VISION-AIDED OBSTACLE AVOIDANCE FOR THE FORMATION OF MULTI-AGENT VEHICLES

by

Zike Wang

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
December 2023

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols Used

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
CMOS	Complementary Metal Oxide Semiconductor
CNNs	Convolutional Neural Networks
CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradients
DOF	Degree of Freedom
DQN	Deep Q-Network
FOV	Field of View
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
IMUs	Inertial Measurement Units
LiDAR	Light Detection of Laser Imaging And Ranging
LMI	Linear Matrix Inequalities
LOS	Line of Sight

MAS	Multi-agent System
PC	Personal Computer
PI	Proportional-integral
RGB-D	Red Green Blue-Depth
RL	Reinforcement Learning
ROS	Robot Operating System
RRTs	Rapidly-exploring Random Trees
SSH	Secure Shell
UAV	Unmanned Aerial Vehicles
VFH	Vector Field Histogram
YOLO	You Only Look Once

Symbols

δ	the difference between two values
$\in$	belongs to
∞	infinity
$\mathbb{R}$	set of real number
$\mathbb{R}^{n\times n}$	set of $n \times n$ Real Matrix
$\mathbb{R}^{n}$	set of $n \times 1$ real vector
$\mathbf{I_d}$	$d \times d$ identity matrix

$\mathcal{E}$	edge set of a graph
$\mathcal{G}$	graph
$\mathcal{L}$	Laplacian matrix
$\mathcal{N}_i$	neighbor set of a node i
$\mathcal{V}$	node set of a graph
Ω	stress matrix
$\otimes$	Kronecker Product
$\rightarrow$	tends to
$\sum$	summation

Abstract

This thesis presents an in-depth study of vision-aided obstacle avoidance for multi-agent formation control, integrating formation control, path planning, and the use of vision sensors. The paper thoroughly reviews the literature on coordination strategies, obstacle avoidance methods, machine learning methods in navigation, and sensor technologies in robotics. It explores the role of different sensors in navigating unknown environments.

The main contributions of the thesis include the design and validation of formation controllers and obstacle avoidance algorithms, supported by simulations and experimental results with mobile robots equipped with an advanced vision sensor. Hardware and software frameworks are firstly discussed, such as the use of the TurtleBot3 mobile robots, Intel RealSense depth camera, and the integration in Robotic Operating Systems (ROS). The problem formulation includes kinematics, a hand position model for mobile robots, and a multi-robot affine formation control system. Algorithms are validated in both MATLAB and Gazebo environments for their efficacy in the goal point navigation, narrow gap passing, and leader-follower obstacle avoidance.

The thesis provides new insights into control laws that prove the effective for maintaining desired formation configurations, demonstrating stability through l_2 norm of formation errors. The research validates the robustness and adaptability of the proposed control algorithms through experiments, showcasing the ability to navigate towards designated goals while achieving precise formation configurations, even when encountering unforeseen challenges in real-world scenarios. The work concludes with successful experimental applications and proposes future investigations into enriched formation control methods and advanced image processing for autonomous navigation. This study is instrumental in advancing the field of autonomous robot formations, offering practical and theoretical insights for the deployment of multi-agent systems in unknown environments.

Acknowledgements

First of all, I would love to extend my deepest appreciation to my supervisor Dr. Ya-Jun Pan. Her guidance, expertise, and unwavering support throughout the entire duration of my Master's program were invaluable. I am truly grateful for her patience, encouragement, and insightful feedback which have significantly shaped the outcome of this work.

I would also like to thank my dedicated committee members, Dr. Ted Hubbard and Dr. Jason Gu for their constructive criticism and helpful suggestions. Their expertise and commitment to academic excellence have been instrumental in refining the research and improving the overall quality of my work. Besides my supervisors, I would like to thank the Advanced Controls and Mechatronics Laboratory research group for their suggestions and comments over the years.

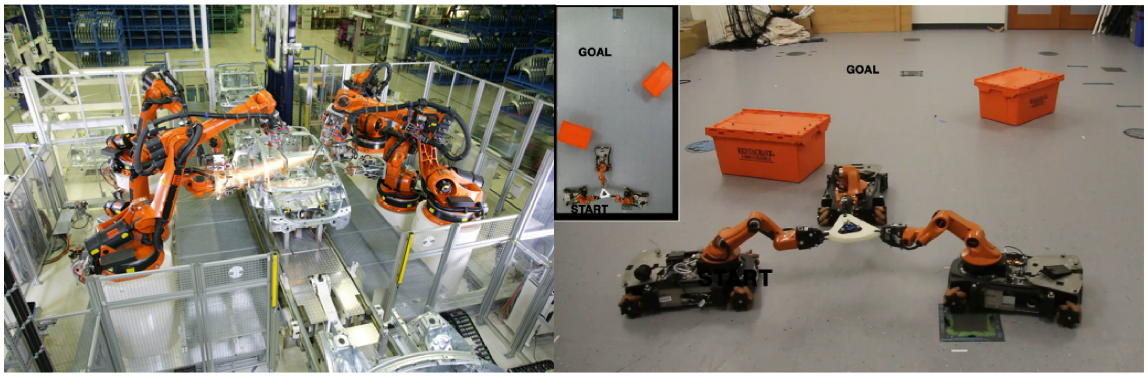
Lastly, I want to thank my loving parents for their unconditional love and endless support throughout my journey. I am also incredibly fortunate to have the most amazing girlfriend Tong Wang by my side. Her love, understanding, and support have been a constant source of inspiration and motivation.

Chapter 1

Introduction

1.1 Research Background

Multi-agent Systems (MASs) consist of multiple autonomous agents that interact with each other and their environment to achieve common goals. A MAS control refers to the design and coordination of these agents' behaviors and actions to accomplish desired objectives. The word "agent" can refer to a mobile robot, a manipulator, or a sensor node. Effective control algorithms are crucial in ensuring the robustness, adaptability, and efficiency of MASs across various domains, including robotics, swarm intelligence, transportation, and rescue missions. During those tasks, MASs provide higher efficiency, low cost, and operational capability compared with a single-agent system. MAS application examples such as assembly and payload transportation can be found in Fig. 1.1.



(a) Assembly line [1]   (b) Transportation task [2]

Figure 1.1. MAS applications

The coordination between the agents is mandatory for the MAS to perform tasks as a group. Traditionally, the MAS coordination algorithm is developed using a centralized structure, with central computers responsible for generating information and scheduling control tasks for all agents, but the computing and communication capabilities required with centralized structures may rapidly increase as the number of agents increases. Central structures are also susceptible to unexpected network issues, and the entire system will fail when the central agent fails. Recently, considerable work has been done on the development of the MASs' distributed coordination control strategy. In distributed structures, each agent has a microprocessor, sensor, and actuator to collect data and perform control functions.

For MASs, two basic coordination research problems are consensus and formation control. A consensus control problem is one where the controllers are designed for each agent solely using the information that is locally accessible, allowing the group of agents to agree on specific quantities of interest. Creating distributed controllers that motivate the entire group to accomplish and sustain a specific geometric pattern of interest is the goal of formation control. A distributed formation control is to keeping the agents in a certain geometric shape relative to each other, which could be a time-varying formation as well.

Another key problem of earlier research is how to let robots "see" their environment. The most common distance sensor used on mobile robots working in an indoor environment is the Light Detection of Laser Imaging and Ranging (LiDAR) sensor and Three-dimensional (3D) vision cameras [3]. LiDAR sensors are frequently used to create high-resolution maps and point clouds, but depending on the mounting location, they can only measure distances at a fixed elevation or in a Two-dimensional (2D) plane. According to its operating principle, the LiDAR sensor sends a laser pulse to the surface of an object and calculates the elapsed time between the emission of the pulse and the received reflection [3]. A robot equipped with a 2D LiDAR sensor cannot detect objects above or below the sensor. One solution is to use more LiDAR sensors to scan at different altitudes, which dramatically increases costs. Therefore, detecting complex environments during navigation using an advanced sensor is essential. As a result, this thesis focuses on the obstacle detection and navigation using a vision sensor for an MAS with an affine formation control for the team overcoming the limitation of existing obstacle avoidance methods.

1.2 Literature Review

1.2.1 Formation Control

Formation control in MASs is a fundamental concept with applications across various domains, including mobile robots, Unmanned Aerial Vehicles (UAV), and autonomous vehicles [4]. The ability to coordinate a group of agents to achieve and maintain a desired formation has been a subject of extensive research. Formation strategies have been observed in nature, such as dolphins frequently swimming in specific formations, which can vary from line formations to tight clusters, in order to aid in communication, hunting, and protection from predators [5].

Many researchers in control and robotics have recently used these formation tactics to apply to robotic systems for a variety of tasks, such as payload transportation [2], object search [6], and forest fire surveillance [7]. A significant amount of research has been done on the control of MASs due to their practical potential in different applications. The theoretical challenges mainly focus on partial and relative information without the intervention of a central controller.

Formation control generally means driving multiple agents to achieve prescribed constraints on their states [8]. Many complex formation control tasks with complex agent dynamics and constraints have been successfully completed by early formation control approaches, including behavior-based [9] and virtual-structure [10]. The characterization of formation control schemes in terms of the sensing capability and the interaction topology naturally leads to the question of what variables are sensed and what variables are actively controlled, to achieve their desired formation [8]. Specific requirements for each agent's sensing capacity are stated in the types of sensed variables. The types of controlled variables, however, are fundamentally linked to the topology of interactions. More precisely, if each agent's position is actively controlled, the agents can move to the desired locations apart from one another. A rigid body can be formed from the agents if the distances between them are actively controlled. Subsequently, the agents must engage in mutual interactions in order to preserve their configuration as a stiff entity. In other words, the kinds of controlled variables dictate the optimal configuration that agents can attain, which in turn dictates the specifications for the agents' interaction topology.

Classifications of formation control could be confusing if the criteria is not clear. According to Ren and Cao [11], the formation control can be classified by whether or not desired formations are time-varying:

- Formation producing problems: The objective of agents is to achieve a prescribed desired formation shape. These problems have been addressed through matrix theory based approach, Lyapunov based approach, graph rigidity approach, and receding horizon approach.

- Formation tracking problems: Reference trajectories for agents are prescribed and the agents are controlled to track the trajectories. These problems have been studied through potential function based approach, matrix theory based approach and other approaches.

The formation control problem could also be classified by the fundamental ideas of control schemes [12] [13]:

- Leader-follower approach: At least one agent plays the leader role and the rest of the agents are designated as followers. The followers track the position of the leader with designed offsets while the leader tracks the desired trajectory.

- Virtual structure approach: The formation of agents is considered as a single object as a virtual structure. The desired motion for the virtual structure is given and determines the motions for the agents.

- Behavioral approach: Desired behaviors are prescribed for agents including cohesion, collision avoidance, and obstacle avoidance. This approach is often related to an amorphous formation control scheme.

Depending on whether or not desired formation shapes are explicitly prescribed, the formation control problem may also be classified as:

- Morphous formation control: Desired formations are explicitly specified by desired positions of agents, desired inter-agent displacements, desired inter-agent distances, etc.

- Amorphous formation control: Without explicitly specified desired formations, desired behaviors such as cohesion, and collision avoidance are given for agents. Amorphous formation control is often related to the behavioral approach.

Based on the observed information, a formation control can be categorized into position, displacement, and distance-based according to types of sensed and controlled variables [8]:

- Position-based control: Agents sense their own positions with respect to a global coordinate system. They actively control their own positions to achieve the desired formation, which is prescribed by the desired positions with respect to the global coordinate system.

- Displacement-based control: Agents actively control displacements of their neighboring agents to achieve the desired formation, which is specified by the desired displacement with respect to a global coordinate system under the assumption that each agent is able to sense the relative position of its neighboring agents with respect to the global coordinate system. This implies that the agents need to know the orientation of the global coordinate system. However, the agents require neither knowledge of the global coordinate system itself nor their positions with respect to the coordinate system.

- Distance-based control: Inter-agent distances are actively controlled to achieve the desired formation, which is given by the desired inter-agent distances. Individual agents are assumed to be able to sense the relative positions of their neighboring agents with respect to their own local coordinate systems. The orientations of local coordinate systems are not necessarily aligned with each other.

Many complex formation control tasks with complex agent dynamics and constraints have been successfully completed by early formation control approaches, including behavior-based and virtual structures. Under the virtual structure approach, human-made control systems impose the geometric relationship between robots and treat the robots as particles in a rigid body. Group motion is made by applying a virtual force field to the virtual structure, which causes each robot to move in the force's direction. However, this method requires a centralized formulation, and is difficult to control the robots in a distributed way. In a behavioral formation control approach, behaviors are prescribed for each agent and the final control input can be determined as a weighted average of each behavior. This approach is relatively easy to implement but no guarantee of system convergence and stability analysis.

Table 1.1. Distinctions of position-, displacement-, and distance-based formation control

	Position-based	Displacement-based	Distance-based
Sensed variables	Positions of agents	Relative positions of neighbors	Relative positions of neighbors
Controlled variables	Positions of agents	Relative positions of neighbors	Inter-agent distances
Coordinate systems	A global coordinate system	Orientation aligned local coordinate systems	Local coordinate systems
Interaction topology	Usually not required	Connectedness or existence of a spanning tree	Rigidity or persistence

In this thesis, the main categorization is considered in characterizing formation control schemes in terms of the requirement on the sensing capability and the interaction topology. As shown in Table. 1.1, the group formation maneuverability is largely dependent on the constraints imposed on the system. For example, displacement-based formation controllers can only be applied to track formations with time-varying translations since the constant displacement constraint also imposes constant orientation and scale the constraints on the group formation. As summarized in Table. 1.1, a position-based control is particularly beneficial in terms of the interaction topology though it requires more advanced sensors. A distance-based control is advantageous in terms of the sensing capability but it requires more interactions among agents. A displacement-based control is moderate in terms of both sensing capability and interaction topology compared to other approaches. Overall, it is a trade-off between the amount of interactions among agents and the requirement on the sensing capability of each agent as shown in Fig. 1.2.

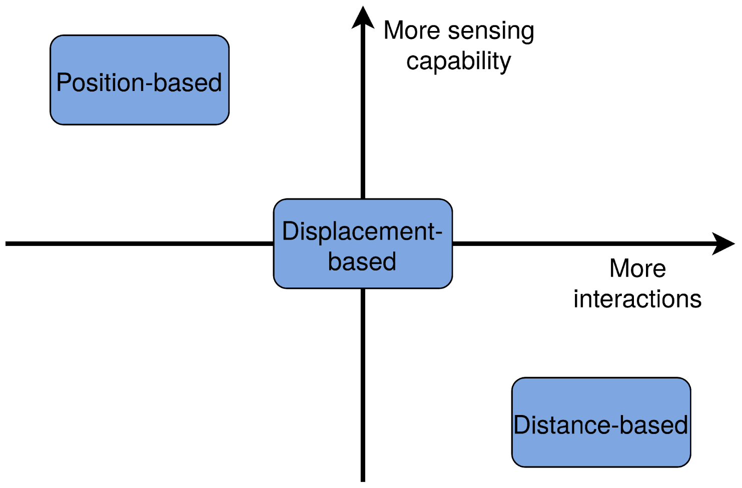


Figure 1.2. Sensing capability vs. interaction topology [8]

Researchers have proposed many methods motivated by the limitations of the three approaches, modifying them in order to achieve desired formation maneuvers. For example, adding a formation-scale estimation mechanism on the displacement-based formation control approach [14], and modifying the distance-based formation control approach to allow the final formation to have an unspecified scale [15]. Nevertheless, those modifications usually result in complicated control and estimation problems and may require additional sensing or communication abilities for each agent. For example, the desired maneuver of each agent must be prespecified in order to track general time-varying formations [16]

It remains an open research problem to define time-varying group target formation for MASs that can dynamically adapt to changes in the environment, such as formation control of multiple quadcopters as in [17] and formation shaping control for MASs with obstacle avoidance and dynamic leader selection as in [18].

1.2.2 Obstacle Avoidance

An obstacle avoidance is a critical aspect of autonomous robotics, enabling robots to navigate complex environment safely and efficiently. This literature review provides an overview of key developments, methodologies, and challenges in the field of obstacle avoidances in robotics.

The obstacle avoidance has been a fundamental challenge in robotics for decades. Early approaches focused on simple reactive behaviors, where robots responded directly to sensory input. While effective in some scenarios, these methods lacked the ability to plan and adapt to changing environments.

Sensor-based obstacle avoidance is a prominent method in robotics. In this approach, robots utilize a range of sensors, including ultrasonic, LiDAR, and vision, to detect obstacles and adjust their trajectories. Studies by Borenstein and Koren [19] on the Vector Field Histogram (VFH) method demonstrated the efficacy of sensor-based approaches in mobile robot navigation.

Path planning algorithms have gained significant attention in recent years. Dijkstra's algorithm, A* search, and rapidly exploring random trees (RRT) have been adapted for obstacle avoidance. Karaman and Frazzoli [20] introduced an efficient RRT-based approach, the RRT* algorithm, which has been widely adopted in the robotics community.

Machine learning and deep learning techniques have revolutionized obstacle avoidance. Researchers have explored Reinforcement Learning (RL) and Convolutional Neural Networks (CNNs) for training robots to navigate complex, dynamic environments. Notable contributions include Deep Q-Network (DQN) method in [21] and Deep Deterministic Policy Gradients (DDPG) method in [22].

While the obstacle avoidance has made significant progress, challenges remain. Adapting to dynamic, real-world environments, handling sensor noise, and addressing the issue of local minima in path planning are areas of ongoing research. Combining traditional methods with machine learning for robust obstacle avoidance is a promising direction.

1.2.3 Path Planning

Path planning algorithms are applied by mobile robots, unmanned aerial vehicles, and autonomous underwater vehicles in order to identify safe, efficient, collision-free, and least-cost travel paths from the start to a destination [23]. The literature review listed the classes of path planning algorithms used today and their potential within automated systems.

Autonomous mobile robots can reduce the contribution of human error and negligence as the cause of collisions. The robots must move from point A to point B safely and efficiently, and path planning is the key in determining and evaluating trajectories [23]. During navigation, robots make use of capabilities that involve modeling the environment and localizing the position, which lead to the four general problems of navigation: perception, localization, motion control, and path planning [24] [25] [26].

This literature review is mainly focused on path planning, which is the determination of a collision-free path in a given environment and often be cluttered in a real-world situation. An appropriate path planning technique must be identified and implemented to accomplish the system's design, and the best-performing technique will vary with the system type and the environment [27]. The complexity of the problem increases with an increase in degrees of freedom (DOF) of the system, the optimal path will be decided based on constraints and conditions [23]. For example, considering the shortest path between points or the minimum time to travel without collisions, minimizes energy consumption. A path planning can be used in known and unknown environments where information is received from internal and external sensors, updating maps to inform the desired motion of the mobile robot [23].

The path planning can be either local or global. A global path planning looks for an optimal path given largely complete environmental information, and it is best performed when the environment is static and known to the robot. The path planner algorithm produces a complete path from the start to the end before the robot tracks the trajectory [28]. A global motion planning is the high-level control for environment traversal [23]. A local path planning is mostly performed in unknown or dynamic environments, while the robot is moving and taking data from local sensors [23]. The robot has to generate new paths in response to the changes in the environment, as obstacles are static or dynamic [28].

Several path planning algorithms that are most frequently used are discussed. The first algorithm is the Dijkstra algorithm which has many variants commonly used in applications, such as Google Maps [29] [30]. To overcome the computational-intensity doing blind searches, A^* and its variants are introduced as the most popular algorithm for static environment [31]. However, A^* is used for the shortest path evaluation based on the obstacles present in the environment, which comprises a selection of node pairs [32]. This makes the algorithm inefficient and impractical in dynamic environments [23]. To make path planning work in dynamic environments, D^* and its variants are introduced. Other path planners such as the Rapidly-exploring Random Trees (RRTs), the Genetic algorithms, the ant colony algorithm, and the Firefly algorithm are introduced to represent some of the foundational algorithms. More path planning algorithms can be found in [33].

According to [34] and [35], the motion planning problem and its algorithms can be classified into search-based and sampling-based, where search-based planning can be seen as two problems: how to turn the problem into a graph and how to search the graph to find the best solution. Example of search-based planning is the Dijkasra algorithm and A^* algorithm. RRT and its variant, however, are sampling-based [35].

The Dijkstra algorithm works by computing the shortest path from the source to vertices among the closest vertices to the source [29]. The algorithm finds the next closest vertex by keeping the new vertices in a priority-min queue and stores only one node in order to find the shortest path [29].

The traditional Dijkstra algorithm finds the shortest path in an acyclic environment which means the path traversed through a sequence of vertices without having the same point as the start and the end vertices, and is able to find the shortest path from the start to every point, relying upon greedy strategy searching on a graph [23]. Many versions of improved Dijkstra have been developed based on specific applications, yet concerns the path solution with formal attention to the pragmatism solution [23]. The modified Dijkstra algorithm aims to find alternate routes where the cost of generating plausible shortest paths is significant. It introduces another component to the classical algorithm in the form of probabilities that define the status of freedom along the graph edges [36]. This variant overcomes the computational shortcomings in the Dijkstra algorithm and becomes suitable in modern applications. Another improved Dijkstra algorithm reserves all nodes with the same distance from the source node as intermediate nodes, followed by a re-search within all the intermediate nodes until traversing successfully find the goal point, all possible short paths can be found after iterations [37].

Due to the nature of the Dijkstra algorithm, nodes that have been previously searched cannot be stored. A storage scheme is introduced to overcome this disadvantage with a multi-layer dictionary implemented, which contains two dictionaries and a list of data structures organized in hierarchical order [38]. One dictionary maps each node and its neighboring nodes, the other dictionary stores the path information of each neighboring pathway [38]. This multi-layer dictionary method allows data structure for the Dijkstra algorithm in an indoor environment application where the Global Navigation Satellite System (GNSS) coordinates and the compass orientation are not reliable, producing the shortest path and the most navigable path at the same time, which is infeasible to compute within the traditional Dijkstra algorithm [38].

Another popular graph searching method for finding the shortest path in a positive and negative weighted graph [39]. Inspired by the Dijkstra which works best for finding the single-source path in a positive weighted graph [39].

In general, the Dijkstra is a reliable algorithm for path planning, but also memory-heavy as it has to compute all the possible outcomes in order to find the shortest path. Due to its limitation, improved variants with a new memory scheme arose to map with a huge cost factor [23]. The Dijkstra algorithm is best suited for a static environment and global path planning as most of the data required are pre-defined.

The A^* algorithm is the most popular graph traversal path planning algorithm, which operates similarly to the Dijkstra algorithm except it prioritizes its search towards the most promising nodes, saving a significant amount of computation time [40].

The A^* algorithm is similar to the Dijkstra but works on the lowest cost path tree from the initial point to the final goal. The base algorithm uses the least expensive path and expands using the cost function defined below

$$F = G + H, \tag{1.2.1}$$

where G is the actual cost from the current node to the start, and H is the heuristic cost of the optimal path from the current node to the goal [31].

The A^* algorithm is widely used in the static environment, gaming industry [41], graph theory, and automatic control [23]. The A^* algorithm is a heuristic algorithm that uses heuristic information to find the optimal path. The A^* algorithm searches for nodes in a map and assigns appropriate heuristic functions for the guidance, such as Euclidean distance, Manhattan distance, and Diagonal distance [42] [43].

$$Euclidean\ distance : \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}, \tag{1.2.2}$$

$$Manhattan\ distance : |x_1 - x_2| + |y_1 - y_2|, \tag{1.2.3}$$

$$Octile\ distance : max|x_1 - x_2| + |y_1 - y_2|. \tag{1.2.4}$$

While using the A^* algorithm, there is a trade-off between calculation speed and path accuracy. Decrease the time complexity in exchange for greater memory, or consume less memory in exchange for slower executions [23]. One example of balancing the trade-off is the application for using the A^* algorithm to find the shortest path in a crowded parking lot [44].

As the A^* algorithm uses a different cost function on top of the Dijkstra algorithm, many improvements and variants of the A^* apply new cost functions, with respect to step sampling or steering costs [45] [46]. Other variants were developed based on the specific applications and there are: Hierarchical A^*, Hybrid A^*, Diagonal A^* and Lifelong Planning A^* [47]. In general, the A^* algorithm is computationally efficient and it is suitable for applications in a static environment, the computational speed and efficiency of the A^* and its variants family depends mainly on the definition of the heuristic cost function [23].

The D^* algorithm stands for Dynamic A^* and it is used to generate a collision-free path with moving obstacles [23]. The D^* Algorithm processes a state until it is removed from the open list, and computes the state sequence along with back pointers to either direct the robot to the goal or update the cost due to the obstacles detected, then place the affected states on the open list [48]. The states in the list are processed until the path cost from the current state to the goal is less than a minimum threshold, changing the cost where the robot continues to follow the new sequence [49].

According to [49], the D^* Algorithm is over 200 times faster than an optimal re-planner, and the main defect of the D^* Algorithm is its high memory consumption when compared with other D^* variants [50]. Those D^* variants improve the computational time and overcome problems such that the robot encounters complicated obstacles [51]. Those common variants are: D^* Lite, Enhanced D^* Lite and Field D^* [52] [53].

Apart from the search-based path planning, there are other dynamic and online algorithms. The RRT algorithm does not require a path to be specified upfront and it expands in all regions, assigns weight to each node then creates a path from start to goal [23]. Its variants are able to cope with non-holonomic constraints and almost any wheeled system, depending on the actual applications [54] [55]. Genetic algorithms help to overcome the limitations that discrete path planning, such as grid-based and potential fields require substantial Central Processing Unit (CPU) performance and significant memory [23] [56]. The ant colony optimization algorithm is inspired by nature, and is based on a heuristic approach by the collective behavior of ants to find the shortest and collision-free path [23] [57]. The Firefly algorithm is an algorithm based on firefly mating behavior, which is a promising swarm-intelligence-based algorithm in order to solve complex continuous and discrete optimization problems inspired by insects [23].

1.2.4 Sensors and Vision for Robotics

This section briefly covers the various sensors available in robotics applications.

An odometry is used in calculating position using the motion of an object such as a wheel encoder and is integrated over time and compared to the initial position to determine the final position of the robot [58]. The problem with this method of determining the position of the robot is that errors can build up over time, leading to a large error between the actual and measured position. Thus, odometries are commonly used combined with other sensors [58].

Gyroscopic systems are inertial sensors that can be used to calculate orientation. Gyroscopes maintain their orientation based on the angular momentum. As the robot moves, the orientation of the gyroscope can be compared to its original orientation to determine the final orientation [58].

Accelerometers are another form of inertial sensors that detect acceleration and can be used to calculate forces acting on a robot. Mechanical accelerometers contain a spring-mass-damper system that measures the position of the mass in the system to calculate the experienced acceleration. Other accelerometers such as piezoelectric accelerometers generate a voltage while being applied a force on a crystal [58].

The system that combines both sensors is called Inertial Measurement Units (IMUs), which provide position and orientation data [58].

There are also other sensors that are commonly applied to robots to provide position data. The Global Positioning System (GPS) uses radio signals from a constellation of orbiting satellites to determine the position of an object. The time delay of radio communication among satellites can be used to calculate position. GPS requires an unobstructed line of sight (LOS) of the orbiting satellites and the accuracy depends on atmospheric conditions and overhead materials. GPS signals can pass through plastic and glass but have trouble passing water and many other materials. GPS is accurate within 1-2m which normally dissatisfy many applications [58].

In order to increase the sensing capability of the robots, advanced sensors are applied. One of the fundamental tasks for robotics navigation control is range finding and identification [58]. Range sensors can be differentiated based on whether they are passive or active. In general, an active sensor emits energy into the environment and measures the environment based on the response [58] [59].

Ultrasonic sensors or sonar sensors emit pulses of sound waves and measure the time taken for the return. Since the speed of sound is known, the distance can be calculated. In this way, multiple readings can be used to construct a map of the environment surrounding the sensor [58]. The advantages of sonar sensors are that they are low-cost, lightweight, low-power, computationally efficient, and can be used in low-visibility environments. Disadvantages include poor directional resolution, slow refresh rates, false readings on angled surfaces, and being less robust than other vision sensors.

Laser range sensors allow robots to generate a 2D map of their surroundings. The system measures the distance to a large number of points within the line of sight and combines them into a map. There are mainly three types of laser range sensors: triangulation, phase-modulation, and time-of-flight [58]. Time-of-flight sensors are also called LiDAR sensors and measure the time it takes for a pulse of light to be reflected.

Vision sensors rely on capturing images of the environment and objects to extract information [59]. The optical image is captured through a lens project on a Complementary Metal Oxide Semiconductor (CMOS) optoelectronic sensor, which converts it into a digital signal. Vision sensors usually include monocular, binocular, and Red Green Blue-Depth (RGB-D) cameras.

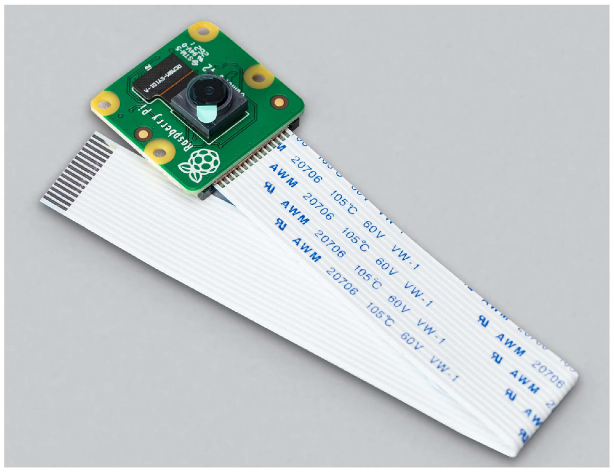


Figure 1.3. Raspberry Pi camera module 2 [60]

Monocular cameras perceive and judge the surrounding environment through flat images taken by one camera, can only obtain 2D information and cannot determine the depth. It relies on complex or computational-heavy algorithms for ranging, requires a large amount of data and highly influenced by the environment and is less accurate [61]. The advantages of monocular cameras are low cost, simple system structure and easy calibration and identification. Fig. 1.3 shows a picture of a monocular camera.

The binocular camera mimics the human eye to achieve the perception of obstacles' distance and size, which can directly obtain the depth information of the scene without distinguishing the obstacle type by performing parallax and stereo matching calculations on two images [62]. The corresponding pixel can be found based on the known camera parameters to calculate the depth of the corresponding point. However, the configuration and calibration are more complex and computationally intensive. Fig. 1.4 shows a picture of a binocular camera.

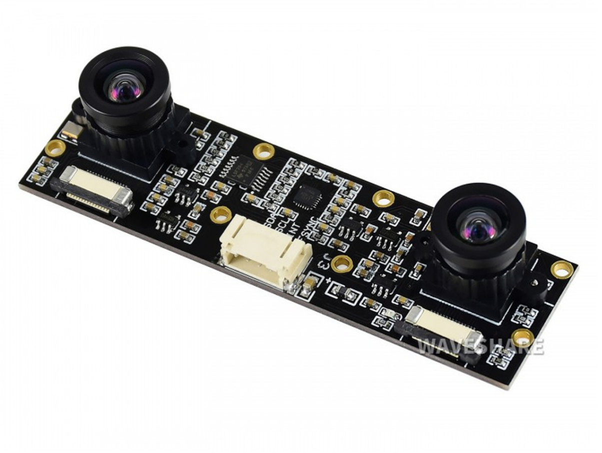


Figure 1.4. IMX219-83 stereo camera [63]

The RGB-D camera is different from the binocular camera that calculates depth by using the parallax method, which measures the depth information of each pixel directly according to the structure light or time of fly. Using this approach can solve the problems of sensitivity to ambient light and dependence on image texture and improve matching robustness. RGB-D cameras can directly perform physical ranging but with a high power consumption. Because it is sensitive to light, translucent objects and reflecting interference, RGB-D cameras are mainly used in indoor applications [64].

1.3 Thesis Objectives and Contributions

This thesis applies a vision sensor on the multi-agent formation control to achieve the navigation in an unknown environment with obstacle avoidance. The proposed approach is novel with few results available in the literature. Although the formation control problem has been studied in [65] and [66], the vision-aided formation control problem has not been addressed extensively in the literature.

The objectives of this thesis are mainly focused on achieving unknown environment obstacle avoidance with a vision sensor on a MAS formed by wheeled mobile robots, using affine formation control navigation. This is an extension of previous work on using a monocular camera combined with CNNs to control a MAS with a low cost [61]. In addition, this thesis also conducts some potential applications on vision sensors.

The main contributions of this thesis are summarized as follows

1. The thesis designed formation controllers based on a MAS with mobile robots and tested the controller using simulations and experiments to verify the effectiveness of the formation.

2. The thesis applied the affine formation controller with obstacle avoidance using an advanced vision sensor to overcome the limitations of regular LiDAR sensor, which provides a practical solution for the MAS navigation and formation control.

3. Experimental results are demonstrated on a team of mobile robots in the lab environment.

1.4 Thesis Outline

This thesis outline is as follows. This chapter provided a general research background and literature review of the research topics in this thesis, as well as the thesis objectives. Chapter 2 introduces the hardware and software used for simulations and experiments in this thesis. Chapter 3 presents the fundamental theories required for this thesis and problem formulation. Chapter 4 studies affine formation controllers designed for MASs under different cases via simulations and results. Chapter 5 extended the simulations to real-world experiments with the vision sensor applied to conducting various tasks. Chapter 6 summarizes the main results of this thesis and suggests areas for future research.

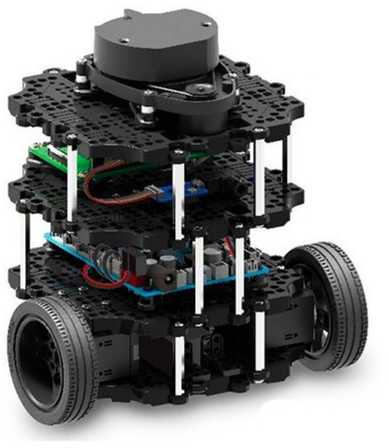
Chapter 2

Experimental Hardware and Software

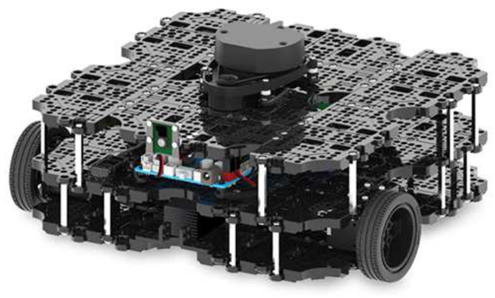
This chapter introduces the general integration of a MAS, hardware, and software used for conducting the simulations and experiments in this thesis. To verify the feasibility of the proposed algorithm, a vision-based affine formation control method is applied to four TurtleBot mobile robots to complete navigation and obstacle avoidance tasks.

2.1 TurtleBot3 Mobile Robots

Three TurtleBot3 Burger robots and one TurtleBot3 Waffle robot from the Advanced Control and Mechatronics Lab at Dalhousie University are used in experiments described in Chapters 4 and 5.



(a) Burger



(b) Waffle

Figure 2.1. TurtleBot3 mobile robots [67]

The Burger robot shown in Fig. 2.1(a), is a two-wheel differential drive robot (with a caster), and it can reach a maximum linear speed of 0.22 m/s and a maximum angular speed of 2.84 rad/s [67]. In addition, it has a maximum payload of 15 kg. The Waffle robot shown in Fig.2.1(b) is also a two-wheel differential drive robot (with 2 casters) and can reach a maximum linear speed of 0.26 m/s and a maximum angular speed of 1.82 rad/s, has a maximum payload of 30 kg [67]. Both robots have an onboard Raspberry Pi as the microcontroller that can communicate with external computers through a wireless network. In addition, these robots can be modeled in an open-source simulation environment called Gazebo to simulate the maneuver and receiving data through the Robot Operating System (ROS).

2.2 Intel RealSense Depth Camera

The vision sensor applied for this thesis is the Intel RealSense depth camera D435i. This camera captures not only traditional 2D images but also 3D spatial information, enabling a more immersive and accurate understanding of the surrounding environment. The Intel RealSense depth camera provides highly accurate dimensional data that fits the purpose of this thesis, using a vision sensor to provide the range information. The depth camera model applied for experiments is the D435i, as shown in Fig. 2.2. The camera has an onboard IMU which is not being used during the experiments.

Figure 2.2. Intel RealSense depth camera D435i [68]

This depth to an RGB scale algorithm was developed for Intel RealSense viewer with post-image processing [69]. In addition, the D435i camera has a software development kit that compiles with ROS and the depth stream information can be obtained through ROS topics, as shown in Fig. 2.4. Although the depth image returns as greyscale images, the ROS topic from the depth stream is an array of distance signals associated with each pixel's coordinate.

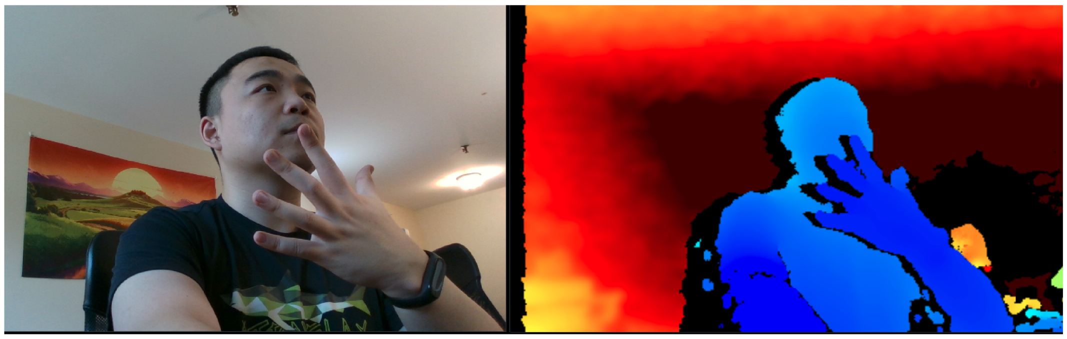


Figure 2.3. Intel D435i color stream (left) and depth stream (right) in RealSense Viewer.

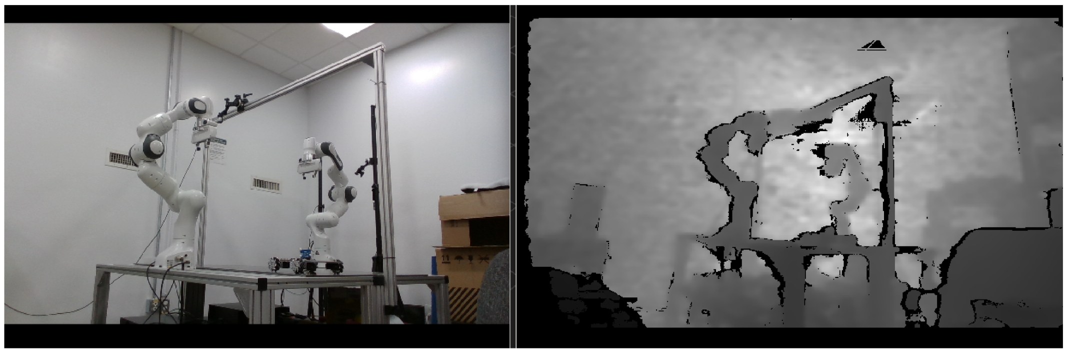


Figure 2.4. Color stream (left) and depth stream (right) in RVIZ.

2.3 Robot Integration

In order to allow the TurtleBot to use the Intel RealSense depth camera, modifications need to be applied. The original TurtleBot robots use Raspberry Pi as the main computing board, it is not powerful enough to handle camera images. Thus, a more vision-friendly computing board Nvidia Jetson Nano is used, as shown in Fig. 2.5. The Nvidia Jetson Nano has a 128-core Maxwell Graphics Processing Unit (GPU) and a Quad-Core CPU, which is able to host a Linux operating system and processes captured images [70].

Figure 2.5. Jetson Nano developer kit [70]

In order to mount the Intel RealSense depth camera and the Jetson Nano board to the leader robot, the TurtleBot Waffle shown in Fig. 2.1(b) is modified from a 2-layer to a 3-layer structure. The original LiDAR sensor is kept on the top although it is not being used for the experiments conducted in Chapter 5. The depth camera is firmly mounted via a 3D printed connector on the second level pointing forward, and the Jetson Nano unit is mounted in-between the second level and the top level on the back of the robot, as shown in Fig. 2.6. A new power supply adaptor, an extra battery, and a solid-state drive are integrated to improve the performance of the Jetson Nano.

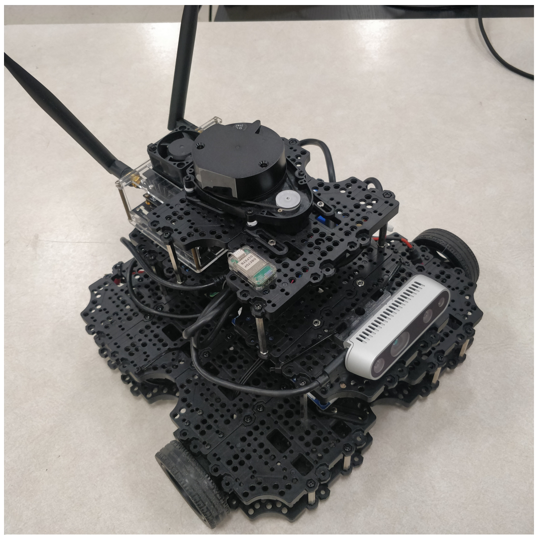


Figure 2.6. Modified TurtleBot Waffle with Jetson Nano and Intel RealSense depth camera

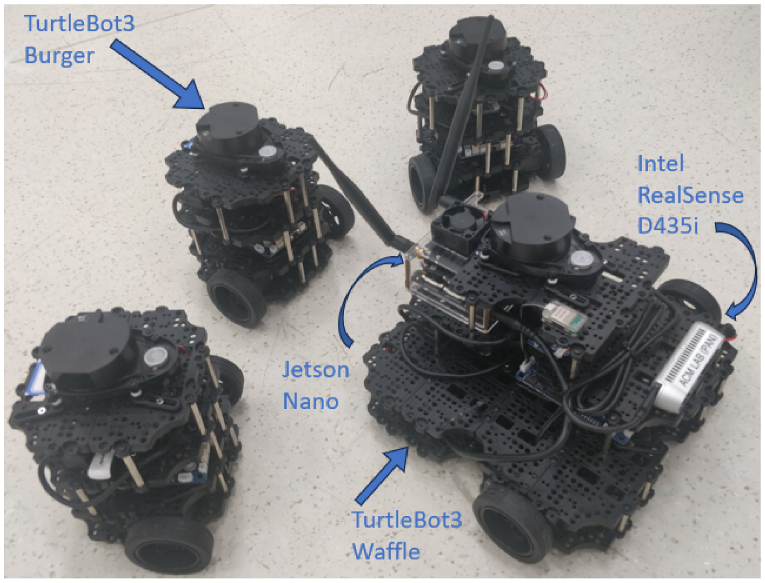


Figure 2.7. Three TurtleBot3 Burger robots, one Waffle robot with Nvidia Jetson Nano and Intel RealSense D435i camera

As shown in Fig. 2.7, the original TurtleBot3 robot usually is equipped with a LiDAR sensor on the top and has a Raspberry Pi single-board computer as the processor or controller. However, no LiDAR sensors were used for this experiment. Instead, one Raspberry Pi camera was added to the front of the leader robot to sense the environment by reading the distance signals of each pixel.

2.4 Communication and Control System

The distributed algorithms developed in this thesis are validated using centralized realizations in the ROS environment. ROS is a free, open-source framework for robot software developers with the intention to overcome challenges, such as distributed computation, software reuse, and rapid testing [71]. ROS runs on Unix-based operating systems such as Ubuntu, enabling services such as hardware abstraction, low-level control, information sharing between processes, and package management, making ROS an effective robotic research system [72]. ROS provides nodes as executables and topics for communicating and controlling components such as controllers and sensors between nodes.

ROS also provides a graphical way of viewing the relationship between nodes and topics. Fig. 2.8 shows a typical communication graph of the navigation of a single TurtleBot communicating with other robots. The MATLAB node receives the states of all robots in ROS by subscribing to the *odom* topics, and the desired control input *cmd_vel* topics are computed in the MATLAB node and published to ROS, which resulted in real-time simulation of robot behaviors in the Gazebo simulation.

In all simulations, controllers are programmed in Simulink and then directly compiled and deployed to the ROS nodes using the MATLAB ROS toolbox by setting the network internet protocol addresses properties appropriately. Note only simulations are applied the MATLAB and Simulink environments.

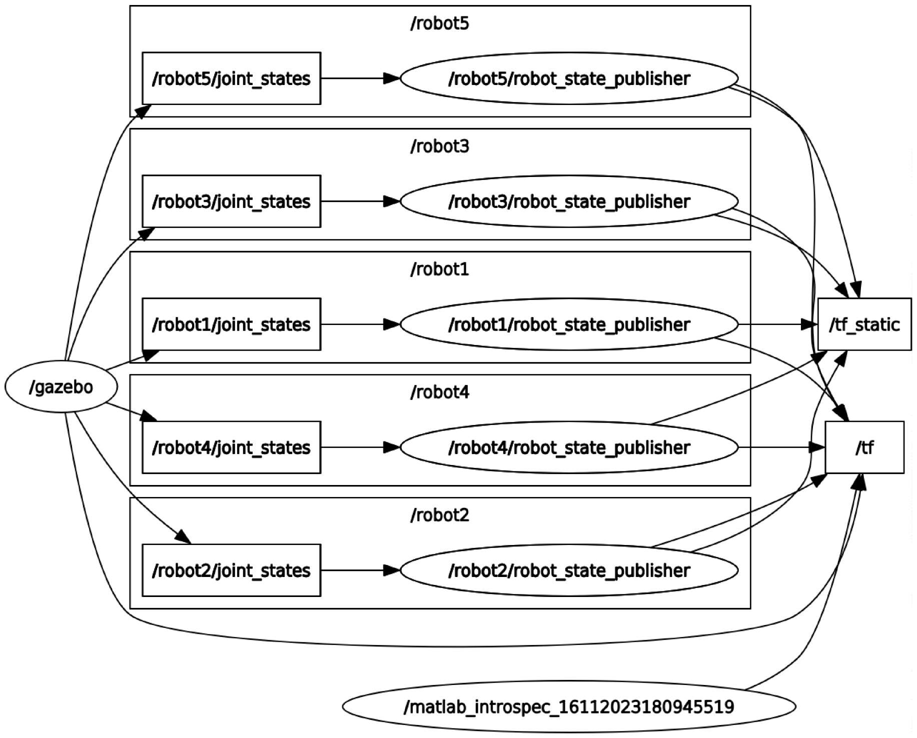

/robot5
/robot5/joint_states
/robot5/robot_state_publisher
/robot3
/robot3/joint_states
/robot3/robot_state_publisher
/robot1
/robot1/joint_states
/robot1/robot_state_publisher
/tf_static
/gazebo
/robot4
/robot4/joint_states
/robot4/robot_state_publisher
/tf
/robot2
/robot2/joint_states
/robot2/robot_state_publisher
/matlab_introspec_16112023180945519

Figure 2.8. Communication graph of ROS through MATLAB and Gazebo

During the experiments, the ROS master node is assigned to the Jetson Nano and runs the main algorithm for calculating control inputs and sharing information with each robot. All robots are connecting to the ROS master via a wireless network, and are associated with different namespace as shown in Fig. 2.8. One computer is connected to the same wireless network to remotely control the ROS master using Secure Shell (SHH) access. The data capture, image processing, and signal calculation are all performed on the Jetson Nano mounted on the leader robot.

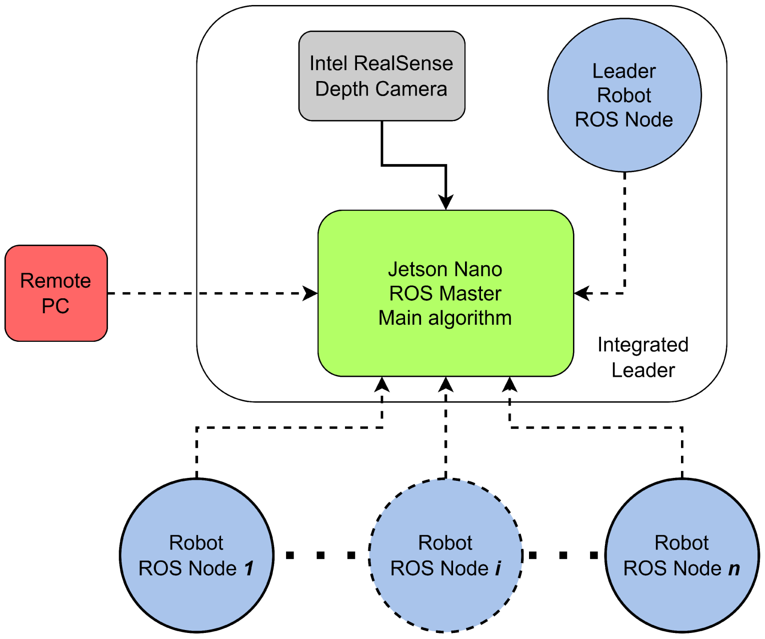


Figure 2.9. Experiment framework diagram

As shown in Fig. 2.9, the integrated leader robot carries the Intel RealSense depth camera and the Jetson Nano, where the Jetson Nano host the ROS Master and executes the main algorithm. The depth camera is directly connected to the Jetson Nano using a cable, as the solid arrow shown in Fig. 2.9. The remote PC and other robots are connected to the Jetson Nano wirelessly as the dashed line arrow shown in Fig. 2.9.

2.5 Simulation Environment

As mentioned above, the simulations are conducted through a simulation software called Gazebo. Gazebo is an open-source and highly versatile simulator in the field of robotics, which provides a dynamic and realistic 3D simulation environment for a wide range of applications [73]. Researchers and developers have used Gazebo to simulate and test various robotics systems, from autonomous drones and self-driving cars to industrial robots and humanoid machines [74]. An example of simulating five TurtleBot Burger mobile robots in the Gazebo environment is shown in Fig. 2.10.

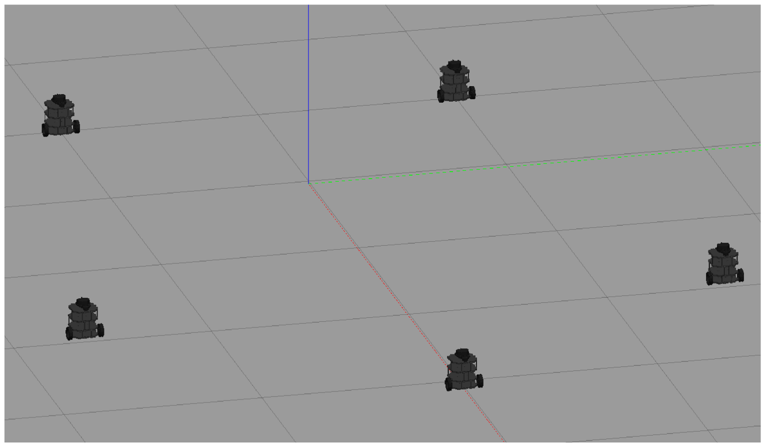


Figure 2.10. Five TurtleBot mobile robots simulated in Gazebo

2.6 Summary

This chapter offers a comprehensive overview of the experimental hardware and software components, from the foundational TurtleBot3 mobile robots to the intricate communication and control systems, showcasing the design, modifications, and capabilities of these robots. This chapter also explores how advanced the Intel RealSense depth camera is in capturing 3D data, and how much potential it could contribute to the environment understanding for MASs. Merging mobile robots and cameras into a seamless and functional unit is essential for conducting the experiments, as well as the communication and control. The simulation environment is also critical because it completes the framework setup as in testing and refining algorithms before implementation on the physical robots.

In summary, this chapter highlights the integration challenges and emphasizes the framework of MAS in both experiments and simulations.

Chapter 3

Problem Formulation

3.1 Kinematics of a Mobile Robot

The kinematic model of a differential-drive mobile robot is shown in Fig. 3.1,

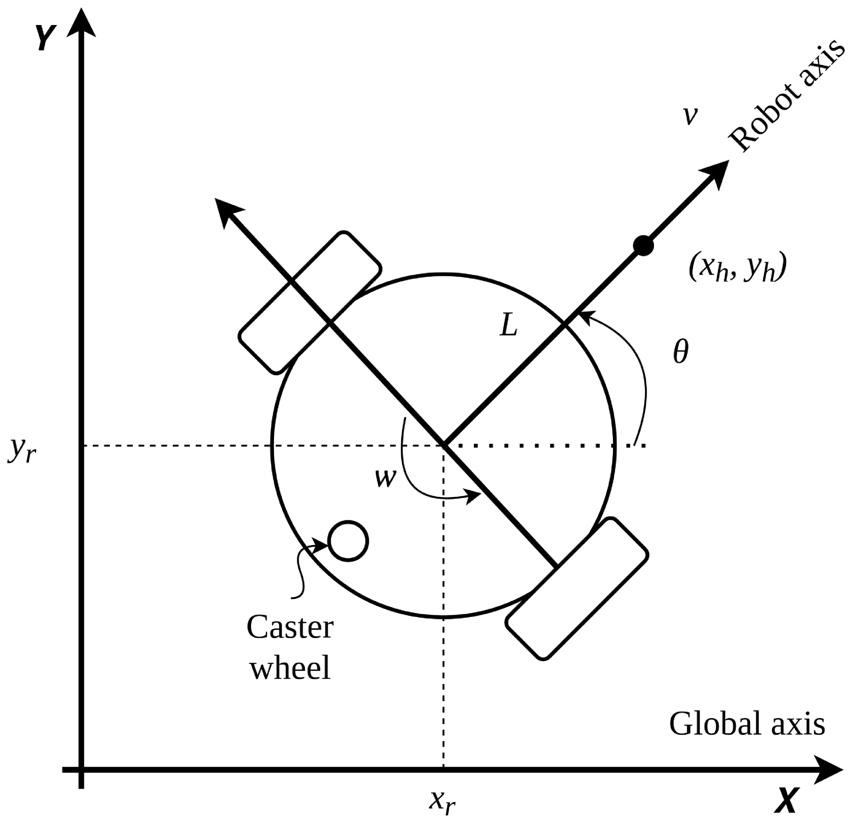


Figure 3.1. Differential-drive mobile robot kinematic model

represented as

$$\begin{cases} \dot{x}_r = v cos\theta, \\ \dot{y}_r = v sin\theta, \\ \dot{\theta} = \omega, \end{cases} \tag{3.1.1}$$

where x_r and y_r represent the position in X and Y directions, θ denotes the robot orientation, v and ω are the linear and angular velocities of the robot in the global frame respectively. The positive direction of v is pointing to the front of the robot and the positive direction of ω is in counterclockwise based on the right hand rule of yaw control. Assume the mobile robot only moves in the 2D plane and the wheels do not slip, the dynamic constraints can be ignored. However, the kinematic model in (3.1.1) is an under-actuated non-holonomic system, the kinematic constraints are not integrable over the center point and there exist no smooth static stabilizing controllers [75]. Thus, a low-level control method is needed to linearize (3.1.1).

3.2 Hand Position Model

Since the differential-drive robot is a non-holonomic system, the formation control on the mobile robots over the center point cannot be stabilized with continuous static state feedback as suggested in [76]. To simplify the controller design, Eq. (3.1.1) has been linearized around a hand position $h = [x_h, y_h]$ that lies at a distance L away from the robot center point $r = [x_r, y_r]$ as shown in Fig. 3.1. The experiment considers the problem of coordinating the hand positions of the robots instead of their center positions, as this simplifies the control problem when the kinematics of the hand position are holonomic for $L \neq 0$. For non-holonomic vehicles: all poses can be achieved in the configuration space, but the paths to reach them can be complex. Let x_{ri}, y_{ri}, θ_i and v_i, w_i denote the position, orientation, linear and angular speeds of the ith robot respectively.

The hand position model can be expressed as

$$\begin{bmatrix} x_{hi} \\ y_{hi} \end{bmatrix} = \begin{bmatrix} x_{ri} \\ y_{ri} \end{bmatrix} + L_i \begin{bmatrix} cos\theta_i \\ sin\theta_i \end{bmatrix}, \tag{3.2.1}$$

differentiating (3.2.1) with time,

$$\begin{bmatrix} \dot{x}_{hi} \\ \dot{y}_{hi} \end{bmatrix} = \begin{bmatrix} cos\theta_i & -L_i sin\theta_i \\ sin\theta_i & L_i cos\theta_i \end{bmatrix} \begin{bmatrix} v_i \\ w_i \end{bmatrix}. \tag{3.2.2}$$

Define the global control input velocity vector as

$$\begin{bmatrix} \dot{x}_{hi} \\ \dot{y}_{hi} \end{bmatrix} = \begin{bmatrix} u_{xi} \\ u_{yi} \end{bmatrix}, \tag{3.2.3}$$

then the relation of actual velocity and the control input becomes

$$\begin{bmatrix} v_i \\ w_i \end{bmatrix} = \begin{bmatrix} cos\theta_i & sin\theta_i \\ -\frac{1}{L_i} sin\theta_i & \frac{1}{L_i} cos\theta_i \end{bmatrix} \begin{bmatrix} u_{xi} \\ u_{yi} \end{bmatrix}, \tag{3.2.4}$$

which can be generalized in the frame of a linear state-space equation $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$ with

$$\mathbf{x} = \begin{bmatrix} x_{hi} \\ y_{hi} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}, A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The kinematic model of the hand position is holonomic, means constraint limits the motion of the system to a manifold of the configuration space, depending on the initial conditions. The formation controller in Chapter 4 will be developed based on the hand position model.

3.3 Obstacle Avoidance

While a mobile robot performs navigation, the path is designed to be efficient. However, path planners need to know or detect the obstacles in the environment so the generated path does not lead the robot to collide with the obstacle. Many methods and algorithms have been developed in past years including the Artificial Potential Field (APF) algorithm which is applied in this thesis.

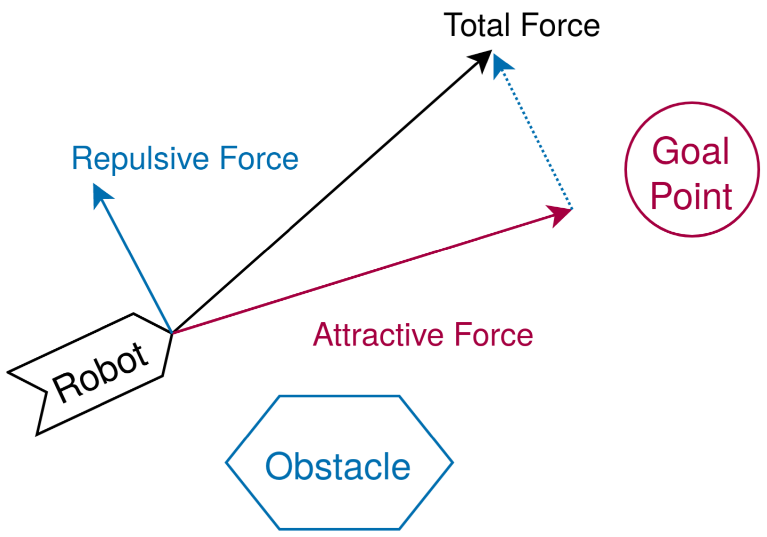


Figure 3.2. Principle of the Artificial Potential Field (APF) algorithm

As shown in Fig. 5.15, the APF algorithm creates an attractive force towards the goal point and applies various repulsive forces from the detected obstacles. The robot moves in the field of forces, where the position to be reached is an attractive goal for the robot and obstacles are repulsive surfaces for the robot part [77].

Commonly, U_{goal} is defined as a parabolic attractor with the formula as [78]:

$$U_{goal} = [dist(q, goal)]^2, \tag{3.3.1}$$

where $dist(q, goal)$ is the distance between the goal and the robot itself. The repulsive force is an inverse of the distance between the robot and the obstacle, which reaches infinity when the robot approaches the obstacle [78]:

$$U_{obstacles} = [dist(q, obstacle)]^{-1}. \tag{3.3.2}$$

Suppose the robot position is at (x_r, y_r), the goal point is at (x_{goal}, y_{goal}), then the attractive force can be calculated as:

$$F_{att} = \begin{cases} F_x = \alpha(x_r - x_{goal})^2, \\ F_y = \alpha(y_r - y_{goal})^2. \end{cases} \tag{3.3.3}$$

Also suppose the detected obstacle is at (x_{obs}, y_{obs}), the repulsive forces can be found as:

$$F_{rep} = \begin{cases} F_x = -\beta(\dfrac{x_{obs}}{x_{obs} - x_r})^2, \\ F_y = -\beta(\dfrac{y_{obs}}{y_{obs} - y_r})^2. \end{cases} \tag{3.3.4}$$

The repulsive force increases inversely proportional to the distance between the robot and the obstacle. Thus, the closer the robot is to the obstacle, the greater the repulsive force generated by the obstacle.

The addition of the repulsive and attractive forces allows obtaining an angle noted $\theta_F = atan(F_x, F_y)$. By a simple comparison of θ_F with the actual orientation θ of the robot, the angular velocity ω is as

$$\omega = k_w(\theta_M - \theta), \qquad -\pi < (\theta_M - \theta) < \pi. \tag{3.3.5}$$

The values of α, β, and k_ω are chosen following various tests and simulations [79].

3.4 Control Objective

The control objective is to autonomously navigate a group of mobile robots through an unknown environment based on the vision sensor of the leaders while avoiding collisions with different sizes of obstacles maintaining a desired formation, adapting its formation according to the environment, and finally reaching a designated goal point efficiently.

In this control objective, the MAS's primary goal is to navigate through an unknown environment while maintaining a consensus formation shape. If there are obstacles in the unknown environment, the MAS should be able to detect them by using a vision sensor and avoid collision for all the agents. The control algorithm should be designed to make real-time decisions, adjusting the robots' formation shape or performing local avoidance to safely and efficiently navigate to the goal. The detailed formation control algorithm will be presented in Chapter 4.

3.5 Vision-aided Obstacle Detection

The vision-aided navigation should apply the advanced vision sensor to detect the obstacles and provide ranging information to the MAS, allowing the leader to make navigation decisions.

As in Chapter 5, the vision sensor used in this thesis provides distance information on each pixel of an image. Obstacle detection means transferring the range signal into the obstacle's location or coordinates according to the MAS pose. As shown in Fig. 3.3, the camera captures two boxes in the front on the left, and the depth camera is able to tell the distance difference between those two boxes in greyscale. The advantage of using a depth camera is it detects complex obstacles such as low-profile obstacles as shown in Fig. 3.4 and overhead obstacles as a desk shown in Fig. 3.5.

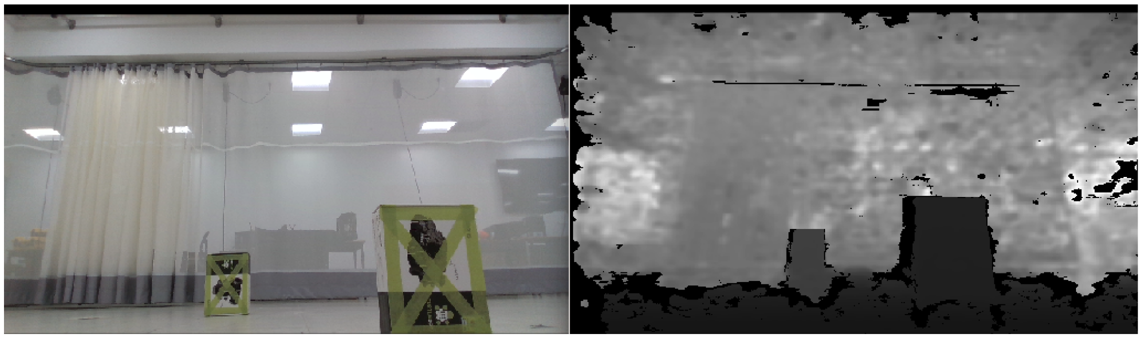


Figure 3.3. Normal obstacle detection

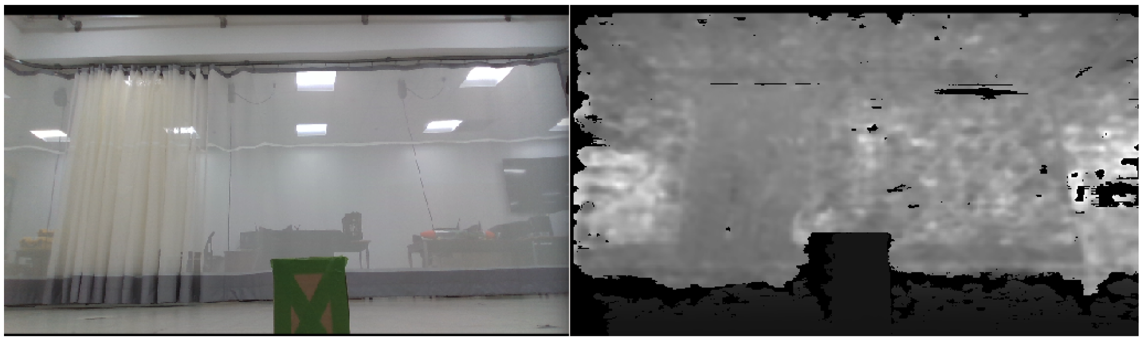


Figure 3.4. Low-profile obstacle detection

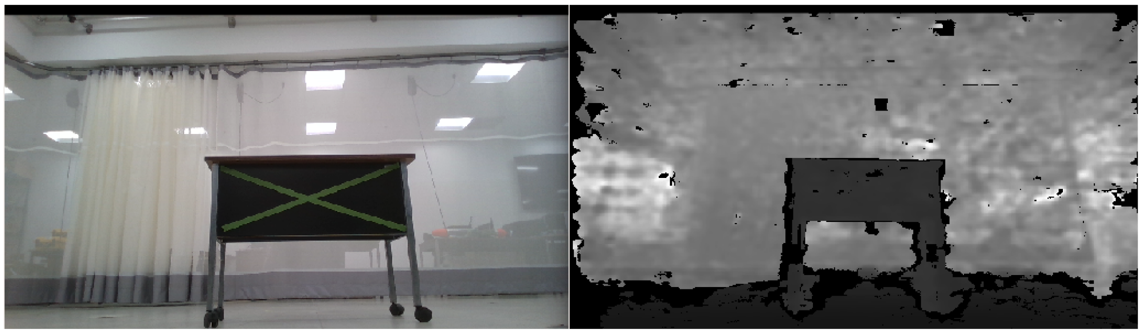


Figure 3.5. Overhead obstacle detection (a desk)

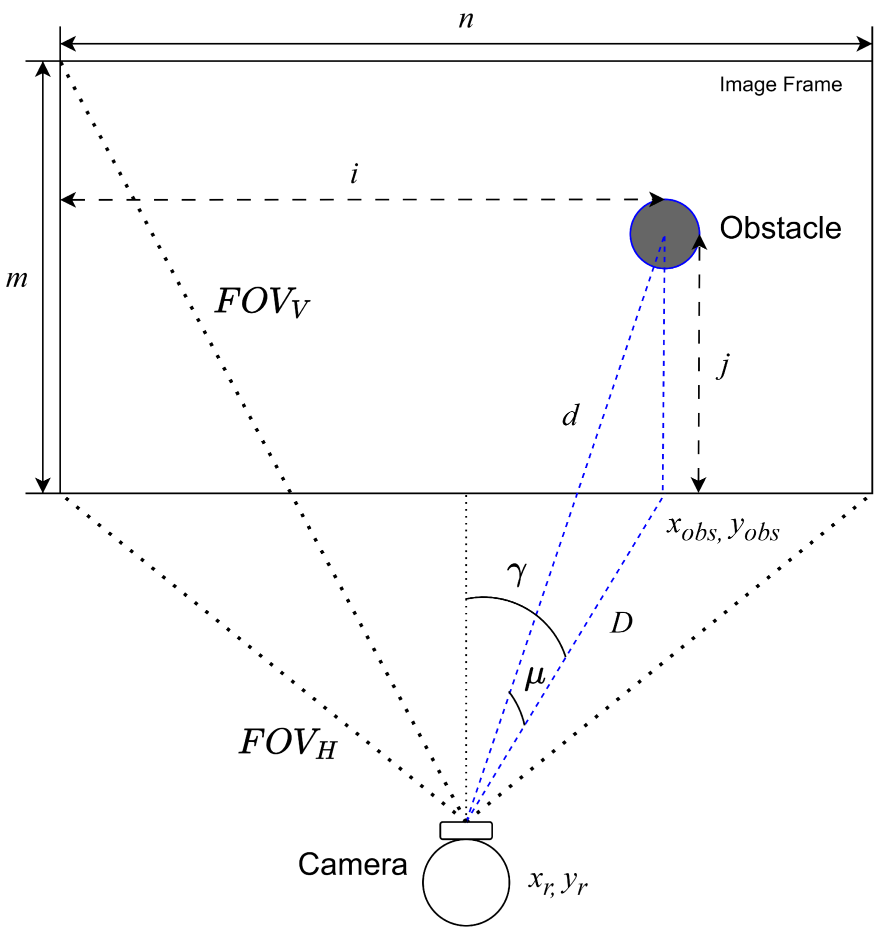


Figure 3.6. Depth camera field of view (FOV) diagram

The output depth image of the camera was transferred to ROS associated with the distance of the objects in the image. To reduce the delay of receiving images, the resolution of captured images was limited to certain horizontal lines of pixels. For example, from Figs. 3.3-3.5, the bottom pixels and the margin pixels on the side are not showing accurate readings and are noisy. The noise shows the distance information on the specific pixel is missing so that it appears to be blacked out on the depth image. Thus, ignoring those data points will not influence the performance of the camera. Each line of pixels is analyzed separately in order to measure obstacles at the same height. Thresholds are set to avoid over-detecting the same obstacle.

In order to transfer the detected obstacle's coordinates to the MAS' frame of reference, the pixel angle of the image was matched with the camera's Field of View (FOV) angle to calculate the obstacle's pose. Consider n as the number of pixels on the width of the image, and m as the number of pixels on the height of the image. FOV$_H$ corresponds to the value of the angle of the horizontal FOV of the camera and FOV$_V$ corresponds to the vertical FOV. The horizontal angle corresponding to each pixel of value i can be obtained:

$$\gamma = \left(\frac{\frac{n}{2} - i}{\frac{n}{2}}\right) \frac{FOV_H}{2}, \tag{3.5.1}$$

where the vertical angle can be obtained:

$$\mu = (\frac{j}{m})FOV_V. \tag{3.5.2}$$

The obtained angle γ is the detected obstacle in the local frame of the camera. With the information on the direct read pixel distance d as shown in Fig. 3.6, the 2D distance D can be calculated as

$$D = dcos\mu, \tag{3.5.3}$$

then with the position of the leader, and the orientation θ, the obstacle position in the global frame can be estimated by the following formula:

$$\begin{cases} x_{obs} = x_r + Dcos(\gamma + \theta), \\ y_{obs} = y_r + Dsin(\gamma + \theta). \end{cases} \tag{3.5.4}$$

3.6 Summary

In this chapter, key aspects that are critical to the algorithm have been comprehensively addressed. The discussion began with an exploration of the kinematics of mobile robots, providing a foundational understanding of the robot's motion dynamics. In addition, the hand position model outlines its significance in manipulating the mobile robot effectively.

The section on the obstacle avoidance clarified the challenges and strategies associated with navigating the MAS in an unknown environment, as the desired results from the control objectives and evaluating the proposed control algorithm at the same time.

Finally, the chapter concluded with an exploration of vision-aided obstacle detection, showing the visual data in enhancing the robot's perception capabilities. This overview not only formulates the problem but also establishes the groundwork for the proposed solution and conducts experiments to address these challenges.

Chapter 4

Multi-Robot Affine Formation Control

In this Chapter, theories of the applied affine formation control and simulation studies are introduced. In [66], the proposed affine formation controllers are model-based with limited simulations and tests conducted. In [17], the affine formation control has been studied on multiple quadcopters and some work has been studied on multiple unicycle-modeled mobile robots as in [80]. In order to verify the robustness of the affine formation controllers proposed in [66], simulations are essential to conduct before applying to experiments with modified controllers, further modifications are made to adapt specific tasks.

4.1 Preliminaries

4.1.1 Graph Theory and Affine Span

Consider a team of n robots in $\mathbb{R}^d$, where $d \geq 2$ and $n \geq d+1$. Let $p_i \in \mathbb{R}^d$ be the ith agent's position and $p = [p_1, p_2, p_3...p_n] \in \mathbb{R}^{dn}$ be the corresponding configuration, d can be considered as the dimension or the degree of freedom. In this thesis, the MAS control problem is focused on mobile robots that can only move in the $x-y$ plane, therefore $d = 2$. If the MAS control problem is applied to aerial vehicles, then $d = 3$ [17].

A fixed graph $\mathcal{G}$=($\mathcal{V}$, $\mathcal{E}$) describes the information flow inside the MAS, where $\mathcal{V}$=$1, 2, 3...n$ means the vertex set and $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}$ means the edge set. If the ith agent accesses the jth agent's information, then it can be regarded as Edge $(i, j) \in \mathcal{E}$. In the graph $\mathcal{G}$, the neighbors of the ith vertex are denoted by $\mathcal{N}_i$=$\{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. In this thesis, the underlying graphs are assumed as undirected, which means the edges are bidirectional, i.e. $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. Thus the undirected graph can be treated as a graph with a sequence of ordered edges from Node i_1 to i_k. A graph has a directed spanning tree if it contains at least one node called the root node which has no parent node and has directed paths from that node to every other nodes [76]. Let m be the number of undirected edges. Denote $\mathbf{I_d} \in \mathbb{R}^{d\times d}$ the identity matrix, $\mathbf{1_n} \in \mathbb{R}^n$ the vector with all entries equal to one.

Given a set of points $\{p_i\}_{i=1}^n \in \mathbb{R}^d$, the affine span $\mathcal{S}$ of these points is [66]

$$\mathcal{S} = \{\sum_{i=1}^{n} a_i p_i : a_i \in \mathbb{R} \text{ for all } i \text{ and } \sum_{i=1}^{n} a_i = 1\}. \tag{4.1.1}$$

$\{a_i\}_{i=1}^n$ are scalars that are not all zero and affinely independent depends on the dimension of the affine span. For example, the affine span of two separated points is the 1D line passing through the two points, and trhe affine span of three points that are not collinear is the 2D plane passing through the three points [66].

4.1.2 Oriented Incidence Matrix

The incidence matrix is used to encode the relation of edges and node pairs in graph theory. An orientation of an undirected graph is the assignment of a direction to each edge with an orientation. The incidence matrix $H \in \mathbb{R}^{m\times n}$ of an oriented graph is the matrix with rows indexed by edges and columns by nodes [65]. An incidence matrix can then be defined in a way that $H[ij] = 1$ if Node j is the head of Edge i, $H[ij] = -1$ if Node j is the tail of Edge i and $H[ij] = 0$ if Edge i and Node j are not connected. In addition, the Laplacian matrix $\mathcal{L}$ for a bi-directional communication topology can be expressed as $\frac{1}{2}H^T H$ [81].

Example 4.1.1 *For the bidirectional communication topology shown in Fig .4.1,*

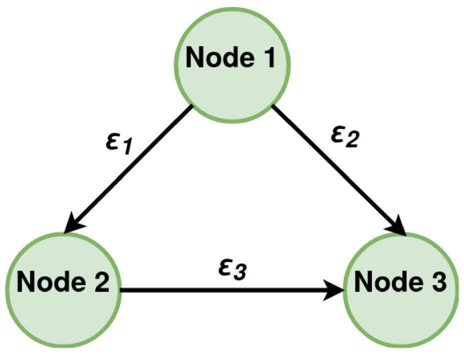


Figure 4.1. A communication topology with three agents as an example

The orientated incidence matrix is given as

$$H = \begin{array}{c} \\ \mathcal{E}_1 \\ \mathcal{E}_2 \\ \mathcal{E}_3 \end{array} \begin{array}{c} \begin{array}{ccc} Node\,1 & Node\,2 & Node\,3 \end{array} \\ \left[\begin{array}{ccc} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{array}\right] \end{array}, \tag{4.1.2}$$

and the Laplacian matrix can be obtained as

$$\mathcal{L} = \frac{1}{2}H^T H = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}. \tag{4.1.3}$$

4.1.3 Kronecker Product

For two matrices, $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{p\times q}$, the kronecker product of A and B is denoted by $A \otimes B$ with a result of a $mp \times nq$ matrix, defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}. \tag{4.1.4}$$

The Kronecker product satisfies the following calculation rules [82]:

$$\begin{aligned} (A \otimes B)^{-1} &= A^{-1} \otimes B^{-1} \\ (A \otimes B)^T &= A^T \otimes B^T \\ (kA) \otimes B &= A \otimes (kB) = k(A \otimes B) \\ (A + B) \otimes C &= A \otimes C + B \otimes C \\ A \otimes (B \otimes C) &= (A \times B) \otimes C \\ (A \otimes B)(C \otimes D) &= (AC) \otimes (BD), \end{aligned}$$

where k is a constant. The first rule holds if and only if both matrices A and B are invertible. The above properties will be frequently used in this thesis.

4.1.4 Stress Matrix

Consider a defined formation as $(\mathcal{G}, p)$, where $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is the associated graph and the vertex i has the position p_i. Define a constant vector r=$[r_1, r_2, r_3 ... r_n]$$\in \mathbb{R}^{2n}$ as the nominal configuration. Then, the associated $(\mathcal{G}, r)$ can be defined as the nominal formation. In the formation $(\mathcal{G}, p)$, all the edges of $\mathcal{G}$ are assigned to a set of scalars $\{w_{ij} \in \mathbb{R} : w_{ij} = w_{ji}\}_{(i,j)\in\mathcal{E}}$, can be positive, negative, or zero in a stress matrix. A stress matrix satisfies an equilibrium:

$$\sum_{j\in\mathcal{N}_i} w_{ij}(p_j - p_i) = 0, i \in \mathcal{V}, \tag{4.1.5}$$

where the vector $w_{ij}(p_j - p_i)$ represents the force applied on Agent i by Agent j through Edge (i, j). An attracting force is $w_{ij} > 0$ along Edge (i, j), otherwise a repelling force when $w_{ij} < 0$. Thus, Eq. (4.1.5) means that the forces applied on Agent i by neighboring agents $j \in \mathcal{N}_i$ are balanced. Note the equilibrium stresses can only be determined up to a scalar factor. The above equation can be expressed in a matrix form as:

$$(\Omega \otimes \mathbf{I_d})p = 0, \tag{4.1.6}$$

where $\Omega \in \mathbb{R}^{n\times n}$ is the stress matrix, which satisfying

$$[\Omega]_{ij} = \begin{cases} 0, & i \neq j, (i,j) \notin \mathcal{E}, \\ -\,\omega_{ij}, & i \neq j, (i,j) \in \mathcal{E}, \\ \sum_{k\in\mathcal{N}_i} w_{ik}, & i = j. \end{cases} \tag{4.1.7}$$

The stress matrix and graph Laplacian matrices have similar structures. The difference is that the stress matrix edge weights can be positive, negative, or zero, while the graph Laplacian matrix edge weights are usually positive [66]. The stress matrix needs to be found ahead using the Linear Matrix Inequalities (LMI) toolbox solver in MATLAB.

According to Eq. (4.1.6), denote $\bar{\Omega} = \Omega \otimes \mathbf{I_d}$ for simplicity. Then the stress matrix could be partitioned according to the partition of leaders and followers as

$$\bar{\Omega} = \begin{bmatrix} \bar{\Omega}_{ll} & \bar{\Omega}_{lf} \\ \bar{\Omega}_{fl} & \bar{\Omega}_{ff} \end{bmatrix}. \tag{4.1.8}$$

Lemma 1: (Generic universal rigidity): Given an undirected graph $\mathcal{G}$ and a generic configuration p, formation $\mathcal{G}, p$ is universally rigid if and only if there exists a stress matrix Ω such that Ω is positive semi-definite and rank(Ω)=$n - d - 1$.

4.1.5 Target Formation

The time-varying configuration of the target formation has the form of

$$p^*(t) = [\mathbf{I_n} \otimes A(t)]r + \mathbf{1_n} \otimes b(t), \tag{4.1.9}$$

where r is the constant configuration, and $A(t) \in \mathbb{R}^{d\times d}$ and $b(t) \in \mathbb{R}^d$ are continuous of t [66]. The desired position of Agent $i \in \mathcal{V}$ in the target formation is $p_i^*(t) = A(t)r_i + b(t)$. With the notion of the target formation, the control problem to be solved in this thesis becomes controlling the group of agents to track the time-varying target configuration so that $p(t) \to p^*(t)$ as $t \to \infty$. A trivial control strategy to solve this problem is to let each agent know $A(t), b(t)$, and r_i so that each agent can track its individual reference trajectory. The disadvantage of the strategy is that it requires $A(t), b(t)$ for all t to be specified in advance and stored on each agent, which is impractical because the information is not able to dynamically respond to unexpected situations such as unexpected obstacles [66].

In order to achieve the target formation in a distributed manner, the leader-follower strategy is adopted. The desired formation maneuvers are merely known by a limited number of agents as leaders, and other agents as followers which only need to follow the motion of the leaders. The affine transformation of the entire formation is achieved by controlling the positions of the leaders. Because the number of leaders is small while testing the formation control law, no specific design coordination for the leaders and simply assume that leaders are being controlled properly. In experiments, the leaders will be controlled by high-level navigation and obstacle detection algorithms and other intelligent decision-making programs. Suppose the position of each leader is equal to the desired value in the target formation, i.e. $p_l(t) = p_l^*(t)$ for all t. Then, the control objective becomes steering the followers such that $p_f(t) \to p_f^*(t)$ as $t \to \infty$. The tracking error can be defined as [66]:

$$\delta_{p_f}(t) = p_f(t) - p_f^*(t) = p_f(t) + \bar{\Omega}_{ff}^{-1}\bar{\Omega}_{fl}p_f^*(t). \tag{4.1.10}$$

4.2 Affine Formation Controller Design

The affine formation controller design in this thesis is based on the distributed affine formation maneuver control laws proposed in [17] and [66]. In order to verify the feasibility and robustness of the controller, all cases are for single or double integrator agents being designed and tested in MATLAB and through Gazebo simulations. This section is separated into three cases based on the increasing task complexity.

In order to verify the controllers proposed or designed in this chapter, multiple simulations were conducted through MATLAB and the Gazebo environment as mentioned in Chapter 5.

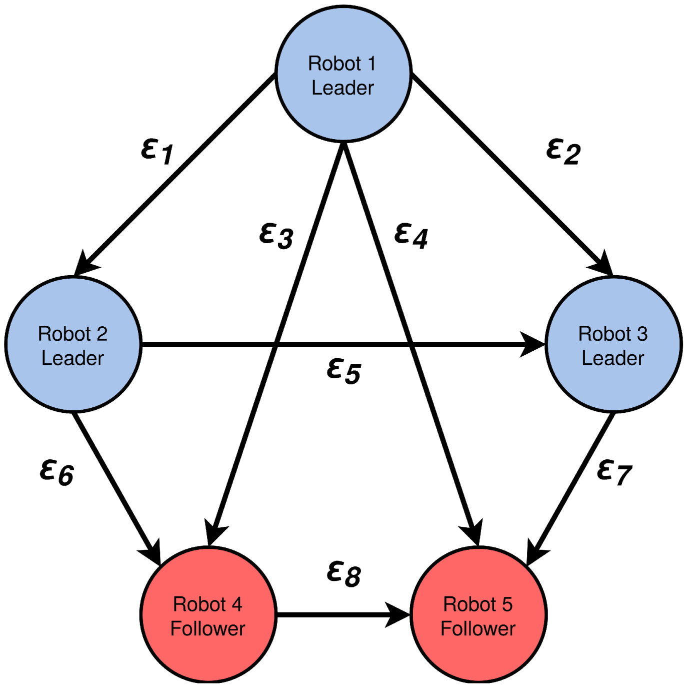


Figure 4.2. Nominal formation configuration for simulations with three leaders (in blue) and three followers (in red)

In this section, simulations were developed with a group of five TurtleBot Burger mobile robots (three leaders and two followers) under a nominal formation configuration shown in Fig. 4.2. Three leaders are selected in this nominal formation to satisfy the affine condition since at least three leaders are required to affinely span in a 2-D space. The communication graph in Fig. 4.2 is universally rigid and the resultant Ω_{ff} is positive definite. The nominal configuration $\{r_i\}_{i=1}^{5}$

for Fig. 4.2 is designed as

$$r_1 = \begin{bmatrix} 0 \\ 2(sin\frac{2\pi}{5} + sin\frac{\pi}{5}) \end{bmatrix}, r_2 = \begin{bmatrix} -1 - 2cos\frac{2\pi}{5} \\ 2sin\frac{2\pi}{5} \end{bmatrix},$$
$$r_3 = \begin{bmatrix} 1 + 2cos\frac{2\pi}{5} \\ 2sin\frac{2\pi}{5} \end{bmatrix}, r_4 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, r_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Thus, according to the oriented incidence matrix introduced in Section. 4.1.2, the incidence matrix for this formation is

$$H = \begin{array}{c} \\ \mathcal{E}_1 \\ \mathcal{E}_2 \\ \mathcal{E}_3 \\ \mathcal{E}_4 \\ \mathcal{E}_5 \\ \mathcal{E}_6 \\ \mathcal{E}_7 \\ \mathcal{E}_8 \end{array} \begin{array}{c} \begin{matrix} Node\,1 & Node\,2 & Node\,3 & Node\,4 & Node\,5 \end{matrix} \\ \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \end{array}. \tag{4.2.1}$$

The normalized equilibrium stress vector for edges $\epsilon_1, ..., \epsilon_8$ can be computed in MATLAB as

$$\omega = [0.5283 \quad 0.5283 \quad -0.2018 \quad -0.2018 \quad -0.3265 \quad 0.3265 \quad 0.3265 \quad 0.2018],$$

and the corresponding stress matrix is

$$\Omega = \begin{bmatrix} 0.6530 & -0.5283 & -0.5283 & 0.2018 & 0.2018 \\ -0.5283 & 0.5283 & 0.3265 & -0.3265 & 0 \\ -0.5283 & 0.3265 & 0.5283 & 0 & -0.3265 \\ 0.2018 & -0.3265 & 0 & 0.3265 & -0.2018 \\ 0.2018 & 0 & -0.3265 & -0.2018 & 0.3265 \end{bmatrix}, \tag{4.2.2}$$

which is the same as the results from [80] that uses the same nominal configuration design.

4.2.1 CASE I: Stationary Leaders

The first case is the simplest case where the leaders are stationary which means the target formation is also stationary, i.e., $\dot{p}_i = 0$ for $i \in \mathcal{V}_l$. The affine formation problem can be solved by the following control law[66]:

$$\dot{p}_i = -\sum_{j \in \mathcal{N}_i} w_{ij}(p_i - p_j), i \in \mathcal{V}_f. \tag{4.2.3}$$

The matrix-vector form of (4.2.3) for followers is:

$$\dot{p}_f = -\bar{\Omega}_{ff} p_f - \bar{\Omega}_{fl} p_l^* = -\bar{\Omega}_{ff} \delta_{p_f}. \tag{4.2.4}$$

If leader velocities are constantly zero, $\dot{p}_l^*(t) = 0$, the tracking error $\delta_{p_f}(t)$ under the control law in Eq. (4.2.3) converges to zero globally and exponentially fast. Substituting Eq. (4.2.4) into $\dot{\delta}_{p_f}$ from Eq. (4.1.10):

$$\dot{\delta}_{p_f} = \dot{p}_f(t) + \bar{\Omega}_{fl} \dot{p}_l^* = -\bar{\Omega}_{ff} \delta_{p_f} + \bar{\Omega}_{fl} \dot{p}_l^*. \tag{4.2.5}$$

Since $\dot{p}_l^* = 0$, the tracking error δ_{p_f} is globally and exponentially stable when $\bar{\Omega}_{ff}$ is non-singular[66]. If the leaders' velocities are not identically zero, the velocities can be viewed as disturbances of the system and can cause non-zero tracking errors. However, since the control law is linear, the tracking error would also be small. Because Eq. (4.2.4) can be rewritten as $\dot{p}_f = -\bar{\Omega}_{ff} \delta_{p_f}$, it can be viewed as a gradient-decent control law for the Lyapunov function:

$$V = \frac{1}{2} \delta_{pf}^T \bar{\Omega}_{ff} \delta_{pf}. \tag{4.2.6}$$

By conducting the time derivative of V and using Eq. (4.2.5),

$$\begin{aligned} \dot{V} &= \delta_{pf}^T \bar{\Omega} \dot{\delta}_{p_f} \\ &= \delta_{pf}^T \bar{\Omega}(-\bar{\Omega}_{ff} \delta_{pf} + \bar{\Omega}_{fl} \dot{p}_l^*) \\ &= \delta_{pf}^T \bar{\Omega}(-\bar{\Omega}_{ff} \delta_{pf}) \leq 0. \end{aligned} \tag{4.2.7}$$

4.2.2 CASE I: Simulation Results

Two scenarios are conducted under Case I. First, assume all the leaders are in the desired position and check whether the followers can achieve the consensus.

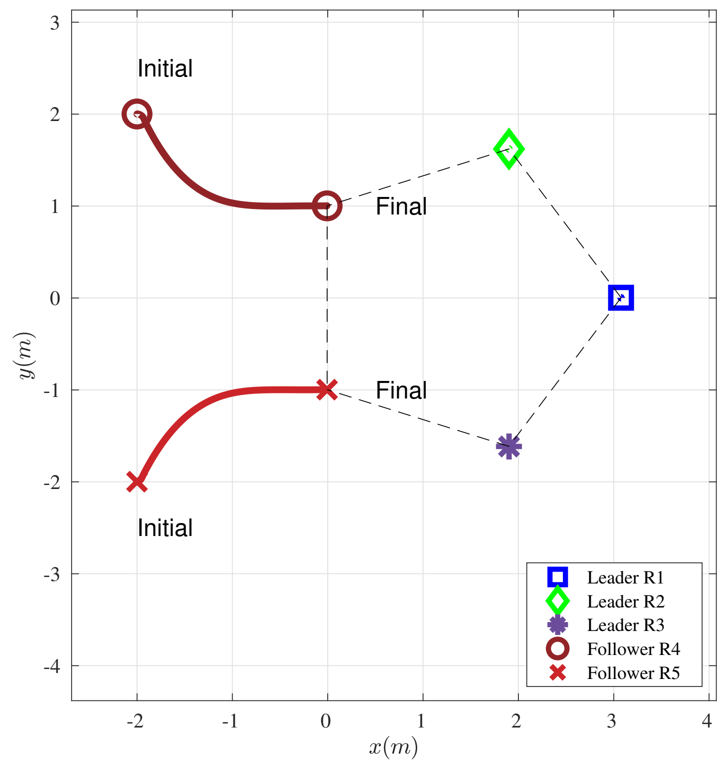


Figure 4.3. Case 1-Scenario 1: Stationary leaders in target positions

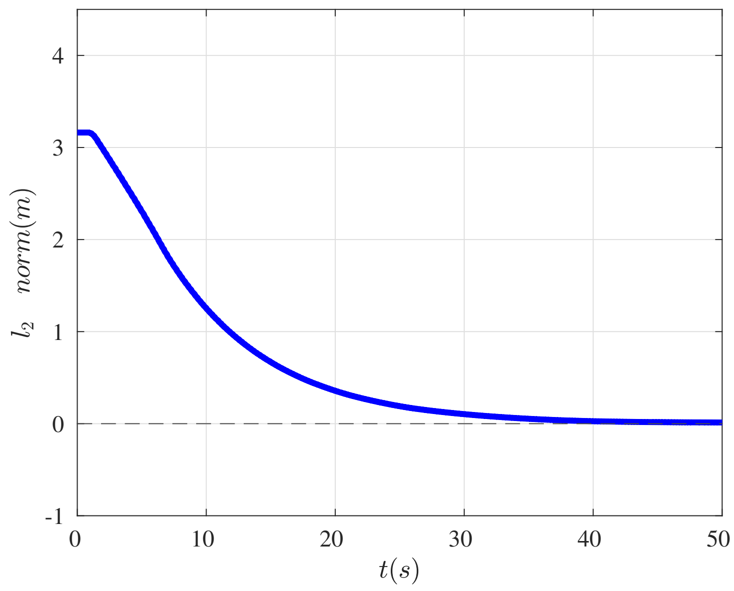


Figure 4.4. Case 1-Scenario 1: l_2 norm of the formation error

As shown in Fig. 4.3, three leaders stay stationary in the desired formation as defined in the nominal configuration, and the two followers start outside of the target formation. The simulation results demonstrate that followers can find their consensus from the initial position to the final position and form the desired target formation. To evaluate the performance of the controller, since there are two follower robots in this simulation, the l_2 norm of formation errors is defined as

$$l_2 = \sqrt{e_4^2 + e_5^2}, \tag{4.2.8}$$

where e_4 and e_5 is the absolute distance error between the actual position to the desired position of Robot 4 and Robot 5. The l_2 norm will be used to evaluate all other controllers in this chapter. The l_2 norm error reduces to zero within 50s as shown in Fig. 4.4. The control profile for both followers in this case is shown in Fig. 4.5.

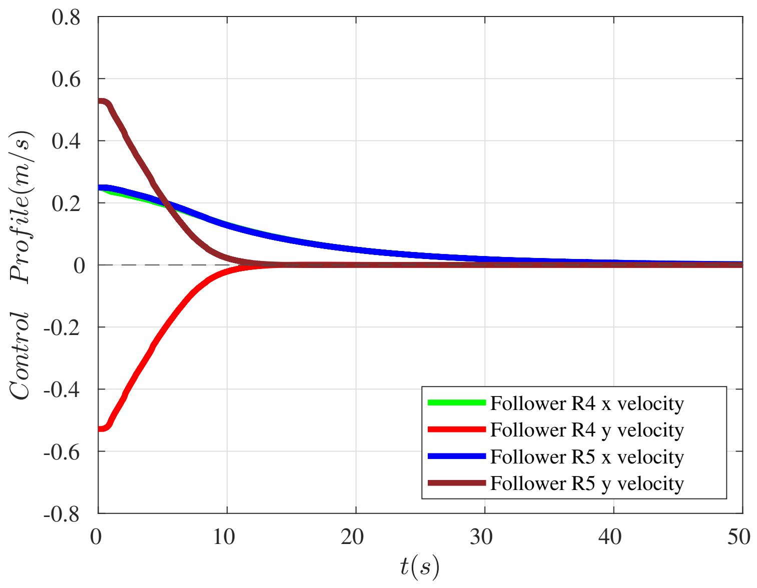


Figure 4.5. Case 1-Scenario 1: Followers' control profiles

For the second scenario, the leaders stay in an arbitrary position which forms an imperfect pentagon shape from the nominal configuration,$r_1 = [4, 2], r_2 = [2, 2], r_3 = [4, 0]$, as shown in Fig. 4.6. The simulation shows the followers are able to join the leaders under the control law in Eq. (4.2.3) to form the real-time formation.

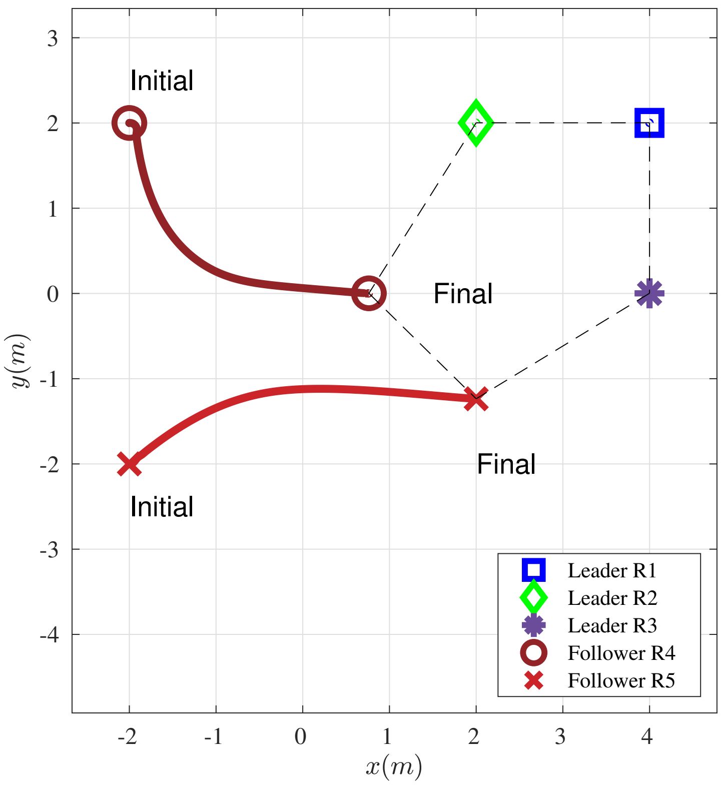


Figure 4.6. Case 1-Scenario 2: Stationary leaders in arbitrary positions

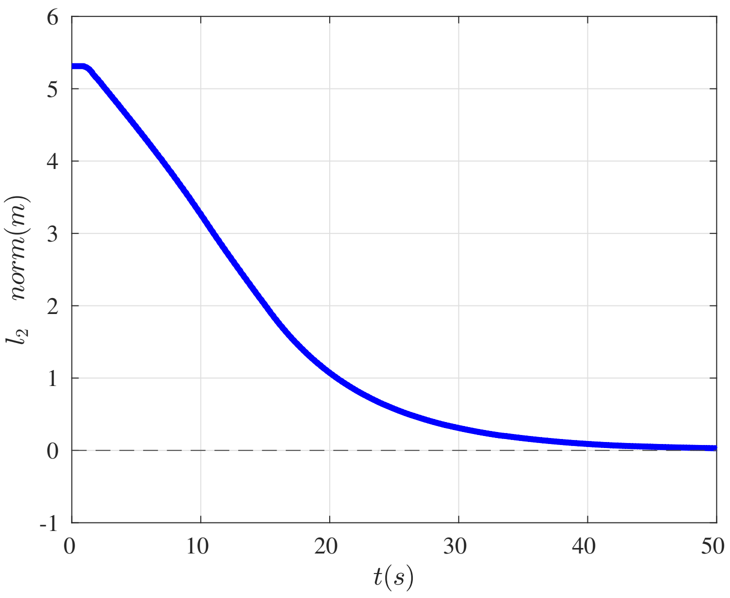


Figure 4.7. Case 1-Scenario 2: l_2 norm of the formation error

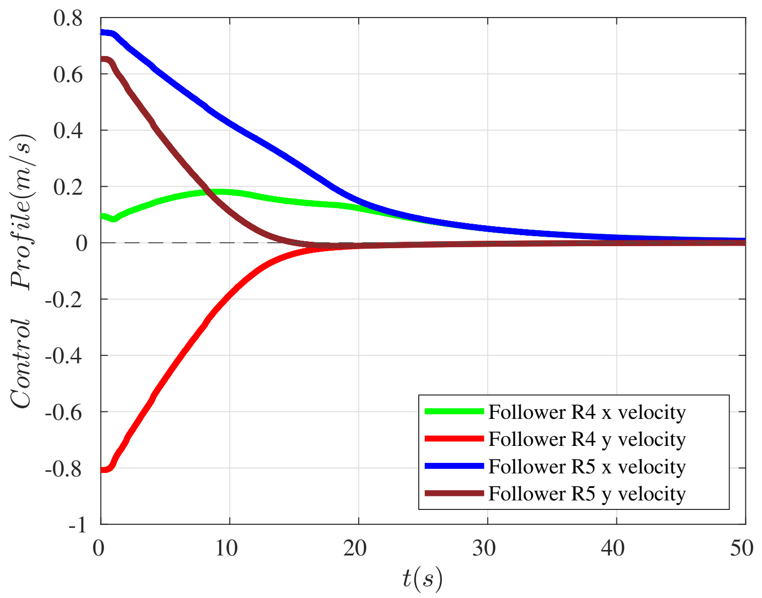


Figure 4.8. Case 1-Scenario 2: Followers' control profiles

Because the leaders' position is not symmetrical compared with the first scenario, the l_2 norm starts at a larger value as both followers' initial positions are further from the desired position compared with Scenario 1. However, the control law in Eq. (4.2.3) still be able to track the error to zero within 50s and result in stable control efforts, as shown in Figs. 4.7-4.8.

4.2.3 CASE II: Leaders with Constant Velocity Motion

The second case allows leaders to move with constant nonzero velocities, the control law in Eq. (4.2.3) is not able to guarantee zero tracking errors. Therefore, an additional integral term is added, as proposed in the following proportional-integral (PI) control law [66]:

$$u = \dot{p}_i = -\alpha \sum_{j \in \mathcal{N}_i} w_{ij}(p_i - p_j) - \beta \int_0^t \sum_{j \in \mathcal{N}_i} w_{ij}(p_i(\tau) - p_j(\tau)) d\tau, i \in \mathcal{V}_f, \quad (4.2.9)$$

where α and β are positive constant control gains. The control law in Eq. (4.2.9) does not require additional measurements compared to Eq. (4.2.3). By defining a new state for the integral term, the control law Eq. (4.2.9) can be rewritten as:

$$\begin{aligned} \dot{p}_i &= -\alpha \sum_{j\in\mathcal{N}_i} w_{ij}(p_i - p_j) - \beta\xi_i, \\ \dot{\xi}_i &= \sum_{j\in\mathcal{N}_i} w_{ij}(p_i - p_j), \quad i \in \mathcal{V}_f. \end{aligned} \tag{4.2.10}$$

The matrix-vector form of (4.2.10) is :

$$\begin{aligned} \dot{p}_f &= -\alpha\bar{\Omega}_{ff}p_f - \alpha\bar{\Omega}_{fl}p_l^* - \beta\xi, \\ \dot{\xi} &= \bar{\Omega}_{ff}p_f + \bar{\Omega}_{fl}p_l^*. \end{aligned} \tag{4.2.11}$$

In this case, the leaders' velocities $\dot{p}_l^*(t)$ are constant, then the tracking error $\delta_{p_f}(t)$ under the action of control law (4.2.9) converges to zero globally and exponentially.

Substituting the control law (4.2.10) into the error dynamics (4.1.10) gives

$$\begin{aligned} \dot{\delta}_{p_f} &= \dot{p}_f + \bar{\Omega}_{ff}^{-1}\bar{\Omega}_{fl}\dot{p}_l^* \\ &= -\alpha\bar{\Omega}_{ff}p_f - \alpha\bar{\Omega}_{fl}p_l^* - \beta\xi + \bar{\Omega}_{ff}^{-1}\bar{\Omega}_{fl}\dot{p}_l^* \\ &= -\alpha\bar{\Omega}_{ff}\delta_{pf} - \beta\xi + \bar{\Omega}_{ff}^{-1}\bar{\Omega}_{fl}\dot{p}_l^*, \end{aligned} \tag{4.2.12}$$

which can be written as

$$\begin{bmatrix} \dot{\delta}_{p_f} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} -\alpha\bar{\Omega}_{ff} & -\beta I_{dn_f} \\ \bar{\Omega}_{ff} & 0 \end{bmatrix} \begin{bmatrix} \delta_{pf} \\ \xi \end{bmatrix} + \begin{bmatrix} \bar{\Omega}_{ff}^{-1}\bar{\Omega}_{fl} \\ 0 \end{bmatrix} \dot{p}_l^*. \tag{4.2.13}$$

Suppose λ is the eigenvalue of the state matrix in (4.2.10), according to [83], it can be obtained as

$$\begin{aligned} det(\begin{bmatrix} \lambda I + \alpha\bar{\Omega}_{ff} & \beta I \\ -\bar{\Omega}_{ff} & \lambda I \end{bmatrix}) &= det\lambda^2 I + \alpha\lambda\bar{\Omega}_{ff} + \beta\bar{\Omega}_{ff} \\ &= det((\alpha\lambda + \beta)(\frac{\lambda^2 I}{\alpha\lambda + \beta} + \bar{\Omega}_{ff})) = 0. \end{aligned} \tag{4.2.14}$$

Suppose σ is the eigenvalue of $\bar{\Omega}_{ff}$. Since $\bar{\Omega}_{ff}$ is symmetric positive definite, $\sigma > 0$. According to Eq. (4.2.14), the results shows that

$$\lambda = \frac{-\beta}{\alpha} < 0, \quad or \quad \frac{\lambda^2}{\alpha\lambda + \beta} = -\sigma. \tag{4.2.15}$$

As a result, the error dynamics is stable and the steady state satisfies [66],

$$\begin{bmatrix} -\alpha\bar{\Omega}_{ff} & -\beta\mathbf{I_{dn_f}} \\ \bar{\Omega}_{ff} & 0 \end{bmatrix} \begin{bmatrix} \delta_{p_f}(\infty) \\ \xi(\infty) \end{bmatrix} + \begin{bmatrix} \bar{\Omega}_{ff}^{-1}\bar{\Omega}_{fl} \\ 0 \end{bmatrix} \dot{p}_l^* = 0, \tag{4.2.16}$$

which follows that $\delta_{p_f}(\infty) = 0$.

4.2.4 CASE II: Simulation Results

When all agents start at the desired target formation, a constant leader velocity is applied as

$$\begin{cases} x_{v_l} = 0.1, \\ y_{v_l} = 0.1. \end{cases} \tag{4.2.17}$$

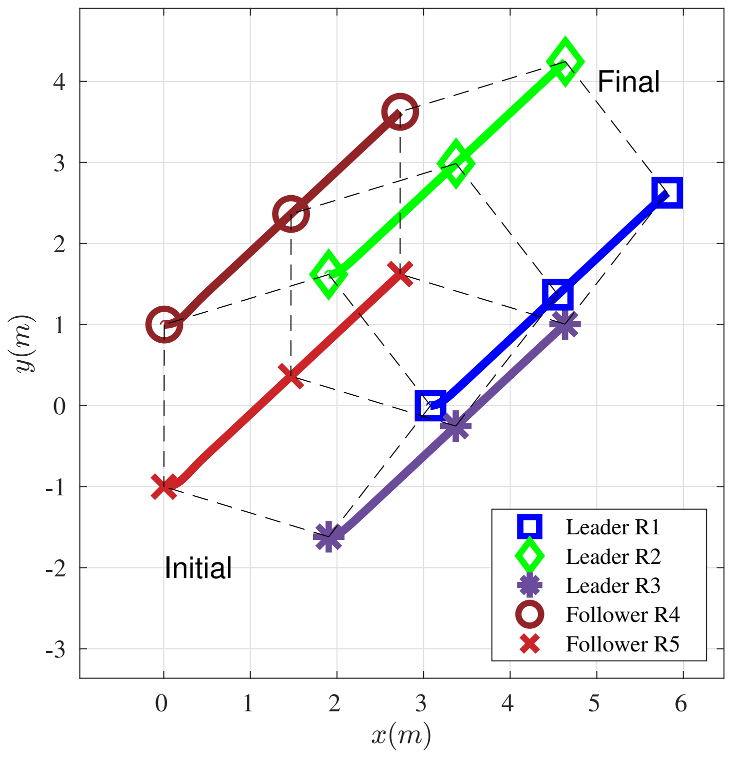


Figure 4.9. Case 2-Scenario 1: Leader with constant velocity (followers in initial desired formation, $\alpha = 10$, $\beta = 1.5$)

As shown in Fig. 4.9, when all robots are in the desired formation, and the leaders are moving with constant velocities, the followers can maintain target formation under the control law in Eq. (4.2.9).

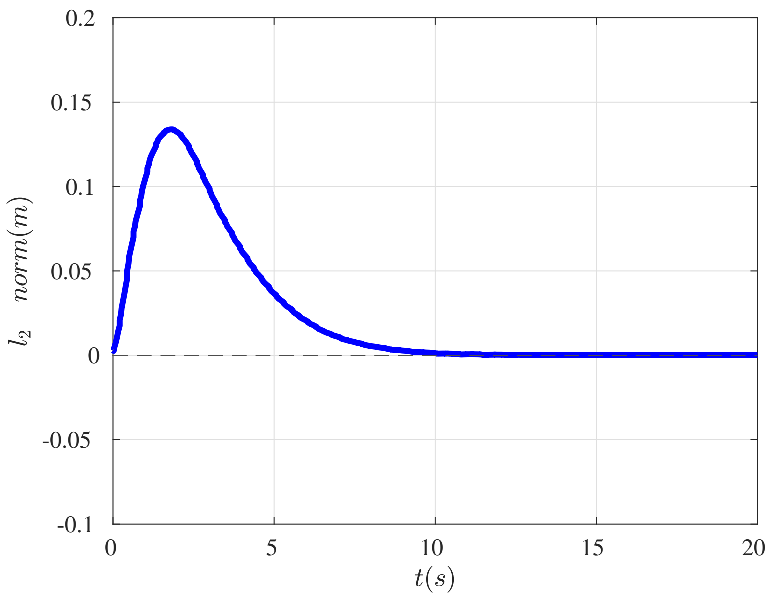


Figure 4.10. Case 2-Scenario 1: l_2 norm of the formation error

The l_2 norm can be tracked to zero within 10s as shown in Fig. 4.10.

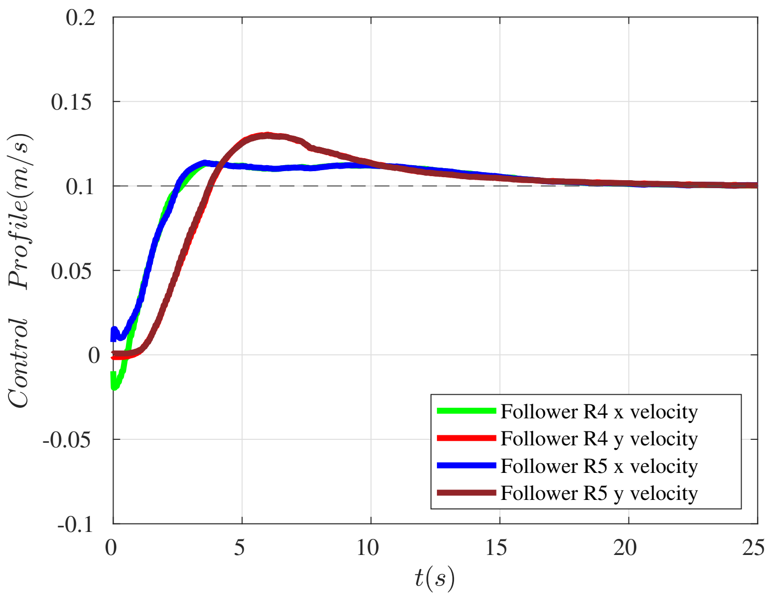


Figure 4.11. Case 2-Scenario 1: Followers' control profiles

Since the leaders' velocities are defined as Eq. (4.2.17), the control law in Eq. (4.2.9) is able to track followers' velocities to $0.1m/s$, as shown in Fig. 4.9. Two followers track leaders' velocities while maintaining the formation successfully.

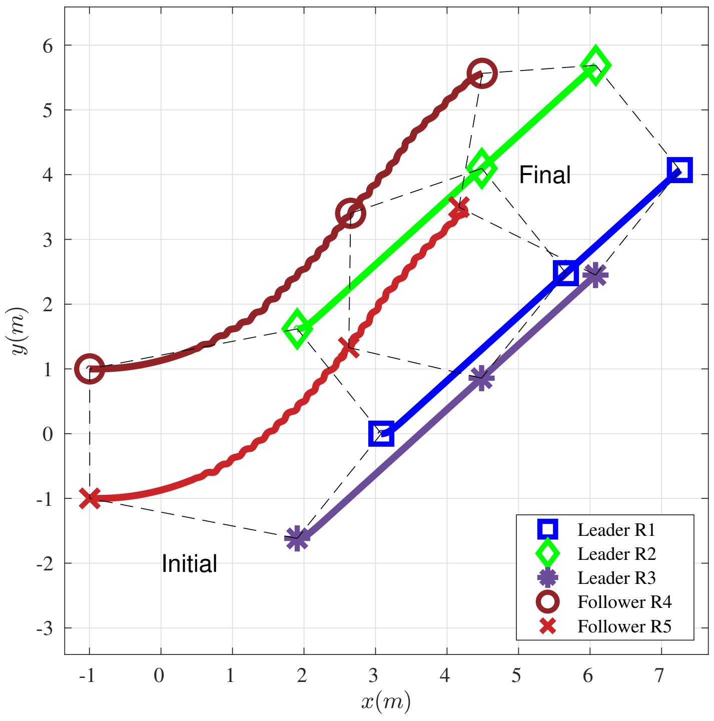


Figure 4.12. Case 2-Scenario 2: Leader with constant velocity (the followers are not in their initial desired formation, $\alpha = 10$, $\beta = 1.5$)

Another scenario to verify the control law would be to let followers starts outside of the target formation, starting at $r_4 = [-1, 1], r_5 = [-1, -1]$ shown in Fig. 4.12.

Although, from the simulation results, the followers are able to join with the moving leaders in an approaching style and stay in the target formation, the path or the velocity control is not feasible. As shown in Fig. 4.12, the followers' movement become wiggling while moving along with the team, shows that the initial velocity input for followers while joining the formation makes the control law in Eq. (4.2.9) not stable. As in Eq. (4.2.9), the two constants $\alpha = 10$ and $\beta = 1.5$ are applied to Scenario 1, which is not optimized for Scenario 2 to lower the disturbances caused by the followers' initial velocities when joined the target formation. However, finding the best control gain or optimizing the PI controller is not in the scope of the simulations. Thus, an extra control gain term or another control gain could be applied to improve the performance. As this scenario would not be applicable in this thesis, further optimization is not discussed.

4.2.5 CASE III: Leaders with Time-varying Velocity Motion

As in real-life applications, the leader velocities cannot be constant all the time. When the leader velocities are time-varying, the PI control law in Eq. (4.2.9) is not able to ensure zero tracking errors. In order to solve this time-varying case, the following control law that requires the velocity feedback is proposed based on [66] with a gain term Q introduced as the formation control novelty to improve the performance as the previous case did poorly on the velocity control:

$$\dot{p}_i = -\frac{Q}{\gamma_i} \sum_{j \in \mathcal{N}_i} w_{ij}[(p_i - p_j) - \dot{p}_j], i \in \mathcal{V}_f, \tag{4.2.18}$$

where $\gamma_i = \sum_{j \in \mathcal{N}_i} w_{ij}$. The non-singularity of γ_i is guaranteed by the affine localizability. Based on the previous definition on the formation stress matrices, $\gamma_i = [\Omega]_{ii}$, and all Ω_{ff} is positive definite because all the diagonal entries are positive, $\gamma_i > 0$ for all $i \in \mathcal{V}_f$.

Thus, if the leader velocity $\dot{p}_l(t)$ is time-varying and continuous, then the tracking error $\delta_{p_f}(t)$ under the action of control law in Eq. (4.2.18) converges to zero globally and exponentially fast. Multiplying γ_i on both sides of Eq. (4.2.18) and omit the constant term Q:

$$\sum_{j \in \mathcal{N}_i} w_{ij}(\dot{p}_i - \dot{p}_j) = -\sum_{j \in \mathcal{N}_i} w_{ij}(p_i - p_j), i \in \mathcal{V}_f. \tag{4.2.19}$$

Now denote $\epsilon_i = \sum_{j \in \mathcal{N}_i} w_{ij}(p_i - p_j)$, Eq. (4.2.19) can be written as $\dot{\epsilon}_i = -\epsilon_i$, which shows that ϵ_i converges to zero globally and exponentially fast. If $\epsilon_i = 0$ for all $i \in \mathcal{V}_f$, then $-\bar{\Omega}_{ff} p_f - \bar{\Omega}_{fl} p_l^* = 0$, which means $\bar{\Omega}_{ff} \delta_{p_f} = 0 \Rightarrow \delta_{p_f} = 0$.

4.2.6 CASE III: Simulation Results

This case is conducted in two scenarios to test the controller in Eq. (4.2.18). The first scenario is a simple case for time-varying leader velocities that is defined by

$$\begin{cases} x_{v_l} = 0.1cos(t/200), \\ y_{v_l} = 0.1sin(t/200), \end{cases} \tag{4.2.20}$$

which allows the leaders to steer in a curvature path within a short period of time (100s). The followers start in the same position as in Case 2-Scenario 2 with an offset from the nominal configuration. The simulation result is shown in Fig. 4.13.

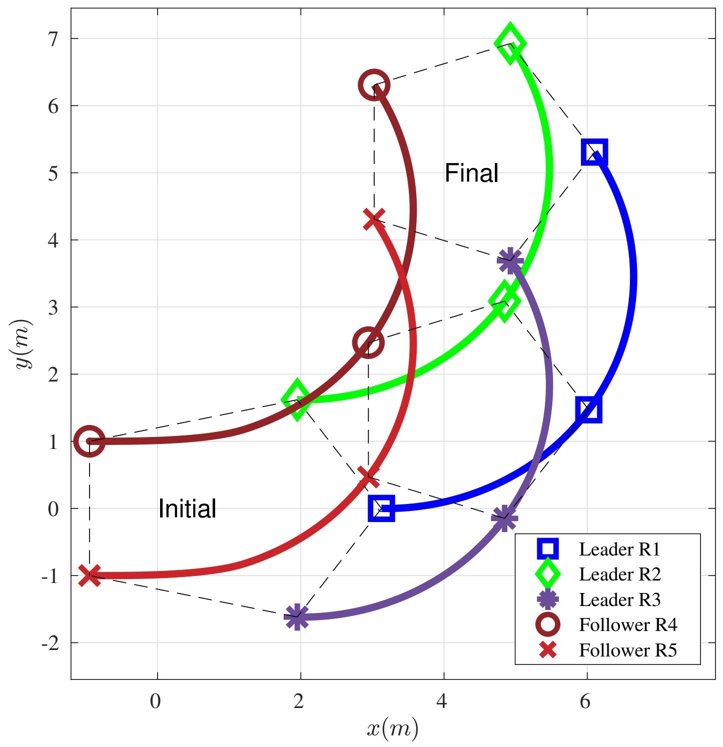


Figure 4.13. Case 3-Scenario 1: Simulation result of time-varying leader velocities

The two followers under the control law in Eq. (4.2.18) can track leaders' velocities while joining and maintaining the desired pentagon formation.

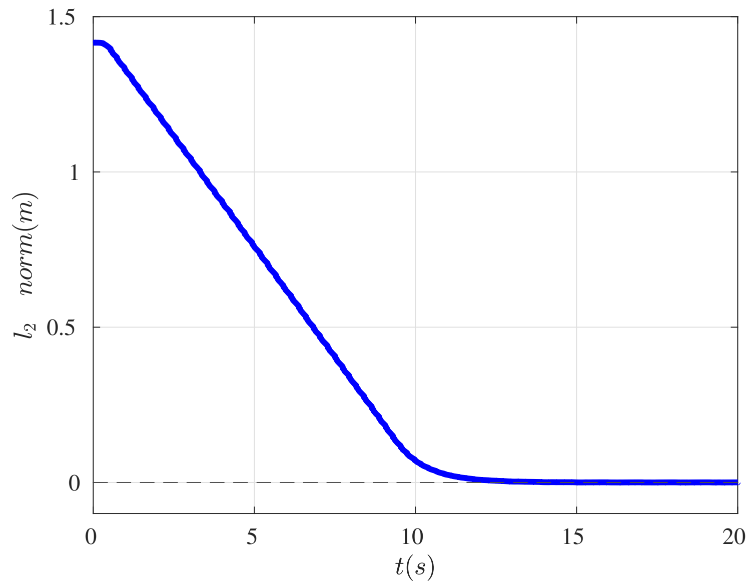


Figure 4.14. Case 3-Scenario 1: l_2 norm of the formation error

Although the follower robots do not initially stay in the target formation, they can track their position error to zero using about 12s, as shown in Fig. 4.14.

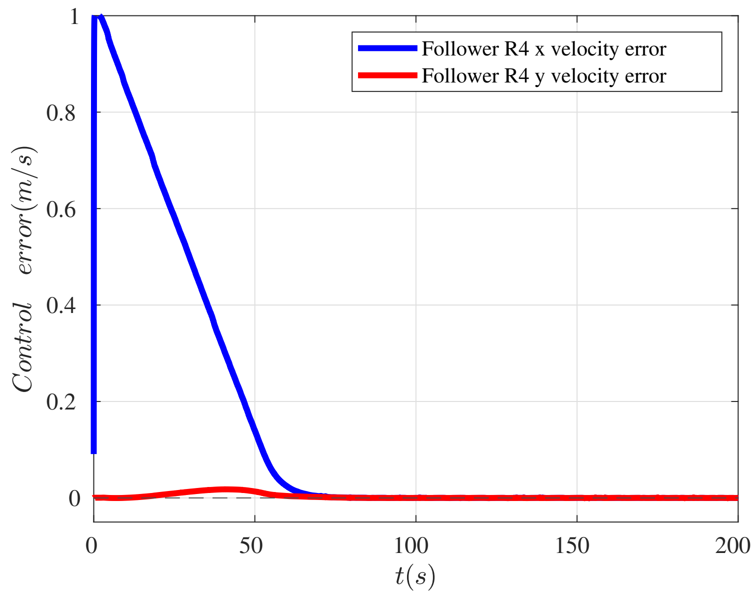


Figure 4.15. Case 3-Scenario 1: Simulation result of Robot 4 control error

Since the leaders' velocities are defined as in Eq. (4.2.20), the control law in Eq. (4.2.18) is able to track the control effort and results in zero velocity difference in both x and y directions.

In addition, changing the target formation in order to overcome different environments, another scenario of scaling is conducted. As shown in Fig. 4.16.

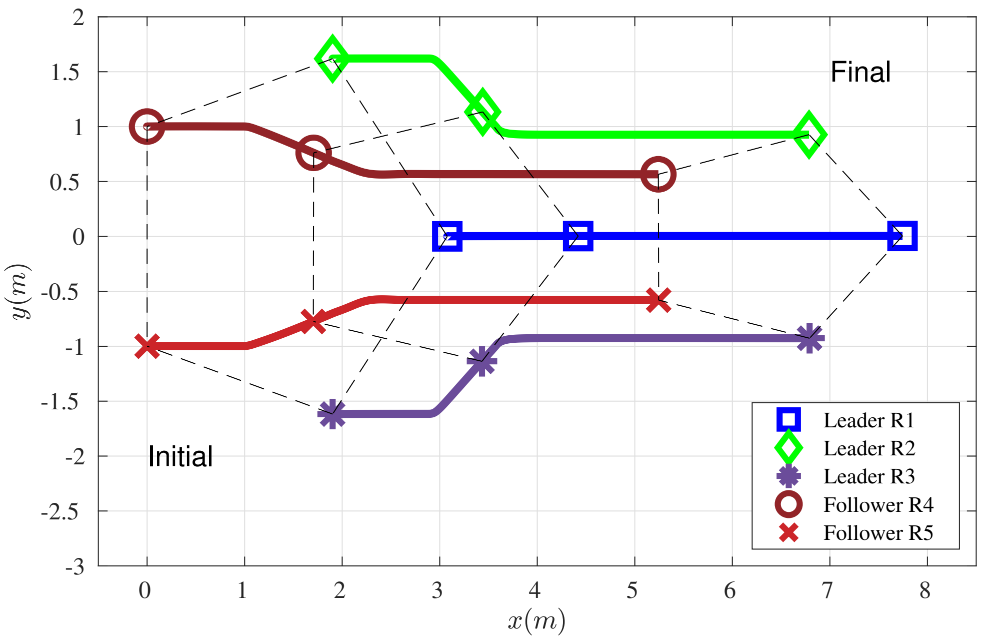


Figure 4.16. Case 3-Scenario 2: Simulation result of the formation change

4.3 Summary

In conclusion, the proposed and designed controllers through simulations have confirmed the effectiveness of control strategies in achieving the desired formation control of MAS. The simulation results have provided compelling evidence that the controllers not only meet the specified performance but also adapted to different navigation scenarios with reliability and robustness. As the affine formation controller in Eq. (4.2.18) is feasible to apply on a real-world MAS, experiments of combining obstacle detection and avoidance with formation control are ready to be performed.

Chapter 5

Experimental Results

Although there are three Pioneer mobile robots in the lab that can communicate through ROS to perform experiments, the experiments are conducted only on the TurtleBot mobile robots for consistency with the simulations and the simplicity of the hardware setup. The experiment implemented the proposed algorithm on four mobile robots, and only one follower was controlled by the proposed affine formation maneuver controller shown in Eq. (4.2.18). A diamond nominal formation configuration was designed $(\mathcal{G}, r)$ as shown in Fig. 5.1.

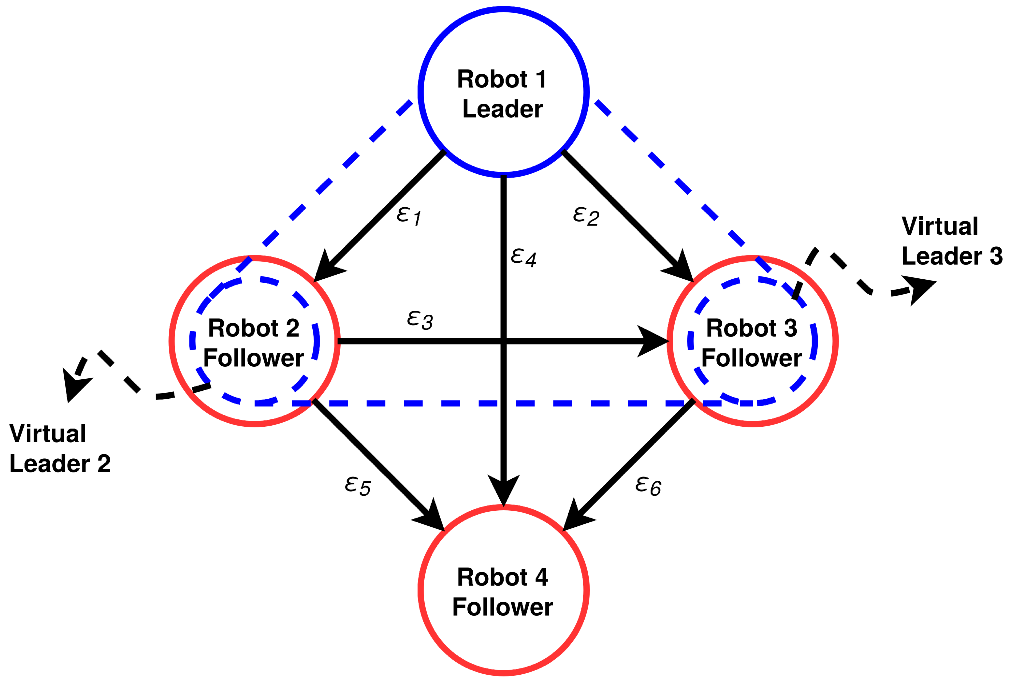


Figure 5.1. Diamond shape nominal formation configuration with leaders (in blue) and followers (in red)

Three leaders, one robot leader and two virtual leaders (in blue dashed line) are selected in order to satisfy Assumption 1. In previous simulations, leaders are assumed to have knowledge of their environments and their paths, which is not true for real-world applications. Thus, virtual leaders are designed to achieve the formation decisions and Robot 2 and Robot 3 become followers as they are tracking the positions of virtual Leader 2 and virtual Leader 3 with obstacle avoidance. Table. 5.1 shows the nominal configuration $\{r_i\}_{i=1}^{5}$.

Table 5.1. Diamond shape formation configuration

Agent	x	y
Robot 1	0	0.5
Robot 2	-0.5	0
Robot 3	0.5	0
Robot 4	0	-0.5

As in the configuration shown in Fig. 5.1, Edges $\mathcal{E}_1, .., \mathcal{E}_6$ define the nodes being neighbors with each other. The incidence matrix $H \in \mathbb{R}^{6\times4}$ for this formation is

$$H = \begin{array}{c} \\ \mathcal{E}_1 \\ \mathcal{E}_2 \\ \mathcal{E}_3 \\ \mathcal{E}_4 \\ \mathcal{E}_5 \\ \mathcal{E}_6 \end{array} \begin{array}{c} \begin{array}{cccc} robot\ 1 & robot\ 2 & robot\ 3 & robot\ 4 \end{array} \\ \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \end{array}, \tag{5.0.1}$$

where "-1" means the robot is sending information on the edge, "1" denotes the robot is receiving information on the edge and "0" means the robot is not on this edge. Using the LMI solvers toolbox in MATLAB [84], the normalized equilibrium stress vector for edges can be computed based on the nominal configuration and the corresponding stress matrix is

$$\Omega = \begin{bmatrix} 0 & -0.4083 & -0.4083 & 0.4083 \\ -0.4083 & 0 & 0.4083 & -0.4083 \\ -0.4083 & 0.4083 & 0 & -0.4083 \\ 0.4083 & -0.4083 & -0.4083 & 0 \end{bmatrix}. \tag{5.0.2}$$

During the experiments, the virtual leaders are commanded to stay behind the leader robot based on the configuration in Table. 5.1, to maintain a rigid formation shape while navigating. Different formation shapes can be defined the same way from Fig. 5.1. In order to perform the narrow gap passing task, a linear shape formation was also designed to produce a different stress matrix.

For low-level control, the hand position model has $L = 0.08m$ for both types of mobile robots as in Eq. (3.2.4). After multiple tests, gain $Q = 0.7$ and $Q = 0.2$ in Eq. (4.2.18) are used for the diamond shape configuration and linear configuration respectively. Leader 1's linear velocity is set to be constant as$0.05m/s$. The experiment was conducted in four tasks. The first task tests the proposed affine formation maneuver controller on simple goal point navigation. The second task tests the depth camera obstacle detection. The third task tests the APF avoidance after detecting obstacles. The fourth task tests the APF avoidance on the affine formation control. The proposed algorithm is shown in the flowchart in Fig. 5.2.

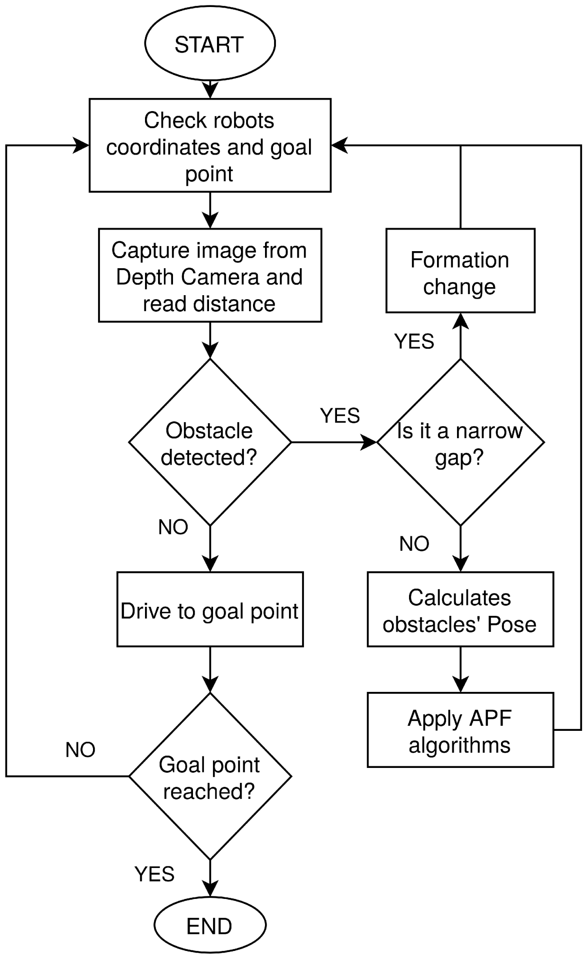


Figure 5.2. Flow chart of the algorithm

5.1 TASK I: Goal Point Navigation

The first task is to let the leader bring the team of robots to navigate from the initial point to the goal point, as shown in Fig. 5.3.

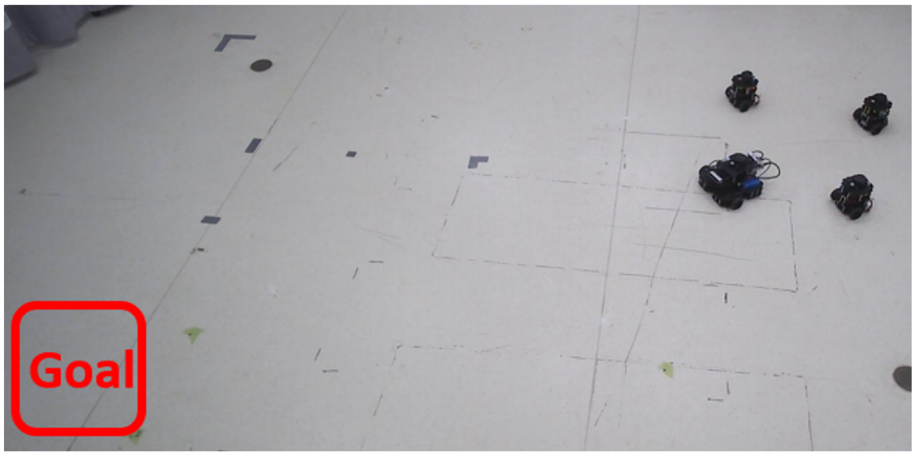


Figure 5.3. Experimental task 1: Goal point navigation experiment environment

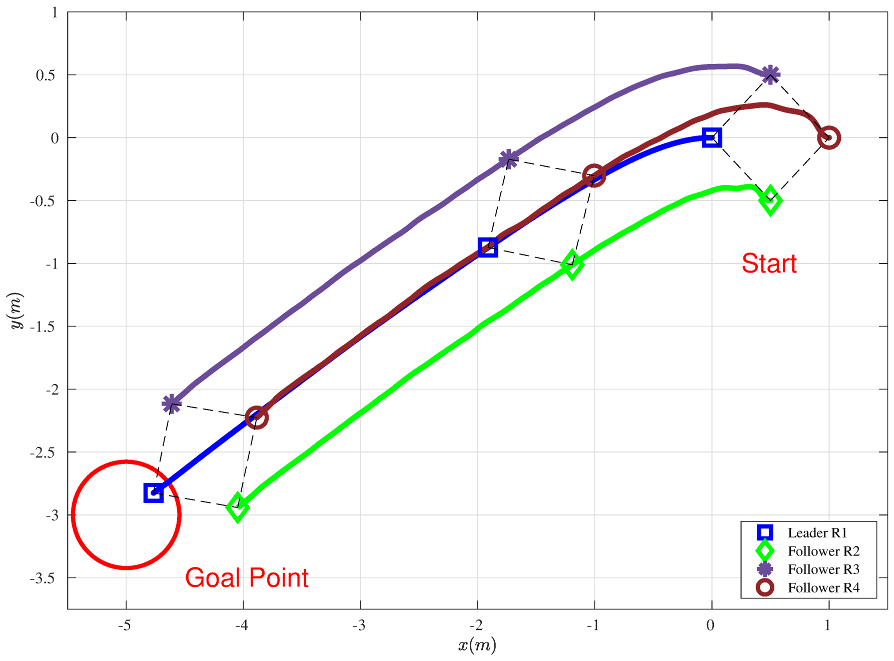


Figure 5.4. Experimental task 1: Goal point navigation

As the result plotted in Fig. 5.4, the leader robot reached the goal point and the follower robots 2 and 3 both tracked the virtual leader's position and followed behind the leader robot. During the navigation, the follower Robot 4 does not have the information about the goal point and reaches the goal point while achieving the desired diamond configuration with other robots. Thus, the controller proposed in Eq. (4.2.18) is feasible and ready to be applied to challenging tasks.

5.2 TASK II: Narrow Gap Passing

The second task is to test whether the leader robot can use the depth camera to detect certain obstacles environment. Here a task with a narrow gap formed by two low-profile boxes is being used for testing. The low-profile boxes appear in the depth image as shown in Fig. 5.5.

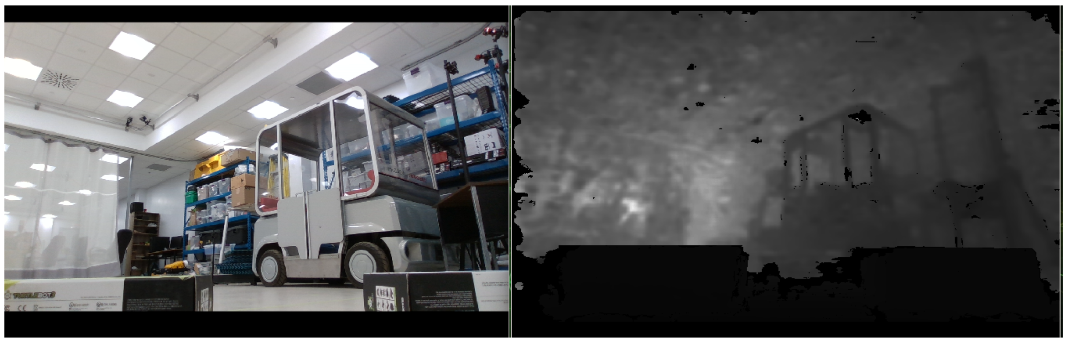


Figure 5.5. Leader depth camera view during narrow gap passing, RGB(left), depth(right)

As shown in Fig. 5.5, although the obstacles are low-profile and will not be detected by the onboard LiDAR sensor, the depth camera received the distance information on both obstacles and decided to perform a formation change, by adjusting the positions of the virtual leaders, as algorithm shown in Fig. 5.2.

The narrow gap passing task requires two formation strategies, as shown in Fig. 5.6 and Fig. 5.7, the team of robots changed its formation from a diamond configuration to a linear configuration before bumping into obstacles.

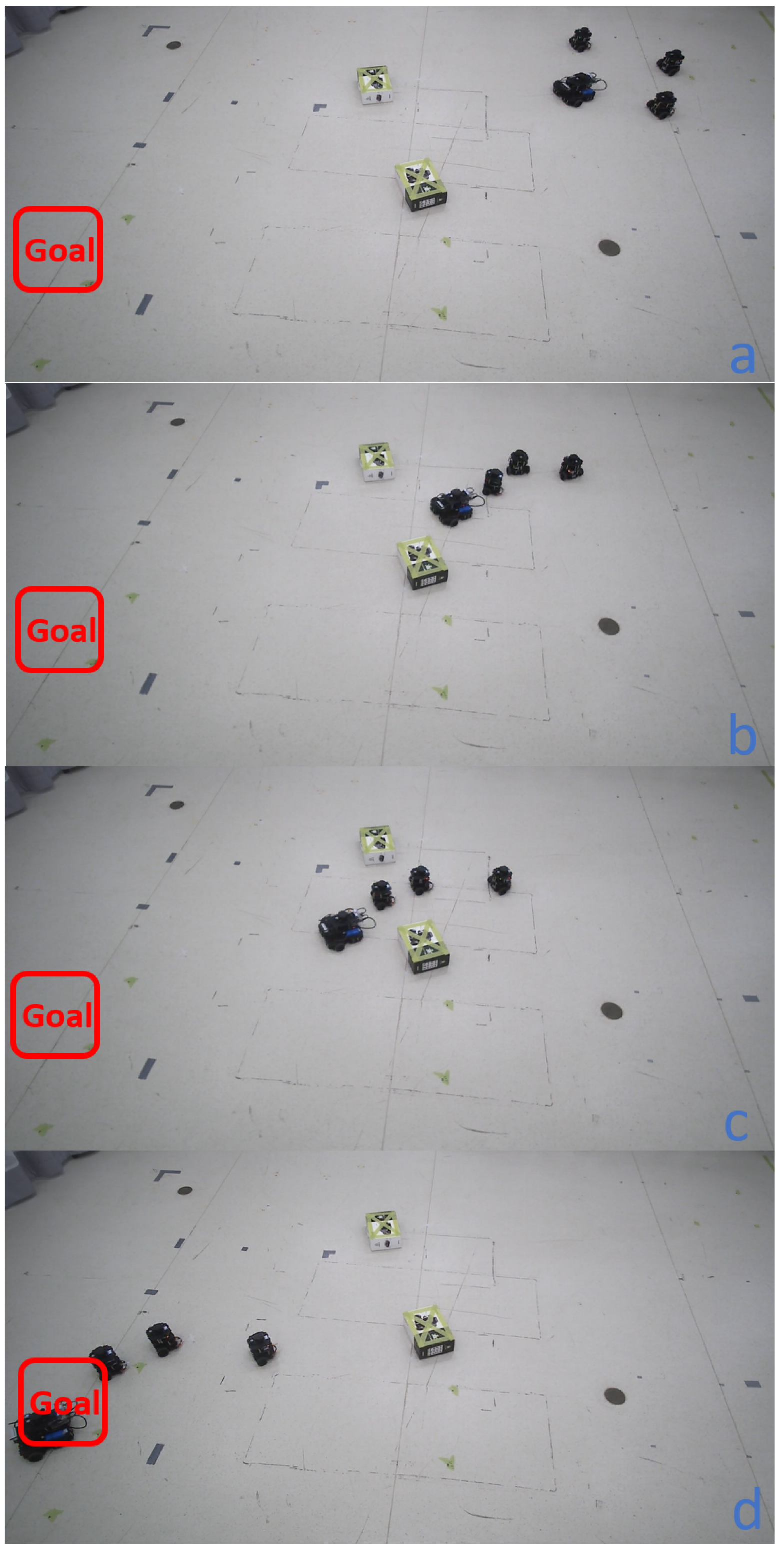


Figure 5.6. Snapshots of experimental task 2: (a) t=20s, (b) t=30s, (c) t=35s, (d) t=50s

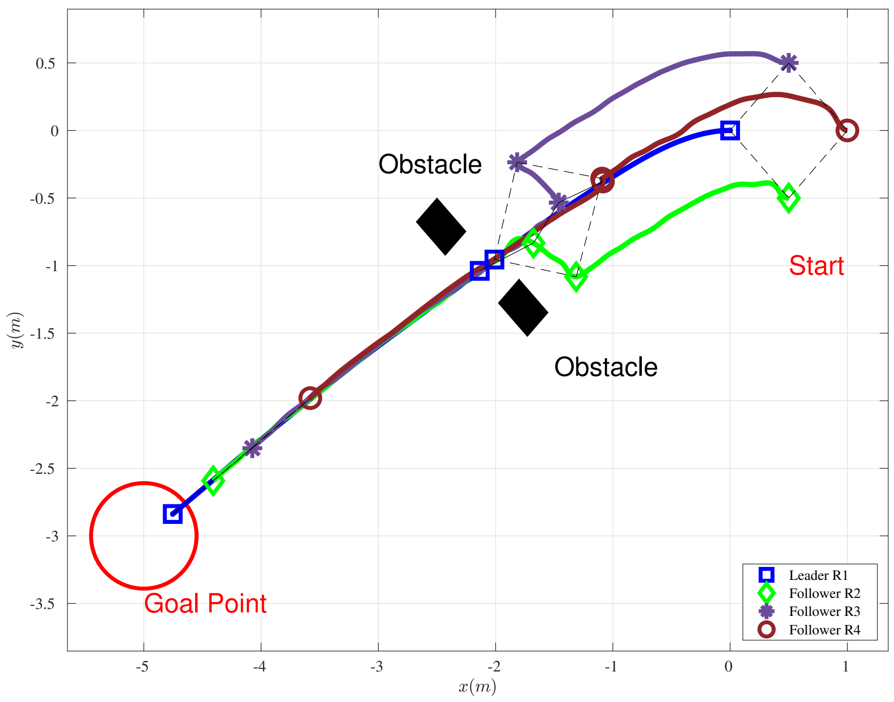


Figure 5.7. Experimental task 2: Narrow gap passing experimental results

In Fig. 5.7, follower Robot 4 adapted both configurations well without running out of formation. In Fig. 5.6, the robots are not perfectly staying in the formation because of the error between the actual encoder with the ROS messages. The reasons could be the motor performance and/or wheels' friction difference. Since the controller was running on the ROS data, Fig. 5.7 demonstrates the proposed algorithm functions as expected. The video of Task I and Task II are shown as in the video of the experiment: https://youtu.be/it1l3p-nODU.

5.3 TASK III: Leader Obstacle Avoidance

The third task mainly focused on the obstacle avoidance part of the algorithm. In this task, three scenarios are set up for the leader robot to perform obstacle avoidance. The first scenario is to set two regular boxes in the environment to test the feasibility of the proposed algorithm. The snapshots of the experiment result are shown in Fig. 5.8.

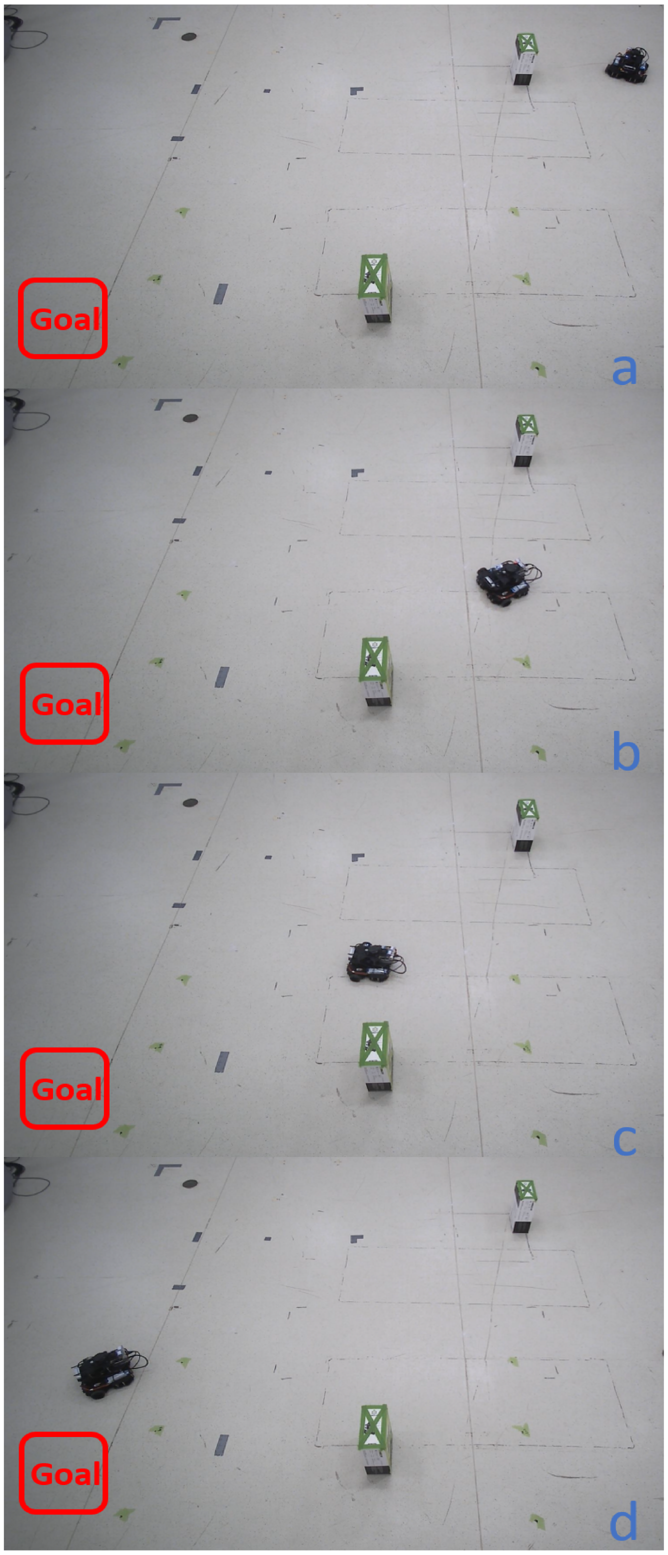


Figure 5.8. Snapshots of experimental task 3-scenario 1: (a) t=20s, (b) t=30s, (c) t=35s, (d) t=50s

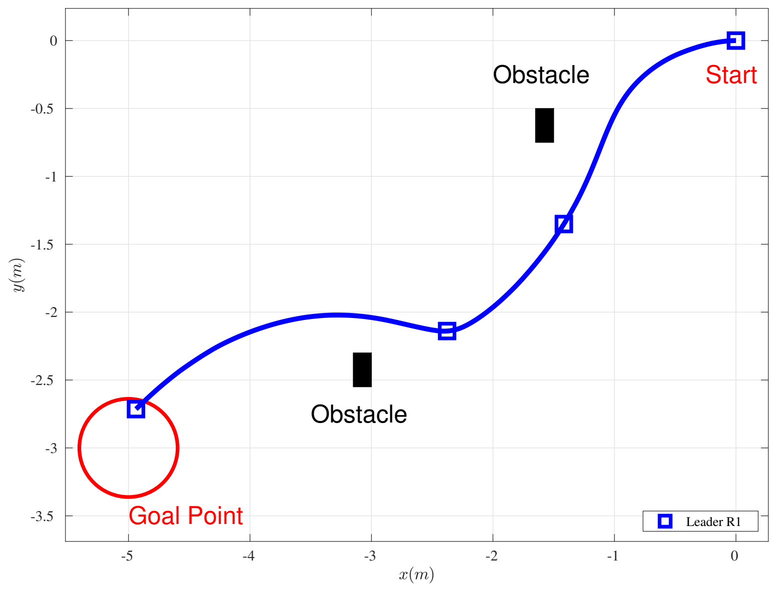


Figure 5.9. Experimental task 3-scenario 1: Two obstacles detection and avoidance

Robot 1 with the depth camera is able to detect multiple obstacles in the unknown environment and apply the algorithm proposed in Fig. 5.2 to avoid both obstacles with the APF, resulting in a smooth path from the start point to the goal point.

However, it is worth mentioning that no external position feedback of state during all the experiments. The leaders' pose deviates from the calculated state due to hardware error and human error from the beginning, which accumulates through the experiments. As the robots' actual position is not aligned with the shared states, the detected obstacles' position could be shifted from its actual position, causing collisions for other robots. For this thesis experiment, thresholds are assigned to estimated obstacles' positions to compensate. The deviation could be seen in both task 1 and 2 experimental results.

The second scenario is to set a low-profile box that is significantly shorter than the robot in the environment. The snapshots of the experiment result are shown in Fig. 5.10. From Fig. 5.11, the robot is able to detect the low-profile box as an obstacle and avoid it using the proposed algorithm.

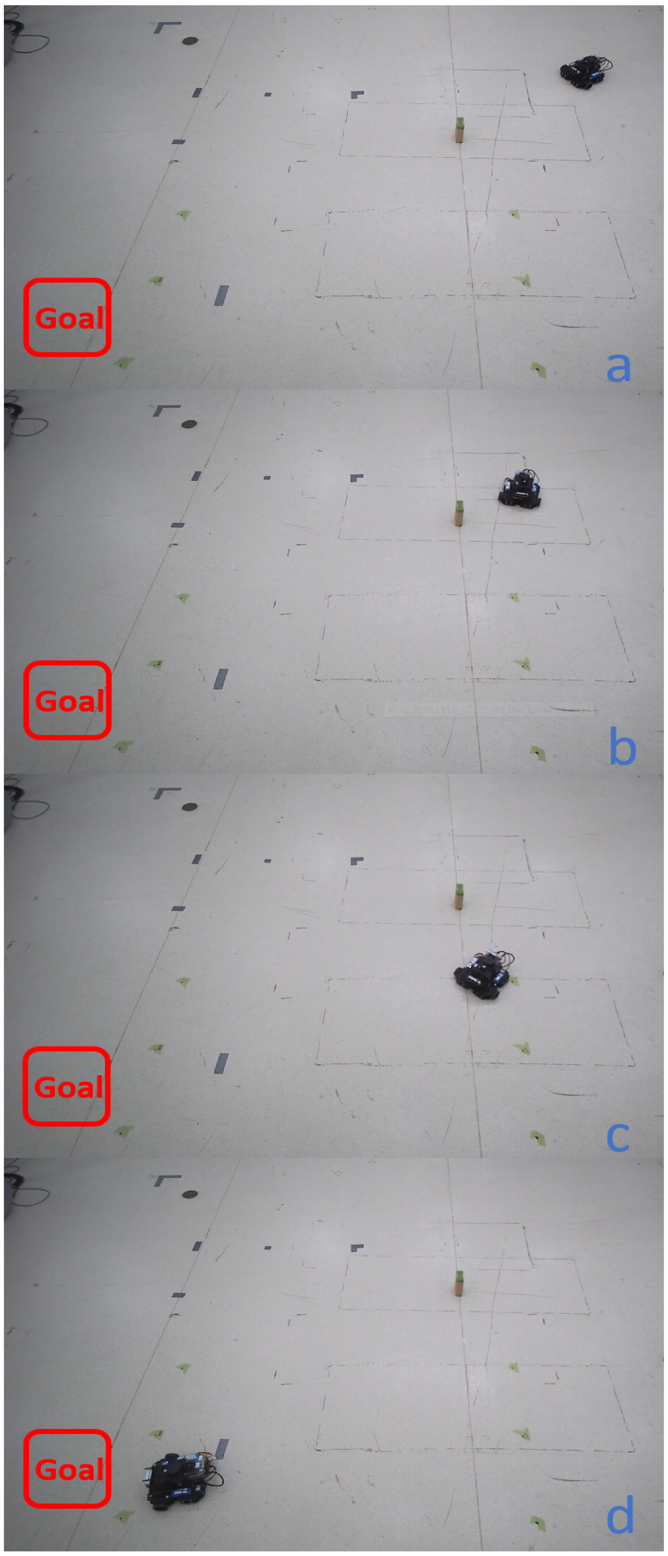


Figure 5.10. Snapshots of experimental task 3-scenario 2: (a) t=20s, (b) t=30s, (c) t=35s, (d) t=50s

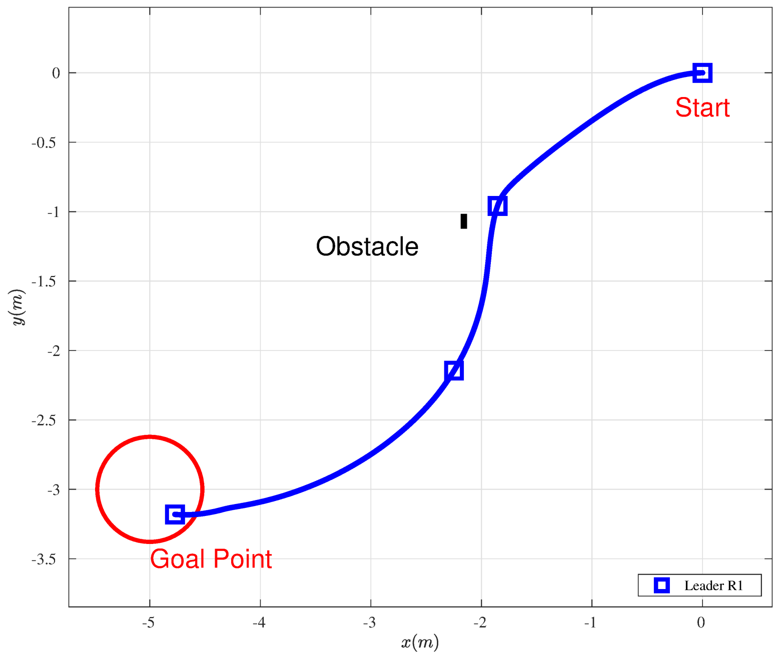


Figure 5.11. Experimental task 3-scenario 2: Low-profile obstacle detection and avoidance

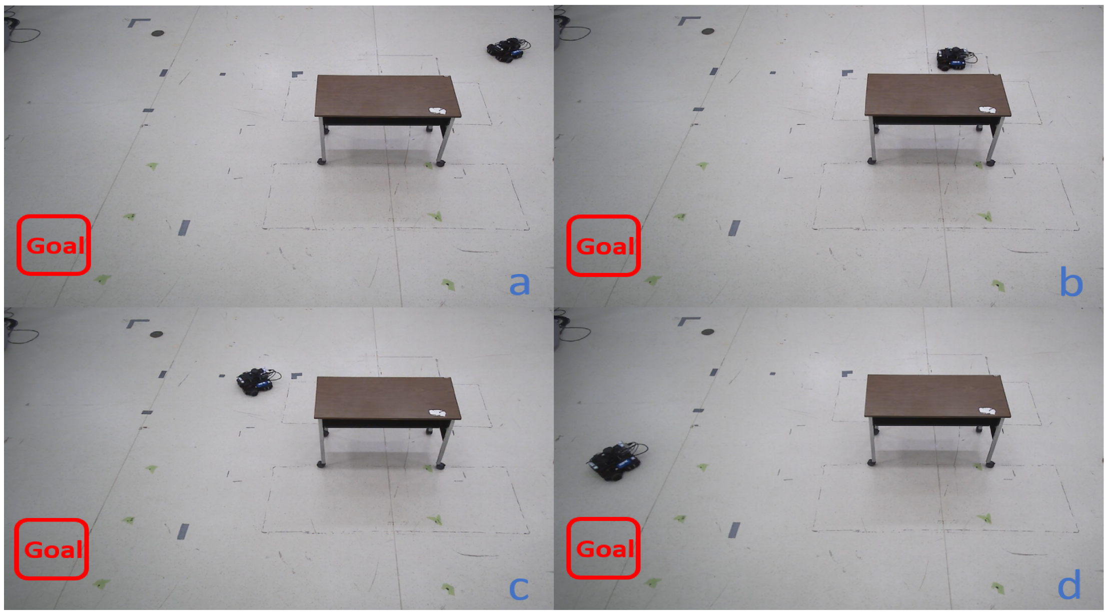


Figure 5.12. Snapshots of experimental task 3-scenario 3: (a) t=20s, (b) t=30s, (c) t=35s, (d) t=50s

The third scenario is to set an overhead obstacle such as a desk on the path of the robot. The snapshots of the experiment result are shown in Fig. 5.12.

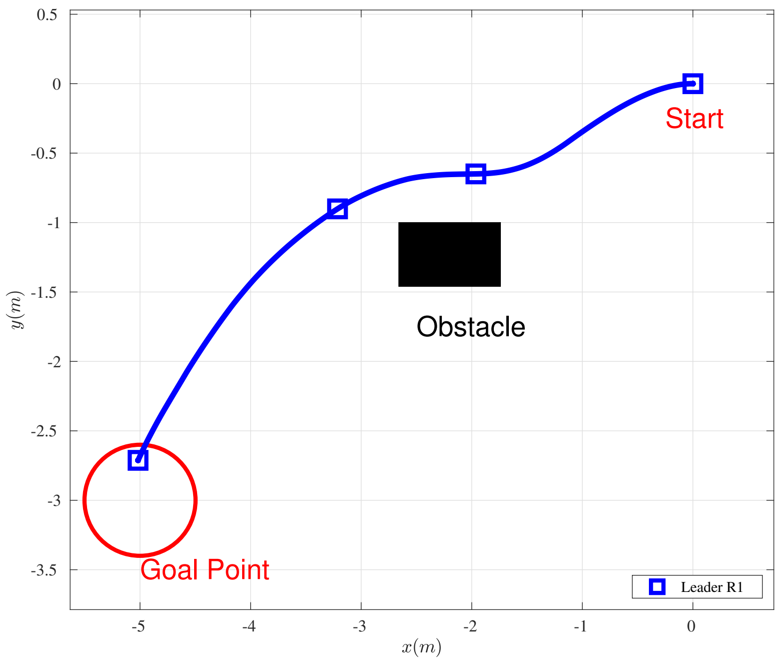


Figure 5.13. Experimental task 3-scenario 3: Overhead obstacle detection and avoidance

From Fig. 5.13, the robot is able to detect the desk overhead as an obstacle and avoid it using the proposed algorithm. Note the robot took a different path than Scenario 1 and Scenario 2, which shows the proposed obstacle avoidance algorithm is able to navigate through most environments.

5.4 TASK IV: Affine Formation Obstacle Avoidance

After both affine formation control and the depth camera obstacle detection experiments are successful, the last task is to deliver both functions for the MAS. In this task, Robots 1, 2, and 3 are applying the APF avoidance algorithm because they are playing the leader role although Robot 2 and Robot 3 only chasing the virtual leaders' position. Robot 4 is only under the affine formation control because of the lack of information and ability to sense the environment.

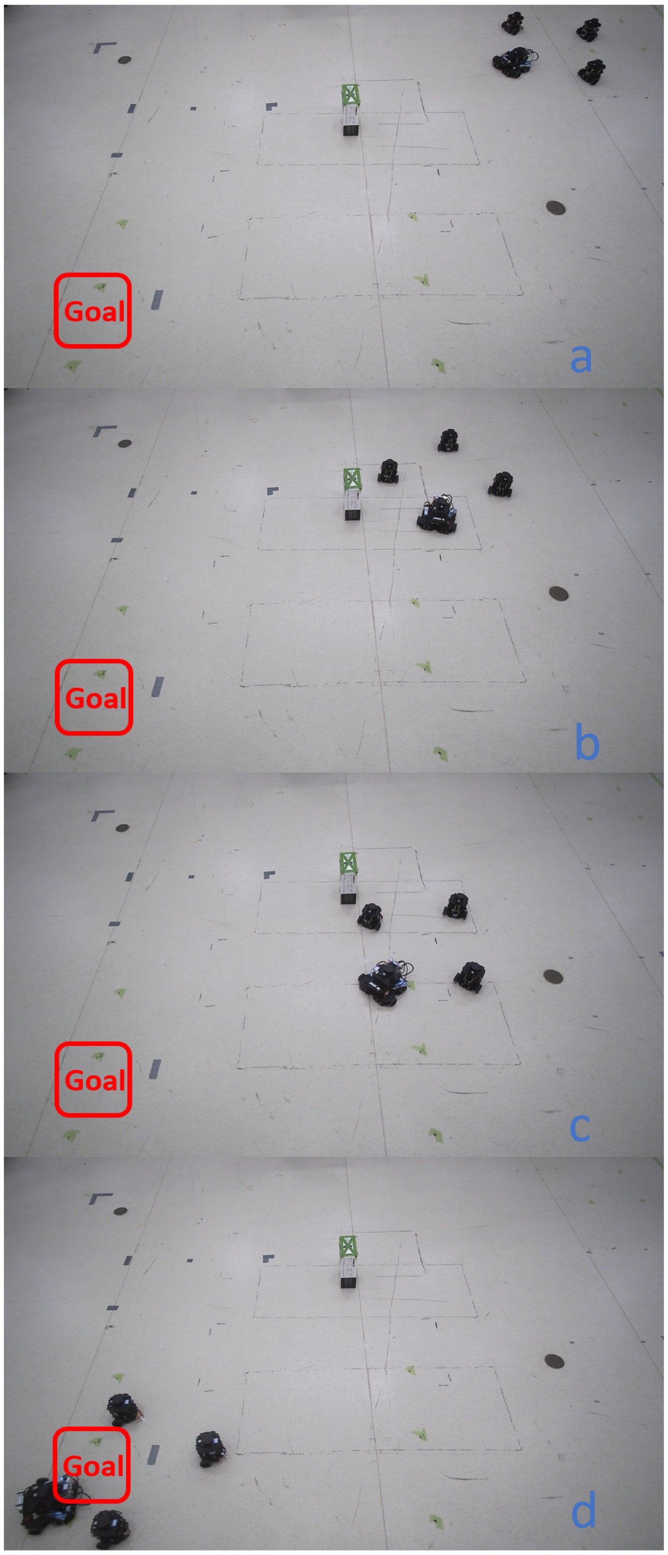


Figure 5.14. Snapshots of experimental task 4: (a) t=20s, (b) t=30s, (c) t=35s, (d) t=50s

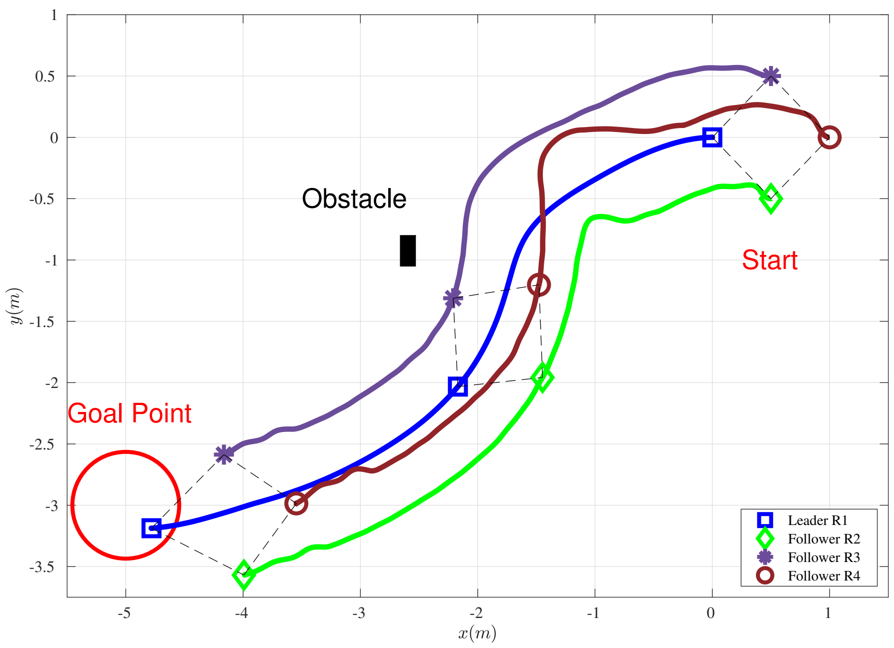


Figure 5.15. Experimental task 4: Obstacle avoidance with affine formation control

From Fig. 5.15, Robot 4 maintains the formation while other robots know how to avoid the obstacle detected by the depth camera. The formation is distorted at the end of the navigation because the leader detects the curtain on the edge of the testing area, applying a repulsive force from the APF algorithms and providing a pushing signal against the moving directions of Robot 2 and Robot 3.

5.5 Summary

In this chapter, the core experimental tasks focused on goal point navigation, maneuvering through confined paths, obstacle sensing and evasion, as well as maintaining precise formation using affine formation control techniques. The proposed algorithm proved its capability to accurately guide the robots to predetermined targets. In the challenging task of navigating through narrow gaps, the algorithm demonstrated adaptability by successfully adjusting the formation to pass through without collisions. This aspect of the experimentation highlighted the flexibility and spatial awareness integrated into the robot team.

When confronted with unexpected obstacles, the algorithm efficiently processed environmental data to dynamically steer the robots away from potential impacts, ensuring safe traversal and consistent formation integrity. This feature was particularly indicative of the robustness of the obstacle detection and avoidance mechanism programmed within the algorithm.

Additionally, the robots' performance in maintaining the diamond shape showed a high level of precision in position control, which is critical for operations that demand strict adherence to formation shapes.

In summary, the extensive experimental results not only validate the efficacy of the proposed algorithm in achieving the required tasks but also demonstrate its potential for practical usage in various real-world applications that necessitate autonomous coordinated robot teams.

Chapter 6

Conclusions and Future Research

This chapter provides a summary of the work presented in this thesis and proposes potential research areas that expand upon the concept of vision-aided for MASs formation control.

6.1 Conclusions

The first part of this thesis introduces the idea of the MAS formation problem and introduces the design and development of an advanced intelligent formation controller employing an affine formation controller. The improved formation controller is verified through a simulation environment and real-world experiments.

The second area of the study concentrates on the establishment of an innovative obstacle avoidance strategy using a depth camera. The strategy is centered on the application of affine formation shift and the APF algorithm. This enables the MAS to not just detect the obstacles in an unknown environment, but also make decisions based on the types of obstacles. The use of a depth camera overcomes the disadvantage of traditional range sensors, which provide more accurate data while maintaining a lower cost.

In conclusion, this thesis presented a practical and effective application for MAS navigation and formation control based on a depth camera. The proposed avoidance controller successfully guides robots to reach their goal point while navigating in an unknown environment. The vision-aided navigation with affine formation control can be extended to any type and number of mobile robots with one leader equipped with the depth camera, it is an advanced system showcasing its potential for enhancing the capabilities of MASs in various applications.

6.2 Future Work

In the future, different formation control methods are required for MASs to perform even more changeling tasks, such as scaling formation or rotating formation. More followers could be included showcasing the advantage of affine formation controller being centralized controlled. Motion capture systems could also be applied to perform an external calibration for the robots' states. Further parameters could be detailed and tuned to improve performances. Since the system applied a depth camera, other advanced image processes could be embedded in order to gain extra information, for example, using the You Only Look Once (YOLO) algorithm to detect certain objects and allow the leader to make mature formation strategies. Machine learning could also be used to help the decision-making progress whether online or offline obstacle detection. The objective of future research is to propose a novel method for MAS to complete challenging tasks in a dynamic environment with other moving mobile robots or moving humans.

Bibliography

[1] Robots. (2023) Design for robotic assemble. [Online]. Available: www.robots.com/articles/design-for-robotic-assembly

[2] J. Alonso-Mora, S. Baker, and D. Rus. (2023) Multi-robot formation control and object transport in dynamic environments via constrained optimization. [Online]. Available: https://www.youtube.com/watch?v=sDNqdEPA7pE

[3] K. Spencer. (2017) Choosing the best sensors for a mobile robot, part one. [Online]. Available: www.fierceelectronics.com/components/choosing-best-sensors-for-a-mobile-robot-part-one

[4] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, vol. 36, no. 7, pp. 1019–1047, 2018.

[5] R. C. Connor, R. A. Smolker, and A. F. Richards, "Two levels of alliance formation among male bottlenose dolphins (tursiops sp.)." *Proceedings of the National Academy of Sciences*, vol. 89, no. 3, pp. 987–990, 1992.

[6] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *Ieee Access*, vol. 8, pp. 191 617–191 643, 2020.

[7] A. Viguria, I. Maza, and A. Ollero, "Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions," *Advanced Robotics*, vol. 24, no. 1-2, pp. 1–23, 2010.

[8] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[9] J. R. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE transactions on robotics and automation*, vol. 19, no. 6, pp. 933–941, 2003.

[10] M. A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous robots*, vol. 4, pp. 387–403, 1997.

[11] W. Ren and Y. Cao, *Distributed coordination of multi-agent networks: emergent problems, models, and issues.* Springer Science & Business Media, 2010.

[12] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on control systems technology*, vol. 9, no. 6, pp. 777–790, 2001.

[13] D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen, "A survey of spacecraft formation flying guidance and control. part ii: control," in *Proceedings of the 2004 American control conference*, vol. 4. Ieee, 2004, pp. 2976–2985.

[14] S. Coogan and M. Arcak, "Scaling the size of a formation using relative position feedback," *Automatica*, vol. 48, no. 10, pp. 2677–2685, 2012.

[15] H. Huang, C. Yu, and Q. Wu, "Autonomous scale control of multiagent formations with only shape constraints," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 7, pp. 765–791, 2013.

[16] X. Dong, B. Yu, Z. Shi, and Y. Zhong, "Time-varying formation control for unmanned aerial vehicles: Theories and applications," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 340–348, 2014.

[17] Z. Huang, R. Bauer, and Y.-J. Pan, "Affine formation control of multiple quadcopters," in *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2022, pp. 1–5.

[18] R. Adderson and Y.-J. Pan, "Formation shaping control for multi-agent systems with obstacle avoidance and dynamic leader selection," in *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*. IEEE, 2022, pp. 1082–1087.

[19] J. Borenstein, Y. Koren *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[20] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1478–1483.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[23] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021.

[24] C. Zheng and R. Green, "Vision-based autonomous navigation in indoor environments," in *2010 25th International Conference of Image and Vision Computing New Zealand*. IEEE, 2010, pp. 1–7.

[25] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in *2006 4th student conference on research and development*. IEEE, 2006, pp. 183–188.

[26] N. B. Sariff and N. Buniyamin, "Ant colony system for robot path planning in global static environment," in *9th WSEAS International Conference on System Science and Simulation in Engineering (ICOSSSE'10)*, vol. 192, 2010.

[27] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[28] G. Klančar, A. Zdešar, S. Blažič, and I. Škrjanc, "Wheeled mobile robotics," *From Fundamentals Towards Autonomous Systems. Butterworth-Heinemann*, 2017.

[29] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[30] Y. Zhang, G. Tang, and L. Chen, "Improved A* algorithm for time-dependent vehicle routing problem," in *2012 International Conference on Computer Application and System Modeling*. Atlantis Press, 2012, pp. 1341–1344.

[31] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[32] T. Dudi, R. Singhal, and R. Kumar, "Shortest path evaluation with enhanced linear graph and dijkstra algorithm," in *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2020, pp. 451–456.

[33] A. Majeed and S. O. Hwang, "Path planning method for uavs based on constrained polygonal space and an extremely sparse waypoint graph," *Applied Sciences*, vol. 11, no. 12, p. 5340, 2021.

[34] S.-B. P. Lab. Sbpl. http://sbpl.net/.

[35] A. T. Le and T. D. Le, "Search-based planning and replanning in robotics and autonomous systems," *Advanced Path Planning for Mobile Entities*, pp. 63–89, 2018.

[36] O. A. Gbadamosi and D. R. Aremu, "Design of a modified dijkstra's algorithm for finding alternate routes for shortest-path problems with huge costs." in *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*. IEEE, 2020, pp. 1–6.

[37] G. Qing, Z. Zheng, and X. Yue, "Path-planning of automated guided vehicle based on improved dijkstra algorithm," in *2017 29th Chinese control and decision conference (CCDC)*. IEEE, 2017, pp. 7138–7143.

[38] S. A. Fadzli, S. I. Abdulkadir, M. Makhtar, and A. A. Jamal, "Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries," in *2015 2nd International Conference on Information Science and Security (ICISS)*. IEEE, 2015, pp. 1–4.

[39] H. I. Kang, B. Lee, and K. Kim, "Path planning algorithm using the particle swarm optimization and the improved dijkstra algorithm," in *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, vol. 2. IEEE, 2008, pp. 1002–1004.

[40] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, 2005, pp. 9–18.

[41] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *International Journal of Computer Science and Network Security*, vol. 11, no. 1, pp. 125–130, 2011.

[42] J. Yao, C. Lin, X. Xie, A. J. Wang, and C.-C. Hung, "Path planning for virtual human motion using improved A* star algorithm," in *2010 Seventh international conference on information technology: new generations*. IEEE, 2010, pp. 1154–1158.

[43] T. Chen, G. Zhang, X. Hu, and J. Xiao, "Unmanned aerial vehicle route planning method based on a star algorithm," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2018, pp. 1510–1514.

[44] L. Cheng, C. Liu, and B. Yan, "Improved hierarchical a-star algorithm for optimal parking path planning of the large parking lot," in *2014 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2014, pp. 695–698.

[45] S. Sedighi, D.-V. Nguyen, and K.-D. Kuhnert, "Guided hybrid a-star path planning algorithm for valet parking applications," in *2019 5th international conference on control, automation and robotics (ICCAR)*. IEEE, 2019, pp. 570–575.

[46] W. Yijing, L. Zhengxuan, Z. Zhiqiang, and L. Zheng, "Local path planning of autonomous vehicles based on A* algorithm with equal-step sampling," in *2018 37th Chinese Control Conference (CCC)*. IEEE, 2018, pp. 7828–7833.

[47] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[48] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent unmanned ground vehicles.* Springer, 1997, pp. 203–220.

[49] A. Stentz and I. C. Mellon, "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, no. 3, pp. 89–100, 1995.

[50] A. Stentz, *The D* algorithm for real-time planning of optimal traverses.* Carnegie Mellon University, the Robotics Institute, 1994.

[51] S. Koenig and M. Likhachev, "D* lite," *Aaai/iaai*, vol. 15, pp. 476–483, 2002.

[52] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Robotics research.* Springer, 2007, pp. 239–253.

[53] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, vol. 3. IEEE, 2002, pp. 3016–3023.

[54] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[55] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM).* IEEE, 2013, pp. 1–8.

[56] H. Burchardt and R. Salomon, "Implementation of path planning using genetic algorithms on mobile robots," in *2006 IEEE International Conference on Evolutionary Computation.* IEEE, 2006, pp. 1831–1836.

[57] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[58] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics.* Springer, 2008, vol. 200.

[59] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, "A survey on vision-based uav navigation," *Geo-spatial information science*, vol. 21, no. 1, pp. 21–32, 2018.

[60] Raspberry-Pi. (2023) Raspberry pi camera module 2. [Online]. Available: www.raspberrypi.com/products/camera-module-v2/

[61] A. Rocchi, Z. Wang, and Y.-J. Pan, "A practical vision-aided multi-robot autonomous navigation using convolutional neural network," in *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2023, pp. 1–6.

[62] T. Mao, K. Huang, X. Zeng, L. Ren, C. Wang, S. Li, M. Zhang, and Y. Chen, "Development of power transmission line defects diagnosis system for uav inspection based on binocular depth imaging technology," in *2019 2nd International Conference on Electrical Materials and Power Equipment (ICEMPE)*. IEEE, 2019, pp. 478–481.

[63] Raspberry-Pi. (2023) Raspberry pi imx219-83 stereo camera. [Online]. Available: https://www.waveshare.com/wiki/IMX219-83_Stereo_Camera

[64] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *2015 international conference on intelligent environments*. IEEE, 2015, pp. 140–143.

[65] S. Zhao and D. Zelazo, "Bearing rigidity and almost global bearing-only formation stabilization," *IEEE Transactions on Automatic Control*, vol. 61, no. 5, pp. 1255–1268, 2015.

[66] S. Zhao, "Affine formation maneuver control of multiagent systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4140–4155, 2018.

[67] ROBOTIS. (2023) Turtlebot3 e-manual. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

[68] Intel. (2023) Intel realsense depth camera d435. [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/

[69] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 1–10.

[70] NVIDIA. (2023) Jetson nano developer kit. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[71] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[72] D. Amanda. (2018) Ros introduction. [Online]. Available: http://wiki.ros.org/ROS/Introduction

[73] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[74] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots: Third International Conference, SIMPAR 2012, Tsukuba, Japan, November 5-8, 2012. Proceedings 3.* Springer, 2012, pp. 400–411.

[75] A. D. Luca and G. Oriolo, "Modelling and control of nonholonomic mechanical systems," in *Kinematics and dynamics of multi-body systems.* Springer, 1995, pp. 277–342.

[76] W. Ren and R. W. Beard, "Distributed formation control of multiple wheeled mobile robots with a virtual leader," *Distributed consensus in multi-vehicle cooperative control: Theory and Applications*, pp. 193–205, 2008.

[77] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles.* Springer, 1986, pp. 396–404.

[78] H. Safadi, "Local path planning using virtual potential field," *McGill University School of Computer Science, Tech. Rep*, 2007.

[79] S. A. Shahbaz and A. A. Anjana, "Autonomous navigation using partial artificial potential fields on differential drive turtlebot," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS).* IEEE, 2018, pp. 121–127.

[80] Y. Xu and D. Luo, "Dynamic affine formation control of multiple unicycle modelled agents," in *Advances in Guidance, Navigation and Control: Proceedings of 2020 International Conference on Guidance, Navigation and Control, ICGNC 2020, Tianjin, China, October 23–25, 2020.* Springer, 2022, pp. 1791–1803.

[81] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks.* Princeton University Press, 2010.

[82] R. A. Horn and C. R. Johnson, *Matrix analysis.* Cambridge university press, 2012.

[83] T.-T. Lu and S.-H. Shiou, "Inverses of 2× 2 block matrices," *Computers & Mathematics with Applications*, vol. 43, no. 1-2, pp. 119–129, 2002.

[84] MathWorks. (2023) Lmi solvers. [Online]. Available: https://www.mathworks.com/help/robust/lmis.html