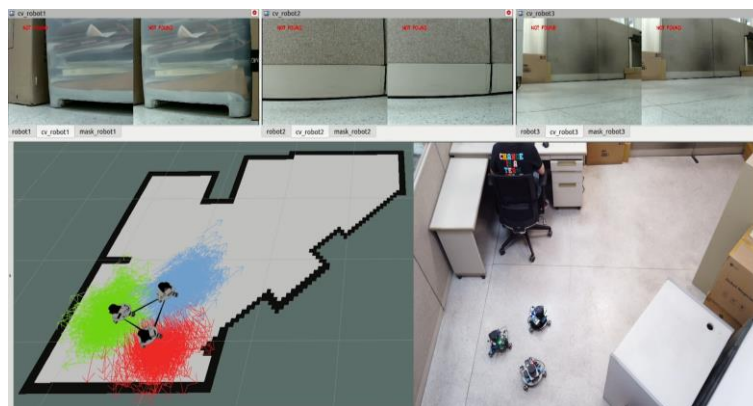


Fig. 32 Robot swarm reached the first waypoint.



Fig. 33. Robot approaching the end of the survey.

Algorithm: Multiple Robot with Formation Control Survey
(Swarm Edge Computing)

Subscribe: initialize_state, robot_status_{*i*}, custom_waypoint
Publish : reference_goal_point (\vec{q}), swarm_status

```

1 if finished initialize_state then
2   read custom_waypoint;
3   for waypoint in custom_waypoint do
4     set swarm_status to FALSE;
5     return swarm_status ;
6     return reference_goal_point ( $\vec{q}$ ) ;
7     while waiting for each robot_statusi do
8       if all roboti reached goal then
9         set swarm_status to TRUE ;
10        return swarm_status ;
11        break ;
12      end
13    end
14 end

```

Algorithm: Multiple Robot with Formation Control Survey
(Robot Edge Computing)

Subscribe: swarm_status, reference_goal_point (\vec{q}) ,
 custom_formation(d_i, α_i, β_i)
Publish : robot_status_{*i*}, goal_pose (\vec{p}_i, θ_i)

```

1 while TRUE do
2   set robot_statusi to FALSE ;
3   return robot_statusi ;
4   read reference_goal_point ;
5    $\vec{d}_i = [ d_i \times \cos(\theta_i), d_i \times \sin(\theta_i), 0 ]$  ;
6    $\vec{p}_i = \vec{q}_i + \vec{d}_i$  ;
7    $\theta_i = \beta_i$  ;
8   return goal_pose ;
9   while not swarm_status do
10    if roboti reached goal then
11      set robot_statusi to TRUE ;
12      return robot_statusi ;
13    end
14 end

```

6.1.2. Experiment #2

In this experiment, most parts of the algorithm are similar to the first experiment. A similar set of 12 reference waypoints are set throughout the workspace. Fig. 34 shows that multiple robots are initializing to find their initial pose. In this experiment, the target object “pink ball” is placed on the floor for the robot swarm to search for. The robot swarm will navigate together around the map until it detects the required object that is in range. When any robot detects the object, that robot would communicate with the “Robot Edge Computing” of other robot to interrupt their motion. The ball position is found by a math formula shown in the “Robot Edge Computing” algorithm on the following page. When the “Robot Edge Computing” publishes the ball_position, it will be shown in the graphical interface as seen in Fig. 35. The algorithm details can be found on the following page.

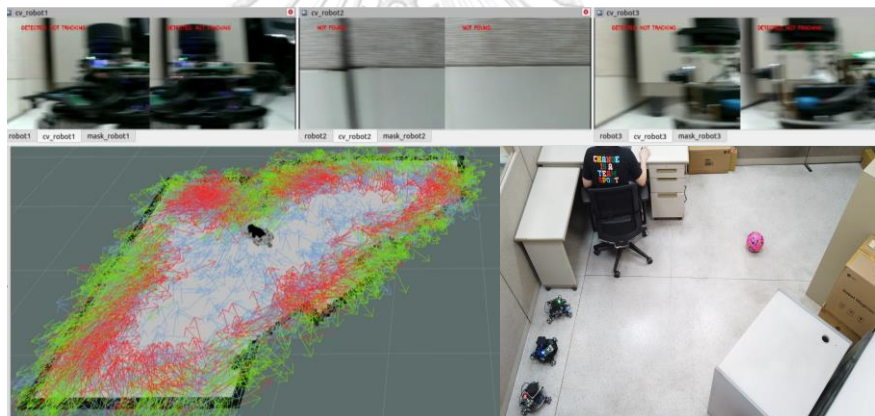


Fig. 34. Multiple robots in the process of initializing their current pose.

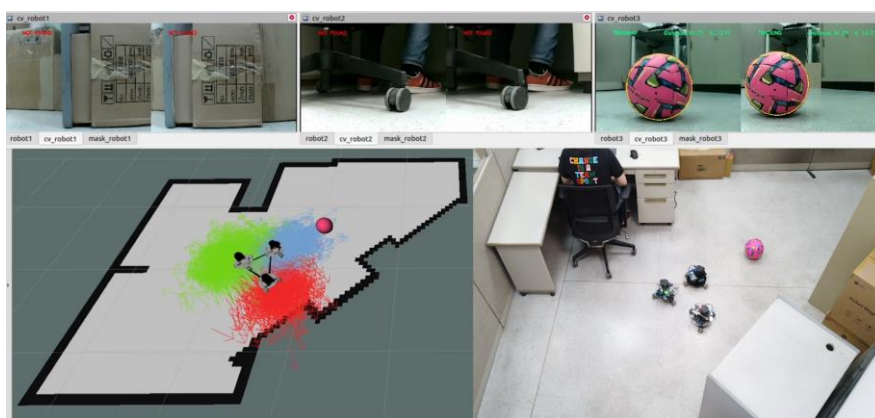


Fig. 35 All robots stop when one robot finds the target object.

Algorithm: Multiple Robot with Formation Control Object Search (Swarm Edge Computing)

Subscribe: initialize_state, robot_status_i, custom_waypoint, camera_interrupt

Publish : reference_goal_point (\vec{q}), swarm_status

```

1 if finished initialize_state then
2   read custom_waypoint;
3   for waypoint in custom_waypoint do
4     set swarm_status to FALSE;
5     return swarm_status ;
6     return centroid_goal_point ( $\vec{q}$ ) ;
7     while waiting for each robot_statusi do
8       if all roboti reached goal then
9         set swarm_status to TRUE ;
10        return swarm_status ;
11        break ;
12        if camera_interrupt then
13          End Program;
14        end
15      end
16    end
17 end

```

Algorithm: Multiple Robot with Formation Control Object Search (Robot Edge Computing)

Subscribe: swarm_status, reference_goal_point (\vec{q}) , camera_interrupt, camera_status, object_depth, custom_formation(d_i, α_i, β_i)

Publish : robot_status_i, goal_pose (\vec{p}_i, θ_i), object_position (\vec{q})

```

1 while TRUE do
2   set robot_statusi to FALSE ;
3   return robot_statusi ;
4   read reference_goal_point ;
5    $\vec{d}_i = [ d_i \times \cos(\theta_i), d_i \times \sin(\theta_i), 0 ]$  ;
6    $\vec{p}_i = \vec{q}_i + \vec{d}_i$  ;
7    $\theta_i = \beta_i$  ;
8   return goal_pose ;
9   while not swarm_status do
10    if roboti reached goal then
11      set robot_statusi to TRUE ;
12      return robot_statusi ;
13    if camera_interrupt then
14      return stop robot ;
15    if camera_status equal "TRACKING" then
16       $q_{i_x} = p_{i_x} + (object\_depth \times \sin(\theta_i))$  ;
17       $q_{i_x} = p_{i_x} + (object\_depth \times \cos(\theta_i))$  ;
18      return object_position( $\vec{q}$ ) ;
19    end
20    End Program;
21  end
22 end
23 end

```

6.1.3. Experiment #3

This experiment is the extension of experiment #2. After the target object is found, that robot would communicate with the “Robot Edge Computing” of other robot to interrupt their motion. The robot which detected the target object will use its stereo camera to detect the depth between the robot and the ball. The position of the ball is sent to the “Swarm Edge Computing” of all robots. The ball became the new reference_goal_point (orange point) and the robot will form a new formation with the parameter ($\beta_1 = -60^\circ, \beta_2 = 180^\circ, \beta_3 = 60^\circ, \alpha_1 = 120^\circ, \alpha_2 = 0^\circ, \alpha_3 = -120^\circ, d_1 = d_2 = d_3 = 0.55$) as seen in Fig. 36. Each robot will move from the position they were interrupted to a new goal pose assigned by the “Robot Edge Computing” as seen in Fig. 37. At the goal pose, the robots will surround the object and the robot’s orientation will face the object and will be able to detect the ball as shown in Fig. 38. The algorithm details can be found on the following page.

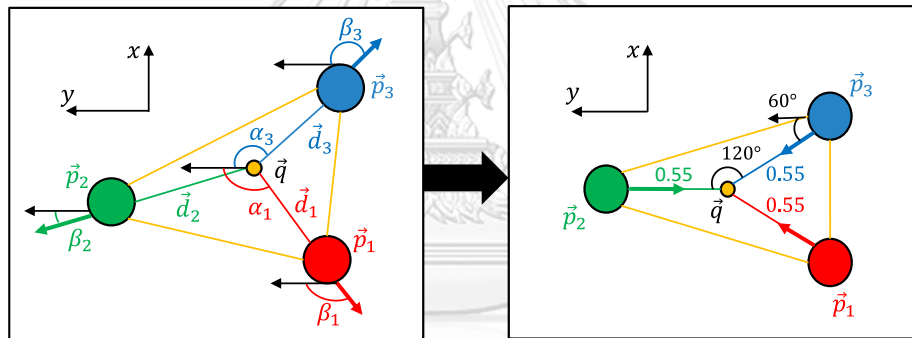


Fig. 36. Formation control parameter when surrounding the target object.

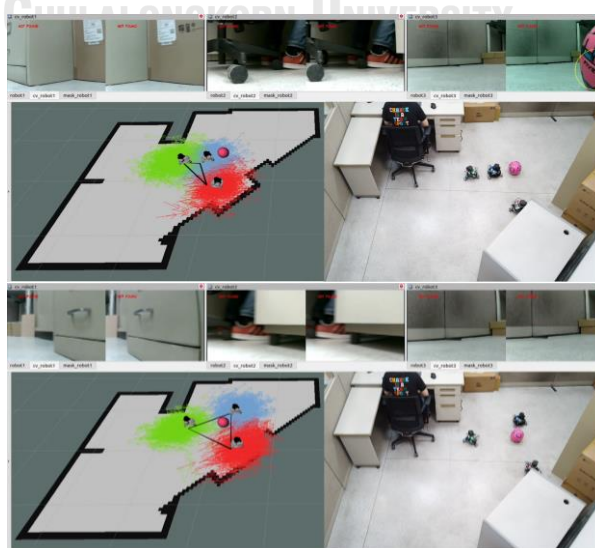


Fig. 37. All robots are in the process of surrounding the ball.

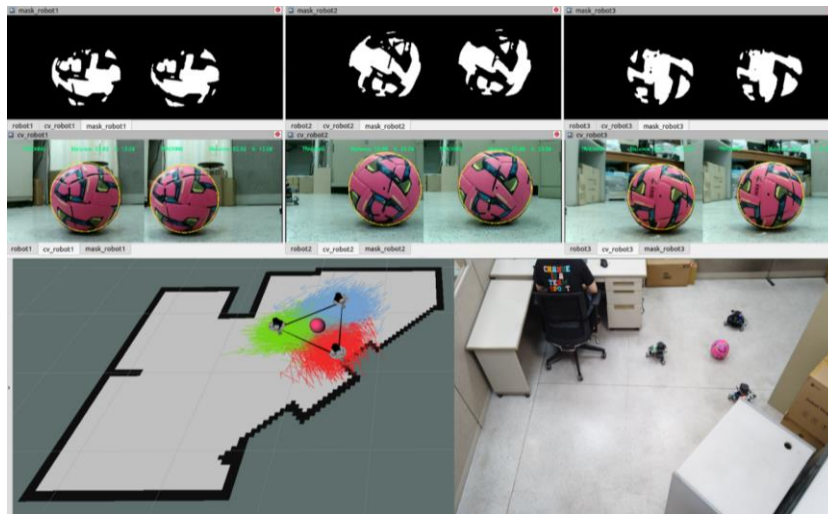


Fig. 38. All robots surrounded and faced the ball.

Algorithm: Send Target Object Position (Swarm Edge Computing)

Subscribe: camera_interrupt, object_position(\vec{q}),
 custom_formation(d_i, α_i)
Publish : reference_goal_point (\vec{q})

- 1 **if** camera_interrupt **then**
- 2 **read** object_position ;
- 3 **return** reference_goal_point ;
- 4 **End Program**
- 5 **end**

Algorithm: Multiple Robot with Formation Control Surround Target Object (Robot Edge Computing)

Subscribe: swarm_status, reference_goal_point (\vec{q}),
 custom_formation(d_i, α_i, β_i)
Publish : robot_status $_i$, goal_pose (\vec{p}_i, θ_i)

- 1 **while** TRUE **do**
- 2 **set** robot_status $_i$ to FALSE ;
- 3 **return** robot_status $_i$;
- 4 **read** reference_goal_point ;
- 5 $\vec{d}_i = [d_i \times \cos(\theta_i), d_i \times \sin(\theta_i), 0]$;
- 6 $\vec{p}_i = \vec{q}_i + \vec{d}_i$;
- 7 $\theta_i = \beta_i$;
- 8 **return** goal_pose ;
- 9 **while** not swarm_status **do**
- 10 **if** robot $_i$ reached goal **then**
- 11 **set** robot_status $_i$ to TRUE ;
- 12 **return** robot_status $_i$;
- 13 **end**
- 14 **end**

6.2. Single Robot Experiment

These single robot experiment as conducted for a single robot to have vision coverage similar to of a three robots survey/search at each waypoint. Therefore, as seen from Fig. 39, at each waypoint the robot will stop at 3 different orientations (seen in blue, green, and red arrow) similar to the orientation of each robot in the multiple robot operation. ($\beta_1 = 120^\circ, \beta_2 = 0^\circ, \beta_3 = -120^\circ$)

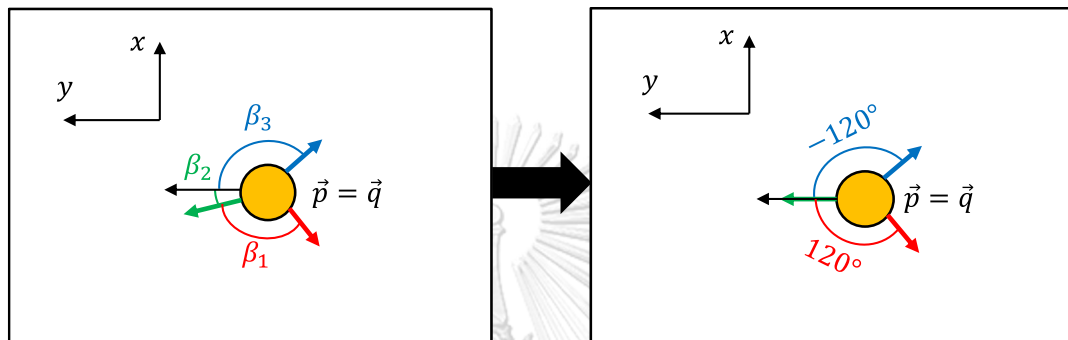


Fig. 39. Single robot orientation assignment.

6.2.1. Experiment #4

In this experiment, a single robot navigates to 12 waypoints with similar coordinates. However, for each waypoint the robot will stop at 3 different orientations. Fig. 40 shows the mobile in 3 different waypoints and 3 different orientations. The algorithm details can be found on the following page.



Fig. 40 Three different orientations at three different waypoints for a single robot operation

Algorithm: Single Robot Survey

```

Subscribe: initialize_state, custom_waypose ( $\vec{q}, \beta_i$ ), robot_status
Publish : goal_pose ( $\vec{p}, \theta_i$ )
if finished initialize_state then
  read custom_waypose ;
  for waypoint ( $\vec{q}$ ) in custom_waypose do
    for wayorientation ( $\beta_i$ ) in custom_waypose do
      set robot_status to WAIT ;
       $\vec{p} = \vec{q}$  ;
       $\theta_i = \beta_i$  ;
      return goal_pose ;
      while waiting robot_status do
        if robot reached goal then
          break ;
        end
      end
    end
  end
end

```

6.2.2. Experiment #5

In this experiment, a “pink ball” is placed in the workspace with the ball position similar to multiple robot search operation (Experiment #2). The robot is assigned to the same 3 orientation in each of the 12 waypoints in Experiment #4. However, when the robot camera detects the target object, the robot will stop its motion and end the algorithm as seen from Fig. 41. The algorithm details can be found on the following page.

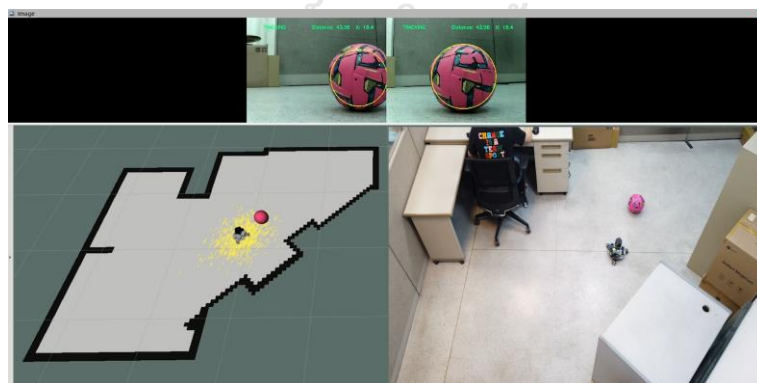


Fig. 41 All robots stop when one robot finds the target object.

Algorithm: Single Robot Object Search

Subscribe: initialize_state, custom_waypose (\vec{q}, β_i), robot_status, camera_interrupt, object_depth

Publish : goal_pose (\vec{p}, θ_i), object_position (\vec{q})

```

if finished initialize_state then
  read custom_waypose ;
  for waypoint ( $\vec{q}$ ) in custom_waypose do
    for wayorientation ( $\beta_i$ ) in custom_waypose do
      set robot_status to WAIT ;
       $\vec{p} = \vec{q}$  ;
       $\theta_i = \beta_i$  ;
      return goal_pose ;
      while waiting robot_status do
        if robot reached goal then
          break ;
        if camera_interrupt then
           $q_x = p_x + (\text{object\_depth} \times \sin(\theta_i))$  ;
           $q_y = p_y + (\text{object\_depth} \times \cos(\theta_i))$  ;
          return object_position ( $\vec{q}$ ) ;
        End Program
      end
    end
  end
end
end

```



Chapter 7: Experimental Results

From the previous chapter, five different experiment scenarios are conducted involving both multiple robots and a single robot. Various data were logged, including the robots' position and orientation, centroid position of multiple robots, duration of operation, and other relevant parameters. The data is imported into MATLAB to plot the results and used to evaluate the effectiveness of the formation control algorithm.

7.1. Robot Survey: Path

Experiment #1 and Experiment#4 were conducted to compare the survey path of the robots in the absence of target objects in the workspace. During these experiments, the robots followed 12 identical waypoint coordinate sets. By examining the robot path plots, shown in Fig. 42 and Fig. 43 respectively, we can observe that the single robot operation demonstrates a more optimized motion, moving in a straight line rather than a curved trajectory observed in the multiple robot formation control. The curved paths of the centroid in multiple robot operation can be attributed to the fact that each robot aims to reach its goal position. However, due to the implemented formation control, the robots are near each other, which restricts their freedom of movement. Each robot perceives the other two robots as dynamic obstacles. Consequently, when one robot is positioned ahead, other robots in the swarm cannot move in a straight line from point to point. This leads to curved or suboptimal paths.

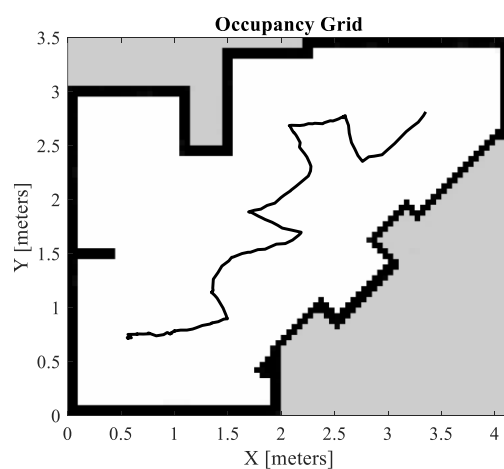


Fig. 42 Multiple robots survey: centroid position (Experiment#1)

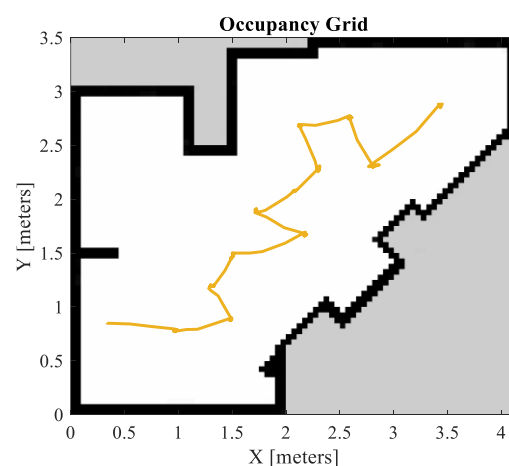


Fig. 43 Single robot position for survey operation (Experiment#4)

7.2. Robot Survey: Vision Coverage and Survey Time

The vision coverage and survey time are compared between the three-robot formation control and a single robot operation (Experiment #1 and Experiment #4 respectively). The robot's orientation can be implied as the robot's camera vision. From Fig. 44, as soon as all three robots have entered formation on their first waypoint, each robot takes charge of a specified range of vision, and the vision of the three robots barely overlaps. It should be noted that the angle of view for each stereo camera is 72 degrees. Therefore, it can be deduced that as the swarm is moving around the workspace, would have a coverage vision of approximately 216 degrees.

However, in a single robot, the algorithm is designed to have vision of coverage similar to multiple robots. The single robot orientation (vision) can be seen in Fig. 45. The robot stops at each waypoint to observe around itself before moving to the next waypoint. The blue circle shows the 12 waypoints with similar orientation, it can be observed that there is a gap in the vision coverage.

Based on the experiment, the proposed formation control algorithm for the three-robot formation took approximately 85.105 seconds to complete the survey of the workspace, whereas it took approximately 187.35 seconds for a single robot to accomplish the same task. Therefore, in this survey scenario, the three-robot formation control was approximately 2.2 times faster than the single robot.

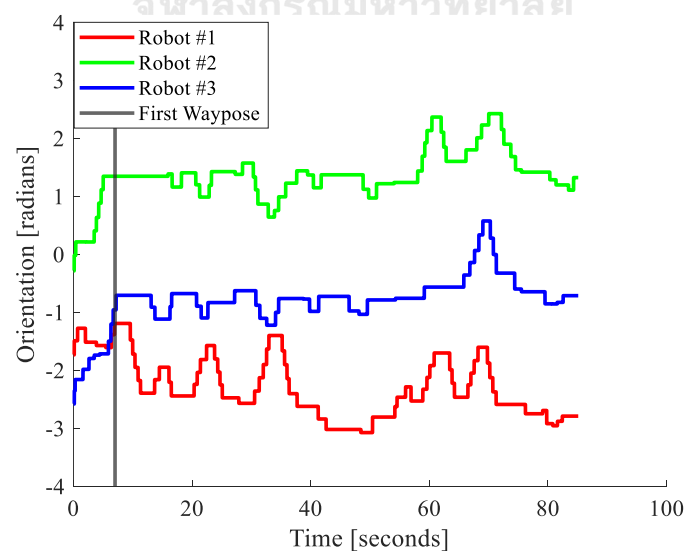


Fig. 44 Multiple robot orientation for survey operation (Experiment#1)

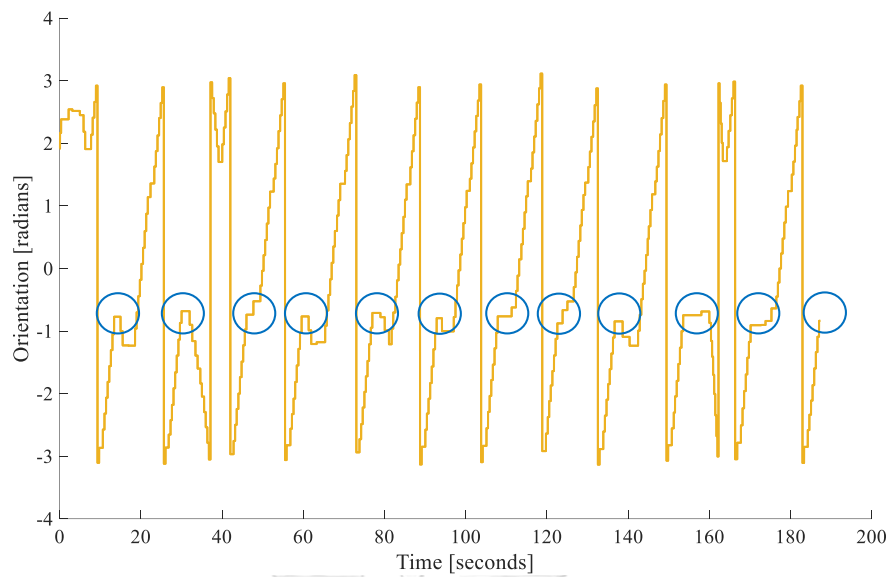


Fig. 45 Single robot orientation for survey operation (Experiment#4)

7.3. Robot Object Search: Position of Detection

Experiment #2 and Experiment #5 were conducted to evaluate the position and orientation of object detection. Both experiments utilized a waypoint system with a similar set of 12 coordinates. Since “Robot #3” detected the object in Experiment #2, “Robot #3” path was then used to compare with a single robot path. The target object is placed in the same position for both experiments. From Fig. 46, it can be observed that “Robot #3” and the single robot have similar distances of motion. However, due to the formation control as specified, the path of “Robot #3” correctly deviates towards the northeast.

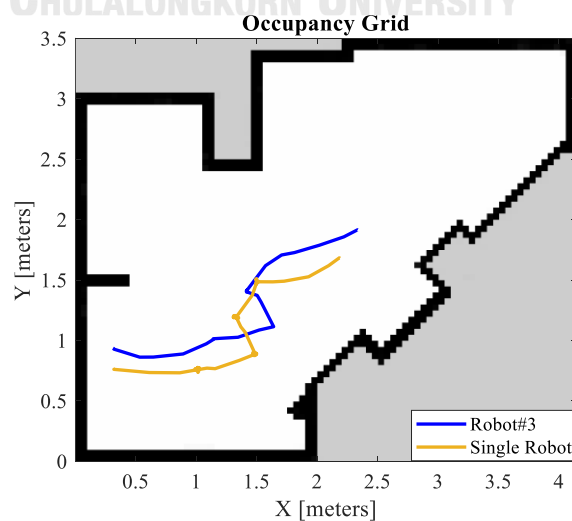


Fig. 46 Path comparison between “Robot#3” and a single robot (Experiment #2 vs Experiment #5)

7.4. Robot Object Search: Orientation of Detection and Survey Time

In terms of orientation (vision), Experiment #2 and Experiment #5 show very close orientations for object detection. In Fig. 47, the orientation of “ROBOT#3” is -0.61 radians, while a single robot’s orientation is -0.65 radians, as indicated by the end points of the orange and blue lines, respectively. Similar to the full survey, formation control is able to control the orientation of the robot to take charge of a certain angle without redundant vision. Regarding the operation time, a single robot took 72.01 seconds to complete the task, while three robots took 35.7 seconds. This indicates that the three-robot operation was approximately 2 times faster than the single robot in the search operation.

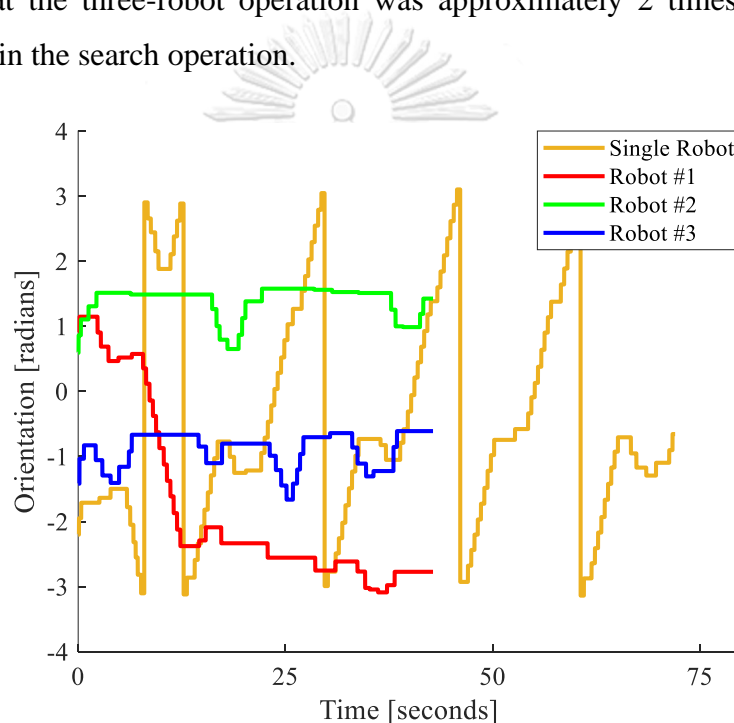


Fig. 47 Robot’s orientation of three robot compared with a single robot (Experiment2 vs Experiment5)

7.5. Robot Swarm Search and Surround Target Object

This experiment is an extension from Experiment#2, demonstrating that the stereo camera could accurately measure the depth of the target object. Fig. 48 illustrates the centroid of the swarm before and after the swarm detected the ball. The target object is positioned at the edge of the path. The robot swarm was able to surround the target object in 16 seconds with the path of each of the three robots shown in Fig. 49.

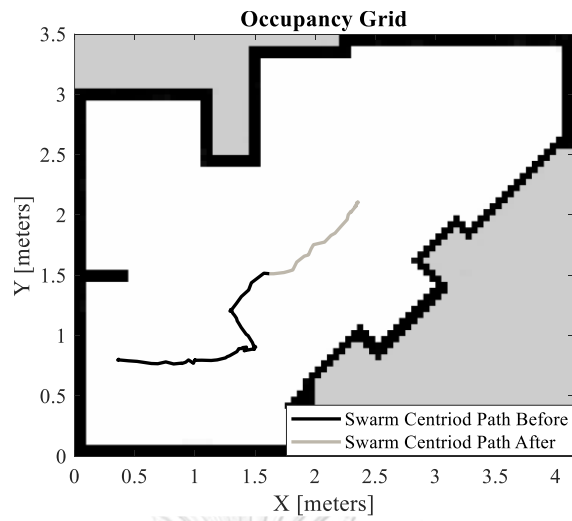


Fig. 48 Robot's centroid position of three robots in a full search operation across the whole workspace. (Experiment#3)

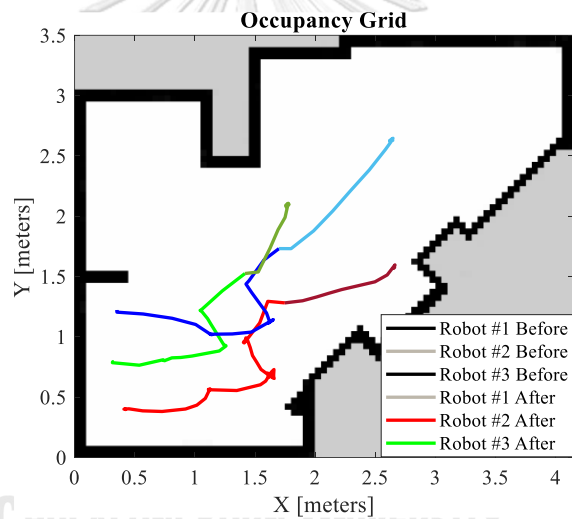


Fig. 49 Robot's position of three robots in a full search and surround target object operation (Experiment#3)

Chapter 8: Conclusion

In this research, we focused on the development of a robotics swarm system for efficient search operation. Starting from the device layer, careful selection of sensors and devices was made to meet the requirements of the machine layer. By integrating the selected devices, we successfully created a standalone robot that could perform desired tasks. The on-board devices were divided into groups based on high-level control and lower-level control functionalities. The lower control handled robot motion, including motors connected to a microcontroller, while the high-level control consisted of sensors connected to a microcontroller. Data from both levels are sent to an edge-level control to perform edge computing.

Once the machine was designed and fabricated, we proceeded to clone it into a group of three robots. To enable effective collaboration among multiple robots, we designed a network architecture comprising three main components: an edge computing computer, edge teleoperation computer, and robot agents. These components were interconnected wirelessly using a mesh Wi-Fi network. Through the implementation of this network architecture, we successfully established a robotics swarm system based on a wireless network.

The research could achieve the five stated challenges. First challenge, deploying the system with high number of robots, keeping each robot alive, and dealing with physical interruption from the outside environment. In this research, each robot is implemented with LiDAR and stereo camera deals with interruption by other robots in the swarm and outside environment. Second challenge, for mobility and endurance, the robot is holonomic with three degrees of freedom (DOF) having no limits to the robot movement in x-axis, y-axis, and ω (orientation) with long power bank battery life. Third challenge, for robots' communication constraints, communication between each robot is connected wirelessly through Raspberry Pi enabling all robots to connect to the same network; capable of managing and controlling many robots. The fourth challenge, the system should be able to scale to a higher number of robots, with the current control algorithm increasing the number of robots into the operation is possible. Fifth challenge, the robustness and adaptability of the system should be considerably high, the system is designed in which each robot

could work as standalone. Therefore, when one robot fails, it is possible for the rest of the robot in the swarm to decide to continue with their operation.

Controlling swarm robots, particularly in a group, posed a significant challenge. To address this, we proposed the utilization of formation control algorithms. Our objective was to optimize the search process by coordinating the movements of the robots in a formation, thereby improving vision coverage and operation time. These algorithms were developed and implemented to regulate the movement and coordination of the robot swarm. To assess the effectiveness and robustness of the proposed formation control algorithm, five experiments were conducted.

The experimental results revealed that increasing the number of robots in a search operation does reduce the operation time by around 2 times compared to a single robot. Furthermore, the adoption of formation control proved advantageous as each robot took charge of a specific range of vision throughout the operation, ensuring comprehensive workspace coverage.

Overall, our research demonstrates the successful development and implementation of a robotics swarm system. Through careful device selection, network architecture design, and the use of formation control algorithms. We improved efficiency and coordination among multiple robots. The findings of this study contribute to the advancement of swarm robotics and offer potential applications in various fields requiring collaborative robotic systems.

REFERENCES

1. Schranz, M., et al., *Swarm robotic behaviors and current applications*. Frontiers in Robotics and AI, 2020: p. 36.
2. Tarapore, D., R. Gross, and K.P. Zauner, *Sparse Robot Swarms: Moving Swarms to Real-World Applications*. Front Robot AI, 2020. **7**: p. 83.
3. Siegwart, R., I.R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. 2011: MIT press.
4. Pfister, S.T., *Algorithms for mobile robot localization and mapping, incorporating detailed noise modeling and multi-scale feature extraction*. 2006: California Institute of Technology.
5. Tsukiyama, T., *RFID based navigation system for indoor mobile robots*. IFAC Proceedings Volumes, 2011. **44**(1): p. 1084-1089.
6. Se, S., D. Lowe, and J. Little. *Vision-based mobile robot localization and mapping using scale-invariant features*. in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. 2001. IEEE.
7. Gryaznov, N. and A. Lopota, *Computer vision for mobile on-ground robotics*. Procedia Engineering, 2015. **100**: p. 1376-1380.
8. Asavasirikulkij, C., et al., *Low Latency Peer to Peer Robot Wireless Communication with Edge Computing*, in *2021 IEEE 11th International Conference on System Engineering and Technology (ICSET)*. 2021. p. 100-105.
9. Field, M., et al. *Motion capture in robotics review*. in *2009 IEEE international conference on control and automation*. 2009. IEEE.
10. Nagymáté, G. and R.M. Kiss, *Application of OptiTrack motion capture systems in human movement analysis: A systematic literature review*. Recent Innovations in Mechatronics, 2018. **5**(1.): p. 1-9.
11. Camacho, E.C., J.G. Guarnizo, and J.M. Calderon, *Design and Construction of a Cost-Oriented Mobile Robot for Domestic Assistance*. IFAC-PapersOnLine, 2021. **54**(13): p. 293-298.
12. Quigley, M., et al. *ROS: an open-source Robot Operating System*. in *ICRA workshop on open source software*. 2009. Kobe, Japan.
13. Chang, L., et al., *Hierarchical multi-robot navigation and formation in unknown environments via deep reinforcement learning and distributed optimization*. Robotics and Computer-Integrated Manufacturing, 2023. **83**: p. 102570.
14. Bloss, R., *Advanced swarm robots addressing innovative tasks such as assembly, search, rescue, mapping, communication, aerial and other original applications*. Industrial Robot: An International Journal, 2014. **41**(5): p. 408-412.
15. Dias, P.G.F., et al., *Swarm robotics: A perspective on the latest reviewed concepts and applications*. Sensors, 2021. **21**(6): p. 2062.
16. Hu, S., H. Chen, and Y. Shao, *Triangular Omnidirectional Wheel Motion Control System*. OALib, 2020. **07**(08): p. 1-8.
17. Liu, Y., et al., *Omni-directional mobile robot controller based on trajectory linearization*. Robotics and autonomous systems, 2008. **56**(5): p. 461-479.
18. Freeman, R.A., P. Yang, and K.M. Lynch. *Stability and convergence properties of dynamic average consensus estimators*. in *Proceedings of the 45th IEEE Conference on Decision and Control*. 2006. IEEE.

19. Asavasirikulkij, C. and M. Hanif. *Human Workload Evaluation of Drone Swarm Formation Control using Virtual Reality Interface*. in *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*. 2023.
20. Verginis, C.K., A. Nikou, and D.V. Dimarogonas. *Position and orientation based formation control of multiple rigid bodies with collision avoidance and connectivity maintenance*. in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017. IEEE.





จุฬาลงกรณ์มหาวิทยาลัย
CHULALONGKORN UNIVERSITY

VITA

NAME Chanun Asavasirikulkij

DATE OF BIRTH 9 July 1999

PLACE OF BIRTH Bangkok, Thailand

INSTITUTIONS ATTENDED Mechanical Engineering, Chulalongkorn University

HOME ADDRESS 8/114 Laddawan Village Ramintra Road Anusaovaree Bangkaen Bangkok 10220

PUBLICATION

[1] Asavasirikulkij, C., Mathong, C., Sinthumongkolchai, T., Chanchaoren, R. and Asdomwised, W., 2021, November. Low latency peer to peer robot wireless communication with edge computing. In 2021 IEEE 11th International Conference on System Engineering and Technology (ICSET) (pp. 100-105). IEEE.

[2] Asavasirikulkij, C., Mathong, C., Sinthumongkolchai, T., Chanchaoren, R. and Asdomwised, W., 2021. A Study of Digital Twin and Its Communication Protocol in Factory Automation Cell. IEICE Proceedings Series, 68(D4-2).

[3] Asavasirikulkij, C. and Hanif, M., 2023, March. Human Workload Evaluation of Drone Swarm Formation Control using Virtual Reality Interface. In Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction (pp. 132-136).