# DYNAMIC OBSTACLE HANDLING IN MULTI-ROBOT COVERAGE

**TAY WING LE**

**UNIVERSITI TUNKU ABDUL RAHMAN**

# DYNAMIC OBSTACLE HANDLING IN MULTI-ROBOT COVERAGE

**TAY WING LE**

**A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Mechatronics Engineering with Honours**

**Lee Kong Chian Faculty of Engineering and Science**
**Universiti Tunku Abdul Rahman**

**April 2024**

# DECLARATION

| | | |
|---|---|---|
| Signature | : | [illegible] |
| Name | : | TAY WING LE |
| ID No. | : | 1902872 |
| Date | : | 29-4-2024 |

## APPROVAL FOR SUBMISSION

I certify that this project report entitled **"DYNAMIC OBSTACLE HANDLING IN MULTI-ROBOT COVERAGE"** was prepared by **TAY WING LE** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Mechatronics Engineering with Honours at Universiti Tunku Abdul Rahman.

Approved by,

| | | |
|---|---|---|
| Signature | : | [illegible] |
| Supervisor | : | Dr Shalini a/p Darmaraju |
| Date | : | 16 May 2024 |

| | | |
|---|---|---|
| Signature | : | [illegible] |
| Co-Supervisor | : | Kwan Ban Hoe |
| Date | : | 16 May 2024 |

## ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor and co-supervisor, Dr. Shalini A/P Darmaraju and Dr. Kwan Ban Hoe for their invaluable advice, guidance and his enormous patience throughout the development of the research. Their expertise and dedication have played a crucial role in guaranteeing the success of this project. It was a tremendous privilege and honour to have worked and studied under their guidance and support.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement, assistance and support which invaluable throughout the entire of research project. Their guidance and feedback have been particularly constructive, aiding in the refinement of my ideas and enhancing the quality of my work throughout this research journey. Once more, I extend my heartfelt thanks to all who contributed to the completion of this project.

## ABSTRACT

The field of robotics has witnessed a growing interest in multi-robot system for coverage task such as static coverage, which presenting a new set of challenges to table. One of the key aspects in multi-robot coverage is dynamic obstacle handling, which involves robots avoiding dynamic obstacle during coverage task while reaching their respective goal positions. This report focuses on developing a multi-robot coverage algorithm with dynamic obstacle handling abilities, known as the Modified Lloyd's algorithm with Velocity Obstacle (VO). This modified algorithm introduces the Velocity Obstacle which calculates avoidance velocities for robots to avoid dynamic obstacle. Unlike the existing Lloyd's algorithm which primarily focuses on achieving complete coverage, the Modified Lloyd's algorithm with VO can determine the time to collision based on the relative distances and velocities of robots and dynamic obstacle. It then generates a new avoidance velocity for robots, enabling them to avoid dynamic obstacles while achieving complete coverage. The simulations were conducted by using MATLAB to demonstrate the superiority of Modified Lloyd's algorithm with VO over existing Lloyd's algorithm in terms of average of total number of collisions between robots and dynamic obstacle during coverage task. All five robots successfully avoided dynamic obstacle during coverage tasks, achieving complete coverage with zero collisions. The Modified Lloyd's algorithm was also tested under two different scenarios to evaluate its functionality. In first scenario, as the aggressiveness of dynamic obstacle increased, the algorithm remained capable of handling them, although there were occasional collisions at very high aggression levels. In the second scenario, where the starting position of dynamic obstacle varied, the modified algorithm consistently handled dynamic obstacle and achieved zero collisions during coverage tasks. The impacts of the modified algorithm's parameters on simulation results were also studied to determine the optimal parameters for achieving better performance. It was found that with number of 500 iterations and a safety margin of 5, the algorithm provided better performance in terms of shorter execution time and lower average number of collisions.

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF SYMBOLS / ABBREVIATIONS

| | |
|---|---|
| UAV | Aerial Robot |
| UGV | Ground Robot |
| VO | Velocity Obstacle |
| MPC | Model Predictive Control |
| CDRM | Constant-distance random movement |
| CVT | Centroid Voronoi Tessellation |
| CCVT | Constrained centroidal Voronoi tessellation |
| OVAC | Obstacle -Aware Voronoi Cells |
| DFP | Density Function Path |
| STC | Spanning Tree-Based Algorithm |
| DFS | Depth-First Search |
| PID | Proportional-Integral-Derivative |
| LQR | Linear–quadratic regulator |
| IAPF | Improved Artificial Potential Field |
| HRVO | Hybrid Reciprocal Velocity Obstacle |
| ORCA | Optimal Reciprocal Collision Avoidance |

| | |
|---|---|
| $p_N$ | nth agent's position in D |
| μ | Mean $\in \mathbb{R}^k$ |
| det (Σ) | Determinant of the covariance matrix Σ |
| x | Data and x $\in \mathbb{R}^k$ |
| K | Maximum number of iterations |
| K | Current iteration |
| $V_i(x)$ | Voronoi cell |
| $\varphi(q)$ | Density function |
| $C\ (V_i\ (x)$ | Centroid of Voronoi cell |
| $u_o$ = | Forward speed of obstacle |
| $\psi_o(t), r_o(t)$ | Heading and heading rate |
| $p_o$(t) | The position of the robot in plane |
| $v_o$(t) | The velocity of robot in plane |

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 General Introduction

Fourth industrial revolution, Industry 4.0 had ushered in an era of extreme digital transformation on a global scale (Ghobakhloo, 2020). The genesis of this transformation can be traced back to Industry 1.0, which emerged in the late 18th century, and was marked by the utilization of water and steam power to drive mechanized production facilities. Industry 2.0, which arose in the early 20th century, utilized electrical power, catalysing rapid advancements in industrial sectors during that era. Industry 3.0, taking shape in the early 1970s, introduced the fields of electronics and information technology, thereby paving the way for automation in production and manufacturing processes (Sherwani,Asad and Ibrahim, 2020). In the modern landscape, Industry 4.0 is the prevailing paradigm, witnessing the advancement of diverse robotic technologies designed to streamline and enhance various aspects of human life across numerous fields. Although automation was introduced during industry 3.0, it only underwent development during industry 4.0 (Karabegovic, 2017).

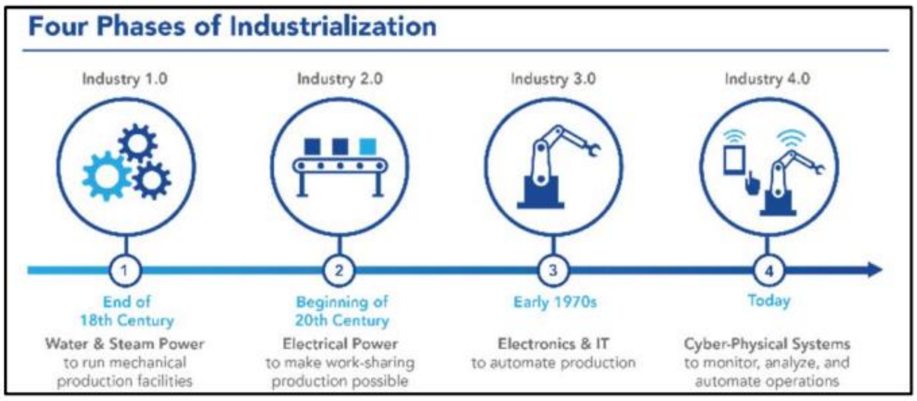

Four Phases of Industrialization
Industry 1.0
Industry 2.0
Industry 3.0
Industry 4.0
1
2
3
4
End of 18th Century
Beginning of 20th Century
Early 1970s
Today
Water & Steam Power to run mechanical production facilities
Electrical Power to make work-sharing production possible
Electronics & IT to automate production
Cyber-Physical Systems to monitor, analyze, and automate operations

Figure 1.1: Four Phases of Industrialization (Sherwani,Asad and Ibrahim, 2020).

Industry 4.0 had brought a swift and remarkable progression in development of robotics. There are some well-known robots used in industry 4.0 which are industrial robot and collaborative robots. The use of these robots instead of humans, especially in agriculture and search and rescue field, brings several benefits. It helps save costs, conserves energy, reduces human mistakes, and allows for continuous work around the clock. Robots can take over risky jobs like pesticide spreading, ensuring worker safety and speeding up work. This also reduces the chances of worker's injuries from hazardous tasks. Industrial robot and collaborative robots are both experience same similarities which are programmable and self-controlled device which consist of mechanical, electric, electronic and hydraulic units to perform a specific complex action. However, industrial robot is huge, large and heavy and used to employed for tasks that involve significant risks, therefore industrial robot usually isolated from human-operated areas and usually only utilized within expansive warehouses and sizeable manufacturing facilities. Collaborative robot or known as co-bot are usually lighter and provides higher mobility and flexibility (Sherwani,Asad and Ibrahim, 2020). Mobile robot, a type of collaborative robot is mostly used in various field because the tasks involve coverage task with dynamic obstacle which require high mobility of robot.

### 1.1.1 Application of Robots

In modern times, a diverse array of fields has incorporated the utilization of robotics, and agriculture stands prominently among them. Agriculture is holding important significance in human society because it provides humans source of food and maintain society activities. The integration of robots into the search and rescue field also has gained importance due to the imperative nature of this field. In search and rescue field, there is zero tolerance for human errors and necessitating a continuous and nonstop work effort. Hence, the implementation of automation becomes imperative as it holds the potential to enhance productivity and operational efficiency. Presently, numerous countries, including Liberia and Niger, suffering with food shortages due to poor and inadequacies in agricultural management. Certain countries have also faced shortages problem in farm labour, causing a direct impact on agricultural production. Agriculture become the most vital providers of occupations and had played an important role in fight against hunger and poverty (Sud et al., 2015). Insufficient consumption of food will lead to an inactive and unhealthy life (Mcguire, 2015).

Traditional agriculture method mainly relies of human labour, and this is a factor that can lead to decrease the productivity of agriculture. Human labourers are constrained by the need for regular daily rest, which can leave crops vulnerable to various risks. These include potential damage from animals like birds. Human also cannot consistently maintain to monitor health status of each individual crop to ensure their well-being and protect them against diseases. Spreading pesticide also is a dangerous and hazardous task for human because accidental inhalation of toxic particles can lead to immediate illness and potentially yield adverse long-term effects (Kim and Son, 2020). Therefore, robotics technology has been implemented in agriculture field as a solution to these issues.

### 1.1.2 Type of Robots

Numerous categories of robots find application across a wide spectrum of industries and domains. Robots come in diverse forms, ranging from substantial and robust robots designed for heavy-duty tasks, such as lifting heavy objects, to more lightweight and agile robots which ideal for tasks like transportation. Unambiguously, for coverage tasks involving dynamic obstacle, mobile robots emerge as the suitable choice. The advancement of sensors and technologies, including machine vision and artificial intelligence, has undergone significant development which lead to robotics had been successfully implemented in many fields and one of it is agricultural task which enable to reduce workload and increase productivity of agriculture (Moorehead et al., 2010). Mobile robots have gained widespread adoption in the various sector due to their higher mobility and flexibility, interpreting them particularly well-suited for the dynamic working environments. In contrast to humans, mobile robots exhibit the capability to consistently execute tasks as well as minimizing errors. This capacity of mobile robot contributes to increasing productivity and efficiencies. Additionally, mobile robots can used in undertaking hazardous tasks, to prevent potential human injuries. Therefore, implementation robot in working environment can grant solutions to difficult problems, increase process productivity and lesser work load (Vasconez,Kantor and Cheein, 2019).

Currently, there are a lot of type of mobile robot has been deployed in coverage task to tailor for different working environment and tasks such as aerial, ground based and aquatic configurations. Unmanned group vehicles (UGV) can be utilized for water distribution and continuous monitoring of individual victims. Through repeated coverage of the same work areas, UGVs can ensure consistent and effective irrigation. Meanwhile, unmanned aerial vehicle (UAV) plays a pivotal role in enhancing security. By repeatedly coverage designated regions, UAVs can prevent unauthorized individuals and safeguarding the working space.

Nonetheless, deploying a single robot in industrial or agriculture field is a high-cost application. This is because designing a singular robot capable of executing an extensive array of tasks, the single robot needs to invent and involve a lot of complex application and development. Furthermore, any operational failures encountered by the single robot, it will cause the process force to stop. To address these limitations and drawbacks of single mobile robot, implementation of multi-robot systems can solve these issues.

### 1.1.3 Multi-robot Systems

In recent years, multi-robot systems have gained substantial attention. This multi-robot systems involve the utilization of multiple robots to collectively address complex and large-scale problems. Communication of multi-robot systems plays a vital role in coordinating the multi-robots. Integrating enhanced communication capabilities into these systems contributes to expedited responsiveness among the robots. There are a lots of factors why multi-robot systems become famous due to their robustness and effectiveness. By employing a multi-robot approach, coverage task can be significantly accelerated and reducing the coverage time. Moreover, the multi-robot configuration facilitates repetitive coverage of the same area within a condensed frame of time. Multirobot systems have excellent advantages compare to single robot in the aspects of resilience and versatility. For instance, a group of multi-robots, each equipped with various types of actuators, sensors and communication devices, can work together to collectively monitor their surroundings (Liu et al., 2023). In pesticide spreading task, large area of agriculture can be divided into few small pieces by using Voronoi diagram and the multi-robot can be deployed to the spraying task through communication to improve performance (Kim and Son, 2020). Sizable sprayers for individual single robot now can be reduced into more lightweight counterparts. This advancement not only yields cost savings but also enhances operational efficiency.

### 1.1.4 Coverage Task with Dynamic Obstacle

The tasks of mobile robots in various field are mostly involved in coverage task. For instance, mobile robots in agriculture field, their tasks such as water distribution, pesticide dispersal, and health status monitoring necessitate the robot's repetitive coverage of specified areas over extended periods. This requirement arises from the robots for uninterrupted and continuous operation to ensure optimal performance efficiency. The working environment can be segmented into different sections, and multi-robot system is strategically deployed to allocate individual robots to cover each designated area. This collaborative approach involves continuous communication among the robots, resulting in improved productivity and operational efficiency. The coverage task that addressed in this project will be static coverage. Static coverage typically refers to the coverage achieved by stationary robots or agents in a given area over a period of time. It also widely uses in various filed such as surveillance, environmental monitoring and so on. In static coverage, the robots will move to a position and remain stationary position throughout the coverage period. The primary objective of static coverage is to ensure that every point within the designated area of interest is covered or monitored by at least one robot or agent. The static coverage can be uniform, where robots are evenly distributed across the coverage area, or it could be optimized based on specific criteria such as maximizing observation efficiency. It implies continuous observation over time with robots or agents remain in fixed positions to monitor the environment or collecting data without movement.

Nevertheless, within the robots working environment, obstacles pose typical challenges. These obstacles can be categorized into two primary types: static obstacles and dynamic obstacles. Static obstacles are defined as obstacles that remain stationary over time, retaining their fixed positions. In response to address these static obstacles, multi-robots are capable of initially identifying their locations during their initial coverage phase. Subsequently, this positional information is communicated to other robots within the same multi-robot system, enabling coordinated avoidance of these static obstacles. Dynamic obstacles are characterized by their continuous motion within the designated area. This movement of dynamic obstacles can be either predictable or random in nature. For instance, the predictable movement in dynamic obstacles, consider a multi-robot system comprising five robots. From the perspective of Robot 1, Robots 2, 3, 4, and 5 are considered dynamic obstacles. These robots' path and coverage are predefined and result in predictable movement. Consequently, the robots must possess the capability to continuously detect the dynamic obstacle as they navigate and cover the area over time and, when necessary, adjust their paths to avoid collisions.

## 1.2 Importance of the Study

Technological advancements have progressed rapidly nowadays, leading to the widespread integration of automation in various field. Automation serves as an effective solution for mitigating human errors, enhancing operational efficiency, increasing productivity, and addressing a numerous of challenges. In agriculture, many nations are currently suffering with dual challenges which are a shortage of food supply and a shortage of labour. In search and rescue field, mobile robot can use to help rescuer to help the victim by keep covering a designated region. Hence, it becomes of importance to prioritize the advance of dynamic obstacle handling in multi-robot coverage as a means to mitigate these critical issues.

Aerial robots (UAV), ground robots (UGV) have been vigorously introduced for automation (Zhang and Kovacs, 2012). The employment of these automated robots, which can address optimize long-term cost and labour shortages (Pedersen et al., 2006). UAVs can use to protection over large areas and inspection rapidly. Hence, application such as monitoring health status involve a lot of UAVs. Besides, UGVs in agriculture are applied for replacing traditional tractors which by combine transplanters and harvesters into a multipurpose UGV system  for transplanting, seeding and mapping of various crops (Ju et al., 2022). The multi-robot systems are playing an important role in various field; therefore, it is a need to handling dynamic obstacle in multi-robot coverage to improve its efficiency and productivity to address the issues encountered in task.

## 1.3 Problem Statement

The prevailing shortages in labour and supply have given rise to a variety of challenges. Consequently, the implementation of robotic automation emerges as a viable and necessary solution to address these pressing issues. However, single robot is not enough to fulfil the work requirement, therefore multirobot is used instead of single robot. Although single robot is low in cost and high scalability, but it is less used in working field due to its short communication range and small processing capacity. In the other hand, multirobot is more robust and higher efficiency compared to single robot but with higher cost. Multirobot also limited due to restricted reliability for computational purposes and understandability. The adoption of multi-robot systems brings forwards a variety of advantages. Nevertheless, the implementation process is remarkably intricate due to several complexities. These complexities include the establishment of communication networks among multi-robot systems, complex decision-making processes, the recognition and navigation around static and dynamic obstacles and the precise determination of coverage areas for each individual robot.

The initial challenge that arises pertains to the partitioning of the working environment into different sections. This imposes a precise allocation of each robot in their designated position and the corresponding area to be covered within that section. The process of subdividing a large working environment into smaller and manageable sections serves to increase operational efficiency. Moreover, it simplifies the deployment of multi-robot systems in coverage task. Following the distribution of specific working environment for each robot, an algorithm becomes essential for guiding these robots from their initial positions or origins, towards a common goal. In this research, the goal is typically represented by the centroid of the designated coverage area. These robots will engage in repeated coverage of their assigned areas until a stop command is issued. During the coverage process, a significant challenge lies in enabling the robots to accurately differentiate the static and dynamic obstacles. Failure to do so can lead to collisions between the robots and these obstacles and resulting in potential losses and disruptions. In real-world working environments, the presence of numerous obstacles necessitates a crucial consideration. Therefore, the implementation of obstacle avoidance mechanisms is important when introducing robots into coverage task. The efficiency of deployment of the multi robots for coverage task in dynamic environments with dynamic obstacle is a big challenge nowadays.

## 1.4 Aim and Objectives

Aim of this project is to develop a robust and efficient model for multi-robot systems to accomplish coverage tasks in environments with dynamic obstacle.

The specific objectives of this project are listed below:

1. To conduct literature review on existing algorithms for coverage tasks in multi-robot systems, specifically focusing on dynamic obstacle handling.
2. To develop a coverage control algorithm with dynamic obstacle avoidance technique.
3. To evaluate the performance of proposed algorithm through MATLAB simulation.

## 1.5 Scope and Limitation of the Study

The development of multi-robot systems for comprehensive area coverage presents a difficult and complicated effort. This project’s scope will be focusing on developing a multi-robot system model to cover a given area with dynamic obstacle avoidance ability. The multi-robot coverage task will be simulated by using MATLAB.

The study is subject to certain limitations which are lacking available research sources on coverage control with dynamic obstacle avoidance in current literature. Consequently, the development of an algorithm has been rendered a challenging task. In order to simply the task and simulation, there has been a cautious reduction in the computational complexity of the multi-robot system. Instead of utilization of heterogeneous systems involving various types of robots, this project will focus on homogeneous systems which comprising solely same type of robots. Additionally, in order to simplify the simulation process, aspects relating to communication and collaboration among the multi-robots have been omitted from consideration.

## 1.6 Contribution of the Study

This project outlines the examination and comparison on the existing coverage control algorithm and dynamic obstacle avoidance technique for multi-robot coverage task with dynamic obstacle. The existing algorithm are evaluated, and a better algorithm will be selected and further developed with the aims to generate an algorithm that can handling and avoid dynamic obstacle in multi-robot coverage. The proposed algorithm was modified and fine-tuned to enhance the final simulation results. The results obtained from the MATLAB simulations will demonstrate that proposed algorithm successfully achieves this project aims and objectives.

## 1.7 Outline of the Report

This report comprises five chapters which are Chapter 1, Chapter 2, Chapter 3, Chapter 4 and Chapter 5. In Chapter 1, the introduction of whole project is discussed. There are several subsections included in Chapter 1 which are general introduction of robots, the importance of study, problem statement, aims and objectives, scopes and limitations of the study, contribution of the study and the outline of the report of this project.

Chapter 2 delves into the literature review of this project. It examines the existing multi-robot coverage control algorithm such as broadcast control, Lloyd’s algorithm and spanning tree-based methods. It also evaluates the existing dynamic obstacle avoidance techniques such as model predictive control, artificial potential field and velocity obstacle. Additionally, the Gaussian distribution for optimizing criteria is also discussed.

Chapter 3 outlines the methodology and work plan for the whole project. The methodology of the proposed Lloyd’s algorithm with velocity obstacle avoidance techniques is explained step by step including necessary information and equations. The mathematical formula of the proposed algorithm is also discussed in this chapter for a deeper understanding. Besides that, Gantt charts for planning of this project part 1 and part 2 are also attached in this chapter.

Chapter 4 presents the results and discussions. The results generated by MATLAB simulations with the proposed algorithm and existing algorithm are evaluated and compared in different scenarios to determine the superior algorithm among the two algorithms. The performance for the proposed algorithm in different scenarios is also compared to determine the suitable parameters in the simulation. Additionally, the impact of proposed algorithm parameters on the simulation results is also included in this chapter.

Chapter 5 discusses the conclusion of the whole report. In this chapter, a short conclusion is constructed, and the limitations of the whole project are presented. The recommendations for future works are included throughout this chapter.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

Handling dynamic obstacle in multi-robot coverage plays a pivotal role in advancing the utilization of multi-robot system across various fields such as industrial field, agriculture field and so on. This handling enables multi-robot system to efficiently cover designated areas over time while mitigating the risk of obstacle collisions. In working environment, the presence of obstacles especially dynamic obstacles is an inescapable reality. Therefore, it is crucial to implement a dynamic obstacle avoidance technique into the coverage algorithm. This augmentation empowers multiple robots to adeptly avoid obstacle to ensure uninterrupted operation in coverage environment.

Numerous coverage control algorithms for multi-robot systems to deploy a group of robots to cover a designated area or region are available. The overall coverage control algorithms can be categorized into six general groups, each with its unique approach. They are grid-based approaches, decentralized approaches, frontier-based approaches, sampling-based approaches, centralized approaches and learning-based approaches. The notably prominent method that used in coverage of multirobot is grid-based approaches which utilizes of cell decomposition and Voronoi based approaches. Cell decomposition involves in division of the workspace into grid cells with each robot will be assigned to cover specific cells then assign each robot to cover specific cell. In the other hand, Voronoi-based approaches is dividing the workspace into Voronoi regions based on position of multi-robots. In the decentralized approaches, swarm intelligence draws inspiration by natural swarm behaviours to enable multi-robots to collaborate collectively in a covering a designated area. Frontier exploration, an example of the frontier-based approaches which robots will explore the frontier areas within their workspace to maximize the coverage efficiency. Sampling-based approach, one of famous method from randomized algorithms. Randomized algorithms are utilizing robot in random movement to explore and cover an area. Reinforcement learning, a novel coverage algorithm that based on learning-based approaches. Reinforcement learning enables to train the robot through reinforcement learning to learn the coverage patterns to adapt the changing environment. Centralized coordination, an algorithm falling under centralized approaches, employs a central controller to allocates coverage task for each robot and regulate their movements to achieve maximum of coverage. These diverse coverage control algorithms offer a wide range of strategies to address specific coverage problem across various applications and environments.

In addition to coverage algorithms, the task of prevent multi-robot systems collide with dynamic obstacle during coverage task necessitate the incorporation of obstacle avoidance techniques. Several dynamic obstacles techniques are employed for this purpose which including velocity obstacles method (VO), model predictive control (MPC) and artificial potential field-based methods (APF). These techniques are seamlessly integrated with coverage control algorithms to construct a comprehensive model for multi-robot systems to be employed in coverage tasks within dynamic obstacle-rich environments.

Numerous researchers have made significant contributions to this field by introducing a numerous coverage control algorithms and dynamic obstacle avoidance techniques for multi-robot systems. In this chapter, different main types of algorithms will be discussed, evaluated and compared.

## 2.2 Multi-robot Coverage Control Algorithm

As previously mentioned, a diverse array of coverage control algorithms and dynamic obstacle avoidance techniques are available for multi-robot systems. In the following subsections of this chapter, commonly used algorithms will be researched, discussed, evaluated and compared to facilitate an informed selection.

### 2.2.1 Broadcast Control

The concept of broadcast control was initially introduced or proposed by Ueda, Odhner and Asada (Ueda,Odhner and Asada, 2007), and initially it had been applied to the context of a biosystem in lactose regulation system (Julius et al., 2008). The first implementation of broadcast control in multi-agent systems was occurred in 2009, introduced by Das and Ghose (Das and Ghose, 2009). Broadcast control refers to the management of multi-robot systems by sending identical and uniform communication signals to all the robots via a centre controller or decision-making unit. Broadcast control is mainly used in addressing problems about motion-coordination for robots to complete a coverage task. Broadcast control is one- to- all broadcast communication instead of agent-to-agent communication. According to Figure 2.1, it is shown that distinction between one-to-all broadcast communication and agent-to-agent communication lies in the manner and direction of information that transmitted during the task. In one -to-all broadcast communication, the multi-robot may not necessitate additional devices and energy for information transmission. This efficiency translated to cost savings and represents a substantial hardware advantage (Azuma,Yoshimura and Sugie, 2013). Although there is advantage in method of information transmission, but this method also introduces a challenge. When a signal is broadcasted from the central control unit to an agent or robot, it can inadvertently affect neighbouring agents, potentially leading to unintended consequences and undesirable situations.

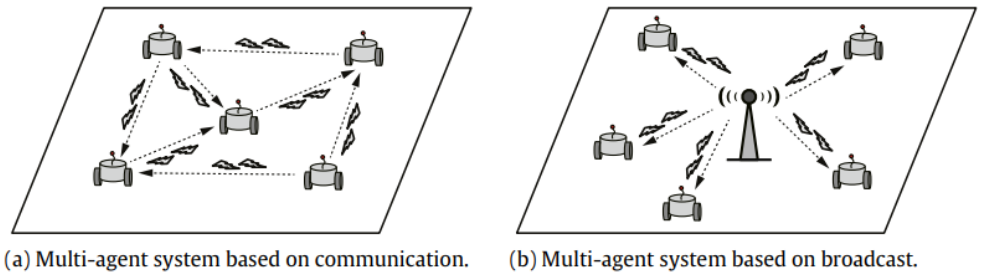

(a) Multi-agent system based on communication.
(b) Multi-agent system based on broadcast.

Figure 2.1: Types of Information Transmission between Multi-agent (Azuma, Yoshimura and Sugie, 2013).

In paper of Azuma, Yoshimura and Sugie in 2013, they proposed a broadcast control which composed of multi-agents, local controller and global controller. The role of local controller is to determine local actions of multi-agents while function of global controller is to observce performance of system and broadcasts a signal to control global behaviourIn the context of solving the coverage task, the local controller directs agents to alternate between random walk and deterministic walk strategies. Meanwhile, the global controller's role is to broadcast the progress or degree of achievement in the coverage task to all agents (Azuma, Yoshimura and Sugie, 2013). The necessatity of randomness in local controller plays an important role because it will affect the effectiveness of broadcast control. In this paper, the researchers conduct an experiment with several mobile robots to evaluate the performance of broadcast control.

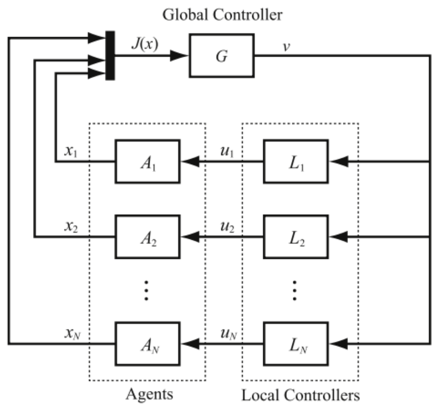


Figure 2.2: Broadcast Control with Local and Global Controller (Azuma, Yoshimura and Sugie, 2013).

The authors handled the experiment in systematic manner, with the first step author overseeing the coverage and local controller aspects. There are four steps which are first step is coverage. Coverage means strategically positioning multi-agent or multi-robot within the environment to ensure occupied coverage area are equal size. In this paper, authors were using objective function and technique of Voronoi diagram. In the second step, the experiment determined the precise locations for each agent by facilitating their convergence at specific rendezvous points. The third step focused on the assignment of individual agent to each designated area without overlap to ensure the efficient coverage distribution. To promote scalability, the last step was assuming local controllers are same for scalability. The consistency in local controllers is essential for large-scale multi-agent systems, as non-identical controllers can pose significant challenges. These local controllers are responsible for directing agents in both random and deterministic movements.

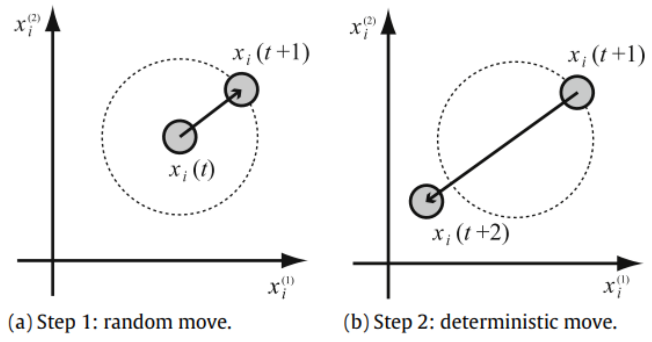


Figure 2.3: Motion of Agents by Local Controller (Azuma, Yoshimura and Sugie, 2013).

Following the configuration of parameters, both numerical simulations and physical experiments were conducted. In Figure 2.4, the circles within the illustration represent the agents, while the lines presented the boundaries of the Voronoi cells assigned to each agent. These simulation results clearly demonstrate the successful achievement of coverage through the proposed broadcast controller. The time evolution of objective function also shown in Figure 2.5, providing further insights into the effectiveness of the approach. After numerical simulation, authors also carried out the experiment in physical mobile robot as depicted in Figure 2.6 with corresponding results was shown in Figure 2.7.

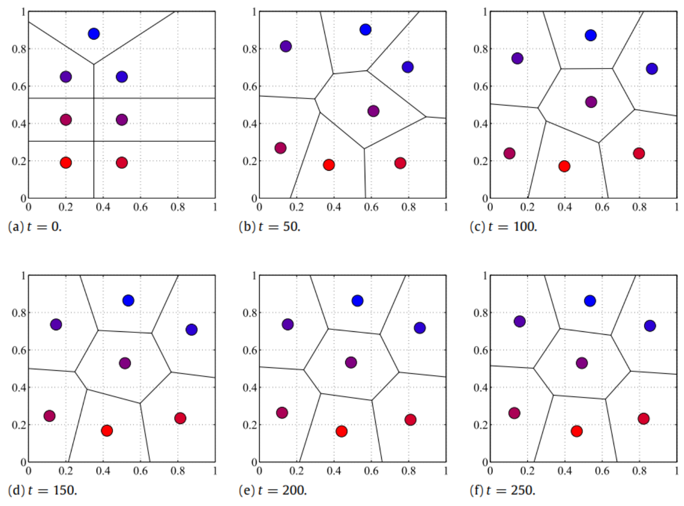


Figure 2.4: Snapshot of Group Position of Multi-agent (Azuma, Yoshimura and Sugie, 2013).

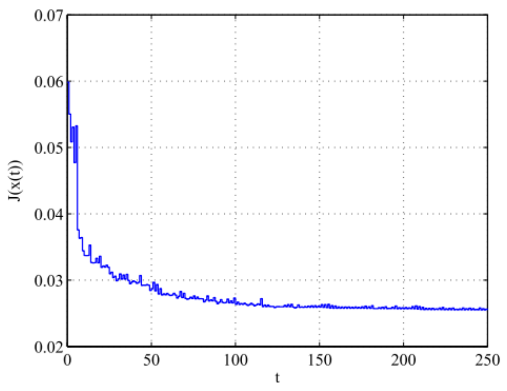


Figure 2.5: Time Evolution of Objective Function (Azuma, Yoshimura and Sugie, 2013).

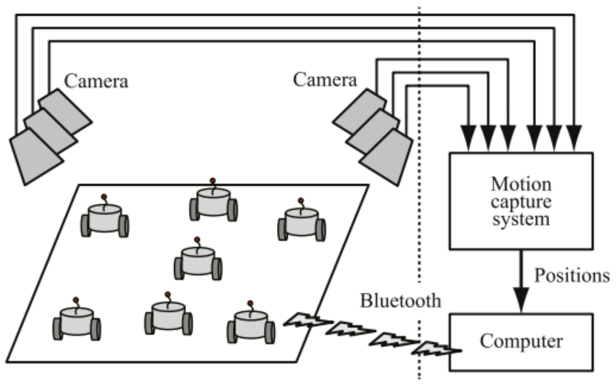


Figure 2.6: Physical Experiment Setup (Azuma, Yoshimura and Sugie, 2013).

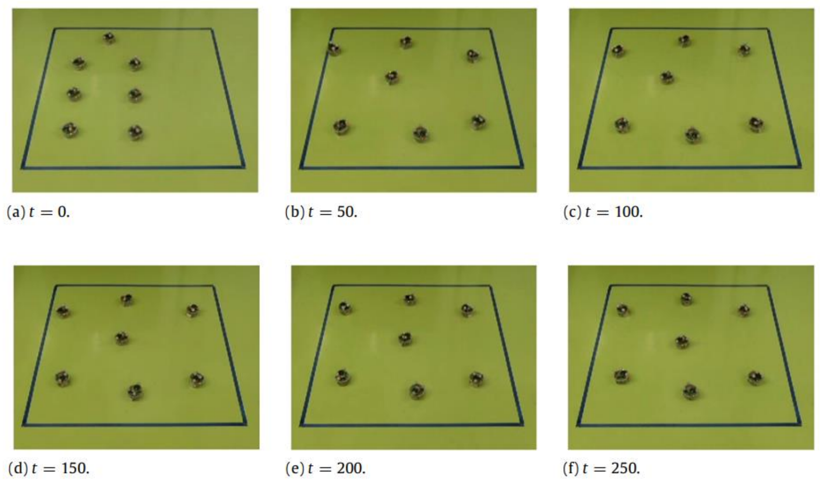


Figure 2.7: Snapshot of Group Position of Multi-agent in Physical Experiment (Azuma, Yoshimura and Sugie, 2013).

Nonetheless, standard broadcast control is susceptible to stability issues in specific motion-coordination tasks related to coverage task. To address this challenge, a modified broadcast control is propesed by Mohamad NorIsmail and Ahmad in 2020. In this research endevor, authors modified standard broadcast control. This modification entails constraining the magnitude of the update vector for the positions of robots to a constant value. Authors also conducted a numerical simulation and performed a comparative analysis involving three methods: standard broadcast control, the modified broadcast control, and constant-distance random movement (CDRM) (Mohamad Nor,Ismail and Ahmad, 2020). The results of these numerical simulations provided compelling evidence that the modified broadcast control method achieved convergence with a high degree of success.

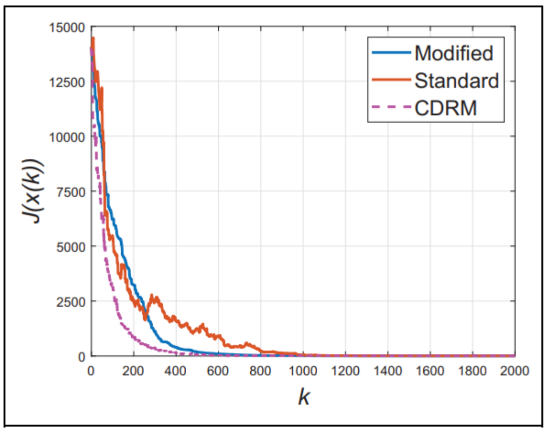
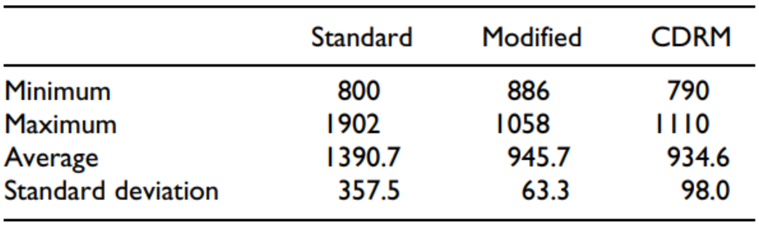


Figure 2.8: Evolution of Cost Function or Objective Function (Mohamad Nor,Ismail and Ahmad, 2020).



| | Standard | Modified | CDRM |
|---|---|---|---|
| Minimum | 800 | 886 | 790 |
| Maximum | 1902 | 1058 | 1110 |
| Average | 1390.7 | 945.7 | 934.6 |
| Standard deviation | 357.5 | 63.3 | 98.0 |

Figure 2.9: Convergence Time for Standard, Modified and CDRM Broadcast Controllers (Mohamad Nor,Ismail and Ahmad, 2020).

During numerical experiment, authors defined the convergence time as the number of iterations required for the objective function to reach its minimum value, which is set at 5. From the Figure 2.8, modified broadcast controller is coverges faster than standard but slower than CDRM broadcast controller. However, from the Figure 2.9, modified and CDRM are better than standard broadcast controller but standard deviation of modified broadcast controller is lower than CDRM. This means that the values does not varies much from average which is better than CDRM broadcast controller (Mohamad Nor,Ismail and Ahmad, 2020).

### 2.2.2 Lloyd’s Algorithm

The Lloyd’s algorithm is a coverage algorithm which based on Voronoi-cell approaches. To comprehend the Lloyd’s algorithm fully, it is essential to first explore into the concept of Centroidal Voronoi Tessellations (CVT) because CVT is fundamental part of Lloyd’s algorithm. Voronoi-based approach is a method to divide the working environment into several regions and each region will have their own mass centre. CVTs represent a special form of Voronoi tessellations within a bounded geometric domain and these domains are generating the points of the tessellations, often refer as centroids or mass centres within. These points were determined with respect to given density function (Du,Emelianenko and Ju, 2006). The natural optimization properties of Centroidal Voronoi tessellation make this concept have been widely used in various applications across in diverse field such as modelling, engineering, data analysis and so on. They are also utilized in the design of optimal vector quantizers. The underlying concept of CVTs shares similarities with the k-means clustering method. Given the modern applications of CVTs in large-scale multi-robot systems, the development of robust and efficient algorithms becomes imperative.

Throughout history, there were numerous of algorithm have been developed, studied and applied across various field (Du,Faber and Gunzburger, 1999). The first algorithm was developed was in 1960s at Bell Laboratories by S. Lloyd which retaining its popularity due to its inherent simplicity and effectiveness (Du, Emelianenko and Ju, 2006). The algorithm later is officiall published and is now commonly known as Lloyd’s algorithm (Lloyd, 1982).

Lloyd’s algorithm is one of the famous algorithms that utilize the concept of CVTs. Lloyd’s algorithm employs a centre-to-cluster and cluster-to-centre iterative process to guide the robots toward the mass centre of their respective Voronoi region points, commonly known as centroid. Initially, the Lloyd’s algorithm could only converge to local optima convergence. However, with advancements in technology, it can now achieve both local and global convergence, expanding its utility and applicability.

In paper of Du, Emelianenko and Ju in 2006, authors had presented a comprehensive analysis of the local and global convergence properties of Lloyd’s algorithm through analtical results and numerical experiments. The paper's focus was on investigating the convergence characteristics of this algorithm within the context of Centroidal Voronoi Tessellations (CVTs). The methodology employed in this research involved the generation points based on predefined density function. The iterative algorithm used in this study involves straightforward steps, starting with the initialization of an initial Voronoi tessellation that corresponds to a previous set of generators. The new set of generators was defined by the centroids of Voronoi regions, and this iterative process continued until a specified stopping criterion was met. In this paper, authors were focused on local and global convergences properties of Lloyd’s algorithm. This paper presents new findings regarding global convergence, including global convergence under specific non-degeneracy conditions, the overall sequence's global convergence within one-dimensional space and the convergence of subsequence based on density functions. For local convergence, authors studied for estimating on convergence rates. As part of their exploration, authors had introduced a concept called constrained centroidal Voronoi tessellation (CCVT). To validate that the energy defined in CVTs remained applicable to CCVT, two experiments were conducted. The first experiment involved six generators constrained to a circular path (1D curve), while the second experiment featured 162 generators constrained to a spherical surface (2D). According to Figure 2.10 ,results of experiments shows that both correspond to the constant density.

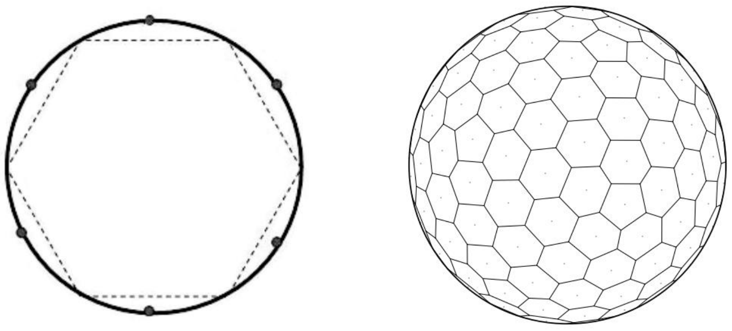


Figure 2.10: CCVTs for One Dimensional and Two Dimensional (Du, Emelianenko and Ju, 2006).

These findings demonstrate that the Lloyd's algorithm, when applied to compute CCVTs, achieves global convergence when dealing with positive and smooth density functions within the bounds of a smooth curve.

Building upon the research conducted by Du, Emelianenko and Ju in 2006, another group of researchers introduced a technique for achieving Voronoi coverage in non-convex environments using a network of interconnected robots. In this paper, the authors introduced a solution for decentralized Voronoi coverage tasks in polygonal environments with non-convex shapes. The algorithm they proposed ensures that it converges to a local optimal solution. In this study, the algorithm utilized was a combination of the classical Voronoi coverage approach, the Lloyd's algorithm, and a local path planning algorithm known as Tangent Bug (Breitenmoser et al., 2010). TangentBug algorithm is originally from family of Bug algorithm and its features is to ensure and compute the motion of multi-robots around obstacles (Kamon,Rimon and Rivlin, 1998). The concept of this novelalgorithm is, Lloyd's algorithm will update the goal position of each robot while TangentBug will responsible to plan a path to the given goal. The collaboration between two algorithms will result in global convergence of multi-robot system to each Voronoi region that optimizes the cost function or objective function. According to Figure 2.11, it showed that Lloyd's algorithm will compute the Voronoi region and updated the current target (level 1) Simultaneously, the local path planning algorithm, Tangent Bug algorithm will computes the path to the taregt to perform obstacle avoidance (level 2). This interaction forms Loop 1 between Level 1 and Level 2, with another Loop 2 occurring within Level 2.

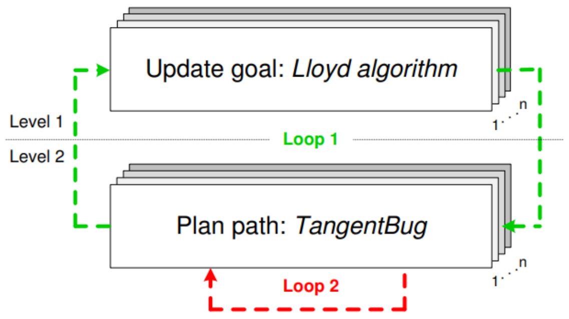


Figure 2.11: Mixture of the Tangent Bug algorithm and Lloyd's algorithm as Control Approach (Breitenmoser et al., 2010).

The authors proposed this new algorithm in response to limitations encountered in previous research involving Lloyd's algorithm. Previous studies focused on convex regions, which posed complex and challenging problems when applied to non-convex regions. Additionally, prior research also did not propose obstacle avoidance technique for Lloyd's algorithm making it unsuitable for realistic physical models. In this paper, authors had created centroid of Voronoi tessellation by using CVTs and described the importance (weighing) of different areas in environment by different density functions (Breitenmoser et al., 2010). Before explored into non-convex environments, authors had presented a series of steps to solve for convex environments. First step constructed the Voronoi partition for generating points for goal of multi-robot systems. Second step was computing the centroids of Voronoi regions, and third step was assigning new locations to the centroids and repeated the process until Voronoi coverage based on Lloyd's algorithm was completed. However, when transitioning to non-convex environments, several potential difficulties arise and cause the coverage problem to become complex. Figure 2.12 shows the possible difficulties faced by robot by using Lloyd's algorithm in non-convex environments.

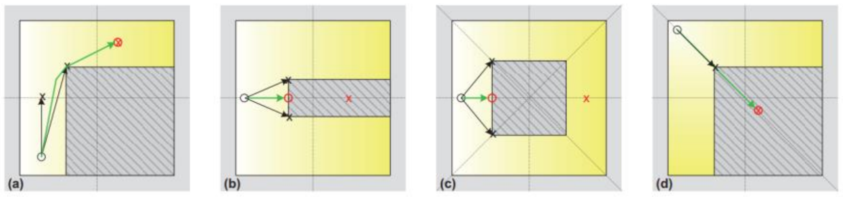


Figure 2.12: Example of Non-convex Environment (Breitenmoser et al., 2010).

In Figure 2.12, certain elements are represented: the white circle indicates the initial position of the robot, the red circle represents the robot's final position, the red cross signifies the Centroidal Voronoi Tessellation (CVT), which serves as the goal or target and the grey region represents obstacles within the environment. Figure 2.12 (a) showed the robot successfully reaches the goal by following a reasonable path calculated based on geodesic distance. The region covered has an L-shape. Figure 2.12 (b) showed the region formed a U-shape, but the robot converged to a minimum point located far from the goal, which is within the obstacle. Figure 2.12 (c) This demonstrates a scenario where the coverage region contained a standalone obstacle, yet the robot became obstructed on its path to the goal. Figure 2.12 (d), similar to (a), this scenario involved a L-shaped region, but the robot departs from the region due to non-gradient sums (Breitenmoser et al., 2010). These issues emerged due to the constraints in non-convex environments of Voronoi coverage when employing geodesic distance. While geodesic distance aids in computing distances along boundaries and avoiding obstacles, as illustrated effectively in Figure 2.12 (a), it simultaneously introduces trade-offs and limitations. For an example robot get trapped in figure 2.12 (b). In this paper, the authors introduce a novel approach for achieving Voronoi coverage in non-convex environments. This approach is constructed upon the foundation of the Lloyd’s algorithm and the Tangent Bug algorithm (Breitenmoser et al., 2010). The numerical results are shown in figure below.

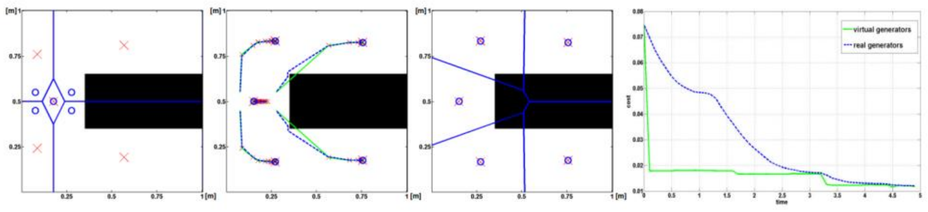


Figure 2.13: Voronoi Coverage in Non-convex U-shaped Environments (Obstacle is represented as Black Colour) (Breitenmoser et al., 2010).

From Figure 2.13, it shows that robots (blue circles) departed from initial position to current target position (red cross) and when it reached the target, Lloyd’s algorithm will compute new target for the robot until convergence is done. The green line represented trajectories of virtual generators.

The Lloyd’s algorithm has gained increasing popularity and there are many researchers are continually developing more advanced versions of Lloyd’s algorithm to adapt it to various situations. In 2019, a research explored the use of multiple sensor for multi-robot system in coverage task with Lloyd’s algorithm, particularly focusing on high order Voronoi (Jiang et al., 2019). Another recent research was carried out in 2021 which the authors introduced the modified version of Voronoi tessellation. The novel Voronoi tessellation was known as Obstacle -Aware Voronoi Cells (OVAC) This research integrated OVAC with the Lloyd’s algorithm to facilitate coverage control while ensuring obstacle avoidance and addressing low-level coverage issues (Abdulghafoor and Bakolas, 2021). In this year 2023, there was a novel approach to dynamic environments coverage based on Lloyd’s algorithm. This approach incorporates Density Function Path (DFP) and a series of Gaussian distribution functions. Authors presented this new approach because they motivated by multi-robot system is widely used in industrial field and some clusters in the working environment need attracted extensive attention by more repeated coverage by multi-robot systems (Yang et al., 2023).

### 2.2.3 Spanning Tree-Based Methods

Spanning tree-based is an algorithm employed to ensure comprehensive coverage of every point within the approximate working environment. It finds application in various tasks such as floor cleaning and lawn mowing (Gabriely and Rimon, 2001). The fundamental working principle of Spanning tree-based algorithm (STC) is diving the environment into different cells corresponding to multi-robot systems and the robots will follow the spanning tree of graph induced by the cell while covering every point in environment precisely (Gabriely and Rimon, 2001). In this paper of 2001, authors presented and analysed three version of STC which are first version is off-line where robot possessed prior knowledge of the working environment, its pre-plans its coverage strategy based on this information. The second version was on-line where the robots utilized their sensor to detect obstacles in real-time and constructed the spanning tree of environments while covering the area. The third version was ant-like where robots operate without prior knowledge of the environment. However, as they traversed and covered the area, they left markings, akin to the trail left by ants. These markings guide their future coverage actions.

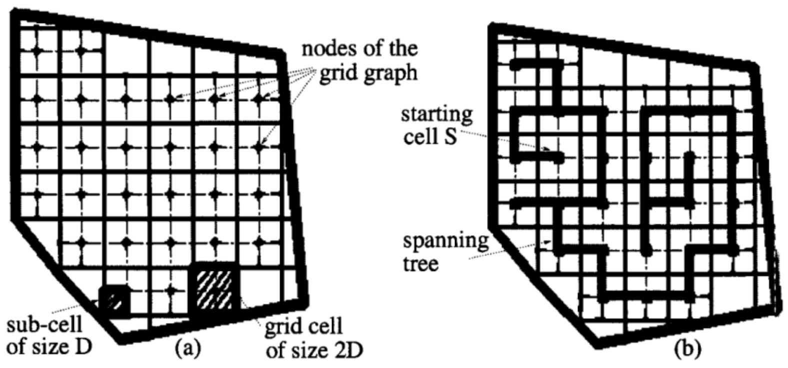
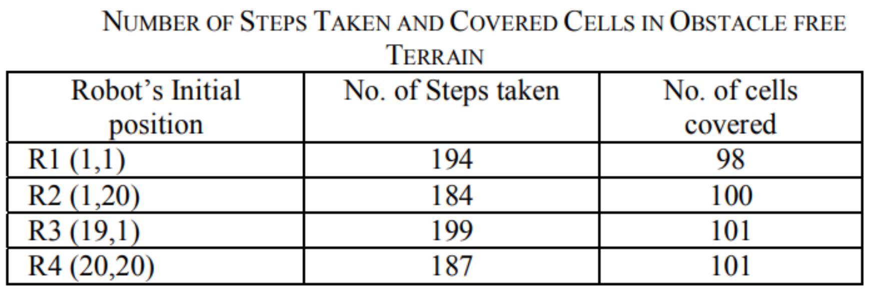


Figure 2.14: (a) Grid Approximation of given Working Environment. (b) A Spanning Tree of Grid of given Working Environment (Gabriely and Rimon, 2001).

In 2008, there were authors presented spanning tree-based algorithms in terrain coverage algorithms. The authors introduced an innovative real-time online coverage strategy for multiple robots, designed to guarantee thorough and dependable coverage of both terrain and unknown environments. Multi-robots will leave marking in the terrain and these marking will sensed by all robots without direct communication with each other (Senthilkumar and Bharadwaj, 2008). In this study, the robots employed a Depth-First Search (DFS) approach to gradually create a spanning tree. This tree was used to track the cells that had been visited and explored during the application of the spanning tree-based algorithm. These explorations were recorded and represented in a matrix format, facilitating subsequent simulation and analysis.


NUMBER OF STEPS TAKEN AND COVERED CELLS IN OBSTACLE FREE TERRAIN

| Robot’s Initial position | No. of Steps taken | No. of cells covered |
|---|---|---|
| R1 (1,1) | 194 | 98 |
| R2 (1,20) | 184 | 100 |
| R3 (19,1) | 199 | 101 |
| R4 (20,20) | 187 | 101 |

Figure 2.15: The Count of Steps taken and Cells covered in a Terrain Devoid of Obstacles (Senthilkumar and Bharadwaj, 2008).

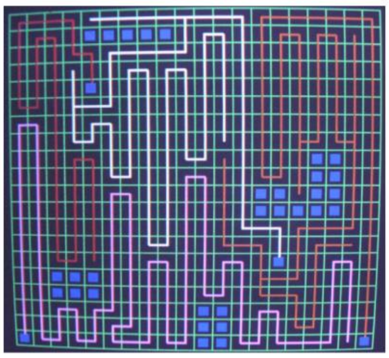


Figure 2.16: Generated Four Spanning Trees in a Terrain with Obstacles (Senthilkumar and Bharadwaj, 2008).

Figure 2.15 clearly illustrates that the initial positions of the robots have a direct impact on the number of cells they can effectively cover. This observation highlights the necessity of optimizing the initial placement of the robots to enhance coverage efficiency. Figure 2.16 presents the resulting spanning tree generated by four robots operating in a terrain that includes obstacles. The application of Depth-First Search (DFS) is shown to successfully achieve complete terrain coverage.

## 2.3 Dynamic Obstacle Avoidance Techniques

When handling dynamic obstacle in multi-robot coverage, it is crucial to consider both coverage control algorithms and dynamic obstacle avoidance techniques. In this sub-chapter, three dynamic obstacle avoidance techniques will be presented, discussed, evaluated and compared.

### 2.3.1 Model Predictive Control (MPC) for Robot Navigation

Model Predictive Control (MPC) is employed as an obstacle avoidance strategy for robots and autonomous vehicles, enhancing safety in unfamiliar environments (Park et al., 2009). The concept of model predictive control (MPC) is generating safe trajectories through non-linear model predictive frameworks. These frameworks employ simplified robot dynamics to forecast the robot's state over a specified look-ahead horizon (Park et al., 2009). The performance index should integrate with the local obstacle information to ensure safety by using a parallax-based method within the framework of non-linear model predictive control. One of the strengths of MPC is its lookahead capabilities, which help reduce the risk of getting trapped in local minima, a common issue encountered with potential field-based methods.

In paper of Park et al in 2009, authors proposed an improved MPC scheme with a controller designated to navigate the vehicle safety to a predefined goal point while avoiding collision with unknown obstacle within the coverage environemnt (Park et al., 2009). The controller consists of two primary components which are first part is focusing on generating safe trajectories for vehicle based on MPC. The second step involves generating actual control inputs for the vehicle based on the safe trajectories generated in the first step. The tracking controller comprises two distinct controllers which are one for handling longitudinal dynamics and the other for managing lateral dynamics. The longitudinal dynamics is using PID controller while lateral dynamics is using LQR controller (Park et al., 2009).

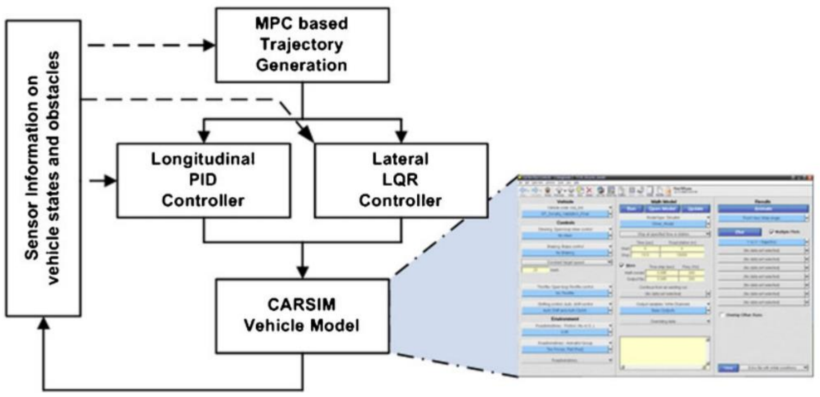

MPC based Trajectory Generation
Sensor Information on vehicle states and obstacles
Longitudinal PID Controller
Lateral LQR Controller
CARSIM Vehicle Model

Figure 2.17: Overall Layout of Controller for MPC (Park et al., 2009).

The concept of a finite horizon, initiated at each time step, refers to the receding horizon principle (Park et al., 2009). This principle is crucial as it aids in generating safe trajectories by integrating sensor-based obstacle information with the performance index. The resulting trajectories may vary each time based on the obstacle information incorporated. While the distance-based method is a common approach to penalize proximity to adjacent obstacles, this paper employs a parallax-based obstacle avoidance technique, which is chosen for its advantages over distance-based approaches. In parallax-based, although the distance between obstacles and vehicle is the same, but the parallax angle will be different depending on the angle.

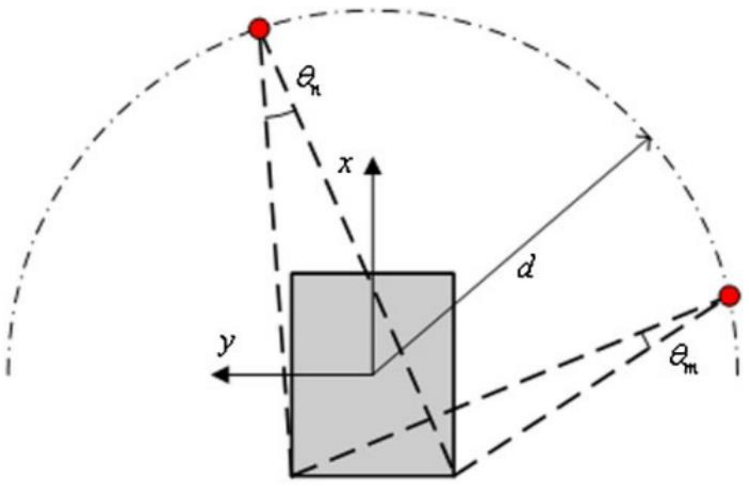


Figure 2.18: Comparing the Parallax Angles of Two Obstacles equidistant from the Vehicle's centre of Gravity, the Obstacle situated in front of the Vehicle exhibits a Larger Parallax Angle. (Park et al., 2009).

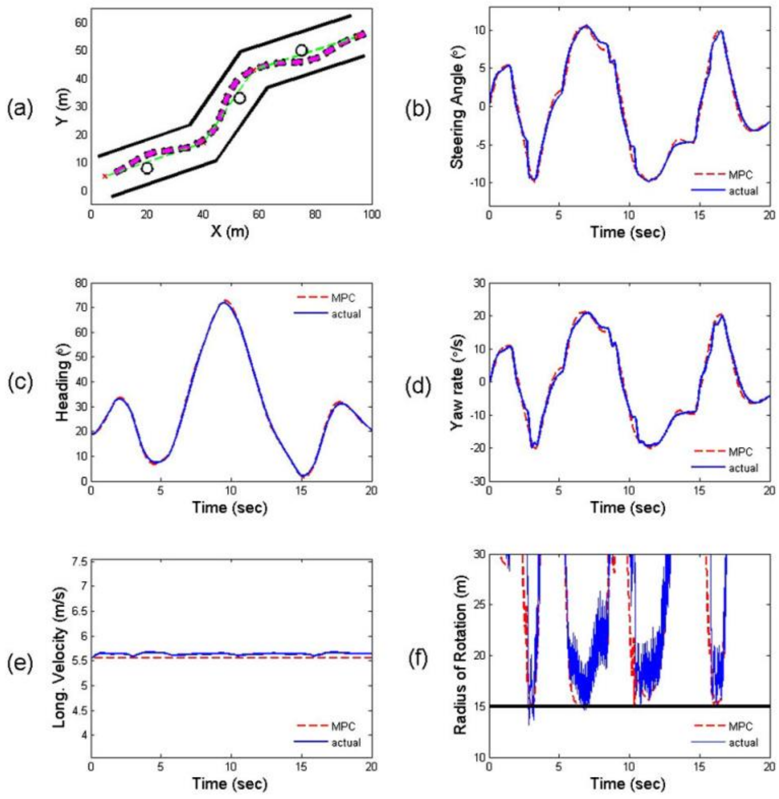


Figure 2.19: Simulation Results in a Static Environment with a Constant Velocity of 20 km/h: (a) Depicts the resulting trajectory, (b) Shows the input history, (c) Displays the heading history, (d) Illustrates the yaw rate history, (e) Demonstrates the longitudinal velocity history, and (f) Presents the instantaneous turning radius along the trajectory (Park et al., 2009).

Figure 2.19 shows the results in numerical simulation in a static environment with constant velocity of vehicle. From Figure 2.19 (a), it is clearly shown that the vehicle is successfully avoid the obstacle.

### 2.3.2 Artificial Potential Field

Collision avoidance is the realm of high-level planning problem. There was an approach introduced by Khatib in 1986 which known as artificial potential field (Khatib, 1986). This method introduces a real-time obstacle avoidance strategy for mobile robots, rooted in the concept of artificial potential fields (Khatib, 1986). Artificial potential field is often referred as virtual force field approach. Artificial potential field find application in various domains including path planning, coverage path planning and coverage control of multi-robot systems. Artificial potential field has the advantages of little calculation amount, simple model and high flexibility (Wu,Su and Li, 2019). The basic idea of artificial potential field is control the movement of robots in the environment as the motion in a virtual potential field (Pradhan et al., 2006).

Artificial potential field (APF) will construct different potential field towards the goal and obstacles in the working environment by controlling the agents or robots through calculating the resultant forces of all potential fields on the robots (Wu, Su and Li 2019). However, APF does have its limitations, one of which is its susceptibility to getting trapped in local minimum. To address these shortcomings, an improved version of APF was proposed in the paper by Wu, Su and Li in 2019. This enhancement focused on two main fields. Firstly, a gain constraint was added to the repulsive potential field model. Secondly, a random factor was introducted to avoid falling into local minimum. Additionally, the concept of B-spline curve was also introduced as optimization to tackle the issues associated with traditional APF. In this research, authors considered a scenario with homogeneous robots operating in a 2D area with the predefined goal and obstacles. Each robot had a unique initial start position, and a predefined goal and obstacles were present in the environment. A threshold distance between the goal and the robots was also established as part of the problem description.

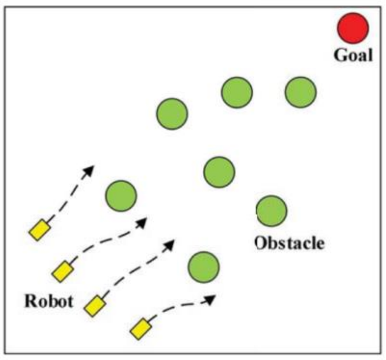


Figure 2.20: Schematic Diagram of the Initial Position of Robots, Position of Obstacles and Goal (Wu, Su and Li 2019).

The concept behind the artificial potential field approach involves the creation of virtual potential field within the space. This field will generate gravitational forces to the robots and influence the movement of robots towards the goals. However, when obstacles are present in the working environment, they generate repulsive forces on the robots to steer them away from potential collisions. The direction of movement of robots can be either deterministic or random, depending on the specific implementation. In the context of the research presented in this paper, the movement direction of multi-robot system is deterministic. It is determined by calculating the resultant force acting on each robot, which is the summation of gravitational forces pulling them towards their goals and repulsive forces pushing them away from obstacles (Wu, Su and Li 2019).

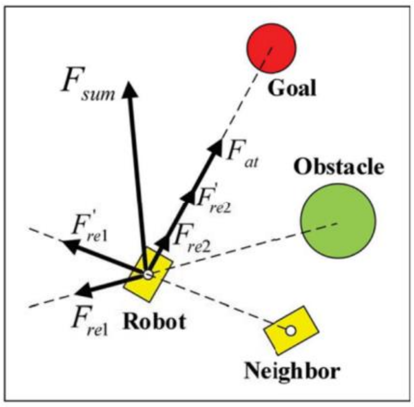


Figure 2.21: Force Analysis of Robot in Potential Field (Wu, Su and Li 2019).

In this paper, the authors had proposed an improved version of the artificial potential field. In the standard APF, the original repulsion model was positively correlated with distance of robots from the goal to ensure robots can still reach their goals even when they are close to them. In contrast, when distance of robots was far from the goal, the distance will have large gain on the repulsive force. If the angle between the gravitational force and the repulsive force is approximately 180 degrees, the oscillation of robots will occur. This is because when robot approaches an obstacle closely, a large and strong repulsive force will be generated due to APF, and this caused repulsive force is larger than the gravitational force. As a result, robot will move away from obstacle due to larger repulsive force. However, the gravitational force is still acting towards the goal, leading the robot to return towards the obstacle, creating a repetitive back-and-forth motion until the angle becomes smaller, allowing the robot to bypass the obstacle smoothly (Wu, Su and Li 2019).

To address this issue in the standard APF, authors proposed a constraint can be added in improved APF which acted as reduction to the angle between the repulsive force and gravitational forces rapidly. Additionally, the standard APF had a drawback which is robot will converge into local minimum. This problem arises because the attraction force by the goal and repulsive force by the obstacle are totally opposite in the angle of robot. This will cause the movement of robot limited to a straight line for round-trip motion (Wu, Su and Li 2019). In this case, robot cannot move to goal without any external force. To mitigate this problem, the authors incorporated a random factor into the resultant force acting on the robot. This random factor introduces variability in the direction and magnitude of the force, helping the robot escape from local minima and enabling more flexible movement (Wu, Su and Li 2019).

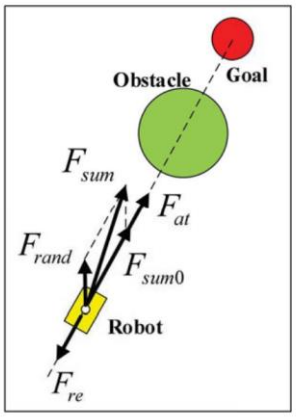


Figure 2.22: Force Analysis after Adding with a Random Factor (Wu, Su and Li 2019).

Authors had also proposed optimization of APF by using B-spline Curve which one of generalization of the Bezier curve. This B-spline curve can reduce variation, geometric invariance and convexity. The results of proposed algorithms are shown in numerical simulation.

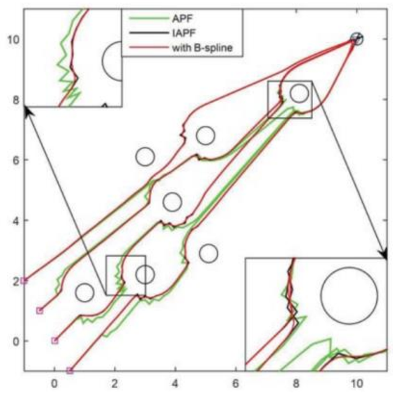


Figure 2.23: Path for Three Algorithms (Wu, Su and Li 2019).

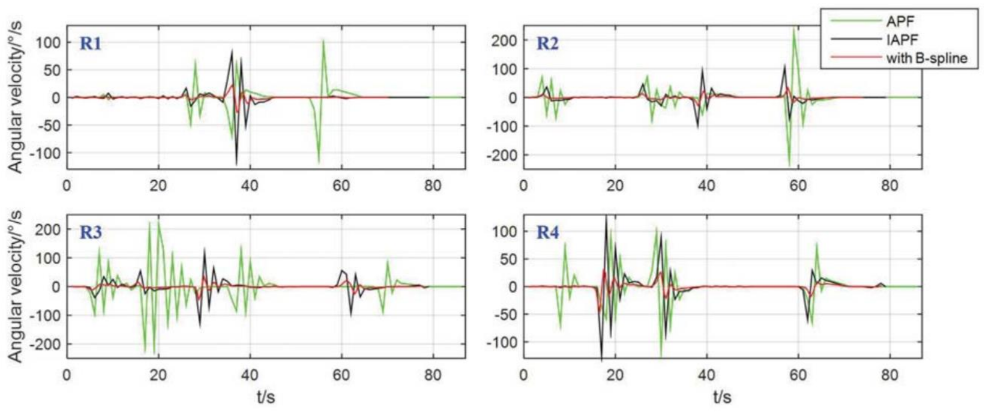
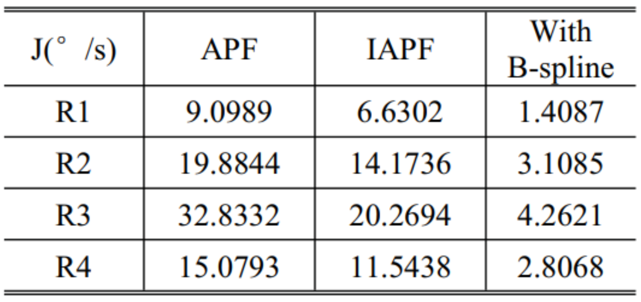


Figure 2.24: Comparison of Angular Velocity for Three Algorithms (Wu, Su and Li 2019).



| J(°  /s) | APF | IAPF | With B-spline |
|---|---|---|---|
| R1 | 9.0989 | 6.6302 | 1.4087 |
| R2 | 19.8844 | 14.1736 | 3.1085 |
| R3 | 32.8332 | 20.2694 | 4.2621 |
| R4 | 15.0793 | 11.5438 | 2.8068 |

Figure 2.25: Comparison of Evaluation Function of Three Algorithms (Wu, Su and Li 2019).

Figure 2.23 provides a comparison of simulation results among APF, IAPF, and B-spline curve optimization. It is evident that IAPF effectively mitigates the oscillation issues compared to standard APF. However, when comparing IAPF to B-spline optimization, it becomes clear that B-spline optimization offers further reductions in oscillations and overall improved performance. In Figure 2.24, the angular velocities of four robots (R1, R2, R3, and R4) are displayed. It can be observed that B-spline optimization results in the lowest angular velocity, indicating that it consumes less energy compared to the other methods. Figure 2.25 illustrates the evaluation function for the three algorithms: APF, IAPF, and B-spline. Both IAPF and B-spline exhibit significantly lower evaluation function values compared to APF. This demonstrates that IAPF and B-spline effectively enhance the performance of the APF approach. The proposed algorithms achieve smoother paths and reduce the angular velocity of robots by guiding them along smaller circles (Wu, Su, and Li, 2019).

### 2.3.3 Velocity Obstacle (VO) Method

Implementing real-time navigation for multi-robot systems in dynamic environments poses a significant challenge, largely because of the presence of dynamic obstacles. To navigate safely and avoid collisions with moving dynamic obstacles, it becomes essential to define controls that guide the robots effectively. Velocity Obstacle approach (VO) has been widely adopted for navigation among dynamic obstacles. The advantage of this approach is it included constraint to compute a collision-free path for a robot to operate in working environments among dynamic obstacles. Velocity Obstacle concept was introduced by Fiorini and Shiller in 1998 and has been extended to multi-agent and multi-robot navigation (Fiorini and Shiller, 1998).

In 2009, a significant research paper extended the concept of Velocity Obstacle to accommodate robots with kinematic constraints. This work was undertaken by Wilkie, Van Den Berg and Manocha (Wilkie,Van Den Berg and Manocha, 2009). This research aims to generalized velocity obstacle approaches to ensure it is safely to adopt in navigation robots among dynamic obstacles. The difference between basic velocity obstacle and generalized velocity obstacle lies in their approach to predicting the future positions of agents or obstacles. In the basic velocity obstacle, the future position of agent or obstacle was assumed approximately as a circle centred at their current position. In contrast, the generalized velocity obstacle predicts future trajectories of agent or obstacle based on their current behaviours and intention. The initial Velocity Obstacle formulation was designed for multi-agents to move along piecewise linear velocities and made an assumption that the dynamic obstacles move at a constant velocity. To assess the performance of this extension, the authors conducted experiments with various parameters. The experiment took place in open environments and the dynamic obstacles were distributed randomly within the region, each with random velocities. The results obtained from these experiments were used to evaluate the proposed approach.

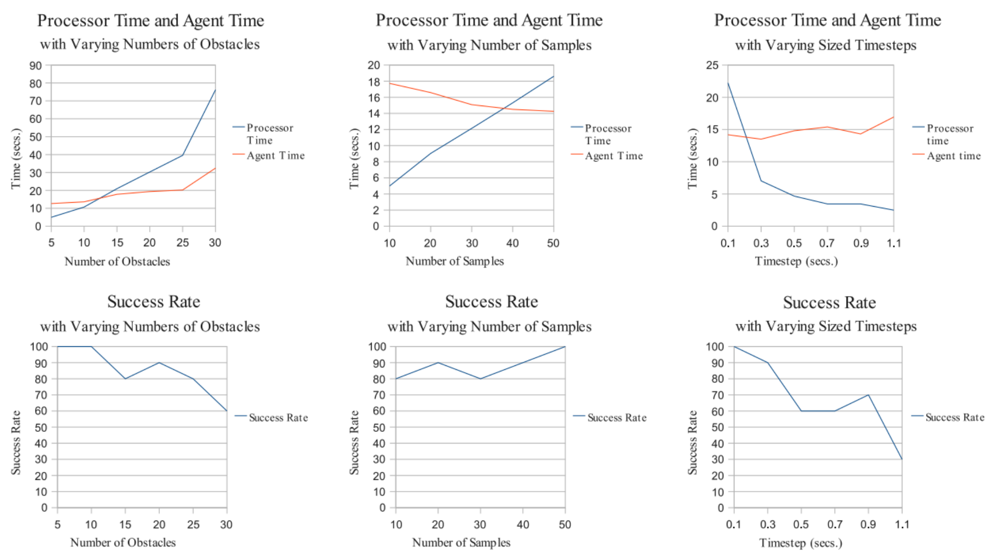


Figure 2.26: Results of Simulation based on Three Different Parameters (Wilkie,Van Den Berg and Manocha, 2009).

Figure 2.26 illustrates the results of the experiment conducted to evaluate the proposed extension of Velocity Obstacle (VO). During the experiment, various parameters were adjusted, including the number of samples taken, the number of dynamic obstacles and the length of the time step. From the figure 2.26, it is shown that when the number of obstacles is increased, the required path will take longer in agent time. The second result obtained was when number of sample increases, the agent or robot will be easier to control the path to the goal. The third result is when the timestep increases, the success rate will drop (Wilkie,Van Den Berg and Manocha, 2009). In this paper, authors successfully generalized the velocity concept for robots to navigate among dynamic obstacles. However, there are some limitations which are the calculated minimum distance is by using numerical mean which will cause inaccuracy in computation.  The second drawback is that due to characteristics of velocity obstacles, the perfect solution may be not selected. In real scenarios, there will be existence of noise which will cause the distortion to the algorithm to calculate the position and distance of dynamic obstacles with robots. The most limitation is this velocity obstacle may not suitable for environment with local minima (Wilkie,Van Den Berg and Manocha, 2009). Despite these limitations, there have been various improved versions and extensions of Velocity Obstacle, including the Reciprocal Velocity Obstacle (RVO), which aim to address some of these challenges.

Reciprocal Velocity Obstacle (RVO) is aimed to create a collision-free and oscillation-free motion with multi-robot or agent navigation. In paper of Juniastuti et al  in 2016, authors proposed a RVO with steering behaviour such as leader following behaviour(Juniastuti et al., 2016).

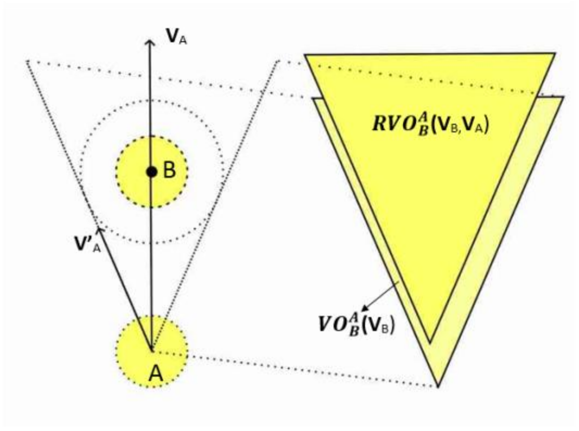


Figure 2.27: Reciprocal Velocity Obstacle from Agent A to B (Juniastuti et al., 2016).

In RVO, the velocity is chosen from a valid velocity which near to a reference velocity. Valid velocity means the velocity which outside of current velocity of agent and also outside of obstacle velocity. This concept can be using in multiagent navigation through RVO.

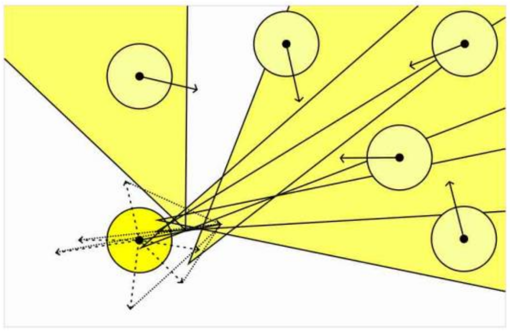


Figure 2.28: Navigation of Multiagent through RVO (Juniastuti et al., 2016).

Leader following meaning refers to a steering behaviour that used in multi-agent system where agents' movement and motion directed to follow leaders. The concept of steering behaviour is originated from Craig W. Reynolds in 1999 (Reynolds, 1999). In this idea, it consists of two behaviours which are arrival and separation  which are followers will stay close only to leader without blocking the movement and accompanied by avoiding collision with other followers (Reynolds, 1999).

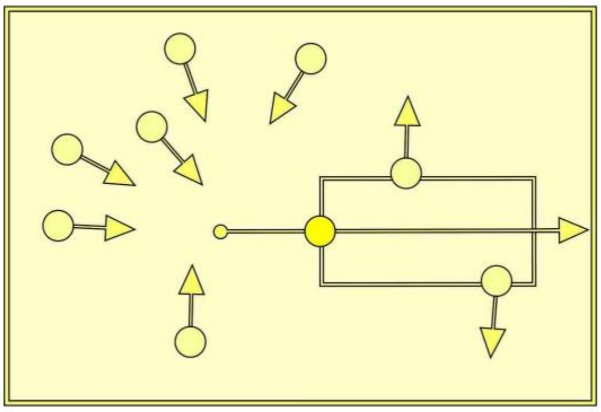
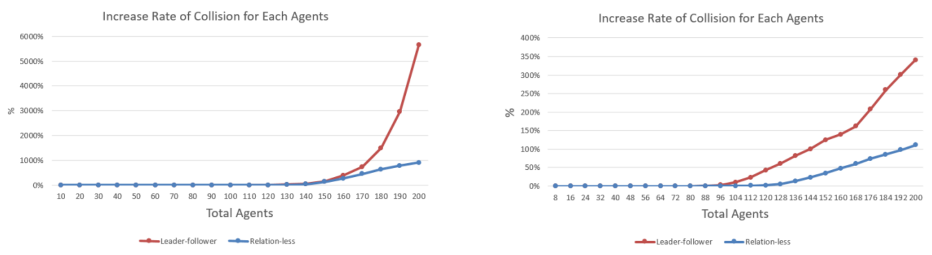


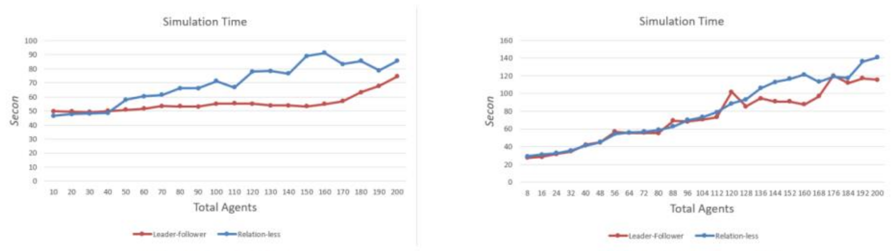
Figure 2.29: Leader-following Behaviour (Juniastuti et al., 2016).

In the simulation, authors had conducted the experiment in two situations which are crossroad scenarios and narrow passage scenarios. The results of simulation are shown in below.



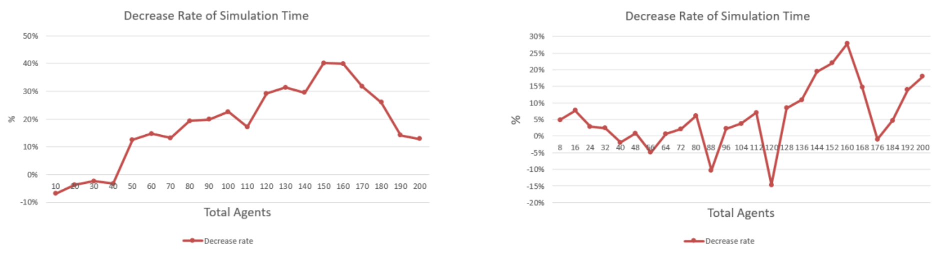
(a) Crossroad scenario (b) Narrow passage scenario

Figure 2.30: Graph of Increase in Agents' collision (Juniastuti et al., 2016).



(a) Crossroad scenario (b) Narrow passage scenario

Figure 2.31: Graph of Simulation Time (Juniastuti et al., 2016).

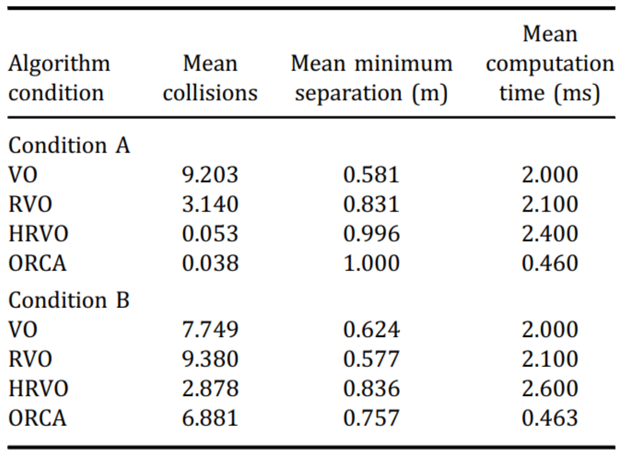


(a) Crossroad scenario (b) Narrow passage scenario

Figure 2.32: Graph of Reduction in Simulation Time by Leader-Follower Implementation (Juniastuti et al., 2016).

The simulation results conclude that implementing a leader-follower approach in scenarios involving narrow passages and crossroads effectively reduces the time required for agents to navigate to their destinations.

In 2019, Douthwaite, Zhao, and Mihaylova conducted experiments to evaluate the effectiveness of two main extensions of Velocity Obstacle (VO) - HRVO and ORCA (Douthwaite,Zhao and Mihaylova, 2019). The evaluation focused on assessing the performance of these collision avoidance approaches in terms of increasing collision scenarios and the computation time required for collision avoidance.



| Algorithm condition | Mean collisions | Mean minimum separation (m) | Mean computation time (ms) |
|---|---|---|---|
| Condition A | | | |
| VO | 9.203 | 0.581 | 2.000 |
| RVO | 3.140 | 0.831 | 2.100 |
| HRVO | 0.053 | 0.996 | 2.400 |
| ORCA | 0.038 | 1.000 | 0.460 |
| Condition B | | | |
| VO | 7.749 | 0.624 | 2.000 |
| RVO | 9.380 | 0.577 | 2.100 |
| HRVO | 2.878 | 0.836 | 2.600 |
| ORCA | 6.881 | 0.757 | 0.463 |

Figure 2.33: Performance of Collision Avoidance Approaches (Douthwaite,Zhao and Mihaylova, 2019).

From the Figure 2.33, Condition A assumes ideal sensing capabilities for the agents, where they have perfect knowledge about the available obstacles in their environment. In contrast, Condition B represents a more realistic scenario where the agents rely on sensors to detect obstacles. From the Figure 2.32, it is clearly shown that HRVO reduced the mean of collision compared to VO and RVO, but ORCA had the lowest mean of collision in condition A. When in condition B, due to uncertainty of sensors in robots, the effectiveness decreasing and cause the mean of collision increase rapidly and higher than HRVO. In condition B, RVO also exhibited a high mean of collisions. This was attributed to its drawback, which involves exacerbating the reciprocal corrections due to the uncertainty in obstacle trajectories (Douthwaite,Zhao and Mihaylova, 2019).

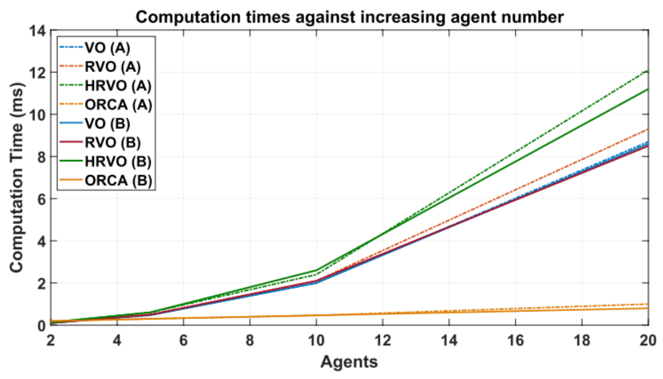


Figure 2.34: Computation Time against the Increasing Agent Number (Douthwaite,Zhao and Mihaylova, 2019).

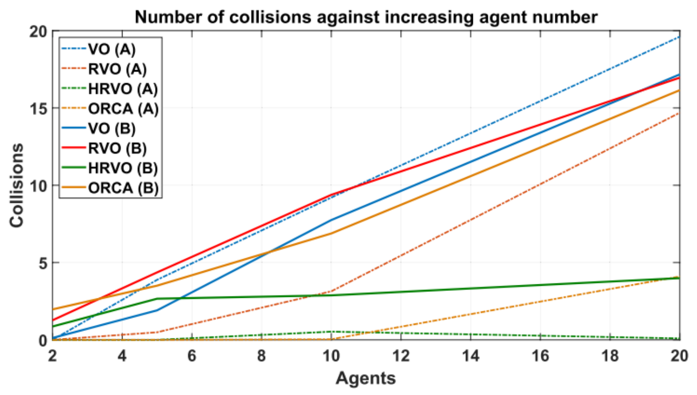


Figure 2.35: Number of Collision against the Increasing Agent Number (Douthwaite,Zhao and Mihaylova, 2019).

## 2.4 Gaussian Distribution

In recent years, there has been a significant increase in efforts from various industries and research fields to efficiently deploy multi-robot systems. These systems are composed of multiple robots operating within a designated domain such as search and rescue, agriculture and so on (Wang,Xie and Agrawal, 2008). The primary objective of this distribution is focusing on achieving the complete coverage and optimize operation time of collection data. With different robots deployed, the coverage and operating time will be different which will cause the system not efficient and decrease the productivity.

One of solution to distribute the multi-robot system in environment are Gaussian distribution. Gaussian distribution has advantages of solving the non-uniform dissipation of energy to deploy large scale of multi-robot.

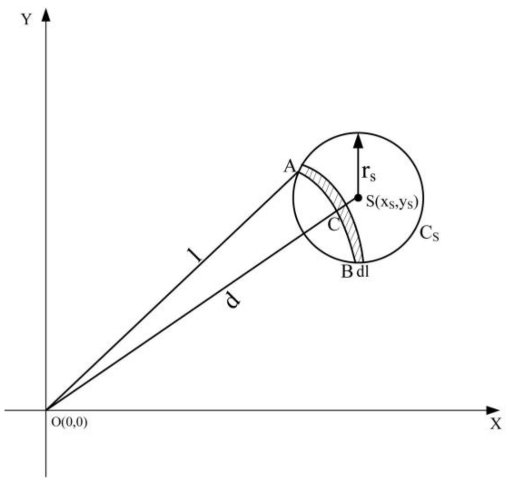


Figure 2.36: Coverage Probability of Point (Wang,Xie and Agrawal, 2008).

From the Figure 2.36, it is clearly shown that the two points which have the same distance will experience same deployment probability. The concentration of points around their mean value will be approximately estimated by using method of continuous probability distribution. The probability density function (pdf) is the mean value when the pdf is located in the peak of bell-shaped.

## 2.5 Summary

The comparison between overall coverage control algorithms and dynamic obstacle avoidance techniques is essential to determine which approach is better suited for solving the problem of modelling multi-robot systems for coverage tasks in dynamic environments.

Table 2.1: Comparison between Coverage Control Algorithms.

| Coverage Control Algorithm | Method Using | Advantages | Disadvantages |
|---|---|---|---|
| Broadcast control algorithm | 1. One to all communication<br>2. Use a centre control unit to send signal to agents to perform coverage task | 1. Do not necessitate additional devices and energy for information transmission.<br>2. Achieve convergence approximately. | 1. Inadvertently affect neighbouring agents, leading to unintended consequences<br>2. Result in an unstable solution in specific motion-coordination tasks related to coverage |
| Lloyd’s algorithm | 1. Using centroid Voronoi tessellation to generate Voronoi partition and assign the centroid | 1. Using k-mean clustering to guide robot towards goal<br>2. Can combine with Tangent Bug to perform obstacle avoidance | 1. Initially only can local optima convergence<br>2. Tangent Bug only suitable for static obstacle |
| Spanning Tree Based Algorithm | 1. Divide the environment into different cells corresponding to multi-robot systems<br>2. Robots will follow the spanning tree of graph induced by the cell | 1. Traversed and covered the area, left markings, akin to the trail left by ants. These markings guide their future coverage actions. | 1. More suitable for single robot<br>2. More suitable for coverage path<br>3. Need modified if applied for coverage control |

Table 2.2: Comparison between Dynamic Obstacle Avoidance Techniques.

| Obstacle Avoidance Techniques | Method Using | Advantages | Disadvantages |
|---|---|---|---|
| Model Predictive Control (MPC) | 1. Generating safe trajectories<br>2. Generating actual control inputs for the vehicle based on safe trajectories that generated | 1. Look-ahead capabilities<br>2. Avoid fall into local minima | 1. Involve with many calculation<br>2. Process complex |
| Artificial Potential Field (APF) | 1. Control the movement of robots in the environment as the motion in a virtual potential field<br>2. Calculating the resultant forces of all potential fields on the robots | 1. Little calculation amount<br>2. Simple model<br>3. High flexibility | 1. Easy fall into local minimum<br>2. Oscillation of motion occurs |
| Velocity Obstacles (VO) | 1. Future position of agent or obstacle was assumed approximately as a circle centred at their current position | 1. Include constraint to compute a collision-free path for a robot to operate in working environments among dynamic obstacles | 1. Calculated minimum distance is by using numerical mean which will cause inaccuracy in computation<br>2. Perfect solution may be not selected and cause more time consuming. |

In summary, the chosen solution for addressing the coverage task with dynamic obstacles is a combination of the Lloyd’s algorithm with the Velocity Obstacle (VO) technique and known as Modified Lloyd’s algorithm with VO. This decision is based on several key factors which are Lloyd’s algorithm offers flexibility and simplicity in terms of modification and integration with other techniques, including Velocity Obstacle. This adaptability is crucial for tackling dynamic obstacle scenarios effectively. Lloyd’s algorithm's underlying concept is relatively straightforward to grab and implement, making it accessible for practical applications. There are many resources available for Lloyd’s algorithm, facilitating its implementation and optimization. Velocity Obstacle technique is chosen because it is explicitly designed for dynamic obstacle avoidance which suitable to solve the problem in this project. VO has various extensions like Reciprocal Velocity Obstacle (RVO), which can further enhance its capabilities and adaptability in complex modelling scenarios.

# CHAPTER 3

# METHODOLOGY AND WORK PLAN

## 3.1 Introduction

In this section, the project methodology and work plan will be discussed. The methodology used for dynamic obstacle handling in multi-robot coverage will be divided into two parts which are first part is algorithm for coverage task and another part is dynamic obstacle avoidance technique. These components will later be integrated into a unified methodology and implemented in a multi-robot system to carry out coverage tasks in the presence of dynamic obstacle. For the algorithm of coverage task, the algorithm used in this project is Lloyd's algorithm due to its stability and flexibility. For the dynamic obstacle avoidance technique, the technique used will be Velocity Obstacle (VO). This is because VO is suitable for handling multi-robot with dynamic or moving obstacle in the working environment. The combination of both parts will be Modified Lloyd's algorithm with VO. The project flow or work plan also will be presented in this section with the detail explanation. Solutions for each problem encountered will be stated and discussed in this section.

## 3.2 Proposed Dynamic Obstacle Handling in Multi-Robot Coverage Algorithm

Lloyd's algorithm and Velocity Obstacle technique will be discussed in this section. The equation, problem setup or assumption will be stated to simply the simulation in order to get the results. The implementation of algorithms into dynamic obstacle handling in multi-robot coverage will be simulated in software MATLAB.

### 3.2.1 Problem Setup and Assumptions

The environment is assumed in a convex domain where $D \subset \mathbb{R}^2$, where $\mathbb{R}$ is the space of dimensional vectors and 2 stands for 2-dimensional (Yang et al., 2023). The function of the convex domain is to be piecewise continuous function. This is to ensure the output value will be finite and not near to zero (non-decreasing). The X is denoted as a convex and compact polygon filed to be covered in dimensional vector and $\varphi(q)$ represents the likelihood of target distribution in the environment which means density function. V = { $V_1$, $V_2$, ..., $V_n$ } is the partition of X and $||.||$ is the Euclidean distance function (Yang et al., 2023). Each robot in the multi-robot system will has their own limitation on the aspects of communication and sensing capabilities. Therefore, the starting positions of robots and dynamic obstacle are assumed known in each reference coordinate system. All robots used is assumed to be differential drive robot and each of them are identical to each other. To address the limitations, the function of domain must differentiable and non-decreasing. The density function φ(q) also differentiable continuous function and is called as bounded. This is to ensure that density function has higher probability to obtain date in the designated domain.

### 3.2.2 Gaussian Distribution

The probability density function of a multivariate Gaussian distribution characterized by its covariance matrix and mean (Yang et al., 2023).

$$p_N(x, \mu, \Sigma) = \frac{\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{\sqrt{\det(\Sigma)(2\pi)^k}} \tag{3.1}$$

where,

$p_N$ = nth agent's position in D

μ = mean ∈ $\mathbb{R}^k$

det (Σ) = determinant of the covariance matrix Σ

x = data and x ∈ $\mathbb{R}^k$

### 3.2.3 Centroid Voronoi Tessellation (CVT)

First, to conduct centroid Voronoi tessellation, CVT, Voronoi partition must be conducted first to divide the working environment into designated number of working areas. The position of agents or robots in the X is represented by x = [ $x^T_1$, $x^T_2$, ..., $x^T_n$ ] and the Voronoi Partition is defined as V(x) = {$V_{1(x)}$, ..., $V_{n(x)}$ } (Yang et al., 2023). Voronoi Partition equation is,

$$V_i(x) = \{q \in X \mid \|q - x_i\| \leq \|q - x_j\|, \forall j \neq i$$

(3.2)

where,

$V_i(x)$ = Voronoi cell

By denoted that density function, *φ(q)* is over X, *φ(q)* will become continuous and measurable. The centroid and mass of Voronoi cell are defined as below,

$$M(V_i\,(x)) = \int_{V_i\,(x)} \varphi(q)\,dq$$

(3.3)

$$C(V_i\,(x)) = \frac{1}{M(V_i\,(x))} \int_{V_i\,(x)} q\varphi(q)\,dq$$

(3.4)

The point in Voronoi partition can be said as centroid if the centroid of $V_i\,(x)$ is point $x_i$. The centroid Voronoi tessellation is successfully conducted.

### 3.2.4 Density Function

The density function is modelled by Gaussian Distribution Model (3.1),

$$\varphi(q) = \varphi(q, \mu, \Sigma) = \frac{\exp(-\frac{1}{2}(q-\mu)^T \Sigma^{-1}(q-\mu))}{\sqrt{\det(\Sigma)(2\pi)^k}}$$

(3.5)

Where,

K = dimension of the function

Σ = sigma or standard deviation

$\mu$ = mean

The density function is constructed based on the Gaussian distribution model (3.1) by its covariance matrix and mean. The density function constructed (3.5) is used to locate set of unknow multi-robot in environment to obtain specific parameters based on expectation.

In this case, bivariate is used, therefore k = 2,

$$q = x = \begin{bmatrix} x_1 \\ \cdots \\ x_k \end{bmatrix}, in\ this\ case, x = \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.6}$$

$$\mu = \begin{bmatrix} \mu_1 \\ \cdots \\ \mu_k \end{bmatrix}, in\ this\ case, \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{3.7}$$

$$\Sigma = \sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \tag{3.8}$$

By combing equation above which are 3.5, 3.6. 3.7 and 3.8, a new equation is formed,

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_x)^2}{\sigma_x^2} - 2\rho\frac{(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} + \frac{(y-\mu_y)^2}{\sigma_y^2}\right]\right) \tag{3.9}$$

### 3.2.5 Objective Function

The objective function is the function to optimize the system, use to minimize or maximize to achieve the goal. The function that reduces cost is called cost function which is one type of objective function (Lee and Egerstedt, 2013). The cost function needs to be as low as possible to make the system become efficiency. The objective function is shown below,

$$H(p, P, t) = \sum_{i=1}^{n} \int f \; ||q - p_N||^2 \; \varphi(q, t) dq$$

(3.10)

where,

$H(p, P, t)$ = The objective function to be evaluated, where:

- p might represent the current position or state of a particular robot.
- P might denote the set of all robots' positions or states.
- t is the time variable.

$n$ = The number of robots or sensors involved.

$\int$ = Integral over a specified domain, likely representing the entire coverage area.

$f =$ A function or a constant factor, potentially representing the influence of distance or some form of weighting related to the efficiency of coverage or sensor effectiveness.

$q =$ A point in the coverage area over which the integration is performed.

$p_N =$ The position of the N-th robot at time t (possibly an error in your equation where you meant $p_i$ corresponding to the i-th robot).

$||q - p_N||^2 =$ The squared Euclidean distance between a point q in the coverage area and the position of the N-th robot, representing how well this point q is covered by robot N.

$\varphi(q, t) =$ A density function or a distribution over the coverage area, possibly representing the importance or priority of covering point q at time t.

The coverage function is also one specific type of objective function. It is widely used in problems where primary focus is on measuring and optimizing coverage quality. The coverage function is shown below (Yang et al., 2023),

$$H(x; \varphi(q)) = \sum_{i=1}^{n} \int_{V_i\,(x)} \|q - C\,(V_i\,(x))\|^2\, \varphi(q)\, dq$$

(3.11)

where,

x = set of agent positions

$\varphi(q)$ = density function

$C\,(V_i\,(x)$ = centroid of Voronoi cell

When the density function is large value and multi-robot positions are distributed in this value, it will be corresponding that position q will be very small and this will cause coverage function be small as well. It can be concluded that when distribution of robot is closet to density function, coverage function is globally minimized (Yang et al., 2023).

### 3.2.6 Lloyd’s Algorithm

Lloyd’s algorithm can be illustrated as,

$$x_i^L(k+1) = C\,(V_i\,(x(k)^L))$$

(3.12)

where,

$x_i^L(k)$ = position of ith agent after kth iteration

As outlined in Chapter 2, Lloyd's algorithm is employed to distribute a set of robots within a predefined bounded space to accomplish a specific coverage objective. In this project, Lloyd's algorithm leverages a density function constructed from the Gaussian Distribution Model (3.9) to compute the density of the space surrounding each robot's position. Using the calculated density function, equations 3.3 and 3.4 are employed to update the weighted centroid. The weighted centroid denotes the centre of mass of the space. Lloyd's algorithm shares a similar concept with the k-means clustering method, utilizing iterations to bring robots closer to the weighted centroid or their respectively goal positions (Du, Emelianenko, and Ju, 2006). After a specified number of iterations, robots will reach their respective goals and accomplish complete static coverage. Employing a small step or gain helps reduce the differential in the initial distribution of robots and mitigates local optimization issues. While more advanced versions of Lloyd's algorithm have been proposed by other authors, as depicted in Appendix C, for the purposes of this project simulation, traditional Lloyd's algorithm is chosen over modified version due to its simplicity. The concept of Lloyd’s algorithm is based on Appendix A, B and C.

### 3.2.7 Velocity Obstacle Technique

Velocity Obstacle Technique (VO) is used to dynamic obstacle handling in multi-robot coverage. From the Lloyd’s algorithm, it is clearly shown that there is no collision avoidance in the algorithm, therefore velocity obstacle technique is implemented with Llyod’s algorithm to create a coverage algorithm with dynamic obstacle avoidance.

The kinematics of nonholonomic multi-robot systems are shown below,

$$\dot{x}(t) = u\cos\left(\psi(t)\right) \tag{3.13}$$

$$\dot{y}(t) = u\sin\left(\psi(t)\right) \tag{3.14}$$

$$\dot{\psi}(t) = r(t) = u\frac{\tan u}{L} \tag{3.15}$$

where,

u = forward speed of robots

$\psi(t), r(t)$ = heading and heading rate

The position of the robot in plane, $\mathrm{p(t)} = [\mathrm{x(t)}, \mathrm{y(t)}]^T$

The velocity of robot in plane, $\mathrm{v(t)} = [\dot{x}(\mathrm{t}), \dot{y}(\mathrm{t})]^T$

The kinematics of nonholonomic dynamic obstacles are shown below,

$$\dot{x}_o(t) = u_o\cos\left(\psi_o(t)\right) \tag{3.16}$$

$$\dot{y}_o(t) = u_o \sin(\psi_o(t))$$

(3.17)

$$\dot{\psi}_o(t) = r_o(t) = u_o \frac{\tan u_o}{L}$$

(3.18)

where,

$u_o$ = forward speed of obstacle

$\psi_o(t), r_o(t)$ = heading and heading rate

The position of the robot in plane, $p_o(\mathrm{t}) = [x_o(\mathrm{t}), y(\mathrm{t})]^T$

The velocity of robot in plane, $v_o(\mathrm{t}) = [\dot{x_o}(\mathrm{t}), \dot{y}_o(\mathrm{t})]^T$

Assumed the forward speed of robots and dynamic obstacles are constant, where u > 0 and the heading rate is controlled and bounded by $|r(t)| \leq r_{max}$ (Haraldsen,Wiig and Pettersen, 2020). The objective of dynamic obstacle avoidance is making the robots have an acceptable distance with obstacles to avoid collisions. The distance can be called as safety distance. By assumed that the velocity obstacle for robot A induced by dynamic obstacles B, is known as $VO_{A|B}$. $VO_{A|B}$ is set of velocities that robot A cannot take, if it taken velocities from the set, it will result in collision with B. The Minkowski sum is generalized by expressing $p_A$ *and* $p_B$ as the centre of A and B and $\beta$ is disc centred at $p_B$ which radius is equal to the sum of radius A and B. If obstacle B is static, there is a collision cone, C (Wilkie,Van Den Berg and Manocha, 2009). However, to simplify and generalized the simulation, cone C is translated by velocity of obstacle B, $v_B$. The velocity obstacle equation is,

$$VO_{A|B} = \{ v \mid \exists t > 0 :: p_A + t(v - v_B) \in \beta$$

(3.19)

The concept of velocity obstacle for kinematically constrained robot, velocity obstacle is achieved over a time interval. However this method cannot guarantee is collision free because robots will be at different positions when they achieved the selected velocities but the velocity obstacle is based on initial position of robots to implement (Wilkie,Van Den Berg and Manocha, 2009). To address this issue, the time interval can be decrease to a small value nut this may results in robots miss the feasible controls. Therefore, generalized velocity obstacle is used and the obstacle in control space is defined as,

$$\{ u \,|\exists t > 0 :: \|A\ (t,u) - B(t)\| < \ r_A + \ r_B\} \tag{3.20}$$

where,
u = set of inputs to the Kino dynamic model

$$t_{\min}(u) = \arg \min_{t>0} \ || \ A(t,u) - B(t) \ || \tag{3.21}$$

where,
$t_{\min}(u)$ = time at which the distance between centre of robot and obstacle is at its minimum for a given control input, u

A finite time horizon is employed to restrict the application of the velocity obstacle to potential collisions that may occur within a specified time frame. The time horizon is $t \ \in [0, t_{lim}]$ (Wilkie,Van Den Berg and Manocha, 2009). The Velocity Obstacle algorithm can refer to Appendix D. The u* is the control for robot if the situation is without any moving obstacles. u* can be known as preferred control. Preferred control can be another algorithm. If robot face dynamic obstacles, it will switch to velocity obstacle control which is u. The concept of VO is based on Appendix D.

### 3.2.8 Dynamic Obstacle Setup

A dynamic obstacle is introduced in the MATLAB simulation. In order to simply the simulation and obtain observable results, only one dynamic obstacle is included, and its velocity and size are fixed. Additionally, the initial position of the dynamic obstacle remains constant rather than being randomized at each start of the simulation. The trajectory of the dynamic obstacle's movement is represented by a fixed function, as shown below,

$$x(t) = x_0 + v_x \times t \tag{3.22}$$

$$y(t) = y_0 + v_y \times t \tag{3.23}$$

Where,

$x_0 \; and \; y_0$ = initial coordinates of the obstacle

$v_x$ and $v_y$ = velocities of obstacle in x and y directions respectively

The trajectory function incorporates random noise to enable the dynamic obstacle to move more aggressively. The random noise is controlled by the aggressiveness of dynamic obstacle which range from 1 to 10. This allows to test whether the robots can effectively avoid collisions when the dynamic obstacle is moving aggressively compared to when it moves non-aggressively. If the dynamic obstacle moves out of the Bound space, it will stop moving.

$$p_{dynamic}(t) = p_{initial} + v \times t + n \tag{3.24}$$

Where,

$p_{initial}$ = initial position vector of the dynamic obstacle.

$v$ = velocity vector of the dynamic obstacle (constant speed).

$t$ = time variable

$n$ = noise vector (random noise x aggressiveness)

### 3.2.9 Flowchart of Modified Lloyd's Algorithm with Velocity Obstacle

Figure 3.1 illustrates the flowchart for Modified Lloyd's Algorithm with Velocity Obstacle.

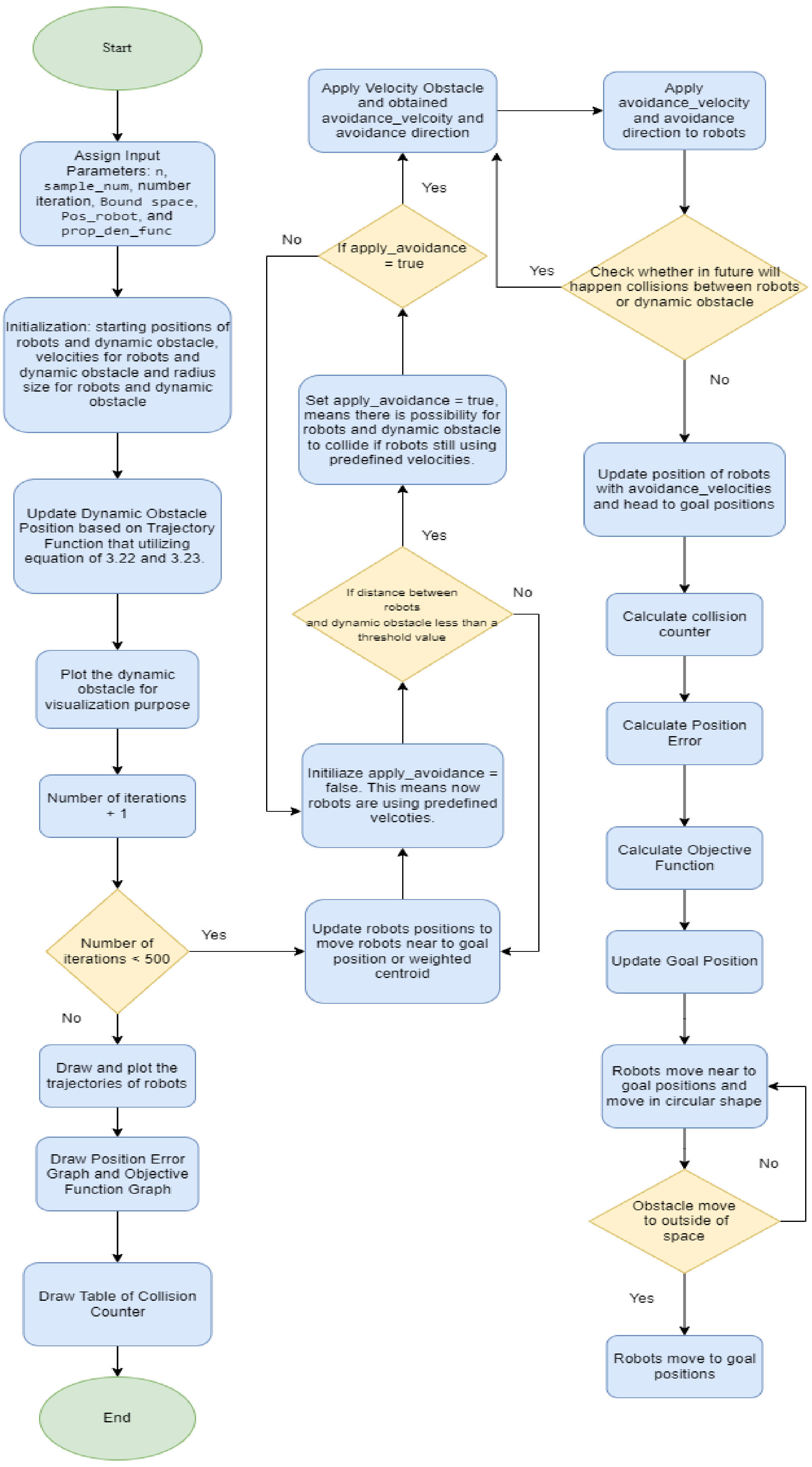

Start
Assign Input Parameters: n, sample_num, number iteration, Bound space, Pos_robot, and prop_den_func
Initialization: starting positions of robots and dynamic obstacle, velocities for robots and dynamic obstacle and radius size for robots and dynamic obstacle
Update Dynamic Obstacle Position based on Trajectory Function that utilizing equation of 3.22 and 3.23.
Plot the dynamic obstacle for visualization purpose
Number of iterations + 1
Number of iterations < 500
Yes
No
Draw and plot the trajectories of robots
Draw Position Error Graph and Objective Function Graph
Draw Table of Collision Counter
End
Update robots positions to move robots near to goal position or weighted centroid
Initiliaze apply_avoidance = false. This means now robots are using predefined velcoties.
If distance between robots and dynamic obstacle less than a threshold value
Yes
No
Set apply_avoidance = true, means there is possibility for robots and dynamic obstacle to collide if robots still using predefined velocities.
If apply_avoidance = true
Yes
No
Apply Velocity Obstacle and obtained avoidance_velcoity and avoidance direction
Apply avoidance_velocity and avoidance direction to robots
Check whether in future will happen collisions between robots or dynamic obstacle
Yes
No
Update position of robots with avoidance_velocities and head to goal positions
Calculate collision counter
Calculate Position Error
Calculate Objective Function
Update Goal Position
Robots move near to goal positions and move in circular shape
Obstacle move to outside of space
No
Yes
Robots move to goal positions

Figure 3.1: Flowchart of Modified Lloyd’s Algorithm with Velocity Obstacle.

## 3.3 Computer Simulation

The dynamic obstacle handling in multi-robot coverage will be tested and evaluated in simulation. Both the existing Lloyd’s algorithm and the Modified Lloyd’s Algorithm with Velocity Obstacle (VO) will be implemented and compared in terms of the average total number of collisions between robots and the dynamic obstacle. The setup of environment is shown in Figure 3.2, where initial positions of robots and dynamic obstacle are fixed. The Modified Lloyd’s Algorithm with VO will also be evaluated in two different scenarios to assess its functionality. In the first scenario, aggressiveness of dynamic obstacle will be varied. Figure 3.3 illustrates the setup environment of first scenario. The second scenario involves positioning the dynamic obstacle at different initial positions within the environment. Figure 3.4 display the setup of second scenario. These simulations will be conducted in MATLAB R2021a on Windows Laptop with AMD Ryzen 7 5800H with Radeon Graphics@ 3.20 GHz processor with 16GB RAM. During these simulations, each robot will consider other robots in same environment as dynamic obstacle. In order to simplify the simulation, some assumptions are to be made. The shape and size of the robots will be denoted as circle while the goal of robots will be represented by a cross. The simulation is conducted in a 2-dimensional workspace. The core simulation technique employed will be the traditional Lloyd’s algorithm coupled with the velocity obstacle approach. If necessary, implementing a modified version of the Lloyd’s algorithm to enhance performance will be considered. The simulation pseudocode can refer Appendix E.

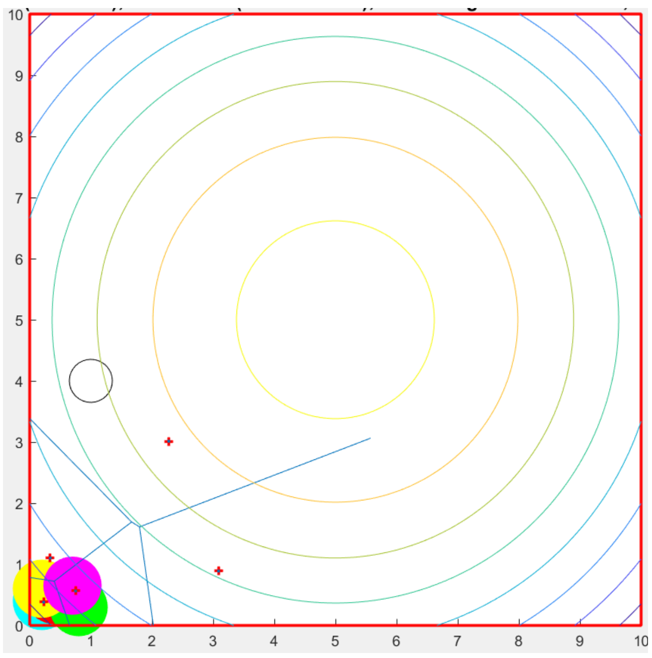


Figure 3.2: Setup Environment where the Starting Points of Robots and Dynamic Obstacle are Fixed.

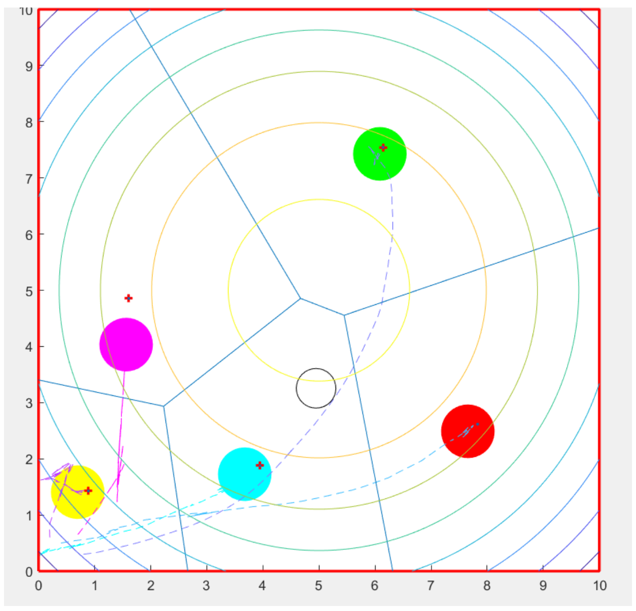


Figure 3.3: Setup Environment in Scenario 1.

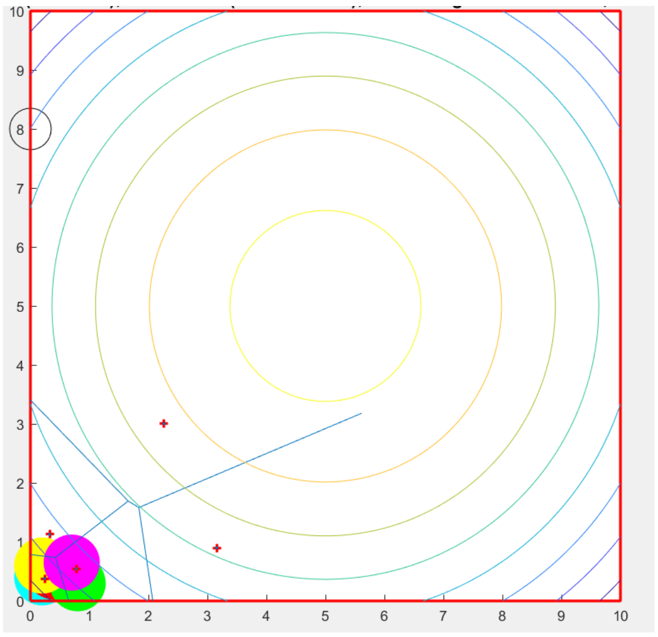


Figure 3.4: Setup Environment in Scenario 2.

## 3.4 Planning and Managing of Project Activities

In this section, the whole planning and managing of project activities will be described. The planning project activities will be represented by Gantt Charts as shown in Table 3.1. This project will be carried out through two semesters; therefore, the project activities will be divided into two parts. The duration of project is total 28 weeks for both part 1 and part 2.

### 3.4.1 Project Part 1

The part 1 of the project is conducted in total 14 weeks. There are five main objectives needed to be accomplished as shown in Table 3.1. The five main objectives are Project Planning, Literature Review, Methodology and Work Plan, Progress Report Writing, and Presentation Slide Preparation. The task first starts with Project Planning which primarily focus project title decision, crafting problem statement, defining aim and objectives and engaging in continuous discussions and meetings with supervisor over a 12-week period. This to ensure to have a better understanding and knowledge about the topic of this project. The second objective, Literature Review, involved an exploration of multi-robot coverage algorithms, dynamic obstacle avoidance techniques, Gaussian distribution, and the construction of a comparison table to enhance the project's knowledge base. The Methodology and Work Plan are the third objective which take approximately six weeks. This involves of research on Lloyd’s Algorithm, Velocity Obstacle, dynamic obstacle setup and flowchart constructing to guide the project algorithm implementation. Next, the four objective is Report Writing. Report writing take over twelve-week period while report checking takes two weeks to ensure good quality of report. Lastly, Presentation Slide Preparation. Presentation slide preparation and checking also encompassed a two-week period. This is to ensure the correctness of presentation slide in format and visual appeal.

### 3.4.2 Project Part 2

In contrast, part 2 of the project encompasses a more extensive scope, comprising five major components that surpass those in part 1. The first critical component involves the Development and Deployment of Modified Lloyd’s Algorithm with Velocity Obstacle for coverage with dynamic obstacle avoidance. This involves coding of Lloyd’s Algorithm, Velocity Obstacle. Additional Feature such as avoidance angle and future position prediction also included during the development. Next, the second part will be Results and Discussion which obtained from MATLAB simulations. These simulations serve the purpose of determining the functionality and stability of the developed algorithms, eventually contributing to the construction of a comprehensive model. The modified algorithm will be compared with existing ones and tested under two different scenarios. Additionally, the relationship between parameters in simulation results will be studied. This analysis phase is expected to take around three weeks and will guide the selection of optimal parameters for refining the algorithms. The third component involves Poster Preparation which takes around three weeks. The fourth element is Final Report Writing which scheduled from Week 3 to Week 14. Report checking will be completed within two weeks to ensure its quality. The fifth component is preparation of presentation slide. Presentation slide preparation based on the content of report and will undergo checking to ensure presentation slide is well prepared.

Table 3.1: Gantt Chart for Project Activities Part 1.

| No. | TASK TITLE | BEGIN DATE | END DATE | JUNE SEMESTER (FYP1) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **1** | **Progress Flow** | | | | | | | | | | | | | | | | |
| **1** | **Project Planning** | | | | | | | | | | | | | | | | |
| 1.1 | Project Title | 25/6/2023 | 27/6/2023 | | 🟩 | | | | | | | | | | | | |
| 1.2 | Crafting Problem Statement, Aims and Objectives | 28/6/2023 | 9/7/2023 | | 🟩 | 🟩 | | | | | | | | | | | |
| 1.3 | Discussion and Meeting with Supervisor | 25/6/2023 | 17/9/2023 | | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| **2** | **Literature Review** | | | | | | | | | | | | | | | | |
| 2.1 | Multi-robot Coverage Control Algorithm | 28/6/2023 | 21/7/2023 | | 🟩 | 🟩 | 🟩 | 🟩 | | | | | | | | | |
| 2.2 | Dynamic Osbatcle Avoidance Techniques | 21/7/2023 | 11/8/2023 | | | | | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | | | | | |
| 2.3 | Gaussian Distribution | 11/8/2023 | 15/8/2023 | | | | | | | | | | 🟩 | | | | |
| 2.4 | Compare algorithms in Table to do selection | 11/8/2023 | 15/8/2023 | | | | | | | | | | 🟩 | | | | |
| **3** | **Methodology and Work Plan** | | | | | | | | | | | | | | | | |
| 3.1 | Research on Lloyd's Algorithm | 23/7/2023 | 30/7/2023 | | | | | | 🟩 | 🟩 | | | | | | | |
| 3.2 | Research on Velocity Obstacle | 2/8/2023 | 15/8/2023 | | | | | | | 🟩 | 🟩 | 🟩 | | | | | |
| 3.3 | Dynamic Obstacle Setup | 16/8/2023 | 20/8/2023 | | | | | | | | | 🟩 | 🟩 | | | | |
| 3.4 | Flowchart of Modified Lloyd's Algorithm with Velocity Obstacle | 21/8/2023 | 31/8/2023 | | | | | | | | | | | 🟩 | | | |
| **4** | **Progress Report Writing** | | | | | | | | | | | | | | | | |
| 4.1 | Report Writing | 3/7/2023 | 17/9/2023 | | | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 4.2 | Report Checking | 13/9/2023 | 17/9/2023 | | | | | | | | | | | | 🟩 | 🟩 | 🟩 |
| **5** | **Presentation Slide** | | | | | | | | | | | | | | | | |
| 5.1 | Preparation Slide | 12/9/2023 | 19/9/2023 | | | | | | | | | | | | | 🟩 | 🟩 |
| 5.2 | Slide Checking | 19/9/2023 | 20/9/2023 | | | | | | | | | | | | | 🟩 | 🟩 |

Table 3.2: Gantt Chart for Project Activities Part 2.

| No. | TASK TITLE | BEGIN DATE | END DATE | JANUARY SEMESTER (FYP2) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| **1** | **Progress Flow** | | | | | | | | | | | | | | | | |
| **1** | **Modified Lloyd'sAlgorithm with Velocity Obstacle Development** | | | | | | | | | | | | | | | | |
| 1.1 | Coding Lloyd's Algorithm | 29/1/2024 | 19/2/2024 | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | | | | |
| 1.2 | Coding Velocity Obstacle | 19/2/2024 | 11/3/2024 | | | | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | |
| 1.3 | Adding Additional Features | 11/3/2024 | 18/3/2024 | | | | | | | 🟦 | 🟦 | | | | | | |
| **2** | **Results and Discussions** | | | | | | | | | | | | | | | | |
| 2.1 | Compare Existing with Modified Algorithm | 25/3/2024 | 31/3/2024 | | | | | | | | | 🟦 | | | | | |
| 2.2 | Test Modified Algorithm under Two Scenarios | 1/4/2024 | 7/4/2024 | | | | | | | | | | 🟦 | | | | |
| 2.3 | Study Relationship between Algorithm Parameters with Results | 8/4/2024 | 14/4/2024 | | | | | | | | | | | 🟦 | | | |
| **3** | **Poster** | | | | | | | | | | | | | | | | |
| 3.1 | Poster Preparation | 25/3/2024 | 6/4/2024 | | | | | | | | | 🟦 | 🟦 | 🟦 | | | |
| **4** | **Final Report Writing** | | | | | | | | | | | | | | | | |
| 4.1 | Report Writing | 12/2/2024 | 28/4/2024 | | | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 | 🟦 |
| 4.2 | Report Checking | 18/4/2024 | 28/4/2024 | | | | | | | | | | | | | 🟦 | 🟦 |
| **5** | **Presentation Slide** | | | | | | | | | | | | | | | | |
| 5.1 | Preparation Slide | 28/4/2024 | 30/4/2024 | | | | | | | | | | | | | 🟦 | 🟦 |
| 5.3 | Slide Checking | 30/4/2024 | 1/5/2024 | | | | | | | | | | | | | | 🟦 |

## 3.5 Summary

This chapter elaborates on the methodology adopted, involving the utilization of both the Lloyd's algorithm and Velocity Obstacles (VO). The initial implementation will focus on the conventional Lloyd's algorithm to execute coverage tasks, streamlining the simulation process. Subsequently, Velocity Obstacles will be integrated with the Lloyd's algorithm to empower the robots with obstacle avoidance capabilities. The deployment of multi-robot systems is achieved through a density function approach, employing Gaussian distribution principles.

These simulations will be conducted within the MATLAB environment, employing distinct parameters. This encompasses variations in the aggressiveness and starting position of dynamic obstacle, the existing and proposed algorithm in terms of collision avoidance. The evaluation of simulation performance will entail a comparative analysis between these different scenarios.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Introduction

This chapter focuses on presenting and discussing the results obtained from the MATLAB simulations using the Modified Lloyd's Algorithm with Velocity Obstacle (VO). This chapter involves comparing the results between the existing Lloyd's algorithm without VO and Modified Lloyd's Algorithm with VO in terms of average total number of collisions between multi-robot system and dynamic obstacle during coverage task. The comparison results will be presented in Section 4.2. Additionally, it will also discuss the Modified Lloyd's Algorithm with VO in two different scenarios which are first scenario is varying aggressiveness of movement of dynamic obstacle and the second scenario is different starting position of dynamic obstacle. This discussion will be presented in Section 4.3. Furthermore, the impact of Modified Lloyd's Algorithm with VO parameters on the simulation results in Section 4.4.

## 4.2 Comparison Between Existing and Modified Algorithms

In this section, the results obtained from the existing Lloyd's algorithm without dynamic obstacle handling and Modified Lloyd's Algorithm with Velocity Obstacle (VO) in terms of total average number of collisions are presented. Both algorithms were implemented using centroid Voronoi tessellation during the navigation of the robots, ensuring that the robots move towards their goal positions during the coverage task.

The existing Lloyd's algorithm was implemented without any dynamic obstacle handling. In this algorithm, the parameters such as number of robots, number of samples, number of iterations, boundary of workspace, velocity of robots, size of robots and initial position of robots were initialized. Additionally, the other parameters such as gain, number of Gaussian were also initialized, while default values were used for others. After initialization, the dynamic obstacle was set up. A dynamic obstacle was introduced with a fixed initial position and velocity and its trajectory is defined by a function which is shown in equation 3.22 and 3.23. During each iteration, the centroid of the

Voronoi was calculated based on the equation 3.3 and 3.4. The density function of the Bound space was calculated by the equation 3.9. During each iteration, the positions of robots were updated to move closer to the centroid of the Voronoi diagram based on the density distribution, allowing the robots to reach their respective goals. In this algorithm, no dynamic obstacle avoidance technique is introduced. The dynamic obstacle moves along its predefined trajectory while the robots move towards their goals during the coverage task. A collision counter graph was constructed to calculate the number of collisions that occur during the coverage task. The collision counter calculated collisions by setting a threshold; if the difference between the radii of the robots and the dynamic obstacle is lower than the threshold, it indicates a collision, and the collision counter records the collision and the iteration in which it occurred. On the other hand, Modified Lloyd's Algorithm with VO, allowing it to handle the dynamic obstacle during the coverage task. Similar to the existing Lloyd's algorithm, the parameters were all initialized. However, in this modified algorithm, the positions of robots do not move directly towards their goals or the centroid of the Voronoi diagram. This is because if robots were to directly avoid the dynamic obstacle and reach their goals, theoretically, they would remain stationary at their goal positions once they are reached. However, if the iterations are not yet finished or other robots have not reached their goals, the dynamic obstacle may still be moving and could collide with the stationary robots that have reached their goals. Therefore, instead of moving directly towards their goals, the robots now move in a circular shape around the goals during each iteration, while the dynamic obstacle remains within the bounds of the workspace. Once the iteration is nearly complete or the dynamic obstacle moves outside of the bounds of the workspace, the robots move towards their goals as usual instead of moving in a circular shape. Velocity obstacle is introduced to enable robots to avoid the dynamic obstacle during the coverage task.

In order to compare the performance of both algorithms in terms of handling dynamic obstacle which known as dynamic obstacle avoidance, each algorithm was simulated five times. They were evaluated based on average total number of collision number throughout the entire iteration. Both algorithms will be using same parameters as shown in Table 4.1. Besides that, in all scenarios, each robot was assigned a unique colour visualization, as illustrated in Table 4.2.

Table 4.1: Simulation Parameters for Both Algorithms.

| **Parameters** | **Value** |
|---|---|
| Bound space | [0,0 ; 0,10 ; 10,10 ; 10,0 ] |
| Initial position of robots | [0.2,0.4 ; 0.4,0.5 ; 0.8,0.3 ; 0.2,0.6 ; 0.7,0.65] |
| Robot radius | 1 |
| Robot velocity | 1.5 |
| Obstacle radius | 0.5 |
| Obstacle velocity | 0.05 |
| Obstacle initial position | [1, 4] |
| Gain | 0.4 |
| Iteration | 500 |
| Aggressiveness | 1 |
| Safety Margin | 5 |

Table 4.2: Colour of each Robot.

| **Robots** | **Colour** |
|---|---|
| Robot 1 | Cyan |
| Robot 2 | Red |
| Robot 3 | Green |
| Robot 4 | Yellow |
| Robot 5 | Magenta |

### 4.2.1 Existing Lloyd’s Algorithm Results without Velocity Obstacle

In this simulation, each robot started from different starting positions as indicated in Table 4.1, with objective of reaching their respective goal positions or centroid of the Voronoi diagram. Figure 4.1 illustrates the final positions of robots and their corresponding goals, and the position of dynamic obstacle after the iteration ends during the first simulation run. The visualization results align with the concept of Lloyd’s algorithm, as proposed by Breitenmoser et al. in 2010. This algorithm involves three steps which are constructing the Voronoi partition to generate points for the goals of multi-robot systems, computing the centroids of the Voronoi regions, and assigning new locations to the centroids. This process is repeated until Voronoi coverage, based on Lloyd's algorithm, is completed (Breitenmoser et al., 2010).

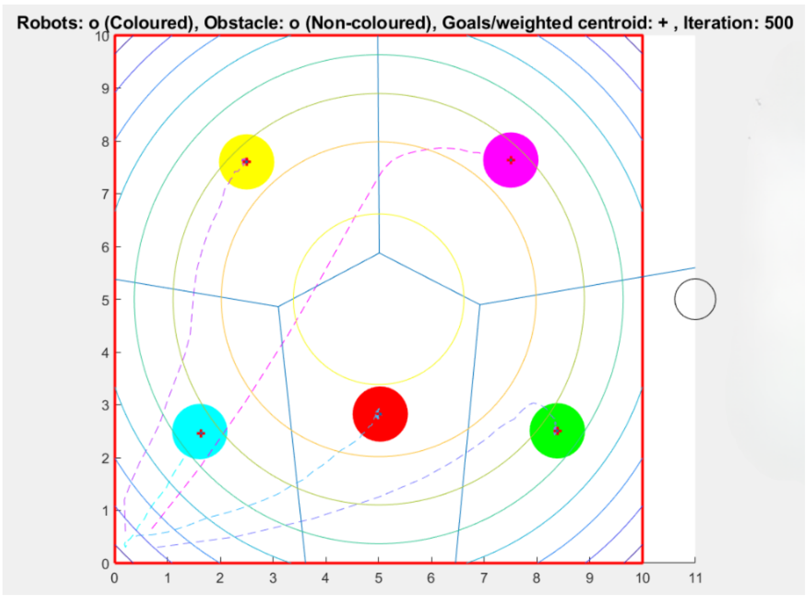


Figure 4.1: Final Positions of Robots and their Corresponding Goals, and the Position of Dynamic Obstacle after Iteration ends during the First Simulation Run.

Figure 4.2 illustrates the density function curve obtained from this simulation. The x and y axes represent the spatial coordinates in the environment, while the z-axis represents the density function value of each point in the environment. This surface plot displays the density function as a continuous surface. The colour in the density function curve indicates the density value of each point. Warmer colours like red indicate higher density, suggesting areas of greater importance or priority for coverage. Cooler colours, such as blue, indicate lower density values. The colour gradient provides a visual representation of the variation in density across different regions of the environment. Areas with warmer colours suggest a higher concentration or density of robots, obstacles, or features, while cooler colours indicate lower density.

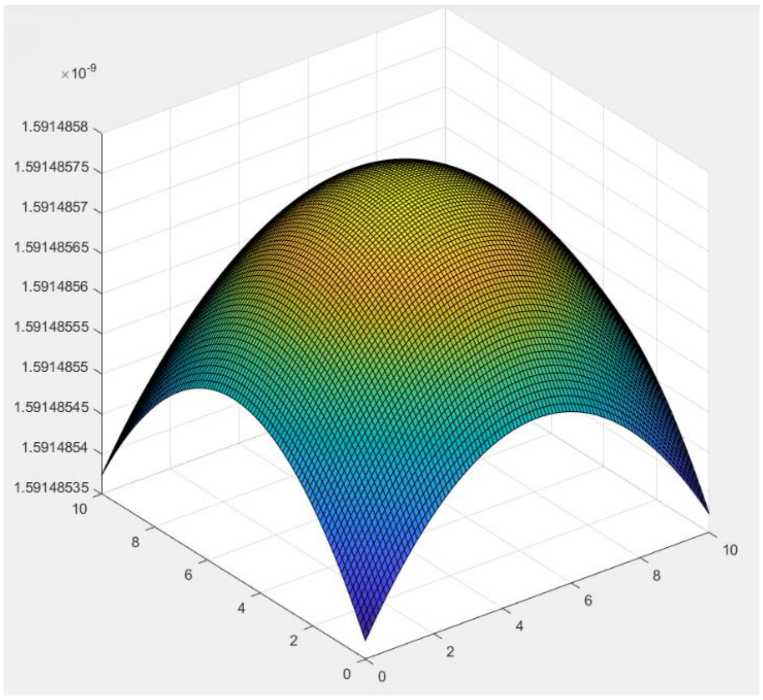


Figure 4.2: Density Function Curve in Existing Lloyd’s Algorithm.

Figure 4.3 shows the position error graph. Position error graph depicts the difference in distance between robots and their respective goals. A higher value on the position error graph indicates that the average difference in distance between the five robots and their respective goals is greater. This suggests that the robots have not yet reached their goals, and the coverage task is incomplete. Conversely, a position error graph showing a value of zero indicates that the average difference in distance between the robots and their goals is zero, signifying that all robots have successfully reached their goals and the coverage task is complete. Figure 4.3 shows that the position error graph reached zero value which indicates that all five robots had reached their corresponding goal positions. Figure 4.4 shows the objective function graph. Objective function represents overall coverage error of the robots’ positions relative to their respective goals. Figure 4.4 demonstrates that the objective function drops from higher value to a lower value, followed by a relatively constant trend once a certain value is reached. This trend suggests that the robots are maintaining relatively stable performance in terms of achieving their goals and covering the area of interest. There is several info can be obtained from the objective function. If the objective function remains almost constant, it indicates that the robots have reached a stable area coverage. The robots are consistently maintaining their positions to cover the entire environment of interest. Additionally, it also indicates that consistent goal achievement has been achieved. This suggests that the robots are consistently reaching their desired goals (centroid of Voronoi diagram) and maintaining their positions over time, demonstrating the effectiveness of the coverage control algorithm in guiding the robots towards their goals.

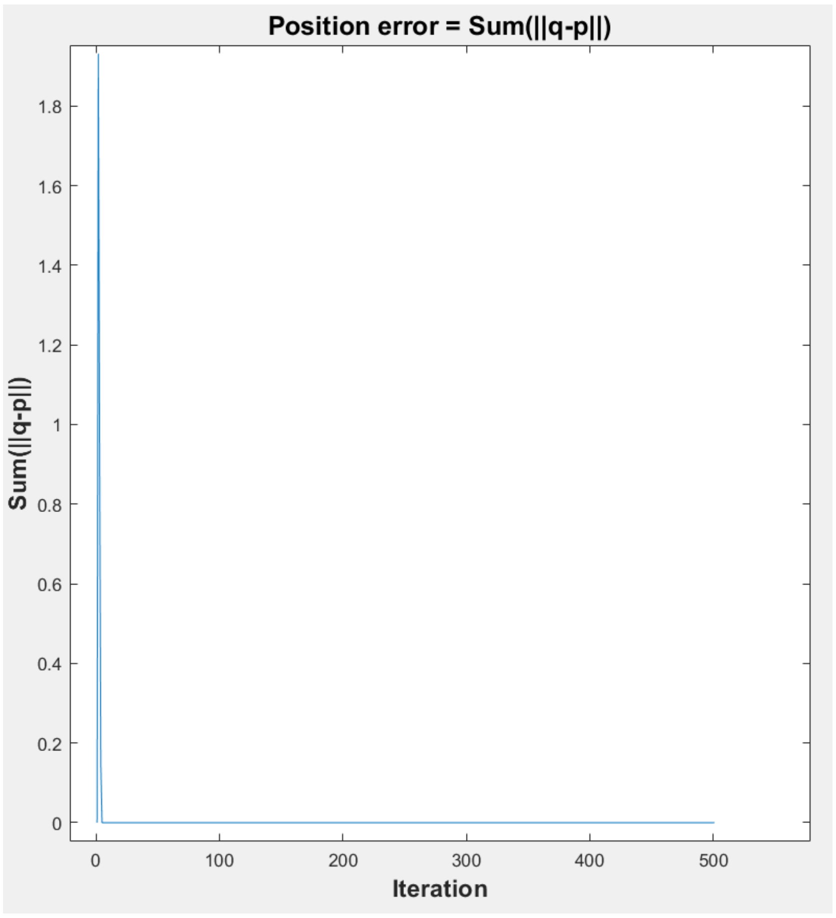


Figure 4.3: Position Error Graph of Existing Lloyd’s Algorithm.

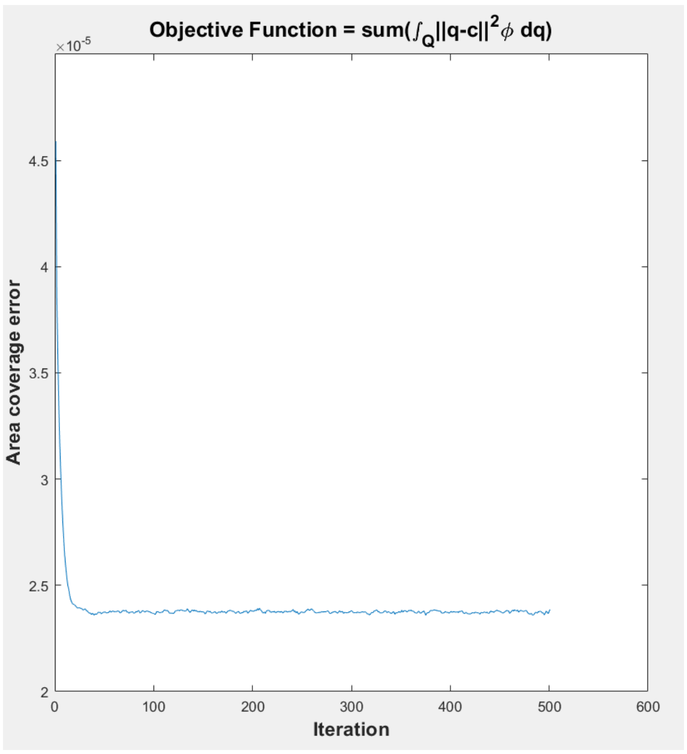


Figure 4.4: Objective Function Graph of Existing Lloyd’s Algorithm.

Figure 4.5 shows that the collision counter graph from the first run simulation. It is obvious from the Figure 4.5 that Robot 2, 3 and 4 had collided with dynamic obstacle during the iteration of coverage task. Table 4.3 presents the average total number of collisions between five deployed robots and dynamic obstacle during the iteration. ‘Sim’ in table represents ‘Simulation’.

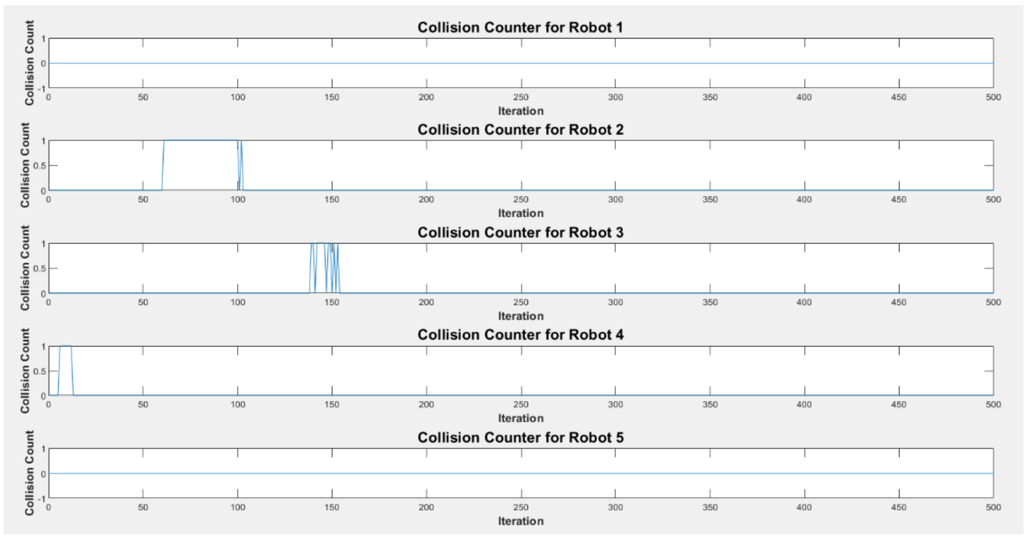


Figure 4.5: Total Number of Collisions between Five Robots and Dynamic Obstacle during Iteration in First Run Simulations in Existing Lloyd’s Algorithm.

Table 4.3: Average Total Number of Collisions between Five Robots and Dynamic Obstacle during Iteration in First Run Simulation in Existing Lloyd’s Algorithm.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 41 | 39 | 37 | 41 | 37 | 39 |
| Robot3 | 11 | 7 | 8 | 4 | 2 | $6.4 \approx 6$ |
| Robot4 | 7 | 7 | 7 | 8 | 7 | $7.2 \approx 7$ |
| Robot5 | 0 | 0 | 0 | 0 | 0 | 0 |

It is obvious that using the existing Lloyd’s algorithm enables the complete and successful execution of the multirobot coverage task, as indicated by a position error of zero, suggesting that all robots have reached their respective goals. Additionally, the relatively constant objective function implies that the robots maintain stable performance in achieving their goals and covering the area of interest. This comprehensive coverage is further confirmed by the visualization of the simulation in Figure 4.1. However, without Velocity Obstacle implementation, existing Lloyd’s algorithm fails to handle dynamic obstacles during the coverage task. Consequently, robots collide with the dynamic obstacle, posing a risk of damage and interruption to their tasks in real-world scenarios. In this simulation, only Robots 2, 3 and 4 collide with dynamic obstacle, with the average total number of collisions of 39, 6 and 7 respectively. This outcome arises because the dynamic obstacle's starting position is [1, 4], and its trajectory lies along the x-axis, intersecting the paths of Robots 2, 3, and 4, leading to collisions. The variability in collision numbers is attributed to the random noise introduced into the dynamic obstacle's trajectory, causing it to move unpredictably along its defined path.

### 4.2.2 Modified Lloyd’s Algorithm Results with Velocity Obstacle

In this simulation, Lloyd’s algorithm is configured with the same settings as described in section 4.2.1, and all other parameters followed those shown in Table 4.1 to ensure a fair comparison. However, in this iteration, the algorithm is enhanced with Velocity Obstacle implementation to handle dynamic obstacles. Velocity Obstacle enables robots to dynamically avoid obstacles during the coverage task.

Figure 4.6 illustrates the final positions of robots and dynamic obstacle after the iteration in this modified algorithm for this first run simulation, while Figure 4.7 shows the density function curve obtained from this simulation. A notable observation is the less orderly trajectory of robots in Figure 4.6 compared to Figure 4.1. This difference arises because, in Figure 4.1, robots move directly to their goal positions, whereas in Figure 4.6, robots move around their goal positions instead of directly advancing towards them to continuously navigate around potential obstacles identified by the Velocity Obstacle method. By comparing Figure 4.1 and 4.6, it is obvious that different robots cover different regions as indicated by the colours of the robots. This discrepancy occurs because, while some robots are avoiding the dynamic obstacle, others, unaffected by the obstacle, proceed to cover their designated regions first. Consequently, the density distribution varies, leading to different robots covering different regions. However, the overall shape and size of the covered area remain similar between Figures 4.1 and 4.6. Figure 4.7 displays a density function curve identical to Figure 4.2, as all settings and parameters remain consistent. Consequently, the density curve exhibits the same pattern in both simulations.

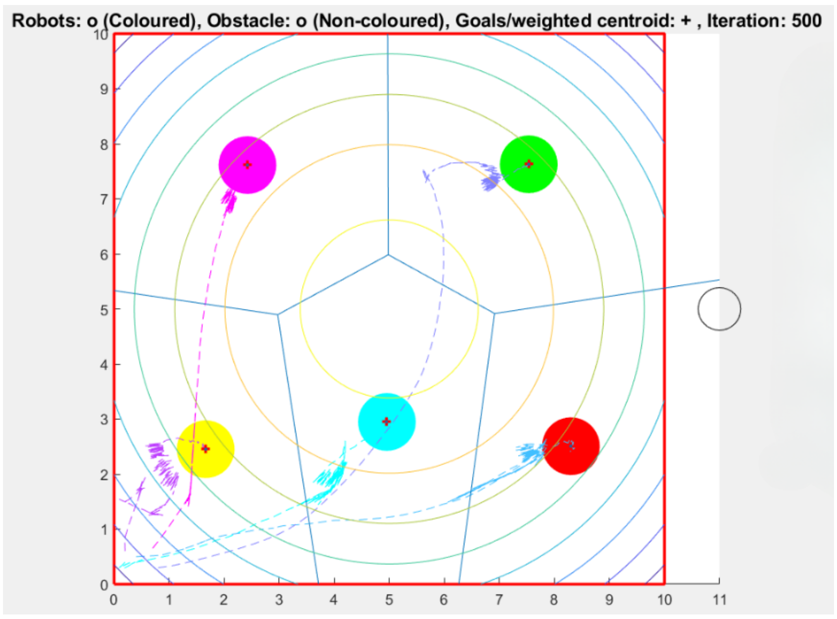


Figure 4.6: Final Positions of Robots and Dynamic Obstacle in Modified Lloyd's Algorithm with Velocity Obstacle during Iteration in First Run Simulation.

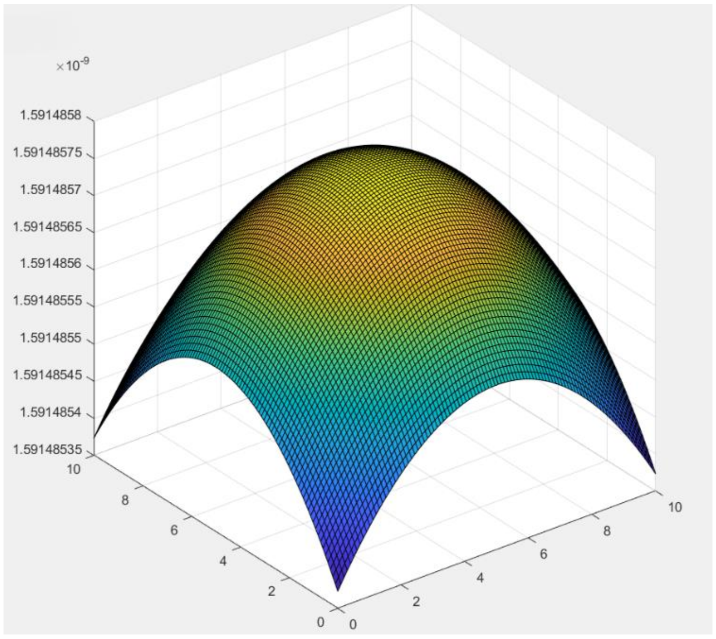


Figure 4.7: Density Function Curve in Modified Lloyd's Algorithm with Velocity Obstacle during Iteration in First Run Simulation.

Figure 4.8 illustrates the position error graph while Figure 4.9 shows the objective function graph during the first-time simulation. In Figure 4.8, it is evident that the position error graph does not reach zero but fluctuates before iteration 180, indicating continuous variations in the average distance between robots and the dynamic obstacle. This fluctuation arises because, before iteration 180, the robots navigate around their goal positions instead of moving directly towards them. Consequently, the average distance between robots and the dynamic obstacle varies as the robots adjust their trajectories. However, after iteration 180, the dynamic obstacle moves outside the Bound space, removing the obstacle from the robots' coverage environment. Consequently, the robots proceed to move directly towards their respective goal positions, resulting in the position error graph reaching zero. Figure 4.9 shows the objective function obtained from the modified algorithm. Unlike the smooth curve observed in Figure 4.4, the objective function graph in Figure 4.9 exhibits fluctuations before iteration 180. This irregularity is attributed to the dynamic obstacle's presence within the coverage environment, necessitating continuous Velocity Obstacle manoeuvres by the robots to avoid collisions. Additionally, before iteration 180, the robots cannot reach their goal positions directly but instead move in circular trajectories around them to maintain avoidance manoeuvres. However, after iteration 180, the objective function remains almost constant and smooth, indicating that the robots have achieved complete coverage of the area of interest.

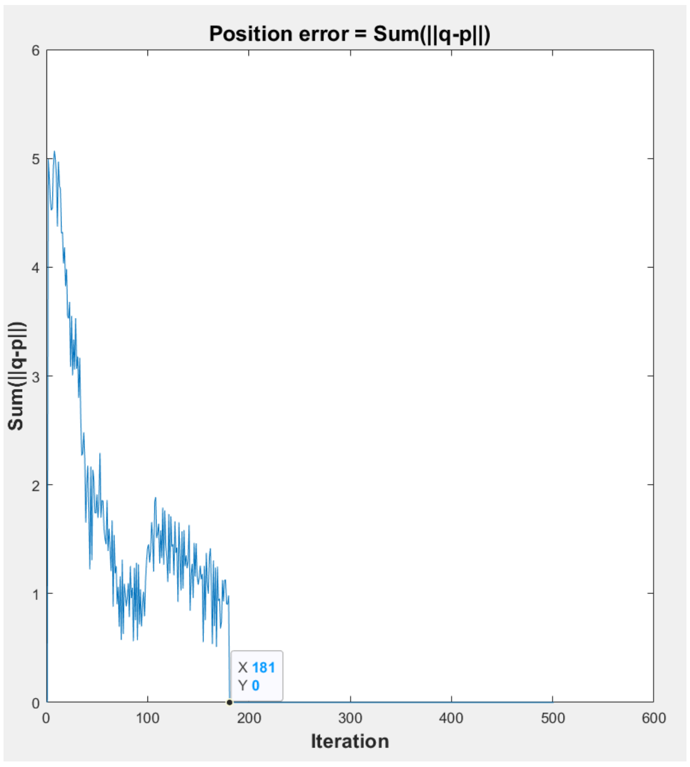


Figure 4.8: Position Error Graph of Modified Lloyd's Algorithm with Velocity Obstacle.

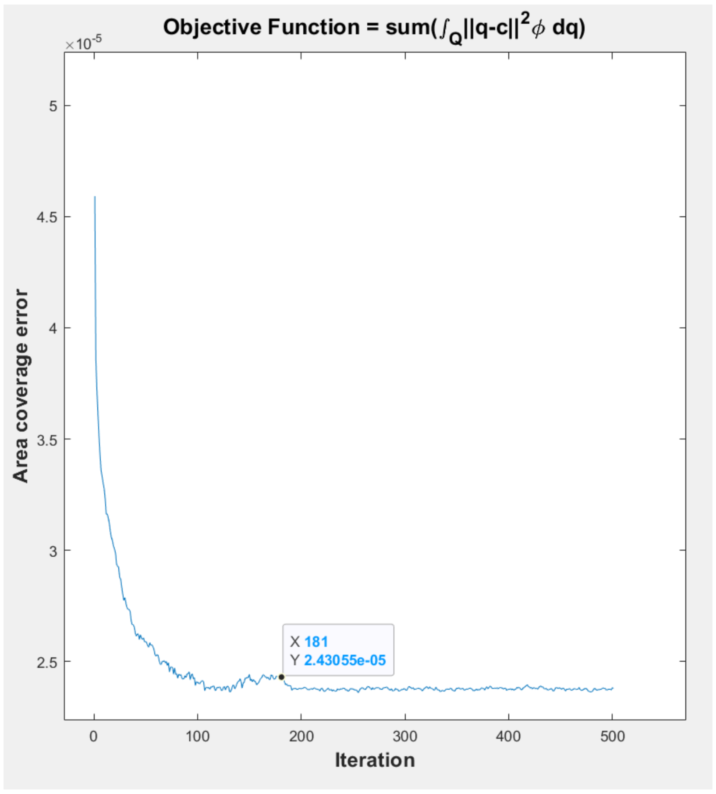


Figure 4.9: Objective Function Graph of Modified Lloyd's Algorithm with Velocity Obstacle.

Figure 4.10 shows the total number of collisions between five deployed robots and dynamic obstacle during the first-time simulation. It is obvious that all five robots successfully avoid the dynamic obstacle during the coverage task. This outcome indicates that the Velocity Obstacle technique has effectively handled the dynamic obstacle in multi-robot coverage, enabling the robots to navigate around the obstacle without collisions. Table 4.4 presents the average total number of collisions between robots and the dynamic obstacle across five simulations using the modified Lloyd's algorithm with Velocity Obstacle. 'Sim' in table represents 'Simulation'. Remarkably, no collisions occurred between the robots and the dynamic obstacle in any of the five simulations. This consistent avoidance of collisions demonstrates the efficacy of the Velocity Obstacle approach in ensuring safe and collision-free multi-robot navigation in the presence of dynamic obstacle.

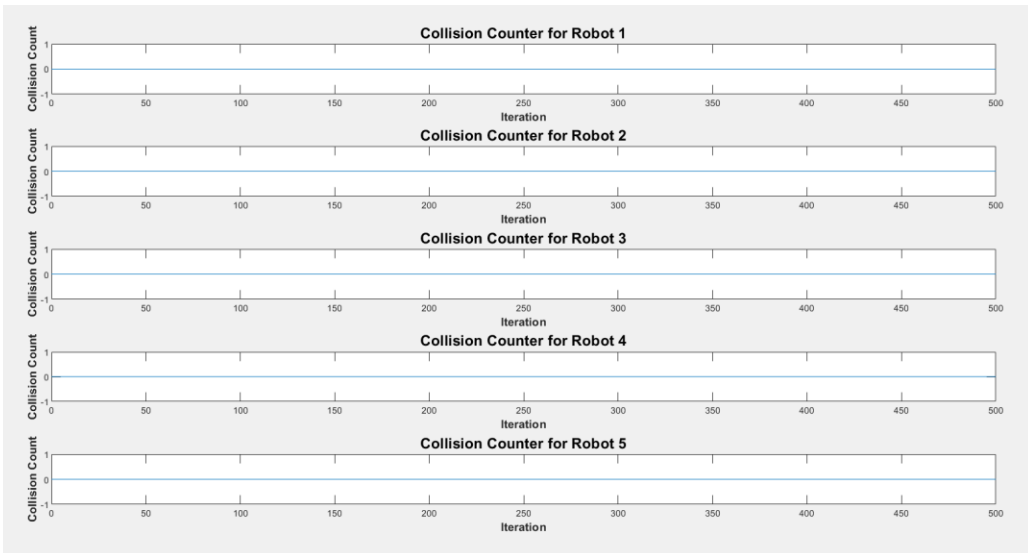


Figure 4.10: Total Number of Collisions between Five Robots and Dynamic Obstacle during Iteration in First Run Simulation in Modified Lloyd's algorithm with Velocity Obstacle.

Table 4.4: Average Total Number of Collisions between Five Robots and Dynamic Obstacle during Iteration in First Run Simulation in Modified Lloyd’s algorithm with Velocity Obstacle.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot5 | 0 | 0 | 0 | 0 | 0 | 0 |

The results from the five simulations employing the Modified Lloyd’s algorithm with Velocity Obstacle demonstrate its success in achieving complete coverage. The position error remains at zero, and the objective function maintains smooth and constant values throughout the iterations. Moreover, the Velocity Obstacle technique effectively handles dynamic obstacle, as evidenced by all robots successfully avoiding collisions during the coverage task. These findings highlight the potential of Velocity Obstacle in real-world scenarios, where it can help prevent damage to robots by enabling collision avoidance.

## 4.3 Modified Lloyd’s Algorithm with Velocity Obstacle under Two Different Scenarios

In this section, the Modified Lloyd’s algorithm with Velocity Obstacle (VO) will be tested under two different scenarios and the results will be compared in terms of the average total number of collisions. This simulation aims to determine effectiveness of velocity obstacles in handling dynamic obstacles in multi-robot coverage across various scenarios. In the first scenario, the aggressiveness of dynamic obstacle will be increased from 1 to 5 and 9. In the second scenario, the starting position of dynamic obstacle will be changed. Each scenario will involve five simulations to ensure the accuracy of the results obtained.

### 4.3.1 Aggressiveness of Dynamic Obstacle (Scenario 1)

In this section, the aggressiveness of the dynamic obstacle's movement will be varied. Higher aggressiveness implies that the dynamic obstacle will move more erratically along its trajectory, while lower aggressiveness indicates smoother movement without much deviation along the trajectory. The aggressiveness of the dynamic obstacle will be adjusted from the original value of 1 to 5 and 9. Each level of aggressiveness will undergo a total of five simulations to calculate the average value. Table 4.5 displays the average total number of collisions observed in the proposed Lloyd’s algorithm with velocity obstacle under an aggressiveness of 5, while Table 4.6 presents the corresponding data for an aggressiveness of 9. ‘Sim’ in each table represents ‘Simulation’.

Table 4.5: Average Total Number of Collisions when Aggressiveness of Dynamic Obstacle is 5.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot5 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 4.6: Average Total Number of Collisions when Aggressiveness of Dynamic Obstacle is 9.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 1 | 0 | 1 | 0 | 1 | $0.6 \approx 1$ |
| Robot2 | 2 | 2 | 2 | 1 | 1 | $1.6 \approx 2$ |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot5 | 1 | 3 | 1 | 1 | 1 | $1.4 \approx 1$ |

From Table 4.5, it's evident that even with an increase in aggressiveness from 1 to 5, the velocity obstacle remains effective in handling the dynamic obstacle. This effectiveness stems from the future position prediction capability inherent in velocity obstacle methodology. By assessing the relative distance between robots and the dynamic obstacle, along with their relative velocities, velocity obstacle algorithms can calculate the time to collision. Leveraging this information, the algorithm can estimate the likely future movements of the dynamic obstacle. This predictive approach is akin to the methodology employed by Wilkie, Van Den Berg, and Manocha in their research, where they assumed the future positions of agents or obstacles to be approximately circular, centred at their current positions (Wilkie, Van Den Berg, and Manocha, 2009). The introduction of a safety margin in the velocity obstacle further enhances the ability of robots to avoid collisions, even with increased aggressiveness. The safety margin, also referred to as the safety distance, represents the minimum distance between robots and the dynamic obstacle necessary to prevent a collision. While a larger safety margin ensures safer avoidance of dynamic obstacles, it may lead to energy wastage or premature avoidance manoeuvres when the dynamic obstacle is still relatively distant.

From Table 4.6, it is obvious that an increase in aggressiveness from 1 to 9 results in collisions between robots and the dynamic obstacle. This occurs because, at an aggressiveness level of 9, the dynamic obstacle exhibits highly erratic movements at high speeds, leaving the robots utilizing the velocity obstacle technique insufficient time to calculate avoidance manoeuvres. Consequently, collisions occur between robots and the dynamic obstacle. However, as indicated in Table 4.6, the average number of collisions remains relatively small, ranging from 1 to 2. This is because, despite the increased aggressiveness, the velocity obstacle algorithm promptly calculates avoidance strategies, minimizing the occurrence of collisions. The results from this scenario demonstrate the efficiency of the velocity obstacle technique in handling dynamic obstacles, enabling robots to avoid collisions during the coverage task, even with heightened aggressiveness levels. However, when aggressiveness reaches excessively high levels, such as 9, the limited time available for calculation may result in some collisions, albeit in small numbers. Nevertheless, once the calculations are completed, robots can successfully avoid the dynamic obstacle in subsequent iterations.

Figure 4.11 shows the final position of robots and dynamic obstacle, along with their trajectory when aggressiveness level is 5. Similarly, Figure 4.12 illustrates the results obtained when the aggressiveness level is 9. A noticeable difference in the trajectories of robots compared to Figure 4.6 is observed in both figures. This variation is attributed to the velocity obstacle algorithm guiding the robots in alternative directions to evade the dynamic obstacle, particularly under higher aggressiveness levels. Furthermore, Figure 4.13 showcases the position error graph and objective function graph for the scenario with an aggressiveness level of 5, while Figure 4.14 presents the corresponding graphs for an aggressiveness level of 9. From both sets of graphs, it is evident that despite the increased aggressiveness of the dynamic obstacle, the multi-robot system still can achieve complete coverage while successfully avoiding collisions.

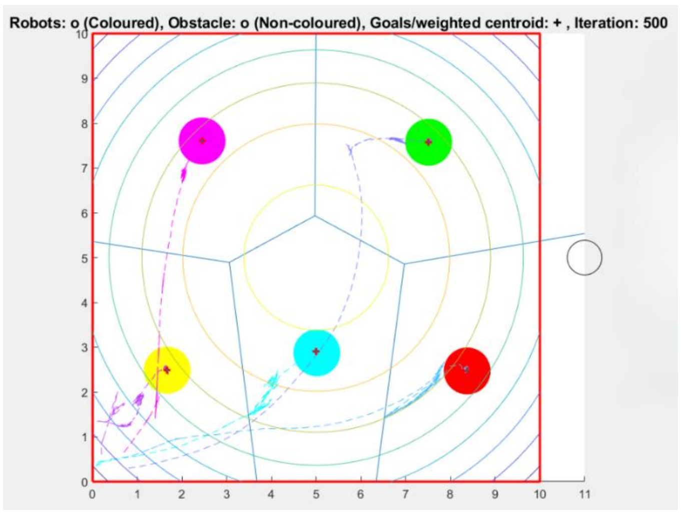


Figure 4.11: Final Positions of Five Robots and Dynamic Obstacle along with their Trajectories when Aggressiveness is 5.

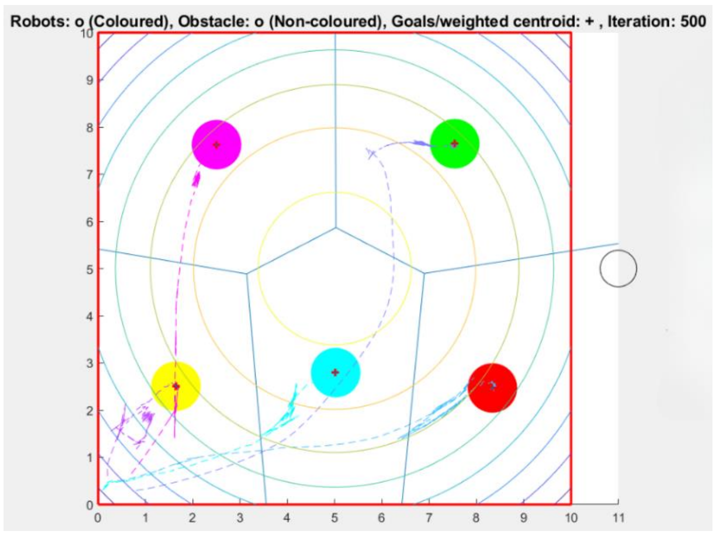


Figure 4.12: Final Positions of Five Robots and Dynamic Obstacle along with their Trajectories when Aggressiveness is 9.

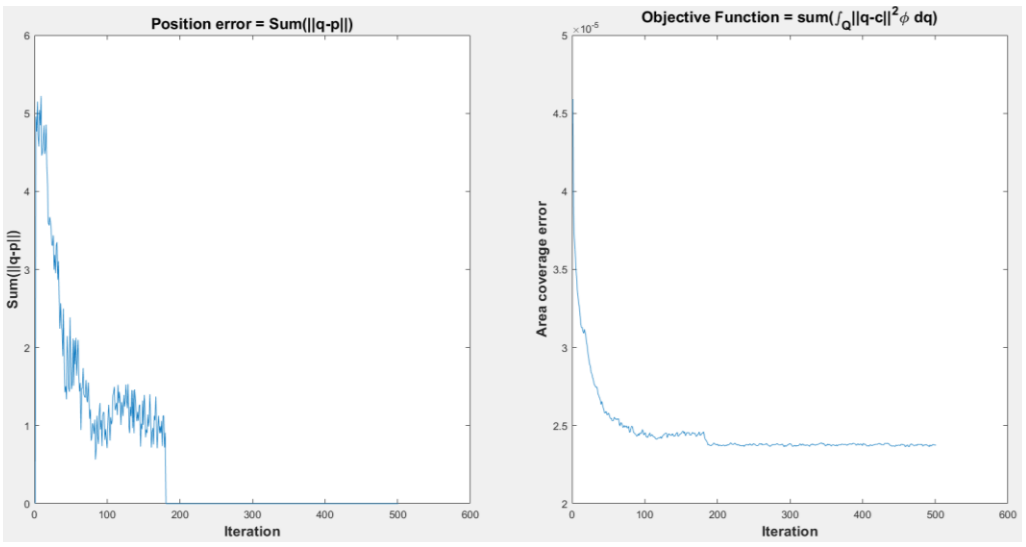


Figure 4.13: Position Error Graph and Objective Function Graph when Aggressiveness is 5.

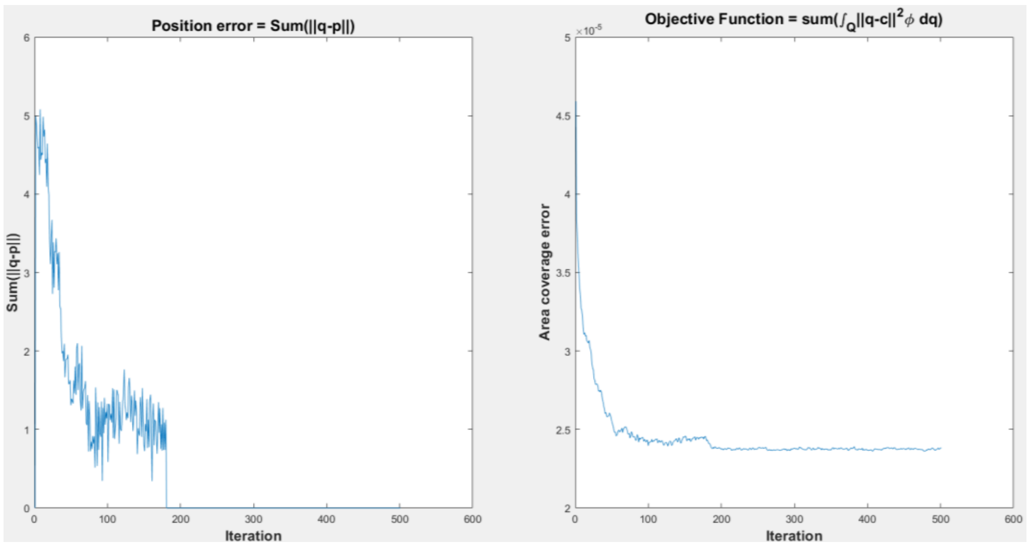


Figure 4.14: Position Error Graph and Objective Function Graph when Aggressiveness is 9.

### 4.3.2 Different Starting Positions of Dynamic Obstacle (Scenario 2)

In this scenario, the dynamic obstacle is initialized with different starting positions during iterations, as indicated in Table 4.7. The parameters such as aggressiveness, number of iterations, and other parameters remain consistent with those presented in Table 4.1. The objective of this simulation is to examine the relationship between the starting position of the dynamic obstacle and the average total number of collisions between robots and the obstacle. By varying the starting position, this simulation aims to assess whether the velocity obstacle algorithm can effectively handle the dynamic obstacle and enable the robots to avoid collisions during the coverage task. Table 4.8 presents the results of the average number of collisions between the five robots and the dynamic obstacle under Case 1 starting positions of the dynamic obstacle. Similarly, Table 4.9 displays the results under Case 2 starting positions. ‘Sim’ in each table represents ‘Simulation’.

Table 4.7: Starting Positions of Dynamic Obstacle for Case 1 and Case 2.

| **Cases** | **Starting Position of Dynamic Obstacle** |
|---|---|
| 1 | [2, 3] |
| 2 | [0, 8] |

Table 4.8: Average Total Number of Collisions between Five Robots and Dynamic Obstacle under Case 1.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot5 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 4.9: Average Total Number of Collisions between Five Robots and Dynamic Obstacle under Case 2.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot5 | 5 | 0 | 4 | 0 | 5 | $2.8 \approx 3$ |

From Table 4.8, it is evident that even when the starting position of the dynamic obstacle changes, the robots can still avoid collisions effectively using the velocity obstacle technique. However, as shown in Table 4.9, Robot 5 collided with the dynamic obstacle when its starting position was [0, 8]. This occurred because Robot 5's initial position was too close to that of the dynamic obstacle, resulting in an immediate collision at the start of the iteration. In such a scenario, the velocity obstacle algorithm did not have sufficient time to perform calculations before the collision. However, as the simulation progressed, Robot 5 managed to avoid further collisions with the dynamic obstacle, demonstrating the effectiveness of the velocity obstacle technique in handling dynamic obstacles during multi-robot coverage tasks. Consequently, the number of collisions remained relatively low.

Figure 4.15 shows the final position of robots and dynamic obstacle along with their trajectory in Case 1, while Figure 4.16 shows the results in Case 2. It is clearly shows that in Figure 4.15, the entire visualization is different compared to Figure 4.6. This is due to variation in the starting position of dynamic obstacle causing the density distribution different. This will result in different goal positions were assigned to robots, and causing the final position of robots is different as well as dynamic obstacle final position In Figure 4.16, the trajectories of robots were different compared to Figure 4.1. This is because the starting point of dynamic obstacle is different, and it will affect movement of robots. However, the robots are still able to avoid dynamic obstacle by using velocity obstacle technique. In Figure 4.16, the trajectory of Robot 5 forms a triangular shape. Initially, Robot 5 moves toward its goal and travels in a circular path. However, leveraging the future position prediction feature of the velocity obstacle technique, Robot 5 anticipates a collision with the dynamic obstacle in subsequent iterations if it remains in its current position, close to the goal. Consequently, Robot 5 adjusts its trajectory, moving backward at an avoidance angle and direction with the avoidance velocity calculated by the velocity obstacle algorithm, thereby successfully avoiding the dynamic obstacle during the coverage task.

Figure 4.17 displays the position error graph and objective function graph for case 1, while Figure 4.18 illustrates the same graphs for case 2. In both figures, it is evident that complete coverage is achieved by all five robots, despite the variation in the starting position of the dynamic obstacle. This indicates that the multi-robots successfully accomplish full coverage while effectively avoiding the dynamic obstacle.

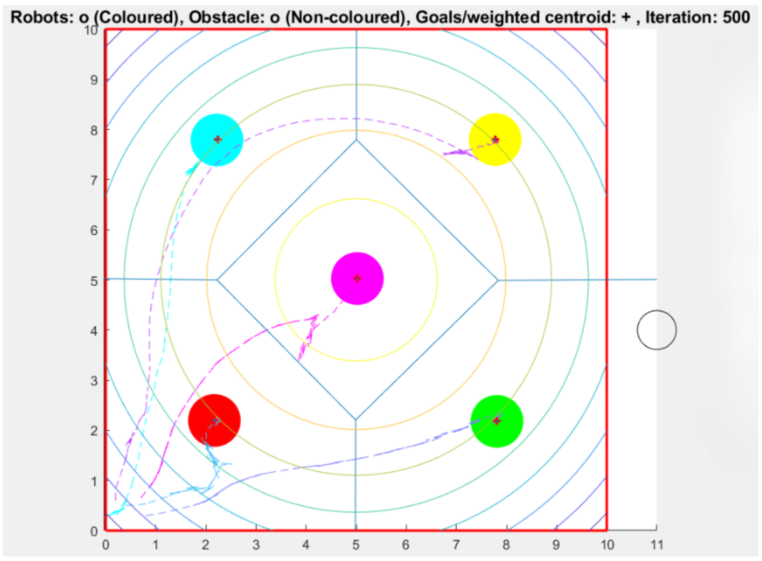


Figure 4.15: Final Positions of Five Robots and Dynamic Obstacle along with their Trajectories in Case 1.

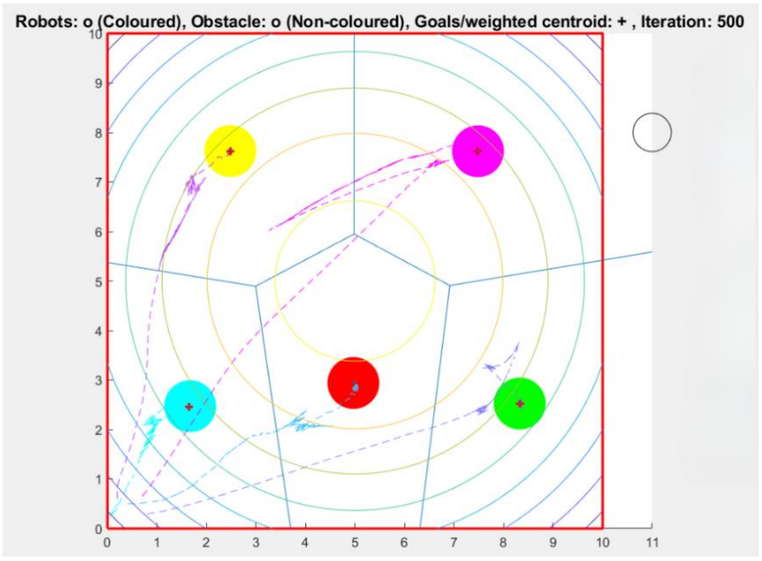


Figure 4.16: Final Positions of Five Robots and Dynamic Obstacle along with their Trajectories in Case 2.

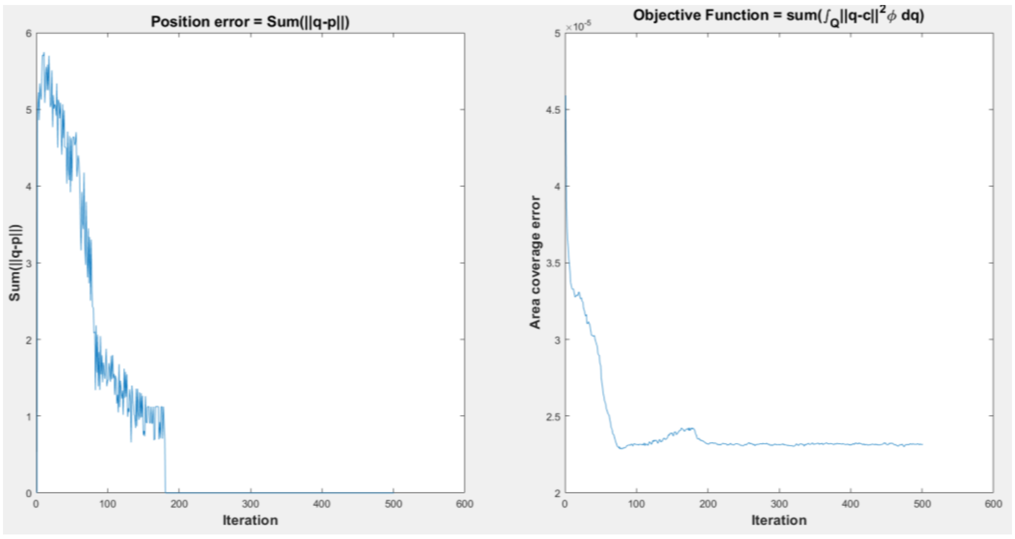


Figure 4.17: Position Error Graph and Objective Function Graph obtained from Simulation in Case 1.

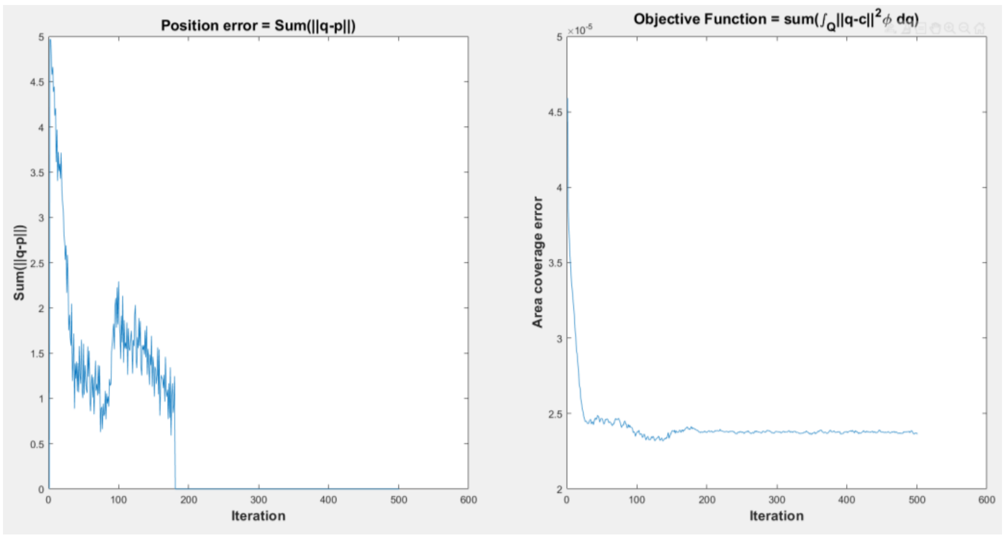


Figure 4.18: Position Error Graph and Objective Function Graph obtained from Simulation in Case 2.

## 4.4 Modified Lloyd's Algorithm with Velocity Obstacle Parameters

This section discusses the impact of parameters of modified algorithm on the simulation result. Various parameters, such as the number of iterations and safety margin, play crucial roles in determining the effectiveness of dynamic obstacle handling. Therefore, it is essential to carefully tune these parameters to enhance performance. In order to assess the influence of the number of iterations and safety margin parameters, the remaining parameters in the simulation are kept fixed based on Table 4.1. This approach allows for the evaluation of how changes in the number of iterations and safety margin affect the simulation results.

### 4.4.1 Number of Iterations

In multi-robot coverage, the number of iterations is a crucial parameter that significantly influences algorithm performance. In Lloyd’s algorithm, the number of iterations determines the convergence of the algorithm. During each iteration, the centroids of Voronoi regions are recalculated based on the mean of the points assigned to each region. As successive iterations progress, these centroids gradually move towards an optimal position that minimizes the variance within each Voronoi region. Therefore, the number of iterations affects how accurately the centroids represent the centres of the Voronoi regions. Sufficient iterations allow ample time for the robots to converge towards the centroids of Voronoi regions, also known as goal positions. In each iteration, robots are constrained to move within specific steps, determined by the gain parameter. The algorithm should stop iterating when there is no change in the Voronoi assignment from one iteration to the next, indicating full convergence. Insufficient iterations may lead to incomplete convergence and failure to accomplish the coverage task. However, increasing the number of iterations also escalates computational costs. Each iteration involves computing distances between points and centroids and updating centroids based on the current Voronoi assignment, requiring more computation time and resources. The simulation was conducted with varying numbers of iterations as shown in Table 4.10. Figure 4.19 shows the final positions of robots and dynamic obstacle when the end of iteration in case 1 while Figure 4.20 shows the same for case 2. Additionally, Figure 4.21 and Figure 4.22 present the position error graphs and objective function graphs for Case 1 and Case 2 respectively.

From Figure 4.19, it is clear that with the 50 number of iterations, the robots have not enough time to reach their goal positions. They are forced to halt halfway when the iteration ends. This indicates that if the number of iterations is too low, Lloyd’s algorithm will fail to fully converge, leading to the failure of the multi-robot coverage task. Furthermore, the position error graph in Figure 4.21 not being zero and the objective function graph not remaining constant in case 1 also indicate that the coverage task is incomplete, with the robots failing to reach their respective goal positions within 50 iterations.

From Figure 4.20, it is evident that the robots have reached their respective goal positions. However, in Figure 4.22, it is clear that after 180 iterations, the robots have achieved complete coverage, as indicated by the position error graph showing zero error and the objective function graph remaining constant value beyond 180 iterations. Thus, iterations beyond 180 are considered excessive and would waste resources. However, to ensure complete coverage and full convergence of Lloyd’s algorithm, the iterations cannot stop immediately after 180 iterations. This is because parameters such as the dynamic obstacle's velocity, random noise, and predefined aggressiveness determine how quickly the dynamic obstacle leaves the Bound space. Only after the dynamic obstacle leaves the Bound space, the robots will proceed to move to reach their respective goal positions, rather than moving in a circular shape around them. Therefore, it is necessary to allow for an additional 2 to 3 times the number of iterations beyond 180, totalling 360 to 540 iterations, to ensure full convergence of Lloyd’s algorithm and achievement of complete coverage. Iterations beyond 540 can be considered excessive, as they increase computational load, time, and resource usage unnecessarily. In Case 2, it can be inferred that an excessively large number of iterations, which is 1000 iterations, does not necessarily improve the solution; instead, it increases the computational power required. Although the number of iterations fluctuates in this simulation, it does not affect the effectiveness of the velocity obstacle technique. It still allowing all robots to successfully avoid the dynamic obstacle during the coverage task in both Case 1 and Case 2.

Table 4.10: Number of Iterations in Case 1 and Case 2.

| **Cases** | **Number of Iterations** |
|---|---|
| 1 | 50 |
| 2 | 1000 |

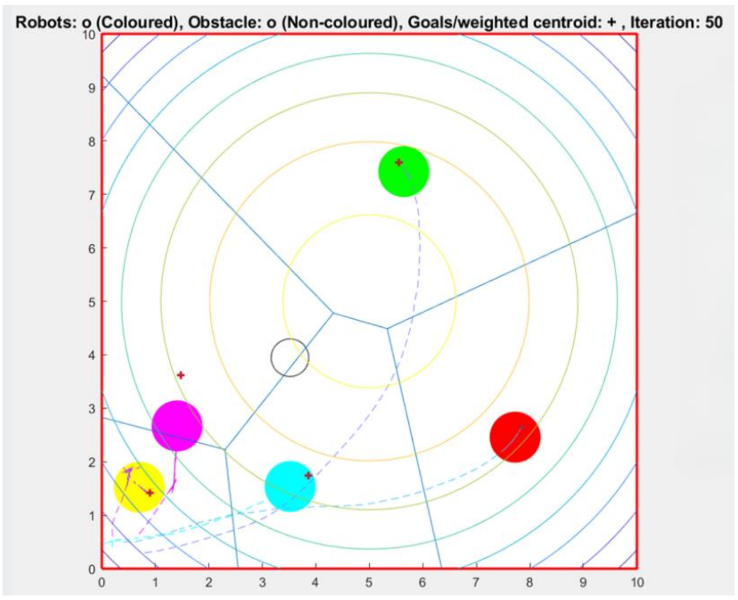


Figure 4.19: Final Positions of Five Robots and Dynamic Obstacle with Corresponding Trajectories at the End of Iteration in Case 1.

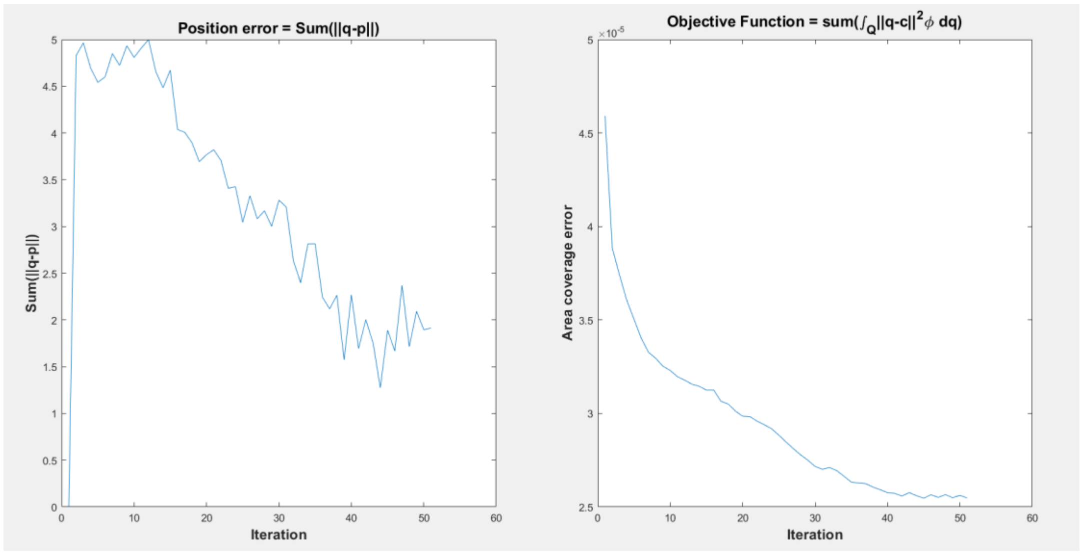


Figure 4.20: Position Error Graph and Objective Function Graph obtained from Simulation in Case 1.

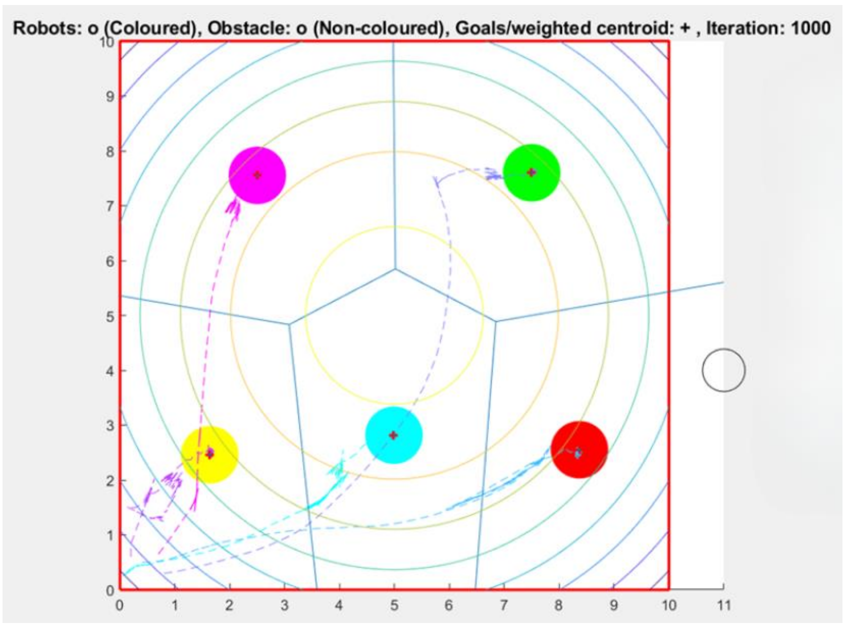


Figure 4.21: Final Positions of Five Robots and Dynamic Obstacle with Corresponding Trajectories at the End of Iteration in Case 2.

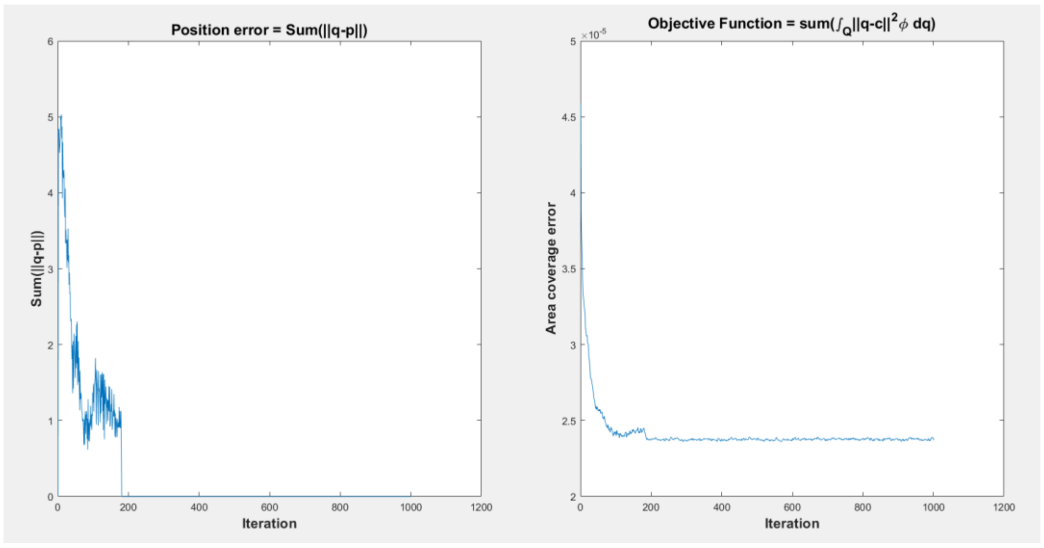


Figure 4.22: Position Error Graph and Objective Function Graph obtained from Simulation in Case 2.

### 4.4.2 Safety Margin

Safety margin, also known as the safety distance between robots and dynamic obstacles, plays a crucial role in ensuring effective collision avoidance. It provides a buffer zone that allows robots and obstacles to avoid collisions even if they are not on a direct collision course. By incorporating a safety margin into the Velocity Obstacle technique, robots have sufficient time and space to steer away from dynamic obstacles, reducing the risk of collision. The adjustment of safety margin is based on uncertainty in robots’ motion, dynamic obstacle motion, aggressiveness and so on. By introducing safety margin into velocity obstacle, it ensures that robots start avoiding obstacle well before a potential collision would occur. However, the size of safety margin should be adjusted wisely based on requirements of scenario. Higher safety margins provide more conservative avoidance behaviour, while smaller safety margins allow robots to navigate closer to obstacles before initiating avoidance manoeuvres. The optimal safety margin depends on various factors such as robot velocity, dynamic obstacle velocity, and the aggressiveness of the dynamic obstacle.

In this section, simulations are conducted based on the safety margin values provided in Table 4.11. Other parameters remain consistent with those listed in Table 4.1. Table 4.12 presents the average total number of collisions between robots and dynamic obstacles in Case 1, while Table 4.13 shows the corresponding results for Case 2. ‘Sim’ in each table represents ‘Simulation’.

Table 4.11: Safety Margin for Case 1 and Case 2.

| **Cases** | **Number of Iterations** |
|---|---|
| 1 | 1 |
| 2 | 20 |

Table 4.12: Average Total Number of Collisions between Five Robots and Dynamic Obstacle under case 1.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 2 | 1 | 1 | 3 | 2 | $1.8 \approx 2$ |
| Robot5 | 25 | 24 | 28 | 25 | 23 | 25 |

Table 4.13: Average Total Number of Collisions between Five Robots and Dynamic Obstacle under case 2.

| **Robots** | **Number of Collisions** | | | | | |
|---|---|---|---|---|---|---|
| | **Sim 1** | **Sim 2** | **Sim 3** | **Sim 4** | **Sim 5** | **Average** |
| Robot1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Robot5 | 0 | 0 | 0 | 0 | 0 | 0 |

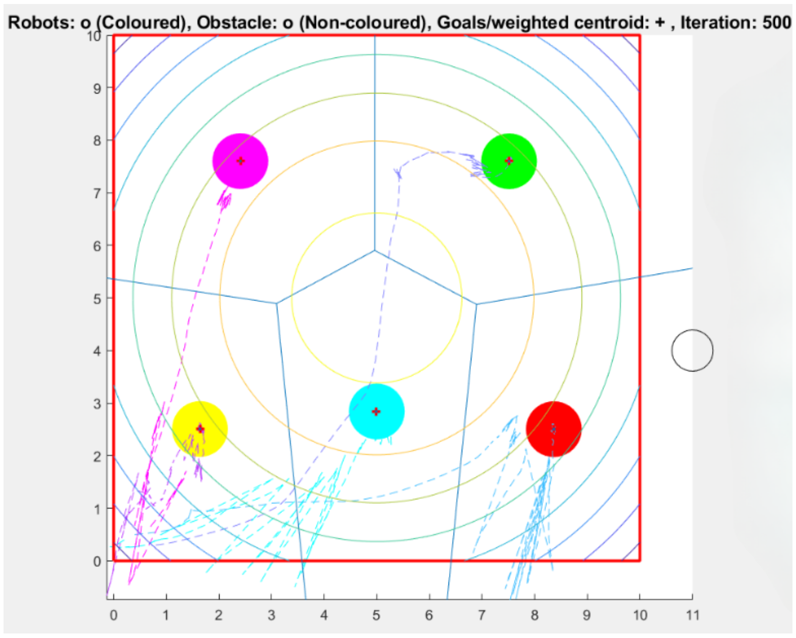


Figure 4.23: Final Positions of Five Robots and Dynamic Obstacle with Corresponding Trajectories at the End of Iteration in Case 2.

From Table 4.12, it's evident that a very small safety margin leads to collisions, with a significant number of collisions occurring, such as 25 collisions for Robot 5. However, Table 4.13 indicates that with a higher safety margin, the average number of collisions is zero, demonstrating successful avoidance of dynamic obstacles by all robots using the Velocity Obstacle technique.

However, setting the safety margin to a very high value can have adverse effects. Figure 4.23 shows the final positions of robots and dynamic obstacle with their trajectories. It is clear that in Figure 4.23, the trajectories of robots are not smooth as trajectories of robots in Figure 4.6. This is because safety margin is excessively large and causing robots need to continuously maintain a large distance between robots and dynamic obstacle. Therefore, when dynamic obstacle moves closer to robots, robots will avoid by using velocity obstacle in a large distance at all times which is not necessarily, leading to unnecessary energy and time consumption. In order to handle the dynamic obstacle, a small distance is enough to avoid dynamic obstacle. A high value of safety margin allows the particles to move more freely and have more space and time to avoid dynamic obstacle which at the same time decrease the chance of collision. However, it should be noted that higher safety margin may cause the robots to overreact the distance and increase the power, time and resources required. On the other hand, a low value of safety margin can lead to lesser waste of power, energy and time but has higher risk in collision occurred between robots and dynamic obstacle during the coverage.

## 4.5 Summary

The performance for both existing Lloyd's algorithm without Velocity Obstacle and Modified Lloyd's algorithm with Velocity Obstacle (VO) was evaluated in terms of average total number of collisions between robots and dynamic obstacle and achievement of complete coverage. The Modified Lloyd's algorithm with VO was also further assessed under two different scenarios which are aggressiveness and different starting positions of dynamic obstacle, with a focus on average total number of collisions and complete coverage by referring to position graph and objective function graph.

The biggest difference between two algorithms is Modified Lloyd's algorithm is implemented with Velocity Obstacle, which is a dynamic obstacle avoidance technique, it can help to handle dynamic obstacle and enable robots avoid it during multi-robot coverage. From the results obtained, existing Lloyd's algorithm that lack of VO, it can still achieve complete coverage, but during the coverage task, robots will collide with dynamic obstacle. On the other hand, Modified Lloyd's algorithm with VO outperforms and no collisions occurred within five simulations. Besides that, the impact of algorithm parameters on the simulation results were examined. It revealed that modified algorithm has the better performance in terms of achieving of complete coverage and zero average number of collisions with number of iteration 500 and safety margin of 5.

# CHAPTER 5

# CONCLUSIONS AND RECOMMENDATIONS

## 5.1 Conclusions

In conclusion, the project has successfully achieved its aim and objectives. A comprehensive literature review of existing algorithms for dynamic obstacle handling in multi-robot coverage has been conducted, accompanied by a summary comparison table. Furthermore, a Modified Lloyd’s Algorithm incorporating Velocity Obstacle has been successfully developed. Its performance had been evaluated using MATLAB. This project aimed to solve the challenges that faced by multi-robot during coverage task, particularly collisions with dynamic obstacles. The significance of this project stems from the increasing demand for multi-robot systems across various fields, necessitating a solution to mitigate collision risks. To achieve this goal, a modified version of Lloyd’s algorithm is introduced which is Modified Lloyd’s algorithm with Velocity Obstacle (VO). Unlike the existing Lloyd’s algorithm which focuses solely on achieving complete coverage without dynamic obstacle avoidance, the modified algorithm with VO incorporates calculations to determine the time to collision. This calculation involves assessing the relative distances and velocities of robots with respect to dynamic obstacles. By utilizing the time to collision and the calculated avoidance direction, an avoidance velocity is calculated and selected. The algorithm determines an avoidance velocity, allowing robots to steer clear of dynamic obstacles effectively. Moreover, VO anticipates the future positions of dynamic obstacles in subsequent iterations, enabling the algorithm to compute the future distance and pre-emptively plan avoidance manoeuvres. The concept of safety margin is integral to this approach, as it determines the minimum safe distance between robots and dynamic obstacles. This ensures that robots have sufficient time to execute avoidance manoeuvres using VO effectively. Overall, the Modified Lloyd’s Algorithm with Velocity Obstacle presents a comprehensive solution to the challenge of dynamic obstacle avoidance in multi-robot systems, enhancing safety and efficiency during coverage tasks.

Simulations were conducted using both existing and modified Lloyd’s algorithm in same scenario, repeated for five times. The results demonstrated that modified Lloyd’s algorithm with VO outperformed the existing Lloyd’s algorithm without VO in terms of average total number of collisions between multi-robot and dynamic obstacle. Furthermore, the Modified Lloyd’s algorithm was also simulated under two different scenarios which are aggressiveness and different starting positions of dynamic obstacle with five simulations each. The results show that even in different scenarios, modified Lloyd’s algorithm with VO still can performed well and effectively handling dynamic obstacles by enabling multi-robots to avoid them. Moreover, the impact of modified Lloyd’s algorithm with VO parameters on simulation results was also evaluated. It was found that number of iterations cannot be too less, as this would prevent achieving complete coverage. At the same time having too many iterations would lead to a waste of computational resources. Thus, the suitable number of iterations for the modified Lloyd’s algorithm with VO was determined to be between 360 and 540 iterations. Additionally, it was observed that the modified Lloyd’s algorithm with VO achieved better performance with a safety margin of 5. Overall, the findings of this project suggest that the modified Lloyd’s algorithm with VO is capable of effectively handling dynamic obstacles during multi-robot coverage, allowing robots to avoid dynamic obstacles and achieve complete coverage.

## 5.2 Recommendations for Future Work

This project can be improved in many ways due to the project’s limitations to enhance its effectiveness. For instance, incorporating the orientation of robots during the coverage task would make the simulation more realistic. In real-life scenarios, robots must be oriented appropriately to detect dynamic obstacles and avoid collisions. To simulate this accurately, sensors with a 360-degree viewing angle could be attached to the robots. This would enable them to detect dynamic obstacles from all directions, ensuring more comprehensive coverage and collision avoidance. Additionally, while the project focused on a single dynamic obstacle due to time constraints and complexity considerations, incorporating multiple dynamic obstacles would provide a more realistic simulation. Dealing with multiple dynamic obstacles introduces additional challenges, such as coordinating avoidance manoeuvres and optimizing navigation paths. However, addressing these challenges would lead to a more robust algorithm capable of handling complex real-world scenarios involving multiple moving obstacles.

Additionally, Velocity Obstacle (VO) is a dynamic obstacle avoidance technique which specifically designed to avoid dynamic obstacle. However, when confronted with static obstacles, VO may not be as effective. In the presence of static obstacles, VO may cause robots to move very slowly or nearly come to a stop while introducing avoidance angles and directions. This can result in a large number of iterations being required for successful obstacle avoidance, which is not an efficient approach. Increasing the avoidance angle and direction could potentially reduce the number of iterations needed, but it may also risk driving robots out of the Bound space if they are facing a static obstacle behind them. To address static obstacles, alternative obstacle avoidance techniques such as Artificial Potential Field (APF) or other methods suitable for static obstacles (Khatib, 1986) can be integrated. These techniques can complement VO in scenarios involving both static and dynamic obstacles, ensuring effective obstacle avoidance while minimizing computational overhead.

Moreover, the algorithm’s performance in a physical environment or real-life environment remains untested, as it has only been simulated and evaluated in simulations by using MATLAB. Thus, the performance of the algorithm can be further evaluated in the physical environment with Robotics Operating System (ROS). Furthermore, it is also important that for an efficient dynamic obstacle handling in multi-robot coverage need to considered multiple aspects such as smoothness of trajectory, execution time, task allocation and so on. In this project, only dynamic obstacle handling and avoiding is considered due to time limitation.

Therefore, recommendations for future work will be involving introducing a greater number of dynamic obstacles and robots, implementing static obstacle avoidance into algorithm, adding 360-degree viewing sensors and extending the algorithm to physical experiment using Robotics Operating System (ROS) to validate its performance and functionality in real-world applications.

# REFERENCES

Abdulghafoor, A. and Bakolas, E., 2021. Distributed coverage control of multi-agent networks with guaranteed collision avoidance in cluttered environments. *IFAC-PapersOnLine,* 54**,** 771-776.

Azuma, S.-I., Yoshimura, R. and Sugie, T., 2013. Broadcast control of multi-agent systems. *Automatica,* 49**,** 2307-2316.

Breitenmoser, A., Schwager, M., Metzger, J.-C., Siegwart, R. and Rus, D., 2010. Voronoi coverage of non-convex environments with a group of networked robots. *IEEE international conference on robotics and automation, 2010.* IEEE, 4982-4989.

Das, K. and Ghose, D., 2009. Positional consensus in multi-agent systems using a broadcast control mechanism. *American control conference, 2009.* IEEE, 5731-5736.

Douthwaite, J. A., Zhao, S. and Mihaylova, L. S., 2019. Velocity obstacle approaches for multi-agent collision avoidance. *Unmanned Systems,* 7**,** 55-64.

Du, Q., Emelianenko, M. and Ju, L., 2006. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM journal on numerical analysis,* 44**,** 102-119.

Du, Q., Faber, V. and Gunzburger, M., 1999. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM review,* 41**,** 637-676.

Fiorini, P. and Shiller, Z., 1998. Motion planning in dynamic environments using velocity obstacles. *The international journal of robotics research,* 17**,** 760-772.

Gabriely, Y. and Rimon, E., 2001. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of mathematics and artificial intelligence,* 31**,** 77-98.

Ghobakhloo, M., 2020. Industry 4.0, digitization, and opportunities for sustainability. *Journal of cleaner production,* 252**,** 119869.

Haraldsen, A., Wiig, M. S. and Pettersen, K. Y., 2020. Vehicle safety of the velocity obstacle algorithm. *59th IEEE Conference on Decision and Control (CDC), 2020.* IEEE, 5340-5347.

Jiang, B., Sun, Z., Anderson, B. D. and Lageman, C., 2019. Higher order mobile coverage control with applications to clustering of discrete sets. *Automatica,* 102**,** 27-33.

Ju, C., Kim, J., Seol, J. and Son, H. I., 2022. A review on multirobot systems in agriculture. *Computers and Electronics in Agriculture,* 202**,** 107336.

Julius, A. A., Halász, Á., Sakar, M. S., Rubin, H., Kumar, V. and Pappas, G. J., 2008. Stochastic modeling and control of biological systems: the lactose regulation system of Escherichia coli. *IEEE Transactions on Automatic Control,* 53**,** 51-65.

Juniastuti, S., Fachri, M., Nugroho, S. M. S. and Hariadi, M., 2016. Crowd navigation using leader-follower algorithm based Reciprocal Velocity Obstacles. *International Symposium on Electronics and Smart Devices (ISESD), 2016.* IEEE, 148-152.

Kamon, I., Rimon, E. and Rivlin, E., 1998. Tangentbug: A range-sensor-based navigation algorithm. *The International Journal of Robotics Research,* 17**,** 934-953.

Karabegovic, I., 2017. Comparative analysis of automation of production process with industrial robots in Asia/Australia and Europe. *International Journal of Human Capital in Urban Management,* 2**,** 29-38.

Khatib, O., 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research,* 5**,** 90-98.

Kim, J. and Son, H. I., 2020. A voronoi diagram-based workspace partition for weak cooperation of multi-robot system in orchard. *IEEE Access,* 8**,** 20676-20686.

Lee, S. G. and Egerstedt, M., 2013. Controlled coverage using time-varying density functions. *IFAC Proceedings Volumes,* 46**,** 220-226.

Liu, X.-F., Fang, Y., Zhan, Z.-H. and Zhang, J., 2023. Strength Learning Particle Swarm Optimization for Multiobjective Multirobot Task Scheduling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Lloyd, S., 1982. Least squares quantization in PCM. *IEEE transactions on information theory,* 28**,** 129-137.

Mcguire, S., 2015. FAO, IFAD, and WFP. The state of food insecurity in the world 2015: meeting the 2015 international hunger targets: taking stock of uneven progress. Rome: FAO, 2015. *Advances in Nutrition,* 6**,** 623-624.

Mohamad Nor, M. H., Ismail, Z. H. and Ahmad, M. A., 2020. Broadcast control of multi-robot systems with norm-limited update vector. *International Journal of Advanced Robotic Systems,* 17**,** 1729881420945958.

Moorehead, S. J., Wellington, C. K., Paulino, H. and Reid, J. F., 2010. R-gator: An unmanned utility vehicle. *Unmanned systems technology XII, 2010.* SPIE, 391-402.

Park, J., Kim, D., Yoon, Y., Kim, H. and Yi, K., 2009. Obstacle avoidance of autonomous vehicles based on model predictive control. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* 223**,** 1499-1516.

Pedersen, S. M., Fountas, S., Have, H. and Blackmore, B., 2006. Agricultural robots—system analysis and economic feasibility. *Precision agriculture,* 7**,** 295-308.

Pradhan, S. K., Parhi, D. R., Panda, A. K. and Behera, R. K., 2006. Potential field method to navigate several mobile robots. *Applied Intelligence,* 25**,** 321-333.

Reynolds, C. W., 1999. Steering behaviors for autonomous characters. *Game developers conference, 1999.* Citeseer, 763-782.

Senthilkumar, K. and Bharadwaj, K., 2008. Spanning tree based terrain coverage by multi robots in unknown environments. *Annual IEEE India Conference, 2008.* IEEE, 120-125.

Sherwani, F., Asad, M. M. and Ibrahim, B. S. K. K., 2020. Collaborative robots and industrial revolution 4.0 (ir 4.0). *International Conference on Emerging Trends in Smart Technologies (ICETST), 2020.* IEEE, 1-5.

Sud, U., Ahmad, T., Gupta, V., Chandra, H., Sahoo, P. M., Aditya, K., Singh, M. and Biswas, A., 2015. Gap Analysis and Proposed Methodologies for Estimation of Crop Area and Crop Yield under Mixed and Continuous Cropping under the project "Research on Improving Methods for Estimating Crop Area, Yield and Production under Mixed, Repeated and Continuous Cropping.

Ueda, J., Odhner, L. and Asada, H. H., 2007. Broadcast feedback of stochastic cellular actuators inspired by biological muscle control. *The International Journal of Robotics Research,* 26**,** 1251-1265.

Vasconez, J. P., Kantor, G. A. and Cheein, F. a. A., 2019. Human–robot interaction in agriculture: A survey and current challenges. *Biosystems engineering,* 179**,** 35-48.

Wang, D., Xie, B. and Agrawal, D. P., 2008. Coverage and lifetime optimization of wireless sensor networks with gaussian distribution. *IEEE transactions on mobile computing,* 7**,** 1444-1458.

Wilkie, D., Van Den Berg, J. and Manocha, D., 2009. Generalized velocity obstacles. *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.* IEEE, 5573-5578.

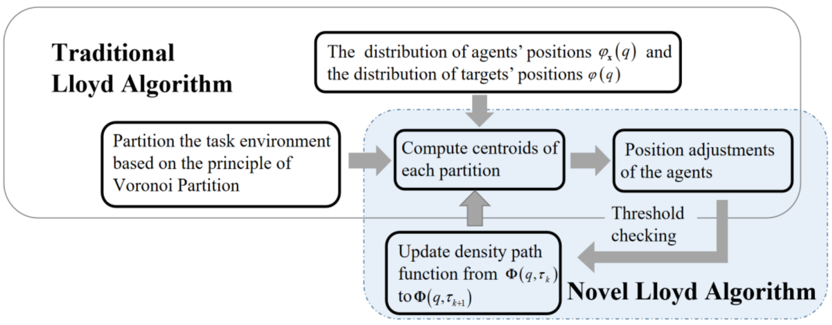
Wu, Z., Su, W. and Li, J., 2019. Multi-robot path planning based on improved artificial potential field and B-spline curve optimization. *Chinese Control Conference (CCC), 2019.* IEEE, 4691-4696.

Yang, Y., Su, M., Fan, H., Liu, L. and Wang, B., 2023. A Constructive Density Function Path Leading to Global Coverage Strategy for A Gaussian Random Field. *IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS), 2023*. IEEE, 249-254.

Zhang, C. and Kovacs, J. M., 2012. The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture,* 13**,** 693-712.

# APPENDICES

Appendix A: Structure chart of the deployment strategy of Lloyd's algorithm (Yang et al., 2023).


Traditional Lloyd Algorithm
The distribution of agents' positions $\varphi_x(q)$ and the distribution of targets' positions $\varphi(q)$
Partition the task environment based on the principle of Voronoi Partition
Compute centroids of each partition
Position adjustments of the agents
Threshold checking
Update density path function from $\Phi(q,\tau_k)$ to $\Phi(q,\tau_{k+1})$
Novel Lloyd Algorithm

Appendix B: Traditional Lloyd's Algorithm(Yang et al., 2023).

**Algorithm 1** Traditional Lloyd's Algorithm

**Require:**

Domain X, initial agents' position are set to x(0), the threshold is set to $\epsilon$.

**Ensure:**

The set of agents' position by x = [ $x^T_1$, $x^T_2$, ..., $x^T_n$ ]

```
1.  Let $e_{max} = \varepsilon$
2.  Compute $e_i = |C(V_i(x)) - x_i|$
3.  **while** $e_i > \epsilon$ ***do***
4.      Generate Voronoi Partition V(x) based on Equation 3.3
5.      **for** i = 1: n **do**
6.          Compute $M(V_i)$ and $C(V_i)$
7.          Update $x_i = C(V_i)$
8.          Compute $e_i = \|x_i - c_i\|$
9.      **end for**
10.     Update $e_{max} = max_i e_i$
11. **end while**
12. **return x** = $[x_1^T, \ldots, x_n^T]^T$
```

## Appendix C: An Advanced Version of Lloyd's Algorithm (Yang et al., 2023).

**Algorithm 2** A Modified Lloyd's Algorithm

**Require:**

Domain X, number of agents n, time step $\Delta\tau$, coefficient of Gaussian function G, a, b α, r, $\epsilon$ ,initial agents' position are set to x(0), the threshold is set to $\epsilon$, k is number of iterations while K is upper limit of iterations.

**Ensure:**

The set of agents' position by x = [ $x^T_1, x^T_2, ..., x^T_n$ ]

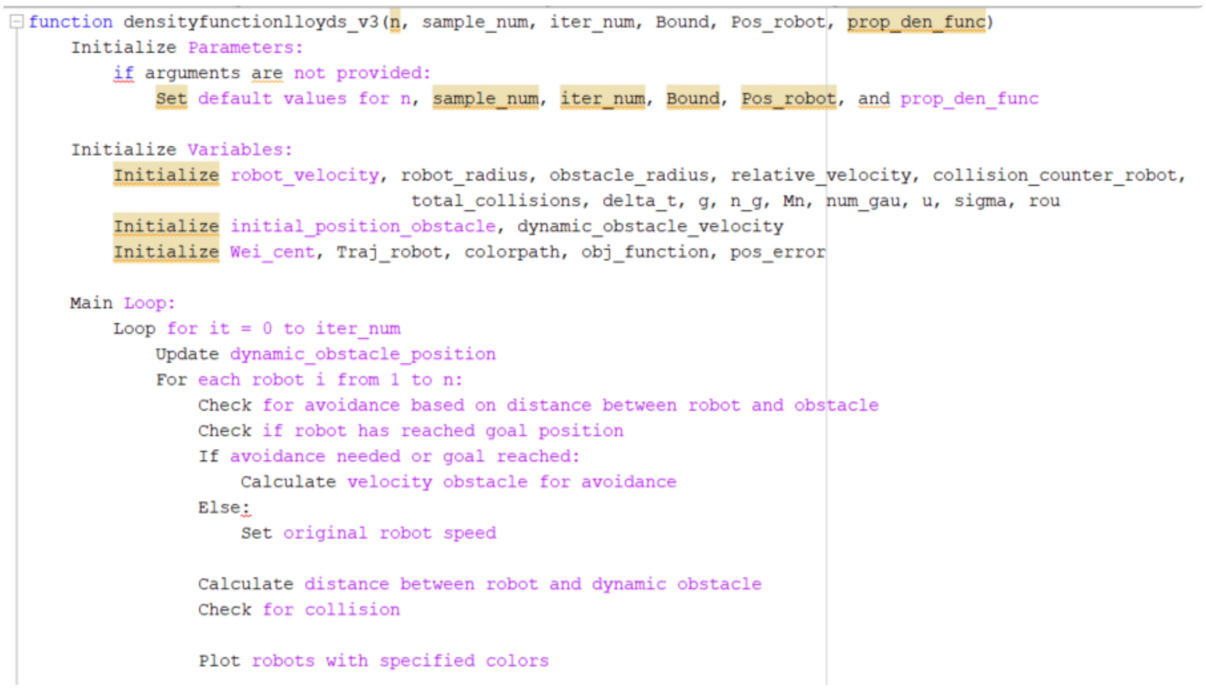
```
1.  Let $e_{max} = \varepsilon$ , k = 0, m = 0
2.  Find $\mu_0, \Sigma_0$ and let $\varphi(q) := \varphi(q, \mu_0, \Sigma_0)$
3.  while $k < K$ do
4.      while $e_{max} > \varepsilon$ do
5.          Let t = $\tau_k$ = k$\Delta\tau$
6.          Compute $\Phi(q,t)$ based on $\varphi(q)$
7.          Replace density function with $\Phi(q,t)$ when t = $\tau_k$
8.          Generate Voronoi Partition V(x) based on $\Phi(q,t)$
9.          for i = 1: n do
10.             Compute $M(V_i)$ and $C(V_i)$
11.             Update $x_i = C(V_i)$
12.             Compute $e_i = \|x_i - c_i\|$
13.         end for
14.         Update $e_{max} = max_i e_i$
15.     end while
16.     Update k = k+1 (increment by 1)
17. end while
18. return x = $[x_1^T, \dots, x_n^T]^T$
```

## Appendix D: Velocity Obstacle to Find Best Feasible Control (D. Wilkie, J. Van Den Berg and D. Manocha, 2003).

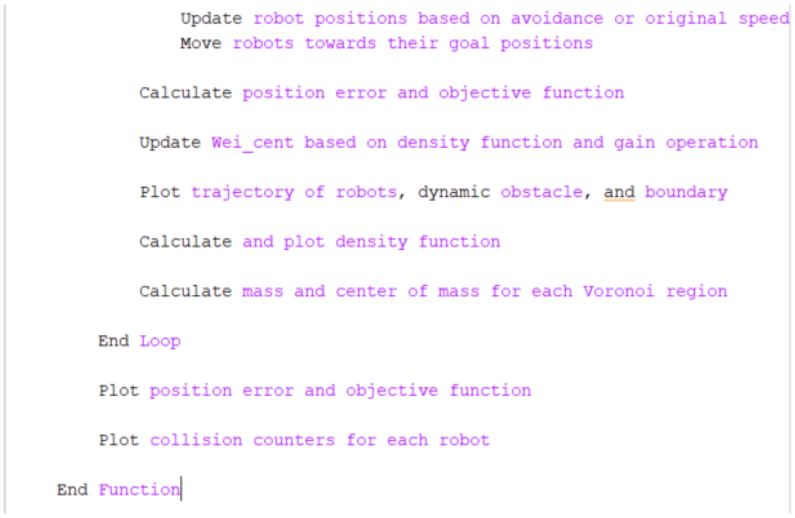
**Algorithm 3** Velocity Obstacle to find best feasible control.

```
1.  for i = 0: n do
2.      u ← sample controls from set of all controls U
3.      $t_{lim}$ ← sample time limit ∈ (0, max)
4.      free ← true
5.      min ← ∞
6.      for all Moving Obstacles B do
7.          let D(t) = distance between A(t,u) and B(t)
8.          $t_{min}$ ← solve min (D(t)) for t ∈ $[0, t_{lim}]$
9.          d ← D ($t_{min}$)
10.         if d < $r_A + r_B$ then
11.             free ← false
12.         end if
13.     end for
14.     if $\|u - u^*\|$ < min then
15.         min ← $\|u - u^*\|$
16.         argmin ← u
17.     end if
18. end for
19. return argmin
```

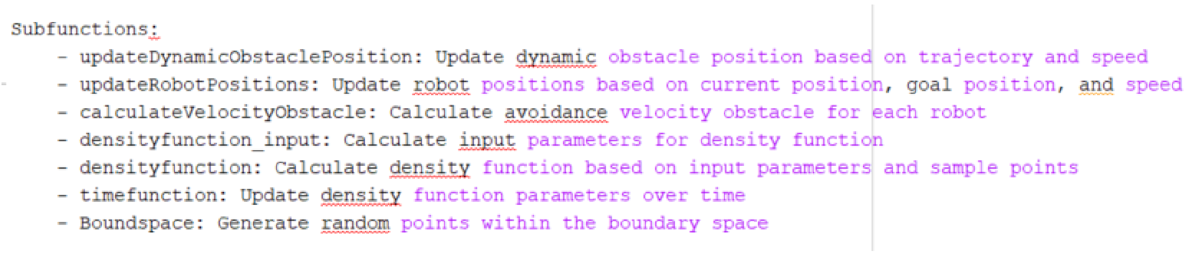
## Appendix E: Pseudocode of Modified Lloyd’s Algorithm with Velocity Obstacle.



```
function densityfunctionlloyds_v3(n, sample_num, iter_num, Bound, Pos_robot, prop_den_func)
    Initialize Parameters:
        if arguments are not provided:
            Set default values for n, sample_num, iter_num, Bound, Pos_robot, and prop_den_func

    Initialize Variables:
        Initialize robot_velocity, robot_radius, obstacle_radius, relative_velocity, collision_counter_robot,
                                   total_collisions, delta_t, g, n_g, Mn, num_gau, u, sigma, rou
        Initialize initial_position_obstacle, dynamic_obstacle_velocity
        Initialize Wei_cent, Traj_robot, colorpath, obj_function, pos_error

    Main Loop:
        Loop for it = 0 to iter_num
            Update dynamic_obstacle_position
            For each robot i from 1 to n:
                Check for avoidance based on distance between robot and obstacle
                Check if robot has reached goal position
                If avoidance needed or goal reached:
                    Calculate velocity obstacle for avoidance
                Else:
                    Set original robot speed

                Calculate distance between robot and dynamic obstacle
                Check for collision

                Plot robots with specified colors
```

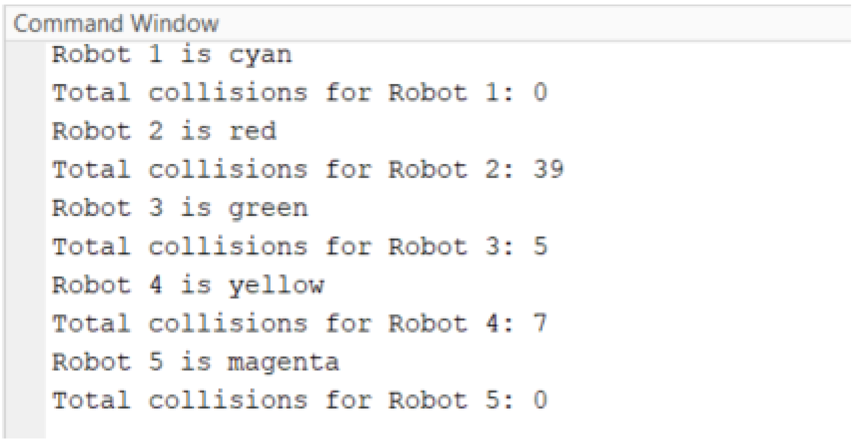


```
                Update robot positions based on avoidance or original speed
                Move robots towards their goal positions

            Calculate position error and objective function

            Update Wei_cent based on density function and gain operation

            Plot trajectory of robots, dynamic obstacle, and boundary

            Calculate and plot density function

            Calculate mass and center of mass for each Voronoi region

        End Loop

        Plot position error and objective function

        Plot collision counters for each robot

    End Function
```



```
Subfunctions:
    - updateDynamicObstaclePosition: Update dynamic obstacle position based on trajectory and speed
    - updateRobotPositions: Update robot positions based on current position, goal position, and speed
    - calculateVelocityObstacle: Calculate avoidance velocity obstacle for each robot
    - densityfunction_input: Calculate input parameters for density function
    - densityfunction: Calculate density function based on input parameters and sample points
    - timefunction: Update density function parameters over time
    - Boundspace: Generate random points within the boundary space
```

Appendix F: Command Window that shows Total Number of Collisions between Robots and Dynamic Obstacle.


Command Window

```
Robot 1 is cyan
Total collisions for Robot 1: 0
Robot 2 is red
Total collisions for Robot 2: 39
Robot 3 is green
Total collisions for Robot 3: 5
Robot 4 is yellow
Total collisions for Robot 4: 7
Robot 5 is magenta
Total collisions for Robot 5: 0
```