UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: SIMULATION OF COLLISION AVOIDANCE ALGORITHMS IN 2D USING VORONOI DIAGRAMS

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor:

Cuenca Macas Leduin José

Tutor:

Ph.D. Pineda Arias Israel Gustavo

Urcuquí, enero del 2022

Urcuquí, 20 de enero de 2022

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2022-00003-AD

A los 20 días del mes de enero de 2022, a las 11:00 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. ARMAS ARCINIEGA, JULIO JOAQUIN , Ph.D.
Miembro No Tutor	Dr. ANTON CASTRO , FRANCESC , Ph.D.
Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.

El(la) señor(ita) estudiante **CUENCA MACAS, LEDUIN JOSE**, con cédula de identidad No. **1105019937**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **Simulation of Collision Avoidance Algorithms in 2D using Voronoi Diagrams** , previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Dr. ANTON CASTRO , FRANCESC , Ph.D.	10,0
Tutor	Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. ARMAS ARCINIEGA, JULIO JOAQUIN , Ph.D.	10,0

Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

CUENCA MACAS, LEDUIN JOSE
Estudiante

Firmado electrónicamente por:
LEDUIN JOSE CUENCA MACAS

Firmado electrónicamente por:
JULIO JOAQUIN ARMAS ARCINIEGA

Dr. ARMAS ARCINIEGA, JULIO JOAQUIN , Ph.D.
Presidente Tribunal de Defensa

Firmado electrónicamente por:
ISRAEL GUSTAVO PINEDA ARIAS

Dr. PINEDA ARIAS, ISRAEL GUSTAVO , Ph.D.
Tutor

Signé électroniquement par
FRANCESC ANTON CASTRO
cn=FRANCESC ANTON CASTRO, c= EC
Date: 2022.01.20 13:37:13 ECT

FRANCESC ANTON CASTRO

Dr. ANTON CASTRO , FRANCESC , Ph.D.
Miembro No Tutor

DAYSY MARGARITA MEDINA BRITO

Firmado digitalmente por DAYSY MARGARITA MEDINA BRITO
Fecha: 2022.01.20 11:53:49 -05'00'

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Leduin José Cuenca Macas**, con cédula de identidad **1105019937**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad del autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, enero de 2022.

Firmado electrónicamente por:
LEDUIN JOSE CUENCA MACAS

Leduin José Cuenca Macas

CI: 1105019937

Autorización de publicación

Yo, **Leduin José Cuenca Macas**, con cédula de identidad **1105019937**, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, enero del 2022.

Firmado electrónicamente por:
LEDUIN JOSE CUENCA MACAS

Leduin José Cuenca Macas

CI: 1105019937

Dedicatoria

A la memoria de mi padre.

Para mi madre, hermanas, familia y amigos.

Leduin José Cuenca Macas

Agradecimiento

Mi agradecimiento a mi tutor Israel Pineda Ph.D. por el apoyo técnico brindado en el desarrollo de este trabajo. Gracias a Fis. Anthony Ramos, Ing. Fernando Zhapa, Ing. Oscar Guarnizo por su buena voluntad para ayudarme cuando me encontré atascado en la implementación del proyecto. Toda mi carrera universitaria no se hubiera completado sin el apoyo incondicional de mi familia. También, agradezco a los amigos que me acompañaron a lo largo de la carrera universitaria, cada uno contribuyó de una forma u otra a este logro. Gracias a mis profesores por ser parte de mi inspiración.

Leduin José Cuenca Macas

Abstract

This work solves the *Collision Avoidance* problem in a simulation of a centralized system of holonomic multi-agents in a two-dimensional space free of static obstacles. For this, we propose an implementation of three modules in an architecture: Threat Assessment Strategy (TAS), Path Planning Strategy (PPS), and Path Tracking Strategy (PTS). The Buffered Voronoi Cells represent the TAS. The PPS modules use two algorithms: the Analytical Geometric Algorithm (AGA) and the Receding Horizons Control (RHC) based on Quadratic Programming (QP) Algorithm. Finally, PTS controls the tracking according to fixed distance magnitudes in each iteration. The analysis of the results considers the computational execution time, the number of steps until convergence, and the calculation of optimal values. Also, these results are compared with the Optimal Reciprocal Collision Avoidance (ORCA) algorithm. In this way, our proposal successfully addresses and solves the collision avoidance problem but takes more execution time and number of steps compared with the ORCA algorithm. Besides, the number of steps of AGA is closer to ORCA, producing promising results with an accuracy of 95%.

Keywords:
Collision Avoidance, Voronoi Diagrams, Convex Optimization, Quadratic Programming, Path Planning, Simulation.

Resumen

El presente trabajo resuelve el problema de *prevención de colisiones* en una simulación de un sistema centralizado de multiagentes holonómicos en un espacio bidimensional libre de obstáculos estáticos. Para ello, proponemos una implementación de una arquitectura con tres módulos para cubrir las siguientes estrategias: Estrategia de Evaluación de Amenazas (EEA), Estrategia de Planificación de Rutas (EPR) y Estrategia de Seguimiento de Rutas (ESR). Las celdas de Voronoi amortiguadas representan la EEA. Los módulos con la EPR utilizan dos algoritmos: el Algoritmo Geométrico Analítico (AGA) y el algoritmo de Control de Horizontes en Retroceso (CHR) basado en Programación Cuadrática (PC). Finalmente, la EPR controla el seguimiento según magnitudes de distancia fijas en cada iteración. El análisis de los resultados considera el tiempo de ejecución computacional, el número de pasos hasta la convergencia y el cálculo de valores óptimos. Además, estos resultados se comparan con el algoritmo de Prevención de Colisiones Recíproco Óptimo (PCRO). De esta forma, nuestra propuesta aborda y resuelve con éxito el problema de prevención de colisiones, pero requiere más tiempo de ejecución y número de pasos en comparación con el algoritmo de PCRO. Además, el número de pasos del AGA está más cerca del algoritmo de PCRO, produciendo resultados prometedores con una precisión del 95%.

Palabras Clave:
Prevención de Colisiones, Diagramas de Voronoi, Optimización Convexa, Programación Cuadrática, Planificación de Rutas, Simulación.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Collision Avoidance (CA) is the process of preventing two or more physical objects from having intersecting boundaries in space-time, taking into account several variables like time, distance, and the parts of the involved objects. In this way, CA is widely studied due to its practical applications, mainly in path planning for ships, autonomous robots, aircraft, and unmanned aerial vehicles, using different mathematical and computational techniques such as geometric analysis, control modeling with optimization, game theory, dynamical systems, and artificial intelligence [4]. Therefore, this problem challenges researchers to simulate the natural ability of complex living beings or processes to avoid physical collisions and react accurately.

The way to deal with CA is usually subject to its immediate application or its limitations. Therefore, the research addresses the problem of CA based on the human presence (such as Advanced Driver Assistance Systems) or absence (such as fully Automated Driving Systems) within the execution of the kinematic process [4]. Also, researchers initially focus on systematizing the flow of information through an architecture [5, 6]. The modules of the architecture are mainly Threat Assessment Strategy (TAS) to ensure collision-free navigation, Path Planning Strategy (PPS) that allows the formulation of the path of mobiles avoiding obstacles, and Path Tracking Strategy (PTS) as a controller of route tracking of a possible re-planned path. TAS calculates the risk of collision in an unwanted event, providing the appropriate information to avoid a collision. The mobile then re-plans the

movement, and the PTS tracks the newly planned route [1, 7].

The construction of the architecture modules depends on several interrelated factors. Thus, the TAS methods are based on time, space, and the parts of the objects, as mentioned before. Besides, selecting a reliable PPS method requires considering several aspects, such as *a priori* knowledge about the environment, obstacles, and trajectory types. In this way, the primary PPS approach and methods are based on the configuration of the space, the introduction of abstract geometric structures in the system, the initial and final positions, computational resources, kinematic complexity, implementation of physical concepts (such as attraction and repulsion), optimization, interpolation, and even artificial intelligence. In the case of the PTS, the methods are based on a similar way of PPS [1].

The management of the information obtained in the movement process is also decisive for categorizing the system where the CA problem is solved. Thus, we can separate the CA systems into centralized and decentralized systems. The first category has a controller to handle all the information. This controller receives the information, processes it according to the architecture specifications, and finally sends it to the agents in motion. On the other hand, each agent in decentralized systems has to process the information perceived in the environment through sensors and communicate individually with the other agents [3].

In the present work, we dealt with CA algorithms for centralized autonomous multi-agent systems. We use the safe distance-based method as TAS, considering aspects like online path planning, without obstacles, and free trajectory to apply a combination of Optimization-based and Geometrical-based Path Planning strategies with breaking-deadlock heuristics as Path Tracking methods. The project proposal involves using Voronoi Diagrams, Analytical Geometrical Algorithm (AGA), and Quadratic Programming (QP) based Receding Horizons Control (RHC) Algorithm. The generated algorithms only require detecting the relative positions with a centralizing character. Therefore, it is very suitable for online deployment, as it does not require a concurrent communication network. We demonstrate the capabilities of our algorithm by comparing it to the Optimal Reciprocal Collision Avoidance Algorithm (ORCA) in a benchmark simulation scenario, and we present the results of over 2160 experimental trials in total.

1.2 Problem statement

An environment represents a physical space that contains tangible objects. In this way, a dynamic environment is a physical space with elements that change their nature over time and the space that contains them. *Movement* is a phenomenon involved in these changes. Nevertheless, the movement, in turn, is limited by the environment. For example, the nature of the environment or other static or dynamic objects prevent direct movement from the starting point to the arrival point of the initially mentioned object. Because of this, the object has to adjust its movement to reach the point of arrival satisfactorily. This satisfaction category has to do mainly with minimizing the wear of the object, the time to reach the end position, and the distance traveled. The distance problem has been approached as a path planning problem, while the object integrity problem has a place in CA.

CA is combined with various techniques to achieve a specific goal. One of these techniques is treating the information resulting from relating the moving object with the surrounding environment, including the movement motivation of the other objects, as a dynamical system [8, 9]. Also can happen that an environmental motivation subordinates the particular movement of all objects. In addition, it is essential to consider how human activity influences the movement of the system. We speak of an automated system when this influence is minimal in the real-time kinematics of the objects but with *a priori* intervention in establishing the objectives and how they are achieved. According to the development of technology, the presence of these systems becomes more notable, making their study essential.

In addition to the physical sciences, we can use mathematical-related disciplines, such as geometry, algebra, and computer technology tools, to solve the CA problem and even address path planning. So, we can analyze and obtain algorithms and numerical results. The theorizing of the problem must be complemented with the implementation in practice since there are variables with a chaotic behavior in reality. Despite this, a theoretical analysis can become the cornerstone in practical systematization. Computer systems are especially considering bringing CA to the digital field and thus represent an advantage in data calculation and analysis. In this way, computer systems can represent an advantage

by simplifying and delimiting the complexity of dynamic environments to understand the phenomena related to their changing nature.

Currently, it is necessary to solve the CA problem in semi-autonomous or autonomous machines, like ships, aircraft, drones, terrestrial vehicles, or robots, increasingly demand a higher level of complexity that must be tackled optimally and effectively. In this sense, CA and problems related to path planning require considerable research to enhance its performance [10, 11]. Algorithms and their respective implementation are analyzed as computer programs to achieve sufficient execution time, distance traveled, and wear of the object [12]. In addition, there is considerable variability in the performance of these computer programs due to the increasing variety of implementations, either mainly by programming languages, architectures, CA strategies, embedded systems, or others.

For all these reasons, this project aims to contribute to the computational analysis of the CA problem for multi-agents in movement in a two-dimensional environment without static obstacles. The proposal to solve the CA problem is based on geometrical structures such as the Voronoi diagrams. Furthermore, we implement AGA and an optimizable version based on QP-RHC to solve the path planning problem. Finally, the results are compared with one of the most used algorithms for CA in multi-agent movement: the ORCA algorithm.

1.3 Objectives

1.3.1 General Objective

Implement a centralized computational simulation for moving multi-agents to avoid collisions.

1.3.2 Specific Objectives

- Develop a CA simulation using Voronoi Diagram, AGA, and QP-based RHC Algorithm in a two-dimensional space free of static obstacles.
- Compare execution time, number of steps, and traveled distance until convergence of the proposed method with other methods found in the literature review.
- Analyze visually and dynamically the results in a two-dimensional simulation.

Chapter 2

Theoretical Framework

This chapter introduces the necessary concepts for understanding the present work. In this way, it starts approaching CA strategy architectures with all its sub-modules, then explains the mathematical and geometrical background of the Voronoi Diagram. Finally, it details the basic Analytical Geometrical Algorithm (AGA), Quadratic Programming (QP) Based Receding Horizons Control (RHC) Algorithm and Optimal Reciprocal Collision Avoidance (ORCA) Algorithm.

2.1 Collision Avoidance

CA is the process of preventing two or more physical objects from intersecting in space-time, taking into account several variables like the time, distances, movement, and the parts of the involved objects. A successful CA demands a complex system that considers various factors such as calculation costs, constraints (vehicle dynamics, obstacles, and environment), and incorporating the most advanced computational and sensing devices. Architectures classify and order the system requirements and provide control over the information flow [1].

There are at least two common types of CA Design Control Architecture [5, 6, 13]. First, the Multi-Layer CA System divides the responsibility of different objectives into layers. This system is the most used and is also known as the Guidance and Navigation Control System. The second is the Unified-Design CA System. This system has two combined blocks for integrated objectives and identical control inputs. From both architectures, it is apparent that a good CA strategies architecture usually comprises several sub-modules,

which include Threat Assessment Strategy (TAS), Path Planning Strategy (PPS), and Path Tracking Strategy (PTS).

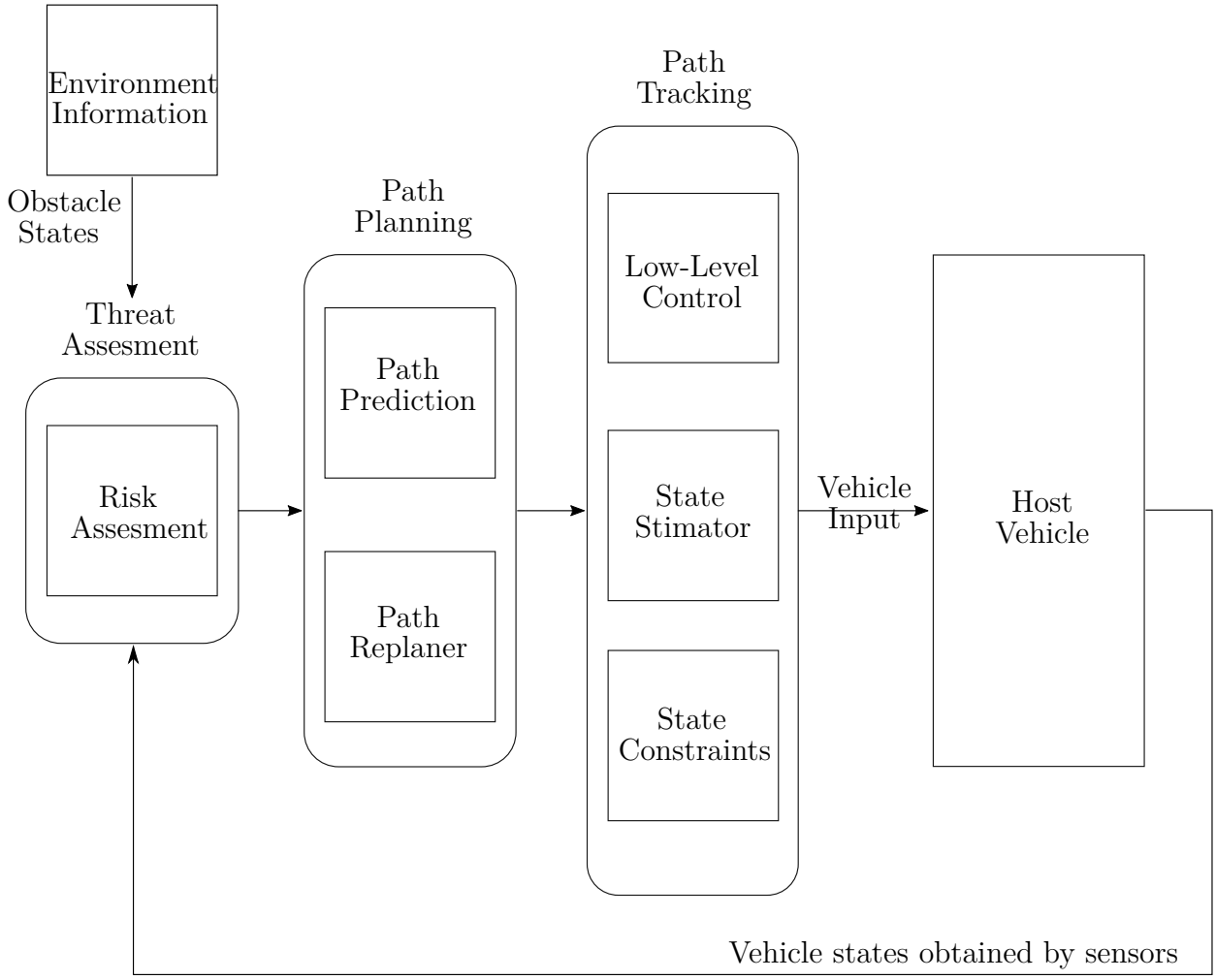


Figure 2.1: Multi-layer CA architecture consists of several modular strategies. Source: [1].

2.1.1 Threat Assessment Strategy

TAS provides an assessment and subsequent warnings of the potential threat to CA. The result of the TAS calculation is the key to triggering the subsequent actions of the AC architecture. In this way, TAS feeds the decision-making strategy on the appropriate action of the moving object. In addition, TAS takes care of the threshold or tolerance limits around obstacles or any physical object in the environment. The philosophy behind TAS is illustrated in Figure 2.2. Once this object is violating certain conditions, the CA system activates the path planning block to re-plan the current trajectory. The risk can be measured by any means, including distance, speed, and acceleration of the moving object

relative to the elements of the environment [4].

The idea of a safe distance threshold is to have an invisible safe region boundary around the physical objects, regardless of their dimension. However, the strategy will be less effective as the dimensions of the obstacles increase. Also, it does not provide sufficient information to output an optimal re-planned path. Ground surface characteristics, weather conditions could also be used to determine the breaking distance. The combinations of this method with other TAS methods ensure a timely CA maneuver activation.

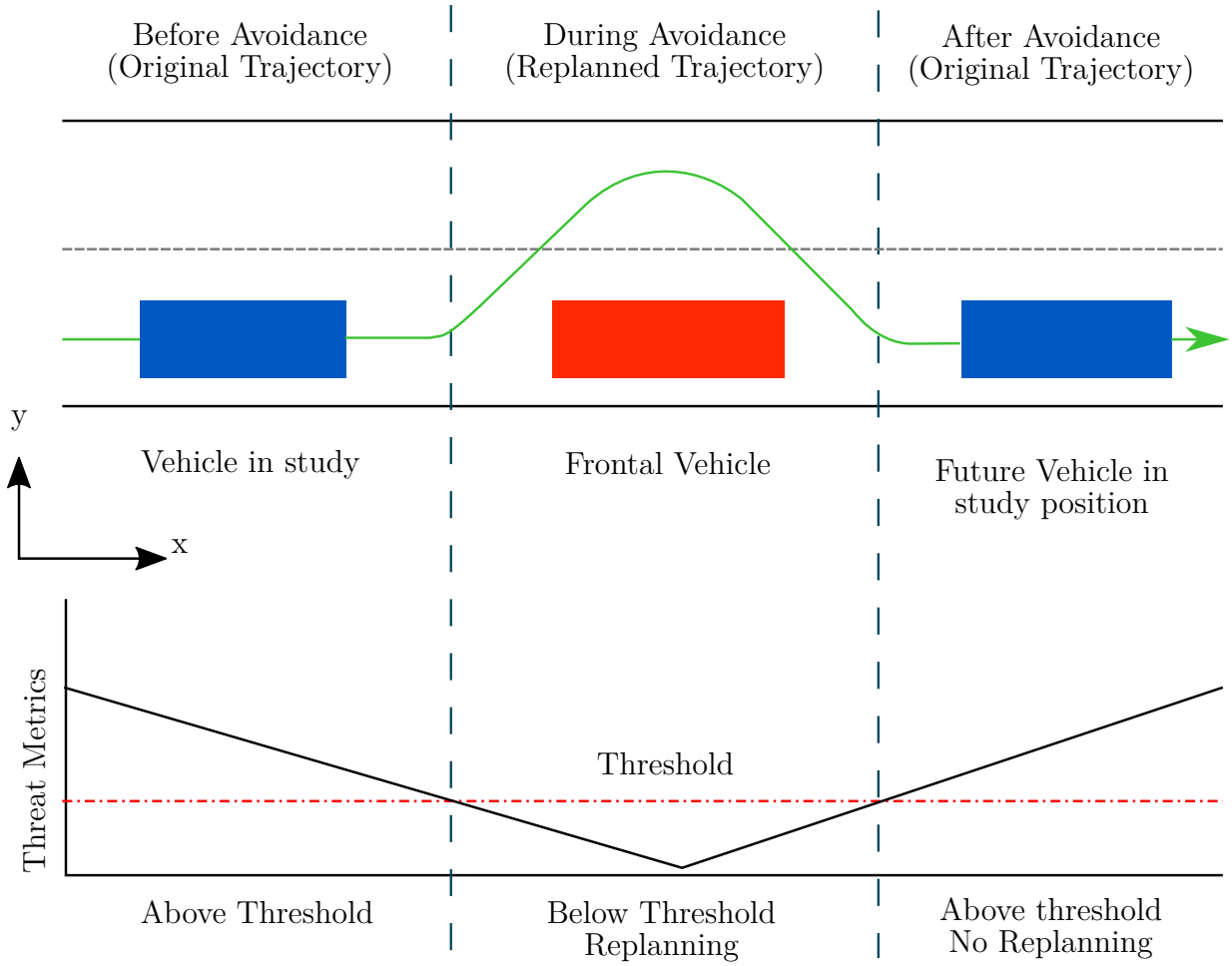


Figure 2.2: Threat assessment general formulations, where the current trajectory of the studied vehicle is re-planned once the threshold is violated. Source: [1].

2.1.2 Path Planning Strategy

The PPS re-plans a collision-free route while the vehicle continues moving once the TAS identifies the potential collision threat. This new route may be different from the previous

route planned by the PPS. In addition, an ideal PPS considers the risk of collisions involved in changing the current kinematics of the vehicle. First, the strategy guarantees enough space for the mobile to maneuver without frontal or side collisions. Then it needs to make sure that there is no potential risk with another obstacle after the maneuver. Finally, the new trajectory must consider the mechanical limitations and internal implications of the moving object. Some considerations for selecting a PPS are presented following.

- **Offline and online path planning:** Offline PPS offers many advantages in specific scenarios, such as previously known environments. However, these strategies do not help if the robot navigates in an unfamiliar environment, like in an urban area with high traffic. Online route planning is preferable in this situation due to its ability to quickly transmit information about nearby robots using sensors. In this way, the path calculation time can be shortened [14, 15].

- **Environment and obstacles:** There are two types of CA environments, which are differentiated by having dynamic or static obstacles. An *a priori* path planner allows the recognition of obstacles. However, for unknown static obstacles, online path planning acts as PPS algorithms. These situations usually involve avoiding multiple static obstacles. Things are more complicated when it comes to dynamic obstacles and even more so in a real-time application because the chance of encountering dynamic obstacles is relatively higher than static obstacles. Therefore, the online path planning for dynamic obstacles must be an algorithm with short computational time and suitable for a routine-tasked robot with static obstacles. In contrast, online path planning is preferable for robots that work in an unknown environment, where collision with dynamic obstacles might occur [1].

- **Trajectory types:** Vehicles and car-like robots move in a non-holonomic way; that is, they cannot make an abrupt movement due to mechanical constraints. On the other hand, the mobiles that can move in any direction have holonomic characteristics. Therefore, PPS ensures navigation through these paths according to the nature of the objects in movement. Furthermore, there are many other types of trajectory creation, for example, navigating through an unfamiliar environment where fast

obstacles may appear and in alley environments. Thus, the selection of precise algorithms is directly related to the production by the PPS of a maneuverable re-planned route [1].

2.1.3 Path Tracking Strategy

PTS algorithms act as a path following controller to ensure the vehicle or mobile robot successfully avoids collisions. A good PTS timely tracks the reference re-planned path by producing the required low-level control actions and output suitable interventions. For this reason, different scenarios demand individual CA actions. Then, we present the following PTS.

- **Geometrical Approach:** A geometrical-based path tracking approach utilizes a look-ahead point and considers predicting future vehicle (x, y) positions [16]. The future positions of the vehicle rely heavily on its current position. Therefore, the distance between its current and future position (look-ahead points) is significant for formulating this approach. Furthermore, the distance is directly proportional to the risk of lane departure after the CA navigation. Thus, ensuring a small look-ahead distance allows the vehicle to follow the path smoothly. This decision follows the dynamic constraints of the vehicle. Besides that, as the geometrical-based approach relies on the location of the vehicle, its speed and velocity are essential elements. Since velocity is the rate of distance change over time, an uncontrollable change in speed produces uncontrolled anticipation of vehicle future positions. Therefore, by discarding velocity, the vehicle might fail to follow the re-planned path correctly. Another drawback of the geometrical approach for the path tracking strategy of the CA system is that it does not ensure an obstacle-free path because it ignores the formulations of safe distance from the vehicle to the obstacle. However, this can be prevented by having a TAS in the entire CA architecture.

- **Model-based Control Approach:** The most well-known method in this approach is Model Predictive Control (MPC). MPC was initially developed for processes in chemical engineering because there was a need to control complex Multiple-Input Multiple-Output systems while respecting the process constraints. However, with

the recent advent of technology, usage has outreached other fields, including CA. An MPC-based controller is a reliable control as it utilizes the knowledge of the vehicle model. Therefore, the vehicle predicts the optimized result (position and speed) at a given prediction horizon at each sampling time [17].

MPC can control under-actuated or over-actuated systems, beneficial, especially in CA, where the intervention of the actuator is crucial in path tracking actions. Besides, the cost function of MPC also includes the system constraints in the optimization problem. Thus, the cost function with system constraints provides an optimization objective solved at each sampling period. This process, in return, promises reliable control actions for CA and gives MPC an advantage compared to other control techniques. Furthermore, MPC can prevent sudden movements of the vehicle when avoiding the collision with MPC.

Another advantage of MPC over other controllers is its intuitive tuning options; this includes the future prediction of the manipulative input, control horizons, and the concept of the constraint that can manipulate the weights on the slip angle, steering angle, and lateral position.

However, for a complex CA situation, the MPC control strategy needs to be improved by utilizing multiple objective functions, reference trajectory, and Nonlinear Model Predictive Control. Due to better computational devices, more complex MPC formulations that were infeasible in the past are now possible.

2.2 Voronoi Diagram

The following explanation is based on the work of Zhou *et al* [18].

Given a set of two or more but a finite number of distinct points in the Euclidean plane, we associate all the locations in that space with the closest members of the set of points using the Euclidean distance. The result is a 2d-tessellation into a set of the regions associated with members of the point set. This tessellation is called *Voronoi Diagram* generated by the point set, and the regions constituting the Voronoi diagram are called *Voronoi cells*. Figure 2.3 shows this description graphically.

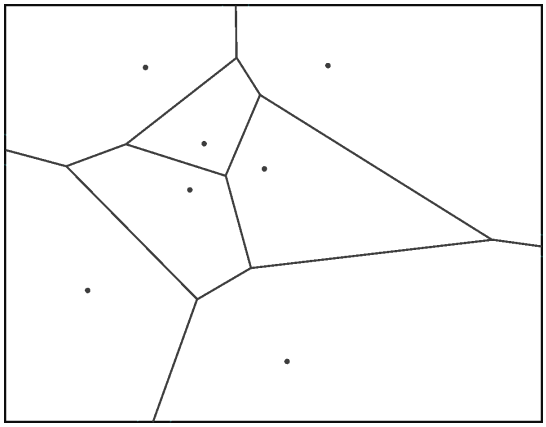


Figure 2.3: Voronoi tessellation generated from a set of points.

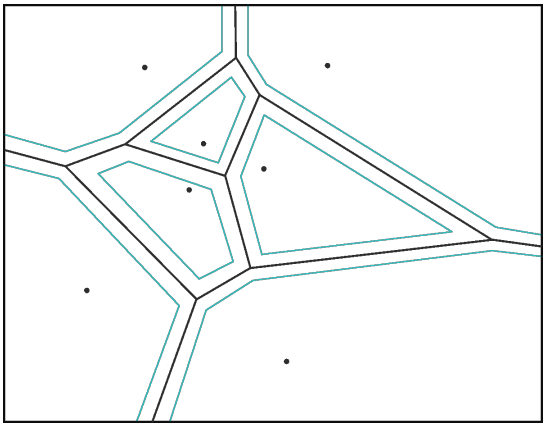


Figure 2.4: Buffered Voronoi Cells colored with light blue.

Supposing that we have a set of n points located in the Euclidean plane. The positions of the n points are labelled by $p_1, \cdots, p_N$ with Cartesian coordinates $((x_{11}, x_{12}), \cdots, (x_{n1}, x_{n2}))$ or location vectors $x_1, \cdots, x_n$. The n are distinct in the sense that they are not in the same coordinates, i.e. $x_i \neq x_j$ for $i \neq j, i, j \in I_n = \{1, \cdots, n\}$. Let p be an arbitrary point in the Euclidean plane with coordinates (x_1, x_2) or a location vector x. Then the Euclidean distance between p and p_i is given by

$$d(p, p_i) = \|x - x_i\| = \sqrt{(x_1 - x_{i1})^2 + (x_2 - x_{i2})^2}. \tag{2.1}$$

If p_i is the nearest point from p or vice versa, we have the relation:

$$\|x - x_i\| \leq \|x - x_j\| \text{ for } j \neq i, i, j \in I_n. \tag{2.2}$$

In this case, p is assigned to p_i (see Figure 2.3).

Definition 1 (Voronoi Diagram) *Let be $P = p_1, \cdots, p_n \subset \mathbb{R}^2$ a set of points, where $2 < n < \infty$ and $x_i \neq x_j$ for $i \neq j$; $i, j \in I_n$. We call the region given by*

$$V(p_i) = \{x \mid \|x - x_i\| \leq \|x - x_j\|;\ \forall j \neq i;\ i, j \in I_n;\ x \in \mathbb{R}^2\} \tag{2.3}$$

the Voronoi polygon associated with p_i, and we call the set given by

$$\mathcal{V} = V(p_1), \cdots, V(p_n) \tag{2.4}$$

the Voronoi diagram generated by P.

We call p_i of $V(p_i)$ the *generator* of the ith Voronoi polygon, and the set $P = p_1, \cdots, p_n$ the *generator set* of the Voronoi diagram $\mathcal{V}$. For brevity, we may write V_i for $V(p_i)$. Also, we may use $V(x_{i1}, x_{i2})$ or $V(x_i)$ when we want to emphasize the coordinates or location vector to the generator point p_i. In addition, we may use $\mathcal{V}(P)$ when we want to explicitly indicate the generator set P of $\mathcal{V}$. Equivalently, we can write Equation 2.3 as

$$\mathcal{V}_i = \left\{ p \in \mathbb{R}^2 \middle| \left(p - \frac{p_i + p_j}{2} \right)^\intercal (p_j - p_i) \leq 0, \forall j \neq i \right\}, \tag{2.5}$$

The safety radius is also important to define the *Buffered Voronoi Cell (BVC)* (see Figure 2.4):

Definition 2 (Buffered Voronoi Cell) *For the set of points in a 2D plane $\mathbb{R}^2$ from Definition 1 and given a safety radius $r \in \mathbb{R}$, the Buffered Voronoi Cell (BVC) of the ith point of this set is defined as*

$$\overline{\mathcal{V}}_i = \left\{ p \in \mathbb{R}^2 \middle| \left(p - \frac{p_i + p_j}{2} \right)^\intercal p_{ij} + r\|p_{ij}\| \leq 0, \forall j \neq i \right\}. \tag{2.6}$$

BVC tries to take into account the physical size of robots. For this, the boundary of its Voronoi cell needs to compress by a safety radius. Some properties of BVC are described in the following lemma:

Lemma 1 (Properties of Buffered Voronoi Cell) *For a set of n points with safety radius r_s we have:*

(i) $\overline{\mathcal{V}}_i \subset \mathcal{V}_i$,
(ii) $\forall p\prime_j \in \overline{\mathcal{V}}_j, i \neq j, \|p\prime_i - p\prime_j\| \geq 2r_s$,
(iii) $\overline{\mathcal{V}}_i \cap \overline{\mathcal{V}}_j = \emptyset, \forall i \neq j$.

The demonstration is also explained in the work of Zhou *et al* [18].

2.3 QP-Based RHC Algorithm

The explanation below is based on the work of Mattingley *et al* [19].

The Quadratic Programming Based Algorithm has its foundations on the Receding Horizon Control (RHC), also known as Model Predictive Control (MPC), a feedback control technique that became popular in the 1980s [20, 21]. RHC applications include various

practical scenarios, such as industrial and chemical process control, supply chain management, stochastic control in economics and finance, revenue management, hybrid vehicles, automotive and aerospace applications.

With RHC, an optimization problem is solved at each time step to determine a plan of action over a fixed time horizon. Then, the first input from this plan is applied to the system. Next time we repeat the planning process, solving a new optimization problem with the time horizon shifted one step forward. The optimization problem takes into account estimates of future quantities based on available information at each time step. The control policy involves feedback since real-time measurements are used to determine the control input.

2.3.1 Receding Horizon Control

The RHC procedure works as follows. At time t, we consider a time interval extending T steps into the future. $t, t+1, \cdots, t+T$. We then carry out the following steps:

1. **Form a predictive model.** Replace all uncertain quantities over the time interval with their current estimates using information available at time t.

2. **Optimize.** The RHC optimization problem takes the form

   Problem 1 (Receding Horizon Control Problem)

$$\text{minimize} \frac{1}{T+1} \sum_{\tau=t}^{t+T} \hat{l}_{\tau|t}\left(\hat{x}_\tau, \hat{u}_\tau\right) \tag{2.7}$$

   subject to:

$$\begin{array}{lrcll} \text{(i)} & \hat{x}_{\tau+1} & = & \hat{A}_{\tau|t}\hat{x}_\tau + \hat{B}_{\tau|t}\hat{u}_\tau + \hat{c}_{\tau|t} & , \tau = t, \cdots, t+T, \\ \text{(ii)} & (\hat{x}_\tau, \hat{u}_\tau) & \in & \hat{\mathcal{C}}_{\tau|t} & , \tau = t, \cdots, t+T, \\ \text{(iii)} & \hat{x}_t & = & \hat{x}_{t|t}, & \end{array}$$

   with variables $\hat{x}_t, \cdots, \hat{x}_{t+T+1}$ and $\hat{u}_t, \cdots, \hat{u}_{t+T+1}$. The objective in problem 1 is a finite-horizon approximation of the infinite-horizon cost function J. In problem 1, we pretend to minimize the estimated objective over the time interval $t, t+1, \cdots, t+T$, subject to the estimated dynamics and constraints. The parameters in this RHC

optimization problem are the estimates $\hat{A}_{\tau|t}, \hat{B}_{\tau|t}, \hat{c}_{\tau|t}, \hat{\mathcal{C}}_{\tau|t}, \hat{l}_{\tau|t}$ for $\tau = t, \cdots, t+T$, and the current state estimate $\hat{x}_{t|t}$. The optimal input trajectory of the RHC optimization problem 1, $\hat{u}_t\prime, \cdots, \hat{u}_{t+T}\prime$, is a *plan of action* for the next T steps.

3. **Execute.** We then choose $u_t = u_t\prime$ to be the RHC input. At the next time step, the process is repeated, with updated estimates of the current state and future quantities.

We assume that $\hat{\mathcal{C}}_t$ and $\hat{l}_t$ are convex, which means that the RHC problem 1 is a convex optimization problem and can be solved using convex optimization tools. Furthermore, problems with nonconvex objectives and constraints can often be handled using sequential convex optimization, where a sequence of convex problems is solved to find local solutions to the nonconvex problem.

For some applications, constraints on the state variables can lead to an infeasible RHC optimization problem. This problem can occur if an unexpected disturbance affects the system, in which case there may not exist any control policy that keeps the system within the constraints. Infeasibility can also occur if the estimated system parameters are inaccurate so that the RHC optimization problem does not reflect the correct system behavior. There are various ways to deal with infeasibility. One strategy is to apply the planned control input from the previous time step. Another method is to allow constraint violations but penalizes them in the objective function. These constraints are referred to as soft constraints instead of the original hard constraints, which cannot be violated. Typically, the softened constraints are constraints on the state variables for which some violation may be acceptable. For example, input constraints, such as actuator limits and trading budgets, usually cannot be violated. Various methods exist for deciding which constraints to relax, based on importance rankings. Although we cannot guarantee that these methods recover feasibility in future time steps, they allow the controller to compute an acceptable control input when infeasibilities occur.

The RHC policy is not optimal except in exceptional cases. In RHC, we replace the infinite horizon average cost J with a finite horizon approximation and uncertain quantities with their estimates. We do not solve the controller design problem, an infinite-dimensional nonconvex problem, and is usually intractable. Thus, RHC may not achieve the minimum possible average cost among policies that respect the constraints. Instead, RHC is a so-

phisticated heuristic, as demonstrated in various applications [19].

2.4 Analytical Geometric Algorithm

This algorithm is a simplification of the QP-Based CA Algorithm described in Section 2.3

The QP Receding Horizon Path Planning in Section 2.3 generates a control policy optimal over the planning horizon at the expense of solving a QP problem online at each time step. Furthermore, we can solve the QP with an Analytical Geometric Algorithm for the particular case of no intermediate cost terms, which executes much faster than the QP, while CA is still guaranteed.

Consider the case where the intermediate state and the control input costs are equal to zero, and the terminal cost exists. However, all the constraints of the optimization problem are the same. This simplification can be considered as a one-step greedy strategy that drives the robot to move to its goal position as soon as possible. With this simplification, the moving object should direct towards a point in the convex polygon borders closest to its goal position. This particular geometric case is almost an algorithm for avoiding collisions as a part of a more extensive swarm guidance algorithm [22].

2.5 Optimal Reciprocal Collision Avoidance Algorithm

The explanation below is based on the article Optimal Reciprocal Collision Avoidance for Multi-Agent Navigation [2].

2.5.1 Reciprocal n-body Collision Avoidance Problem

The basis of the Optimal Reciprocal Collision Avoidance is the problem of reciprocal n-body CA consists of avoiding collisions among multiple decision-making entities. This problem is defined as follows. Let there be a set of n disc-shaped robots sharing an environment. Each robot A has a current position p_A, a current velocity v_A, and a radius r_A. These parameters are part of the external state of the robot, i.e., we assume that other robots can observe them, as we can see in Figure 2.5 (a). Furthermore, each robot has a maximum speed v_A^{max} and a preferred velocity v_A^{pref}, which is the velocity the robot would assume

had no other robots been in its way (for instance, a velocity directed towards the goal of the robot with a magnitude equal to the preferred speed of the robot). Again, we consider these parameters part of the internal state of the robot and can not be observed by other robots.

The task is for each robot A to independently (and new simultaneously) select a new velocity v_A^{new} for itself such that all robots are guaranteed to be collision-free for at least a preset amount of time r when they would continue to move at their new velocity. As a secondary objective, the robots should select their new velocity as close as possible to their preferred velocity. The robots cannot communicate with each other and can only use observations of the current position and velocity of the other robot. However, each robot may assume that the other robots use the same strategy to select a new velocity. Note that this problem cannot be solved using central coordination, as the robot itself only knows the preferred velocity of each robot.

2.5.2 Preliminaries

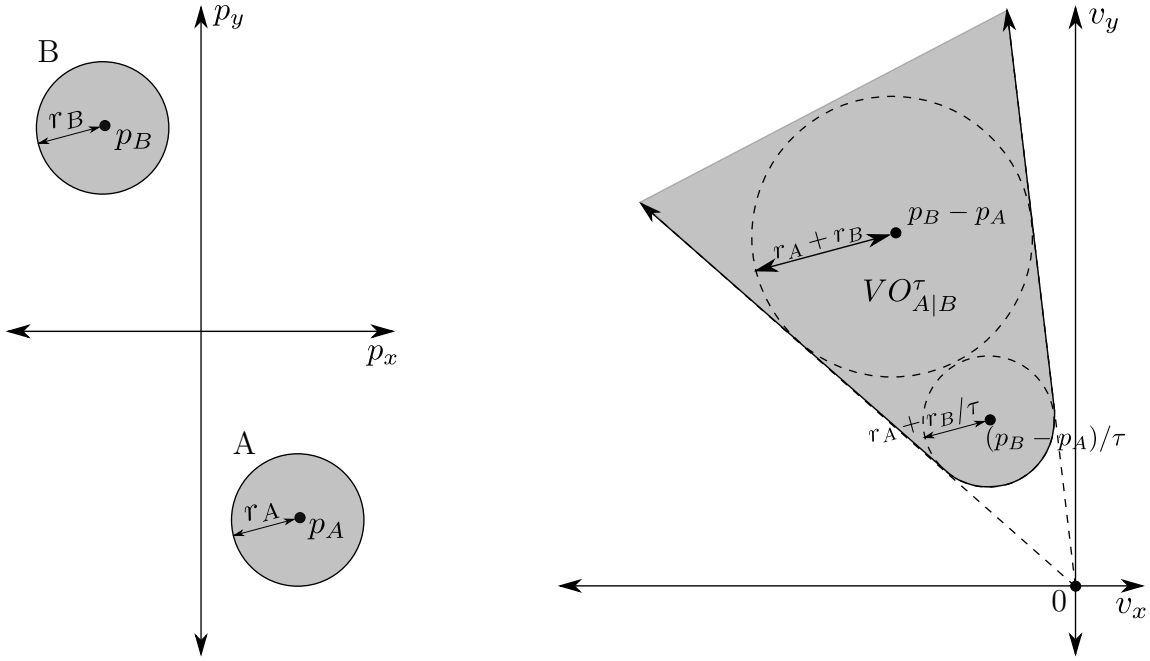
For two robots A and B, the relative velocity $\vec{v}_{B|A}$ (also $\vec{v}_{BA}$ or $\vec{v}_{B\ rel\ A}$) is the velocity of B in the rest frame of A. In this way, the velocity obstacle $VO^{\tau}_{A|B}$ for A induced by B in a time window τ is the set of all relative velocities of A concerning B that results in collisions between both robots before time τ. It is formally defined as follows. Let $D(p, r)$ denote an open disc of radius r centered at p:

$$D(p, r) = \{q | \|q - p\| < r\}, \tag{2.8}$$

then:

$$VO^{\tau}_{A|B} = \{v | \exists t \in [0, \tau] \text{ defined by } tv \in D(p_B - p_A, r_A + r_B)\}. \tag{2.9}$$

The geometric interpretation of velocity obstacles is shown in Fig. 1(b). Note that $VO^{\tau}_{A|B}$ and $VO^{\tau}_{B|A}$ are symmetric in the origin.



(a) A configuration of two robots A and B (b) The velocity obstacle

Figure 2.5: Geometrical interpretation for ORCA. Source: [2].

2.5.3 Optimal Reciprocal Collision Avoidance Definitions

The explanation below is based on the article A Novel Collision-Free Navigation Approach for Multiple Nonholonomic Robots Based on ORCA and Linear MPC [23].

Let v_A^{opt} and v_B^{opt} be the optimization velocities of A and B, respectively. Nominally, the optimization velocities are equal to the current velocities, such that the robots have to deviate as little as possible from their current trajectories to avoid collisions. Let $\partial VO^{\tau}_{A|B}$ the velocity obstacle in the boundary of its respective open disc, and u be the vector from $v_A^{opt} - v_B^{opt}$ to the closest point on the boundary of the velocity obstacle:

$$u = \left(\underset{v \in \partial VO^{\tau}_{A|B}}{\operatorname{argmin}} \|v - (v_A^{opt} - v_B^{opt})\| \right) - (v_A^{opt} - v_B^{opt}), \tag{2.10}$$

and let n be the outward normal of the boundary of $VO^{\tau}_{A|B}$ at point $(v_A^{opt} - v_B^{opt}) - u$. Consequently, u is the smallest change required to the relative velocity of A and B to avert collision with τ time. The increment $\frac{1}{2}u$ means that agent A takes half the responsibility of avoiding potential collision with agent B, while the remaining half is taken by agent B to share the responsibility of avoiding collisions among the robots in a fair way. Hence,

the set $ORCA^{\tau}_{A|B}$ of permitted velocities for A is the half-plane pointing in the direction n of starting at the point $v^{opt}_A + (1/2)u$. More formally:

$$ORCA^{\tau}_{A|B} = \left\{ v| \left(v - \left(v^{opt}_A + \frac{1}{2}u \right) \right) n \geq 0 \right\}. \tag{2.11}$$

This set is illustrated in Figure 2.5 (b). By this definition, the chosen new relative velocities do not enter $VO^{\tau}_{A|B}$, and consequently the velocities of the agent can be smoothed.

Each robot A performs a continuous cycle of sensing and acting with time step δt. In each iteration, the robot acquires the radius, the current position, and the current optimization velocity of the other robots. Based on this information, the robot infers the permitted half-plane of velocities $ORCA^{\tau}_{A|B}$ for each robot B. Thus, the set of ORCA velocities for agent A for all robots is the intersection of the half-planes of $ORCA^{\tau}_{A|B}$ induced by each other robot B:

$$ORCA^{\tau}_{A} = D(0, v^{max}_A) \cap \bigcap_{B \neq A} ORCA^{\tau}_{A|B}. \tag{2.12}$$

Note that $D(0, v^{max}_A)$ includes the maximum speed constraint on the robot A.

At last, the new velocity v^{new}_A is selected from $ORCA^{\tau}_{A}$, which is the closest to its preferred velocity v^{pref}_A amongst all velocities inside the region of permitted velocities:

$$v^{new}_A = \underset{v \in ORCA^{\tau}_A}{\operatorname{argmin}} \|v - v^{pref}_A\|. \tag{2.13}$$

If the ORCA algorithm only uses a simple robot model, then the kinematics are ignored. Consequently, the robot reaches its new position:

$$p^{new}_A = p_A + v^{new}_A \delta t \tag{2.14}$$

2.6 Multi-agent Navigation

Generating plans is a crucial problem in multi-agent collaboration, a highly active field of research, and numerous approaches have been proposed over time. An important task is developing decision-theoretic methods to have frameworks to describe, analyze, and understand planning strategies. We can categorize the systems based on the different

views of planning approaches into two classes: centralized and decentralized.

Agent behavior is often viewed in terms of collective intentions and actions. Therefore, a plan often describes a mapping from the states of the global system to the collective actions of the agents. This plan specifies the troubleshooting strategy of the agents in a unified manner. We call this a Centralized Policy (CP) for multiple agents.

In a decentralized system, the agents partially observe the global system state and make local decisions. Thus, the planned mapping from local knowledge to local actions is called a Decentralized Policy (DP) for multi-agent. Clearly, in this decentralized view, each agent needs its own DP, unlike in the centralized view where one CP specifies the joint actions of the agents [3].

2.6.1 Centralized Policies

In decision-theoretic terms, a multi-agent Markov decision process model can describe the centralized view of the system. In this way, a standard Markov decision process consists of a set of global states S, a set of joint actions A (each joint action specifies one action for each agent), a transition probability matrix $Pr(s\prime|s,a)$,the probability that the system moves from state s to state $s\prime$ after joint action a, and a reward function $r(s)$ that specifies the global utility received when the system is state s. This framework corresponds to a computational model where at any stage t, the system is described through its current global state s, which is made up of the current local states of the agents. The system then takes a joint action and evolves into one of the possible following global states based on completing the joint action. In this framework, a CP, a mapping from global states to joint actions, is precisely a policy for a Markov decision process. Figure 2.6 shows how problem-solving is carried out under such a policy (in a two-agent system with agents X and Y). The expected utility of the CP can be calculated using standard policy evaluation algorithms for Markov decision processes.

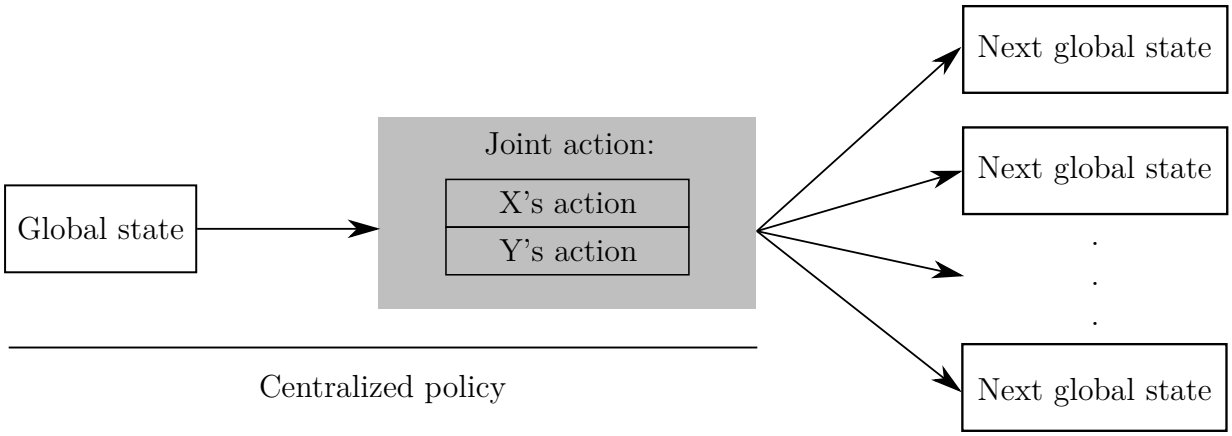


Figure 2.6: Centralized view and policy. Source: [3].

2.6.2 Decentralized Policies

In the decentralized view, an agent cannot see the local states of other agents and local actions and has to decide the following local action on its own. Thus, each agent has a partial view of the global state of the system, and different agents have partial views. However, of course, this does not necessarily mean that the agents are isolated. Instead, an essential ability of decentralized cooperative agents is their ability to communicate.

Communication expands an agent's partial view by exchanging local information not observed by other agents. So, communication is the action of agents updating each other with local state information and thus collectively discovering the current global state. In other words, the agents synchronize themselves so that they all observe the current global state. Therefore, a DP must deal with communication explicitly, mainly when communication incurs a cost or when continuous communication is not feasible. The system is modeled by each agent having an individual state space, its own local action set, and local state transition probability measure but uses a global reward function to connect the effects of the actions in the agents. Thus, it is not a standard Markov decision process. However, the DP under this framework now also explicitly includes communication decisions. Specifically, the DP should define what local action to take based on the current local knowledge of the agent, but also whether a communication is needed after its local action is completed. Typically, such decisions are based not only on current local state information but also on the history of the agent, including past states and past communications.

Figure 2.7 shows the computation model under this decentralized view: at any stage

t, each agent first decides what local action to take (per the DP). Then, decides whether there is a need for communication when the action finishes. Next, the agents enter a sub-stage where all communications occur (if any). Finally, when the communication sub-stage finishes, each agent enters the next stage with an updated set of local knowledge (one of several possible local knowledge sets based on the outcomes of actions in both agents).

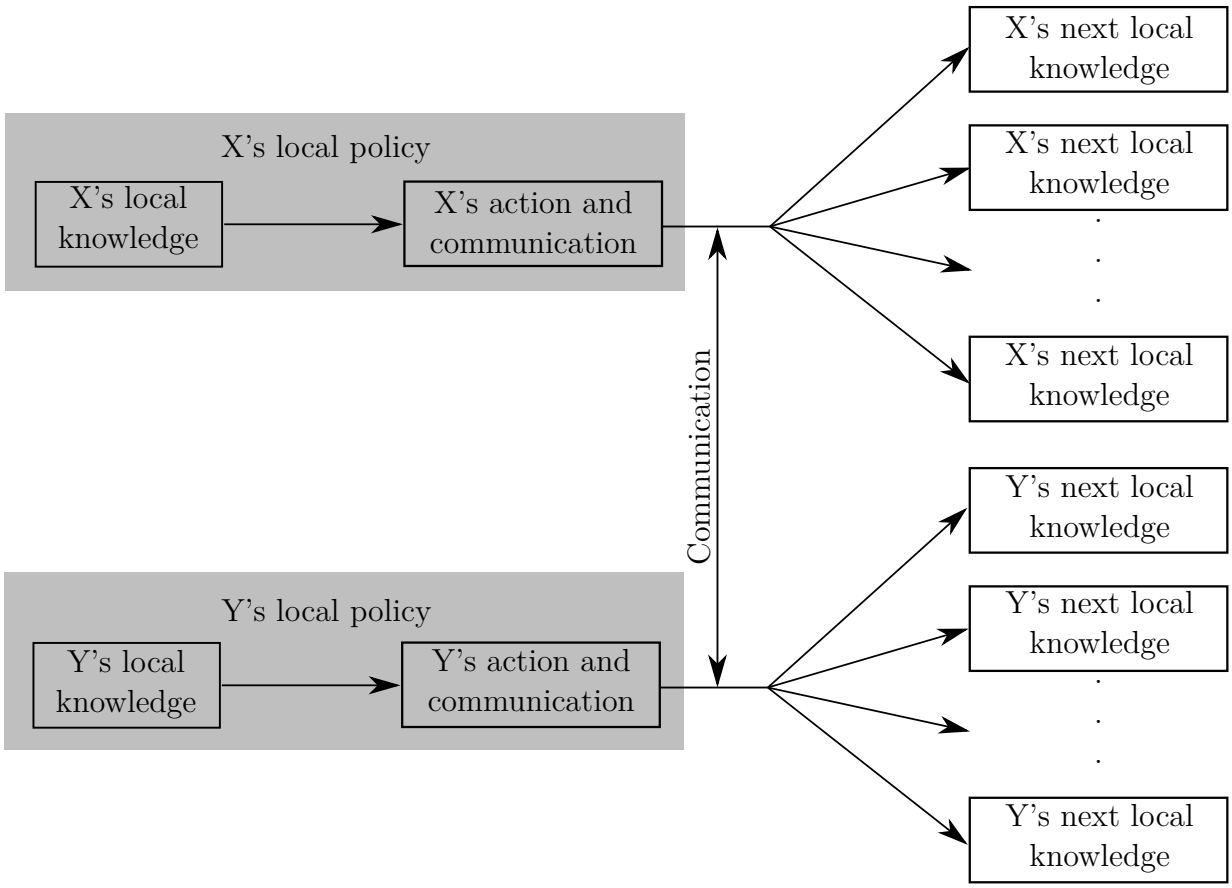


Figure 2.7: Decentralized view and policy. Source: [3].

The fundamental difference between CP and DP is how the agents relate to the global state of the environment. In this way, CP assumes the global state as a starting point, while the agents in DP do not. As a result, DP must deal with communication decisions and historical information, making it more complex than CP. However, most multi-agent systems are distributed in nature, and the agents are generally autonomous, meaning that each agent is a decision-maker for itself. Therefore, a centralized view simplifies information management (for example, assuming agents see the big picture instead of the partial view), imposing some solid assumptions or unique mechanisms to ensure the observability of the

state of the global system. On the other hand, if a decentralized vision is adopted, the objective is to develop agents with partial visions that can carry out actions effectively and implement cooperative problem-solving strategies.

Of course, this does not mean that centralized policies are invalid. On the contrary, CP and DP are related. For example, decentralized models have an advantage in their power to represent reality, but at the same time, they have complex handling of information. Generally, solving a standard Markov decision process is of PSPACE complexity, but solving a decentralized Markov decision process is of NEXP time complexity, a class of higher complexity. As a result, heuristic approaches and approximation methods for developing PD are essential. On the other hand, CP is easier to solve, and there are systematic methods to obtain them. So it is very convenient to find ways to derive DP directly from CP.

Chapter 3

State of the Art

The information presented in this chapter was based on the reviews of González *et al.* [24], Hamid *et al.* [1], and Dahl *et al.* [4].

3.1 Threat Assessment Strategies

Accurate deterministic prediction for moving objects uses model-based state propagation, perfect measurements, and a complete understanding of the intention of each mobile. In reality, perfect results of calculations rarely appear, so simplifications in formulating the problem are necessary. For example, it is reasonable to assume that vehicles continue to follow the current route for a specific time. Another simplification is to assume that the measurement data is free of noise. Therefore, using reasonable simplifications, the Threat Assessment problem can be rephrased using Threat Metrics (TM) based on the unique future behaviors of different traffic participants. The metrics we use are Time To Collision (TTC) and Distance-based [1].

3.1.1 Time-to-Collision

A widely used TM is Time-to-Collision (TTC) between two objects. The TTC values are thresholds that decide whether to enable a warning or automatic intervention [25, 26, 27]. The most used calculation is to divide the distance to the closest object by the relative speed. Also, Jansson [28] proposed a more general definition of TTC where he uses relative distance and relative velocity, given a constant relative acceleration. Moreover, Falcone *et al.* [29] used Time-to-Lane Crossing (TLC) for lane departure warning systems.

Different authors have also proposed several similar metrics. For instance, the Inverse TTC (ITTC) increases with collision risk [30]. In the work of Kiefer *et al.*, [31] the ITTC concept derives a forward-collision warning system based on experimental data on maneuvers involving last-second braking and steering.

As an alternative to TTC, Noh and Han [32] introduce Inter-Vehicle-Time (IVT), defined as the time of a mobile, given the current mobile's velocity, to travel the distance equal to the relative distance obstacle ahead. Thus, a short relative distance and a high mobile velocity yield a low IVT time. An essential difference between the IVT and TTC is that IVT is calculated based on the host's velocity, while TTC is calculated based on the relative velocity. Hence, as highlighted by Noh and Han, IVT is a good complement in situations where the TTC metric falls short in identifying the threat. For example, when two vehicles are traveling at the same velocity but close to each other, TTC attains large values even though the situation is hazardous. Another relevant TM is the Time-to-Manoeuvre (TTM) (also referred to as Time-to-X), which is the time waiting to initialize an automated maneuver to avoid a collision, e.g., Time-to-Brake (TTB), Time-to-Steer (TTS), and Time-to-Kickdown (TTK).

Moreover, Time-to-React (TTR) defines the last point in time at which an evasive trajectory still exists. For instance, Sontges *et al.* [33] have used a set-based approach to determine TTR. On the other hand, Tamke *et al.* [34] present another approach where a modified binary search algorithm computes approximated values for different TTMs to derive the TTR for multiple object scenarios.

3.1.2 Distance Domain

The disadvantage of TTC, which does not consider the obstacle's exact dimension risk in its calculation, is resolved by the safe distance-based TAS. The idea of a safe distance threshold is to have an invisible safe region boundary around an obstacle, regardless of its dimension [35, 36, 37]. A commonly used distance-based metric is the Minimal Safe Distance (MSD), defined as the minimum distance between the host and the obstacle [32]. This metric aims at situations where spatial margins are essential, e.g., when queuing at a traffic light or in a traffic jam. Ali *et al.* [36] constrain the distances from each vehicle edge to the obstacle's center of gravity to create an imaginary rectangular safe region. It

is considered the threshold and must be below a specific value for the CA maneuver to be activated. However, the drawback of this strategy is evident when avoiding more significant obstacles. It does not provide sufficient information to output an optimal re-planned path.

Nevertheless, this method, like TTC and other TAS methods, ensures a timely CA maneuver activation. For example, Brunson *et al.* [38] described Time Headway (TH), a combination of time and distance-based TAS measurement. They calculated the range of two vehicles and divided it by the host vehicle speed. However, as in previous cases, it does not ensure the best timing for the CA maneuver to be taken [39]. The main disadvantage of TH is that it oversimplifies the traffic scenario by only considering the vehicles traveling at the same consistent speed. On the other hand, by uniting the time domain with the distance, we can obtain the speed multidomain, which is used mainly for multi-agents in motion. In the following subsection, we discuss it along with other approaches.

3.1.3 Multi-Domain

In complex scenarios, however, a single Threat Metric may not be enough to characterize a situation thoroughly, and therefore multiple Threat Metrics may be needed. For example, let a scenario consist of two vehicles traveling next to each other with approximately the same velocity. With only Time-To-Collision, the threat level is low even if the inter-vehicle distance decreases to a minimum. Hence, Noh and Han [32] proposed using a combination of Threat Metrics like Time-To-Collision, Inter-Vehicle-Time, and Minimal Safe Distance to reflect better the current threat level. As research progresses in this area, implementations of a more complex CA architecture appear that provide TAS action combined with Path Planning Strategies [40, 41]. For example, Bauer *et al.* [40] combined the vehicle environment information with the trajectory planning strategies to output a safety corridor for the trajectory re-planning action. In addition, Balachandran *et al.* [41] provide a combined warning and overriding system that proposes haptic feedback to the driver before the CA maneuver occurs. Though it has many advantages, the aforementioned unified method needs precise model information and future prediction of the states.

3.1.4 Optimization Methods

Dynamic optimization has become a standard tool for decision-making in many practical problems and a wide range of areas. For example: providing fuel-efficient cell systems [42], sustainable energy systems with optimization techniques in power generation [43], or new mathematical modeling techniques [44]. A popular optimization framework widely used in the literature is Model Predictive Control (MPC). While optimization-based approaches may often require high computation power, it is possible in some cases to leverage the structure of the problem to retrieve an explicit control law, as we present in this work with the Analytical Geometrical Algorithm. This action can allow off-line pre-computation of the explicit feedback policies, reducing the online computation in Receding Horizon Control setup to a function evaluation, therefore avoiding the online solution of complex optimization programs. This characteristic is of particular interest for safety-critical and time-critical constrained applications such as automotive threat-assessment algorithms [45] [46].

The MPC-based methods can assess the threat level for forwarding CA, i.e., another mobile is ahead [47, 48]. In practice, especially for non-holonomic mobiles, the solution violates some of the constraints, as relaxing the problem's hard constraints guarantees the feasibility of the MPC problem. For example, the lateral acceleration metric combines with a different metric to cope with more complex situations. Then, the number of violated constraints is defined, such that an intervention is only triggered if the level of violation of the constraints or the lateral acceleration exceeds a given threshold. Gray *et al.* combined a look-ahead driver and a vehicle model and used an MPC controller to compute the minimum steering action necessary to avoid an obstacle [49]. Such uncertainties are considered probabilistic constraints to cope with modeling uncertainties on the driver's behavior. By computing an upper bound on the uncertainties' deviation, the constraints on the MPC problem can be refined according to, which yields a robust control scheme.

Gutjahr and Werling [50] proposed a grid occupancy and optimization-based method to find the optimal braking point for collision avoidance at low speeds, e.g., parking applications. Hult *et al.* [51] leveraged the structure of the problem in order to derive a hierarchical decomposition of the original optimization problem separating the central co-

ordination problem from the local optimal control problems on each vehicle for conflict resolution at traffic intersections. As a result, the authors claim to reduce demands on computational capabilities and information exchange significantly. More precisely, assuming that vehicles are following a predefined path through the intersection, the combinatorial part of the problem (i.e., defining the vehicle crossing order and collision-free time slots that are feasible under the vehicle dynamics and physical constraints) is separated from the problem of finding the appropriate control inputs for a given crossing order. Based on this decomposition, Hult *et al.* [52] focused later on the properties of the underlying coordination problem. First, the authors formulated a finite-time optimal control problem. Next, they proposed a primal decomposition, showing that standard sequential quadratic programming efficiently tackles the problem such that most computations can be performed in a distributed manner by the vehicles. Finally, Zanon *et al.* [53] tried to tackle some of the communication aspects of a distributed Sequential Quadratic Programming (SQP) framework for solving the optimization problem. To reduce the communication time and burden, the authors proposed an asynchronous algorithm where the sensitivity of the optimal control problem is only updated for a subset of all agents at each step and show how one can decide on this subset of agents to optimize the contraction properties of the algorithm.

3.2 Path Planning Strategies

Intelligent vehicle development was minimal before the 90's because of reduced investments in the field [54]. The Intelligent Transportation Systems (ITS) concept was born from the evolution of information technology applicable to vehicle automation. Different research centers worldwide focus on this end (e.g., California PATH, Parma University, among others), improving intelligent vehicle systems.

Shladover *et al.* [54] and Behringer and Muller [55] describe the first automated vehicles, dating back to the transition between the '80s and '90s. So, Shladover *et al.* describe longitudinal control systems (including vehicle the following control, inter-vehicular communications and a comparison between different methods), and lateral control systems (considering lateral vehicle dynamics and magnetic sensors as path reference with no path

planning involved) in order to improve the Advance Highway Systems (AHS). Finally, Behringer and Muller described the architecture proposed for the VaMoRs-L vehicle in the PROMETHEUS project, which could perform automated driving, aided by vision and path generation algorithms.

Following these first implementations, different control architectures appear for automated driving. Path planning in mobile robotics has been a subject of study for the last decades. Most of the authors divide the problem into global and local planning [56] [57] [58]. Many navigation techniques have been taken from mobile robotics and modified to face the challenges of road networks and driving rules. According to their implementation in automated driving, these planning techniques belong to any of these four groups: graph search, sampling, interpolating, and numerical optimization.

3.2.1 Optimization-based Approach

This method group is developed due to observations of the navigation behavior from animals. The Particle Swarm Optimization (PSO) method is one of the algorithms which fall under this category [59, 60]. It contains part of the evolutionary computation and relates to the genetic algorithms. This method creates a collision-free path in configuration space by connecting surrounding random nodes. In this sense, Saska *et al.* [61] configure the robot to use the PSO method combined with Ferguson Splines (one of the Spline-Based Approaches). Thus, the PSO has the potential to manage minimal local problems. Along with his team at AutoNOMOS Project, Rotter has applied the swarm behavior concept for PP of an autonomous vehicle [62]. The implementation includes selecting swarm members, velocity matching, path planning based on the trajectories of the swarm members, data abstraction from each swarm member to generate clustered data, cluster-based path planning, and finally smoothing the re-planned trajectory with linear regression [62, 63].

This approach is helpful in urban driving, especially in heavy traffic, where each position of the vehicles in the current moment can be considered a swarm member to generate a collision-escaping path. This consideration finally creates a collision-free path for each of the swarm members in the trajectory. One of the drawbacks of the PSO method is that the data may have become obsolete at a particular time [62]. Consider a situation where a vehicle is trapped in a traffic jam for a long time. The data may become worn out

due to the long waiting time where the surrounding environment of the host vehicle is different from the frontal swarm members, which might have already escaped the heavy traffic congestion. Future researchers must consider this before utilizing this approach by considering an advanced path planning plan.

The following method of this approach is Rapidly-Exploring Random Tree (RRT) [64]. RRT is fast becoming a favorite with researchers since its introduction. Reduces computational cost by finding the trajectory to prevent collisions without knowing details about environments and obstacles. The reason is that it is a sampling-based algorithm [65]. However, RRT is not adequate to solve obstacle avoidance problems because it does not ensure asymptotic optimality [65, 66]. Gong wrote about the fundamental of RRT [67]. In this way, RRT depicts the environment of a vehicle as to the configuration space with a tree. Then, the current vehicle trajectory is re-planned by fulfilling the constraints computed with several differential equations. These newly re-planned paths allow the vehicle and robot to avoid obstacles by utilizing the combined information between the environments and several optimization mechanisms [64].

3.2.2 Geometrical-based Approach

One example of a geometrical algorithm approach is the visibility graph, also known as road-mapping path planning [68]. It is a method that consists of a graph of various locations, e.g., a set of positions, obstacles, and targets. The graph contains nodes that symbolize the robot and obstacle (point) locations, and these nodes are connected where their connection is called edge. The motivation is to find the shortest path to the goal for the robot to cross it in case an obstacle appears on the plane. The path can be built by connecting all the edges present from the start point to the end. The connection is not made if an obstacle exists between or in the middle of two edges (points). Thus, if the resulting path exists, it does not collide with any obstacle.

Therefore, theoretically, this is an excellent method for an obstacle avoidance system [69]. However, the biggest drawback of this algorithm is the routes created by this group of methods. For example, the visibility graph method is not collision-free as the routes could still contact the edges of obstacles [70]. This disadvantage is quite similar to the follow-the-carrot situation [71]. Besides, if this method is implemented for vehicle CA, it

is not safe as it does not consider the safe distance from other vehicles.

In this approach, notable examples of algorithms are the three-dimensional geometric algorithm [72], the probabilistic roadmap [64], Voronoi diagram-based roadmap [73].

3.3 Multi-agent Navigation

One of the critical challenges that a group of mobile agents, such as service robots or video game characters, face in a shared environment is arriving at their target locations while avoiding collisions with the obstacles and each other. In general, two approaches to solve the multi-agent navigation problem can be identified: centralized and decentralized.

3.3.1 Centralized Control

Centralized approaches assume that a central controller possesses all the information about the agents and the environment and can communicate with the agents. The centralization of the external information of each agent in motion may be due to individual sensory limitations and the ease that it would entail to collect information about the movement of the agents. On the other hand, the centralization of information facilitates optimization problems, which could also be carried out individually. For example, the controller creates a joint collision-free plan and then lets the agents execute it.

One of the main advantages of such an approach is solid theoretical guarantees, as proposed by Jankovic and Santillo [74]. Their work is carried out on the analysis of Control Barrier Functions, a model-based feedback control method that can be formulated as a quadratic program (QP) and solved online using real-time capable solvers. However, at the same time, they used a decentralized variant called Complete Constraint Set to analyze differences in performance.

3.3.2 Decentralized Control

Collisions are typically avoided reactively when agents have limited communication and sensory capabilities in decentralized settings, relying on local observations/communications. Deadlocks are likely to occur in numerous scenarios involving navigation through tight passages or confined spaces due to the egoistic behavior of the agents, and as a result,

the latter can not achieve their goals. To this end, Dergachev and Yakovlev [75] apply the locally confined Multi-Agent Path Finding (MAPF) solvers that coordinate sub-groups of the agents that appear to be in a deadlock. Furthermore, they present a way to build a grid-based MAPF instance, typically required by modern MAPF solvers.

Recently, Velagapudi *et al.* [76] presented a decentralized version of the prioritized planning technique for teams of mobile robots, which can utilize the distributed computational resources to reduce the time needed to find a solution. However, since the algorithm proceeds in globally synchronized rounds, faster-computing robots have to wait at the end of each round for the longest-computing robot, and thus the distributed computational power may not be used efficiently.

Chapter 4

Methodology

This chapter presents the procedures implemented to reach the objectives, together with a justification for their use. In that way, this chapter starts with an explanation of the chronological phases for problem resolution. Then, we present the model proposal, which includes the construction of the Voronoi Diagram, the implementation of the Analytical Geometrical Algorithm (AGA), the Quadratic Programming (QP) based Receding Horizons Control (RHC) Algorithm and the Optimal Reciprocal CA (ORCA) Algorithm. Finally, we describe the data sets and the methods for analyzing the results.

4.1 Phases of Problem Solving

The Figure 4.1 shows the work-flow which had been taken to perform this project.

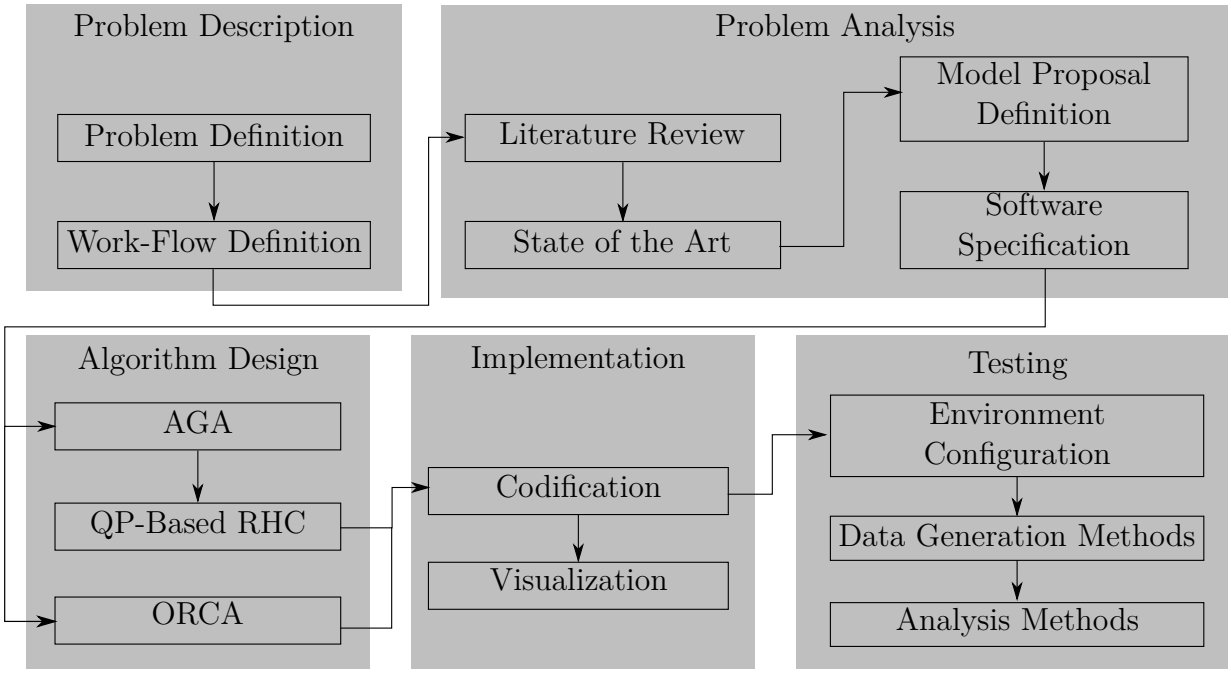


Figure 4.1: Phases of problem solving

4.1.1 Description of the Problem

This phase acts as a basis for the development of the project because it presents the general and specific objectives, the derived problems, and how to approach them. In this way, we define CA and its general characteristics. Then we detect limitations in the implementations, such as the execution time and the number of steps until the system converges. Subsequently, we define a model proposal to address this problem. To do this, we define an incremental technique to design and code the geometric structures and algorithms progressively. This style helped detect errors quickly. Chapters 1 and 4 consolidate the steps mentioned in this phase.

4.1.2 Analysis of the Problem

This phase consists in approaching the background and fundamental knowledge to understand CA. In this way, we can propose a more consistent simulation model. Chapters 2 and 4 contain the essential information in a systematic and orderly way. In Chapter 1, we analyze the CA problem, the generation of the Voronoi Diagram, and the CA algorithms. In this sense, we decided to approach the CA problem using the most common architecture

and two of its three sub-modules: Threat Assessment and Path Planning. We decided not to include Path Thracking due to the nature of the simulation and the absence of physical elements. In Chapter 2, we describe the state of art in CA. Finally, we decided on the hardware and software specifications for the simulation.

4.1.3 Algorithm Design

This phase approaches the different alternatives of the model proposal. The algorithms selected to analyze were the AGA, the QP-based RHC, and the ORCA. The explanation of these algorithms appears in Chapter 3 and Section 4.2.

4.1.4 Implementation

This phase consists of planning and executing the coding of the proposed model. We decided to use the Python programming language due to its functionality and widespread use in scientific computing. Therefore, the selected development environment was Anaconda, a complete and robust platform. On the other hand, Cython is the language in which the ORCA algorithm is implemented.

4.1.5 Testing

This phase focuses on measuring and analyzing the performance of the proposed model and the algorithm to be compared. In this way, we test the satisfactory construction of the Voronoi Diagram. Subsequently, we test the performance of the AGA parameters to define our final optimal proposal, both in AGA as QP-based RCH algorithm, and compare it with the ORCA algorithm. To do this, we had to establish how to generate the data for experimentation. Finally, the evaluation of results considers some techniques to obtain information about their performance. Details of all these steps appear in Section 4.4.

4.2 Model Proposal

We propose a simulation based on CA algorithms in a two-dimensional multi-robot environment. The simulation uses Voronoi Diagram as part of its TAS. Also, the PPS contains AGA or the QP-based RHC algorithm. Therefore, we take some heuristics as PTS. On

the other hand, the QP-based RHC algorithm is similar to AGA: both implementations converge according to the variation of the input parameters, but with the difference that it has a solver for the QP problem. Furthermore, all the simulation is centralized. Finally, the comparison of the performance of both implemented algorithms with the ORCA algorithm allows us to determine the performance of the proposal.

4.2.1 Two-Dimensional Environment

The environment is represented by a two-dimensional Cartesian plane. This environment has several characteristics, which are listed below:

- Both main axes contain the real set of numbers for the generation of coordinates.
- The position of the robots is any real coordinate along the execution of the implementations. The established restrictions of movement constrain the positions. In the same way, the Voronoi vertices belong to real two-dimensional space.
- The initial and final positions of the simulated robots form a circular figure, the center of which is at the point (0, 0).
- The initial position of the robot with its respective final position is antipodal.
- Let N be the number of robots, the distance d from each initial and final position to (0,0) is equal to:

  $$d = \left(4 \times \sqrt{N}\right) + \delta,\ \delta \in \left[-\frac{1}{0.4 \times N}, 0\right).$$

  The formula of the distance is selected to generate configurations similar to the work of Zhou *et al.* [18].
- The construction of the Voronoi Diagram follows the mathematical guidelines presented in Section 2.2.
- There exist an approximation for Voronoi infinity vertices for practical purposes. The magnitude generated for their respective infinity edges is equal to 20 times the number of simulated robots, *i. e.* if there exist five robots, then the magnitude is equal to 100.

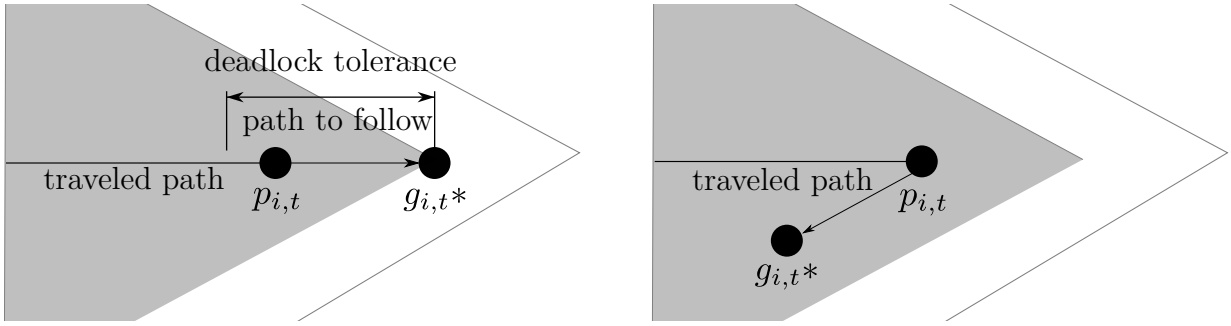
- There are no static obstacles or passages, just moving robots that can be defined as reciprocal obstacles.
- The precision of the numeric values depends on the programming language. In this case, it depends on Python.

4.2.2 Heuristics to deal with Deadlock

Deadlock is an imminent problem in CA. Deadlock happens when some robots block the paths of each other so that they cannot reach their goal. For those robots whose goal positions are not inside their own Buffered Voronoi Cells (BVCs), in a deadlock situation, each robot must be at the closest point to the goal position on its BVC. The closest point in the BVC of robot i, g_i^*, to the goal $p_{i,f}$, must be either at a vertex, or on an edge such that a line from $p_{i,f}$ to g_i^* is perpendicular to this edge.

No existing algorithm can provably avoid deadlock without central computation to our best knowledge. Most distributed algorithms attempt to alleviate the problem through sensible heuristics. Similarly, we propose two heuristic that perform well in practice to solve deadlock. At the same time, we establish a deadlock threshold value due to the limited range of numerical data types of programming languages.

- **Right-Hand Rule (Or Left-Hand Rule):** Each robot always chooses to detour from its right side when encountering other robots in deadlock situations. Figure 4.2 shows the basics of the heuristic. If the application of this heuristic causes the robot to leave its BVC, then we prefer not to move the robot in that step. This preference is made until the movement of other robots can break the deadlock or allow the movement of the blocked robot.



(a) A deadlock situation in a vertex (b) Application of right-hand rule heuristic

Figure 4.2: Dealing with deadlock through right-hand rule heuristic

- **Previous Positions:** Previous positions could ensure a high level of breaking deadlock situations, abrupt and zigzag movements. In this case, we analyzed the distance between previous positions with the closest point in the BVC. Figure 4.3 shows how dealing with deadlock taking into account three previous positions.

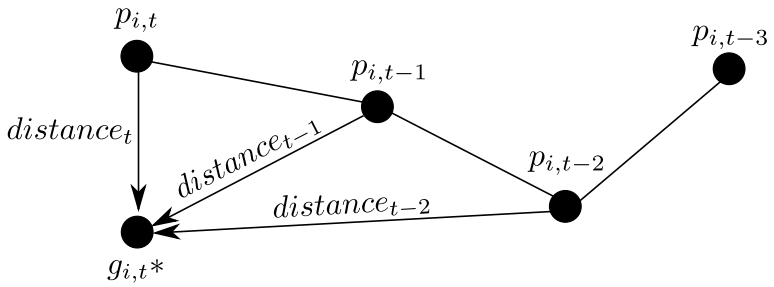


Figure 4.3: Dealing with deadlock taking into account distance of previous positions

4.2.3 Simulated Robots

The simulated robots is instantiated from a Python class with attributes detailed in Table 4.1.

Table 4.1: Attributes of the class Robot

Attribute	Datatype	Details
bvc	List of coordinates	List containing the BVC vertices
cell		List containing the Voronoi cell vertices
closer_edge		List containing the coordinates of the closer edge
neighbors_pos		List of robots positions which are next to the agent
path		List of previous positions
closer_point	Coordinate	Closer point in the BVC borders
pos_f		Final position
pos_i		Current position
color	Word	Color name
deadlock	Truth value	True if agent is in deadlock situation
inside		True if the final position is inside the BVC
dist_total	Real positive number	Total distance from initial to final position
dist_travel		Distance traveled along execution
id	Integer number	Robot number identification
neighbors_id	List of id	List of robots id which are next to the agent

4.2.4 Analytical Geometrical Algorithm

The algorithm works based on geometrical principles without an advanced optimization policy. However, this algorithm served as a base for the QP-based RHC algorithm. Furthermore, in this basic version, the algorithm prevents deadlock situations taking into account the proposed heuristics. The procedure to follow is detailed below:

1. Initialize the environment together with the robots.
2. Generate the Voronoi Diagram structure for the current positions.
3. Next, check the Collision Free Configuration for initial and final positions.
4. Then, generate the BVCs structure for the current positions too.
5. Get the most closer point from each BVC to the respective final point.
6. Check deadlock situation using actual and previous positions. If it exists, then apply the right-hand rule heuristic to the robots involved.
7. Move the robot to the closest point in its cell.

The procedure continues until the final position is on the BVC edges or inside the cell. Then, the robot should move directly to the final position. The explanation of Collision Free Configuration and generation of the closest point to final in a cell mentioned in the procedure is below.

- **Collision Free Configuration.** For the group of N robots with the same safety radius r, *A collision free configuration* is one where the distance between positions of robot p_i and robot p_j satisfies:

$$\|p_i - p_j\| \geq 2r, \forall i, j \in 1, 2, \cdots, N, i \neq j. \tag{4.1}$$

- **BVC Closest Point.** Let $\mathcal{V} = (\epsilon, e)$ represents a convex polygon in $\mathbb{R}^2$, where ϵ is the set of edges and e is the set of vertices. For any point $g \in \mathbb{R}^2$, the closest point $g^* \in \mathcal{V}$ to g is either g itself, or on an edge ϵ^* of $\mathcal{V}$, or is a vertex e^* of $\mathcal{V}$.

Algorithm 1 outlines the basic AGA procedure. Algorithm 1 outlines the basic AGA procedure. The input parameters are the number of robots N, the magnitude of the safety radius r, the magnitude of movement m in each step, the magnitude of deadlock tolerance δ, the magnitude of movement ϵ to break the deadlock situation, and the number of previous

positions ω to evaluate if a deadlock situation exists.

Algorithm 1: Analytical Geometric

Data: N, r, m, δ, ϵ, ω, robots

```
if not Collision Free Config(P,r) or not Collision Free Config(G,r) then
    Change the initial or final configuration;
    break;
end
while not current positions = final positions do
    vor = Generate Voronoi Diagram(P);
    bvc = Generate BVC(P, r, vor);
    for agent in robots list do
        if not Final position inside BVC(agent, bvc) then
            closest point = Minimum distance function(agent, bvc);
            deadlock = Distance(agent, closest point) < δ;
            cycle = 1;
            while cycle < ω and not deadlock do
                if Distance(agent path, closest point) < δ then
                    deadlock = true;
                    closest point = Apply right-hand heuristic(agent, bvc, ϵ);
                end
                Increment cycle in 1;
            end
        else
            closest point = Move to the final position(agent)
        end
        P = Move the robot to its closest point(agent, closest point, m);
    end
end
```

4.2.5 QP-based RHC Algorithm

Until now, with the AGA, the robot computes at each step its BVC, plans a path within its cell using basic geometric definitions, then moves along that path according to a magnitude. The robot position, BVC, and planned path evolve together throughout the execution of the algorithm. The next step in our work is implement the QP-based RHC algorithm. Based on the Subsection 2.3.1 and the work of Zhou *et al.* [18], we propose the following optimization model: assume a planning horizon with T steps, and for robot i at each time instance, we denote the positions of the planned trajectory as $\overline{p}_{i,1}, \overline{p}_{i,2}, \ldots, \overline{p}_{i,T}, \forall i \in \{1, 2, ..., N\}$, where N is the number of robots moving in the environment. The algorithm requires each robot to solve a QP problem at each implementation step, e.g., for robot i, the optimal value $\overline{p}_{i,T}$ inside its BVC concerning to the its final position is given by solving the following QP problem:

Problem 2 (Receding Horizon Path Planning)

$$\min_{\overline{p}_1,\ldots,\overline{p}_T} J_i = \sum_{t=0}^{T-1} \left(\left(\overline{p}_{i,t} - p_{i,f}\right)^{\intercal} Q \left(\overline{p}_{i,t} - p_{i,f}\right) + u_{i,t}^{\intercal} R u_{i,t} \right) + \left(\overline{p}_{i,T} - p_{i,f}\right)^{\intercal} Qf \left(\overline{p}_{i,T} - p_{i,f}\right) \tag{4.2}$$

subject to:

$$\begin{array}{lrcll}
\text{(i)} & \overline{p}_{i,t+1} & = & A\overline{p}_{i,t} + Bu_{i,t}, & t = 0, \cdots, T-1, \\
\text{(ii)} & \overline{p}_{i,t} & \in & \overline{\mathcal{V}}_i, & t = 1, \cdots, T-1, \\
\text{(iii)} & \overline{p}_{i,0} & = & p_i, & \\
\text{(iv)} & \|u_{i,t,x}\| & \leq & u_{x,max}, & t = 0, \cdots, T-1, \\
\text{(v)} & \|u_{i,t,y}\| & \leq & u_{y,max}, & t = 0, \cdots, T-1.
\end{array}$$

In Problem 2, the cost function J_i is a summation of intermediate state costs and a terminal cost. In Equation 4.2, $p_{i,f}$ is the final position, $\overline{p}_{i,0}$ to $\overline{p}_{i,T}$ and $u_{i,0}$ to $u_{i,T-1}$ are the path, and inputs to be planned, respectively. The positive definite or semi-definite matrices Q, R, and Q_f are weight factors to balance among the three costs. The decision variables for this standard QP problem are $\overline{p}_{i,1}$ to $\overline{p}_{i,T}$. The constraint (i) ensures that the path is feasible with the robots dynamics. In the present work, the holonomic robots do not need to fix the matrix values because they can move in any direction, and their velocity is the same. The constraint (ii) restrains the planned path to be inside of the

corresponding BVC $\overline{\mathcal{V}_i}$ of robot i. This constraint can be written explicitly in the form of a set of linear inequalities:

$$\overline{p}_{i,t}^{\intercal}\epsilon_j \leq 0, \quad t = 1, \cdots, T, j \in \{1, \cdots, N\}, j \neq i, \tag{4.3}$$

where ϵ_j is a vector representing an edge of $\overline{\mathcal{V}_i}$ that separates the robot i from the robot j. Constraint (iii) makes sure the planned positions starts from the current position of the robot, and, finally, the lower and upper bounds for the input $u_{i,t}$ are written component-wisely in constraints (iv) and (v).

The procedure is similar to the presented in Section 4.2.4 for AGA but different for getting the optimal and closer point inside BVC. For this, we use a Python-embedded modeling language CVXPY to solve the QP problem [77]. It allows expressing a convex problem naturally that follows the mathematical conventions rather than in the restrictive standard form required by solvers. The usage of CVXPY is similar to an usual Python library. Our implementation of the solver is presented in the Appendices and in the GitHub repository of the present work [78]. Finally, the necessary parameters for the correct implementation of the solver are detailed below:

- **External Parameters.** The main input parameters are the instances created from the robot class detailed in Subsection 4.2.3, the safety radius and the receding horizon steps T. As the code is implemented, other secondary parameters appear to display and debug it.

- **Internal Parameters.** The Table 4.2 shows the required parameters that should be initialized using CVXPY specifications inside the code of the solver. As we can see, these parameters are related to the cost function presented in the Equation 4.2 of the Problem 2.

Table 4.2: Internal parameters of the QP-based RHC solver

Parameter	**Datatype**	**Details**
m	Integer number	Dimensions
A	(2×2) matrix	Defines robots dynamics
B	$(2 \times m)$ matrix	
R	$(m \times m)$ positive semidefinite matrix	Balancing costs
Q	(2×2) positive semidefinite matrix	
Q_f		
$\overline{p}$	$(2 \times T + 1)$ matrix	Planned path
u	$(m \times T)$ matrix	Planned input
$u_{max,x}$	Real non-negative number	Lower and upper bound
$u_{max,y}$		

As the procedure, the pseudocode is similar to Algorithm 1. For this reason, the Algorithm 2 contains the most fundamental parts of the pseudocode to solve the QP problem and get

the closer coordinate in the robot cell to its final position.

Algorithm 2: QP Solver

Data: agent, r, T

Result: $\overline{p}_T$

initialize m, A, B, R, Q, Q_f, $\overline{p}$, u, $u_{max,x}$, $u_{max,y}$;
cost = 0;
constraints = [ ];
p_i = agent initial position;
p_f = agent final position;
Add to constraints($\overline{p}_{i,0} = p_i$);
for *t in range(T)* **do**
    Sum to cost $((\overline{p}_t - p_f)^\intercal Q(\overline{p}_t - p_f) + u_t^\intercal R u_t)$;
    Add to constraints($\overline{p}_{t+1} = A\overline{p}_t + Bu_t$);
    Add to constraints($\|u_{0,t}\| \leq u_{max,x}$);
    Add to constraints($\|u_{1,t}\| \leq u_{max,y}$);
    for *each position of agent neighbors* **do**
        p_j = neighbor position;
        Add to constraints($p_t - \frac{p_i+p_j}{2}^\intercal (p_j - p_i) + (r * \|p_j - p_i\|) \leq 0$);
    end
end
Sum to cost($(\overline{p}_T - p_f)^\intercal Q_f(\overline{p}_T - p_f)$);
for *each position of agent neighbors* **do**
    p_j = neighbor position;
    Add to constraints($p_T - \frac{p_i+p_j}{2}^\intercal (p_j - p_i) + (r * \|p_j - p_i\|) \leq 0$);
end
Solve the CVXPY problem();

4.2.6 ORCA Algorithm

In this work, we use the RVO2 library as the ORCA algorithm implementation [79]. It is an open-source implementation and has a simple API for third-party applications. In this way, the user specifies static obstacles, agents, and the preferred velocities of the agents. The

simulation is performed step-by-step via a simple call to the library. Thus, the simulation is fully accessible and manipulable during runtime. The algorithm ensures that each agent exhibits no oscillatory behaviors.

For our purposes, we use the Cython-based Python bindings of Liu, which can be found as a GitHub repository [80]. This variant should be installed together with all its dependencies in an Anaconda environment. Finally, the algorithm is used in Jupyter Lab. Our implementation of the ORCA algorithm is presented in the GitHub repository of the present work [78].

4.3 Analysis Method

We defined the ways to study the model proposal in this subsection. A series of experiments evaluate and compare the performance of the model proposed in Section 4.2. These experiments were designed with specific research intentions:

1. Execution of pseudo-random spatial configurations of the robots positions to check the consistency in the construction of the Voronoi Diagram.

2. Identification of suitable values for parameters in AGA and QP-Based RHC Algorithm.

3. Comparison of the proposed algorithms and ORCA using the best performing AGA and QP-Based RHC Algorithm.

We defined a set of measures for testing the algorithms. These measures helped us understand the algorithm performance in terms of duration and calculation of optimum values. The performance measures are the following.

- **Execution Time (ET)** is a measure of execution duration in units of time. It is the time from starting movement until all robots reach the final positions. This measure depends on the computer and the programming language. In this way, we used a timer function of Python from the built-in timer library for getting the initial and final time. Then, the execution time follows is represented as:

$$ET = t_f - t_i. \tag{4.4}$$

Thus, ET is the total execution time, t_f is the final time, and t_i is the initial time.

- **Steps Number (ST)** is a measure of duration in terms of iterations. It is the number of iterations until all robots reach the final position. This measure depends on the algorithm design, mainly of the movement magnitude.

- **Effectiveness in distance traveled (ED)** measures how close the path traveled is to the shortest distance, understood as a straight line from the starting point to the end. This effectiveness is represented as:

$$ED = \frac{\sum_{i=1}^{N} a_i}{\sum_{i=1}^{N} b_i}. \tag{4.5}$$

  In this equation, N is the total number of robots, a_i represents the shortest distance for robot i from its started position to the final position, and b_i is the total distance traveled by robot i. This measure is only used for AGA parameters analysis.

- **Sum of Cost (SC)** represents the sum of the means of all the previous measures for the selection of the best AGA parameter value. The SC follows the next model:

$$\min \left\{ \frac{ET_i}{\sum_{j=1}^{k} ET_j} + \frac{ST_i}{\sum_{j=1}^{k} ST_j} + \frac{1 - ED_i}{\sum_{j=1}^{k}(1 - ED_j)} \middle| i \in \{1, \cdots, k\} \right\}, \tag{4.6}$$

  where k represents the number of values that one of the parameters can take.

4.4 Experimental Setup

The experiments were executed in a computer with the following specifications:

- Processor: Intel® Core™ i7-10750H CPU @ 2.60GHz 2.59 GHz
- CPU Cores: 6
- Logical processors: 12
- RAM: 15.5 GB
- System type: 64-bit

- Operating System: Ubuntu 20.04.2

The implementation of our work was carried out in the Anaconda Development Environment using the software detailed in Table 4.3.

Table 4.3: Software used in the implementation of the CA algorithms

Software	**Details**	**Version**
Python	Programming Language	3.6.13
CVXPY	Modeling language for convex optimization problems	1.1.15
Cython	Programming Language	0.21.1
CMake	Cython compiler	3.16.3
RVO2	Library to implement ORCA algorithm	2.0.2
Conda	Package, dependency and environment management	4.10.3
Jupyter Lab	Web-based interactive development environment	3.1.4
Scipy	Mathematical and statistical library	1.5.4
Numpy	Library for numbers, strings, records, and objects	1.19.5
Matplotlib	Library for visualization	3.3.4
Pandas	Library for data analysis	1.1.5
Seaborn	Data visualization library based on Matplotlib	0.11.2

Chapter 5

Results and Discussion

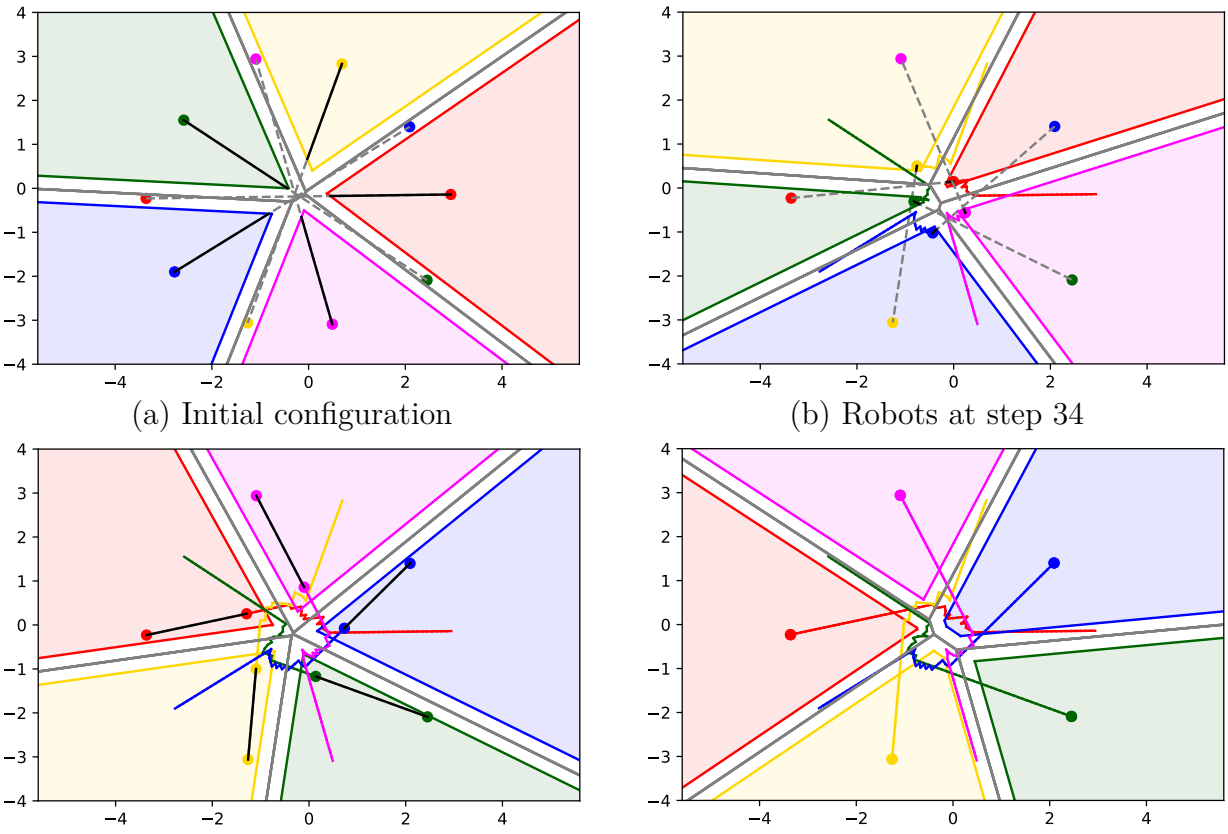
We document our experiments of the model proposed in this chapter. For the analysis of the model, we follow a progressive study. First, we present the constructed Voronoi Diagrams. Then we work with the basic version of the Analytical Geometric Algorithm (AGA) to test previously defined geometric structures, deadlock situations, parameters performance and have a basis for the next steps. The parameters are the magnitude of the safety radius r, the magnitude of movement m in each step, the magnitude of deadlock tolerance δ, the magnitude of movement ϵ to break the deadlock situation, and the number of previous positions ω to evaluate if a deadlock situation exists. Later, we worked with the Quadratic Programming (QP) based Receding Horizons Control (RHC) Algorithm. Finally, we used both AGA and QP-based RHC Algorithm for comparing with the Optimal Reciprocal Collision Avoidance (ORCA) Algorithm.

Computational limitations must be taken into account before running the codes on the machine. For this reason, the hardware is a limitation of the number of simulated robots, mainly for the QP-based RHC implementation. Furthermore, we describe some results for each robot configuration first, and then we discuss these results. The codes used in this work can easily be found as a repository on GitHub [78].

5.1 Construction of Voronoi Diagram

The construction of the Voronoi Diagram was developed successfully in Anaconda Development Environment. The Voronoi class of Scipy Spatial data structure was elemental to generate the attributes related to the coordinates of the Voronoi vertices.

The Figure 5.1 shows one of the experiments with all the corresponding geometric structures. The Voronoi Diagram is drawing with thick gray lines. The dashed gray lines intersect with each other, are the straight lines from the robots' current positions to their goal positions. Also, the executed trajectories, Buffered Voronoi Cell (BVC), and goal positions have the same color as the robot, and the thick dark lines are the planned paths from our algorithm for each robot in its cell. We can see in Figure 5.1 (a) an initial configuration with five robots and a safety radius equal to 0.3. In Figure 5.1 (b), we can see the robots in the middle of the execution. In Figure 5.1 (c), we can see all robots direct to their goal position. Finally, Figure 5.1 (d) shows the final state of the system.



(a) Initial configuration

(b) Robots at step 34

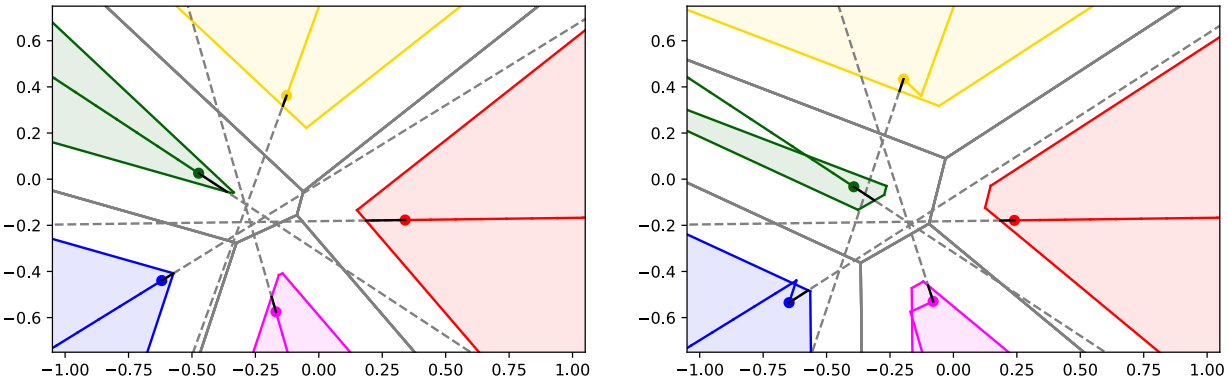
(c) Robots at step 51

(d) Robots at step 76

Figure 5.1: Visualisation of AGA execution with the respective BVC generation

5.1.1 Deadlock

The blocking situation was addressed satisfactorily in all the experiments. In Figure 5.2 (a), we can see blue, cyan, and magenta robots in deadlock situation. Next, we can see in Figure 5.2 (b) the next state after dealing with deadlock using right-hand rule heuristic.



(a) Yellow, blue and magenta robots in deadlock.

(b) Positions after using the right-hand rule heuristic.

Figure 5.2: Deadlock situation

5.2 Evaluation of AGA Parameters

The description of the evaluated parameters appears in the opening paragraph of the current Chapter. The method used to analyze the parameters is in Subsection 4.3. Also, the default values of each parameter are shown in Table 5.1. Finally, the number of experiments to evaluate each of the parameter values was equal to 30.

Table 5.1: Default parameters values to evaluate the AGA performance

Parameter	N	r	m	δ	ϵ	ω
Value	5	0.1	0.1	0.05	0.1	1

5.2.1 Safety Radius

In Figure 5.3 we can see the increase of the average execution time along with the the safety radius magnitude. However, there are some outliers regardless of the safety radius value. Also, the number of total system steps increases as the safety radius value increases. Finally, we prove that obviously the AGA is effective when the safety radius is smaller because the safety radius prevent the direct movement from initial to final position.

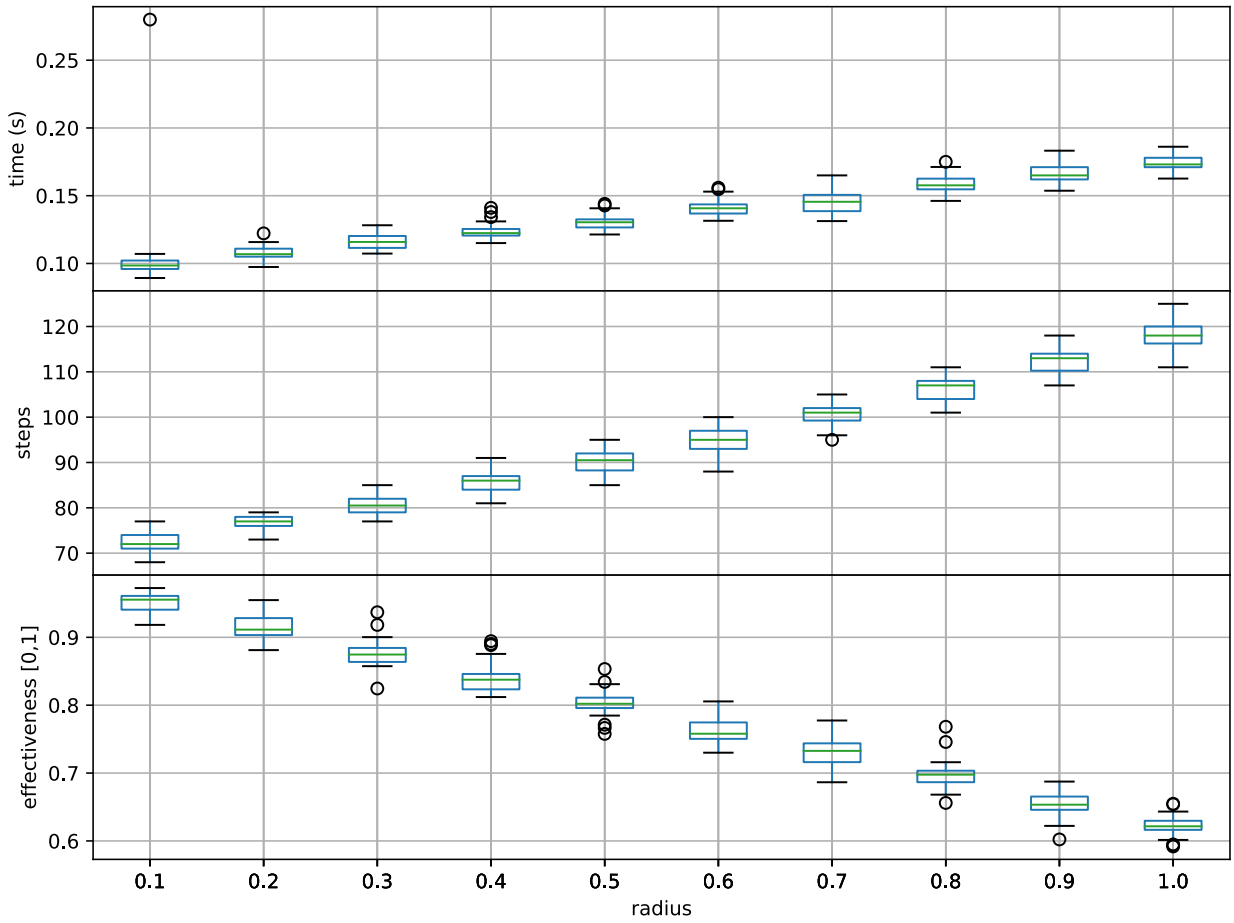


Figure 5.3: Time in seconds, number of steps and effectiveness vs size of the safety radius

5.2.2 Deadlock Tolerance

Figure 5.4 shows the results of our experiments. As we can see, the measure of execution time, number of steps, and sum of costs have similar behavior. In this way, the results of these measures resemble a parabola whose vertex is at a tolerance of approximately 0.6. Furthermore, we can realize that effectiveness is inversely proportional to the magnitude of tolerance. Finally, in the sum of the cost graph, we observe that the optimal value of the tolerance according to the measurements made is 0.6.

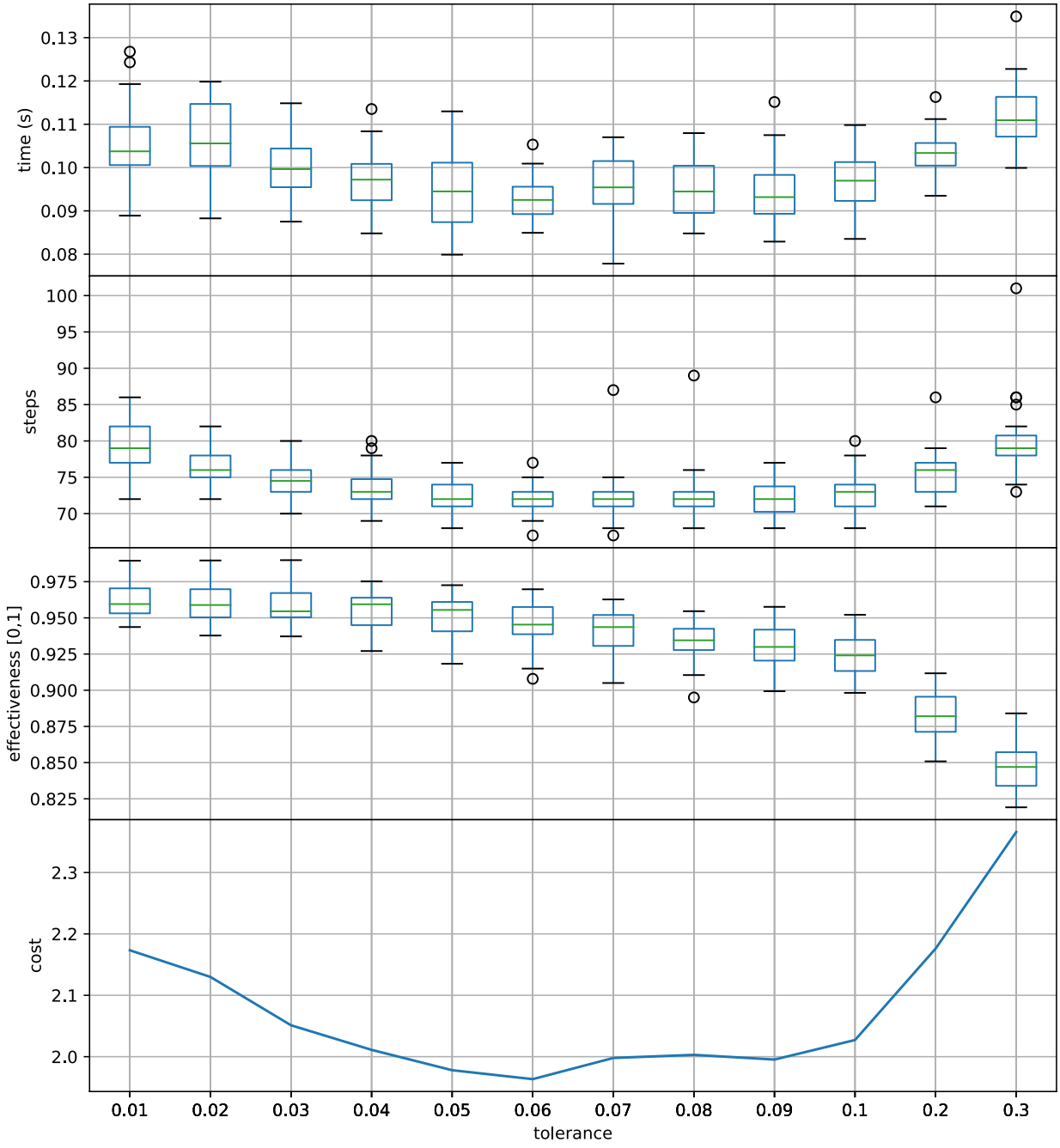


Figure 5.4: Time in seconds, number of steps, effectiveness and the cost vs size of the deadlock tolerance

5.2.3 Normal Movement

The relationship between time, number of steps, and effectiveness with movement magnitude is presented in Figure 5.5.

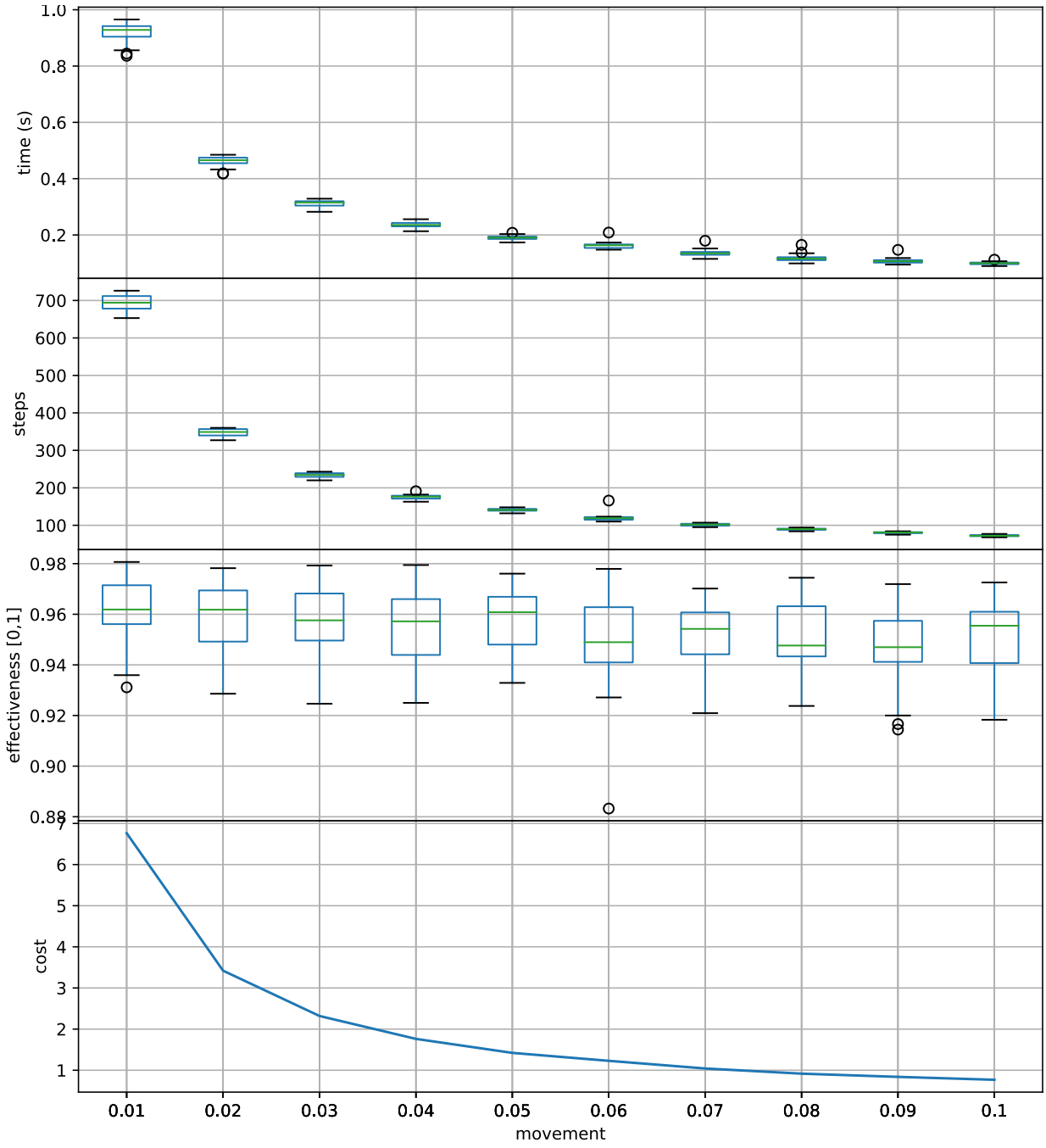


Figure 5.5: Time in seconds, number of steps, and effectiveness vs movement magnitude

As we can see, the mean execution time, number of steps and cost decrease along with the increment of movement value in a logarithmic form. However, the effectiveness of the algorithm appears to be independent of the magnitude of movement.

5.2.4 Movement in Deadlock

The relationship between time, number of steps, and effectiveness with movement in deadlock is presented in Figure 5.6.

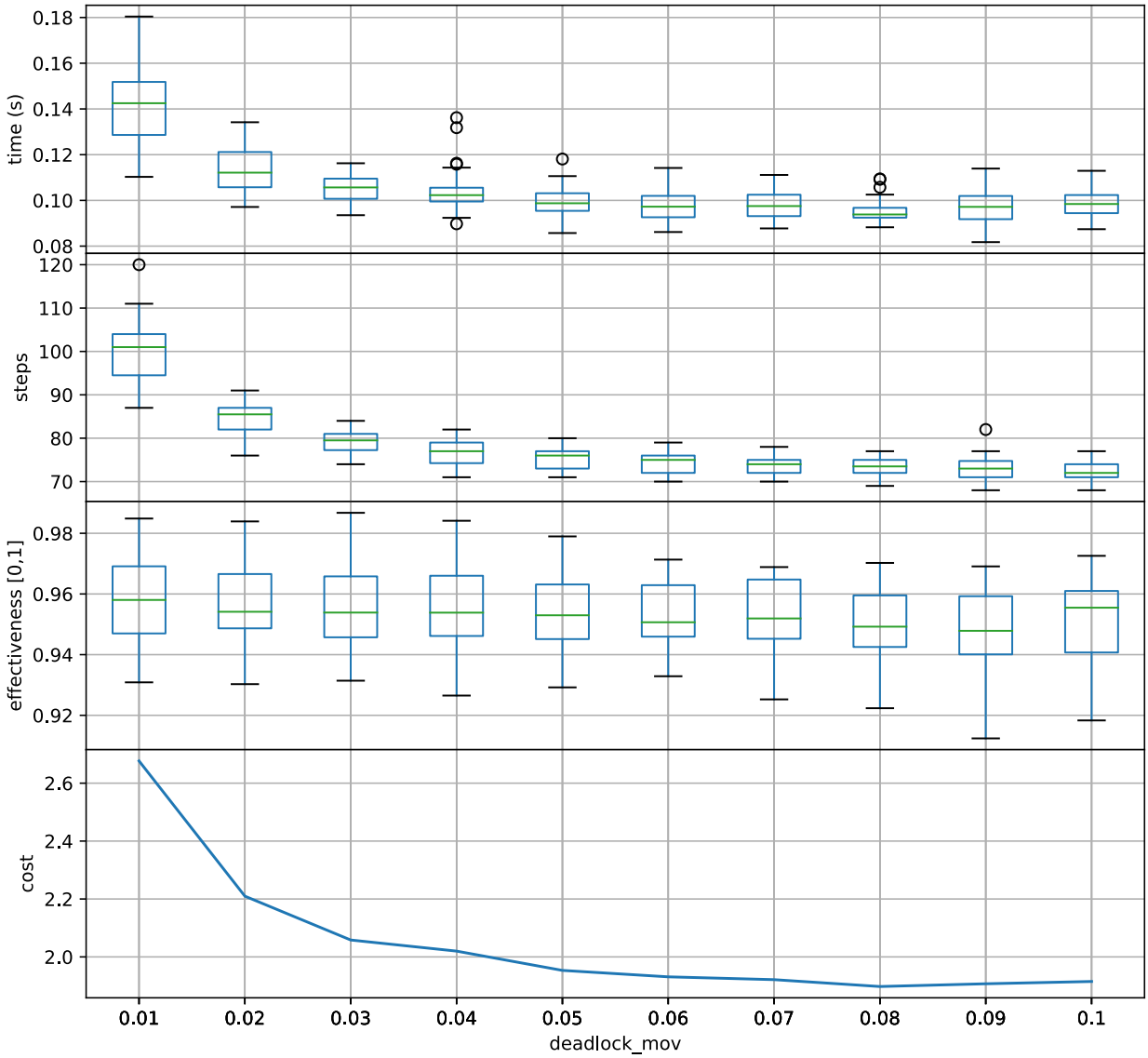


Figure 5.6: Time in seconds, number of steps, and effectiveness vs magnitude of movement in deadlock

As we can see, the mean execution time, number of steps, and cost decrease along with the increment of deadlock movement value in a logarithmic form. This compartment looks like the one seen in the normal movement variation, but with the difference that 0.08 is the most optimal value. Finally, the efficacy of the AGA appears to be less when the movement in the deadlock is significant. If the value of the movement in the deadlock situation is

greater than the normal movement, the code could fail. For this, the analysis stops in the default movement magnitude in experiments, which is 0.1.

5.2.5 Previous Positions

The relationship between time, number of steps, and effectiveness with previous position is presented in Figure 5.7.

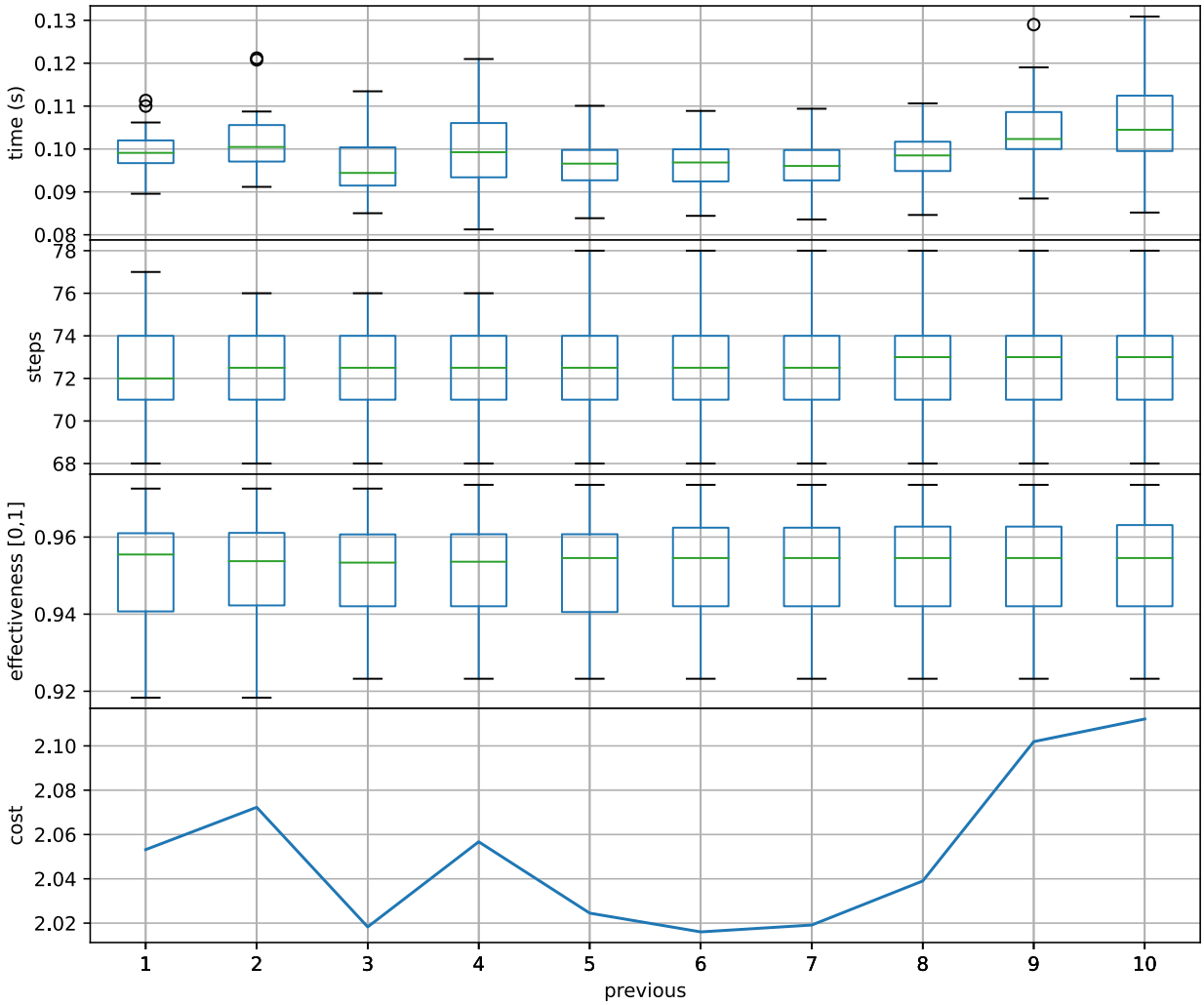


Figure 5.7: Time in seconds, number of steps, and effectiveness vs number of previous positions to break deadlock

As we can see, the execution time does not follow a growth or decrease pattern until the value 5. From the value 5, the average execution time tends to increase as the number of reviewed positions to establish a neutral situation increases. In the same way, the average number of steps of the system until convergence begins to stabilize from the value

5, and then a tiny variation occurs in the value 8 that stabilizes again. Regarding the effectiveness, it has slight variations as the number of previous positions of the robots changes. Finally, the cost graph reveals the variation of the number of positions: there is an irregular pattern up to the value number 6; after this value, there is a trend of cost growth directly proportional to the execution time.

5.3 QP-Based RHC Implementation

The implementation in Python of the solver is in the repository of the present work [78]. The default values of each parameter are shown in Table 5.2. The number of experiments to evaluate each of the parameter values was equal to 30.

Table 5.2: Default parameters values to evaluate the QP-based RHC algorithm performance

Parameter	N	r	m	δ	ϵ	ω
Value	5	0.1	0.1	0.05	0.1	1

Figure 5.8 shows the number of steps according to the number of receding horizons steps.

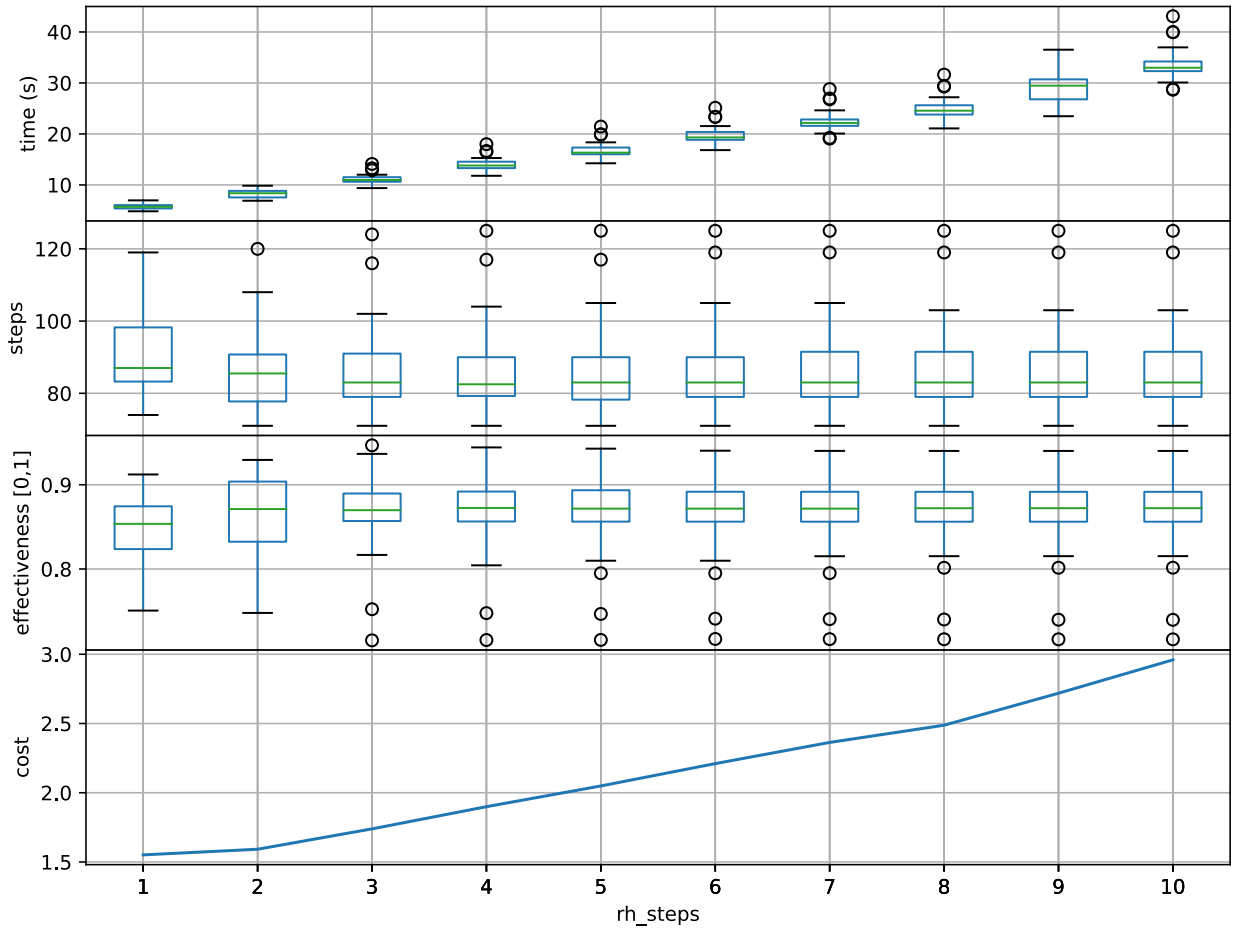


Figure 5.8: Steps vs number of receding horizons steps

The results are similar to Figure 5.7 since there is a growing trend of the execution time proportional to T and instability in the step patterns and effectiveness up to a particular value. In the case of steps, the results begin to stabilize from value 6. In the case of effectiveness, the results tend to be the same from value 5 onwards.

5.4 RVO Library Implementation and Performance

Figure 5.9 shows the results of implementing the ORCA Algorithm compared to the performance of the AGA.

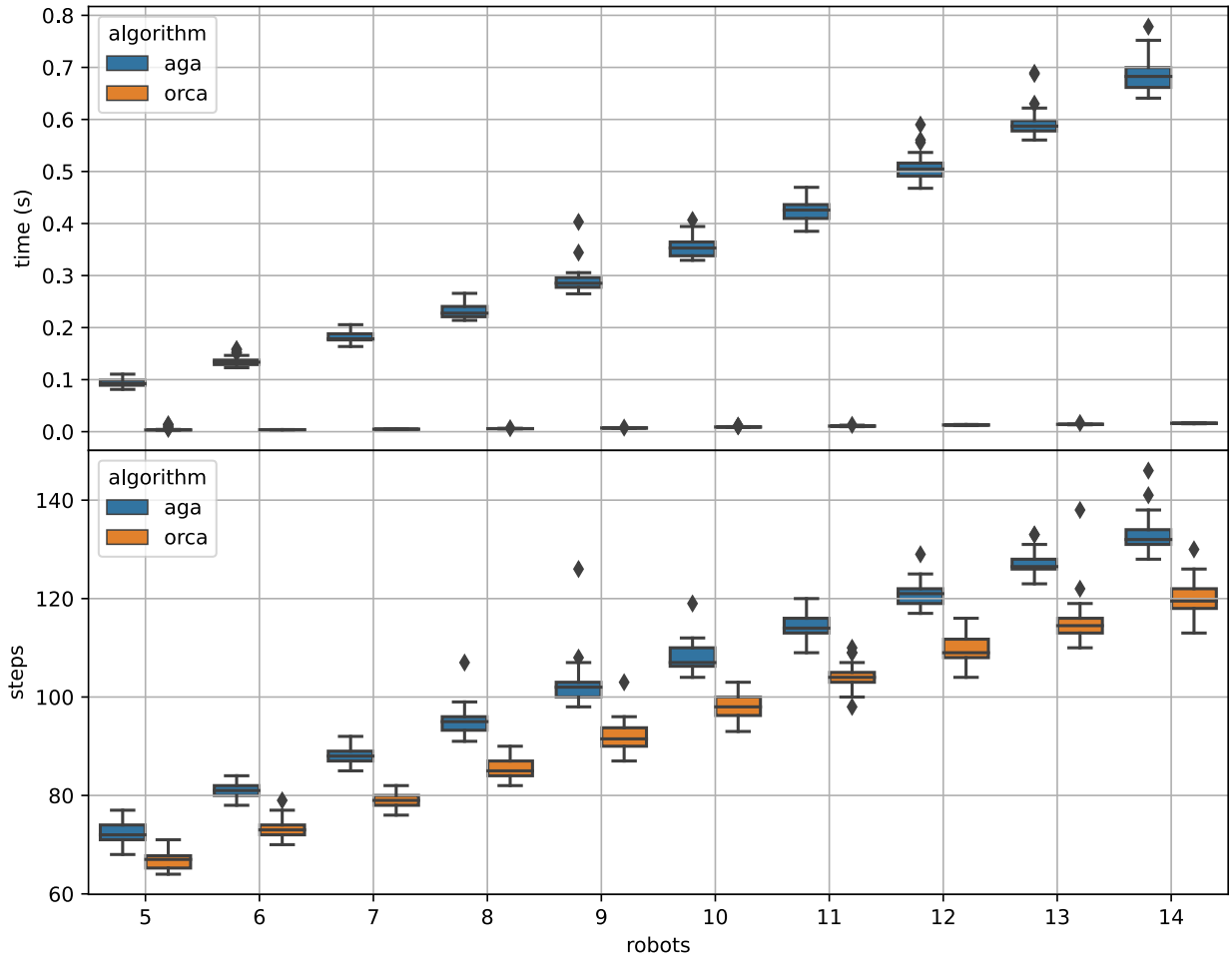


Figure 5.9: ORCA and AGA performance in time and number of steps vs number of robots

As we can see, both the time and the number of steps of ORCA are less than AGA for a range of values from five to fourteen robots. Thus, the differences between ORCA and AGA are more evident over time. However, for the number of steps, AGA comes close to ORCA.

Chapter 6

Conclusions

6.1 Conclusion

This work presented a CA simulation for multi-agents in movement based on Voronoi Diagram, AGA, and QP-based RHC algorithm in a two-dimensional space free of static obstacles. The design and construction process began with the two-dimensional Voronoi Diagram, tested throughout the work, and showed efficiency in generating the structure in all the final experiments. Then, based on the generation of the environment and the geometric structure, the AGA was implemented and converged to the solution. Each of the algorithm parameters demonstrated the behavior of the algorithm after varying its input values, which allowed us to understand it and work with optimal values. Following this, the QP-based RHC algorithm, especially its solver, was implemented using the AGA and the Voronoi Diagram as the base platform. These two algorithms and the geometric structure were compared with the ORCA algorithm, allowing us to know how effective our proposal was concerning one of the most used CA implementations currently. Finally, the kinematic simulation could be observed through graphics generated from libraries of the Python language.

The experiments showed that the AGA produces the best performance results on time and steps to reach the goal position of robots. Conversely, the QP-Based RHC Algorithm had low performance. By comparing our versions with the ORCA Algorithm, the results suggested that algorithms have an acceptable accuracy with values above 95%. Nevertheless, our proposal presented more variable accuracy results comparing ORCA Algorithm.

This loss of accuracy was compensated with the low execution time, with values around 10ms in the first and second routing stages.

On the other hand, we concluded that the QP-Based RHC Algorithm produces softer paths because of the optimization. Another conclusion was that the maximization or minimization of parameters has no relationship with the generation of the optimal value inside each cell. Furthermore, the CA guarantee was proved in practice.

Finally, the qualitative results of the graphic simulation showed the convergence of our proposal. The robots were able to reach the final position through the generation of Voronoi Diagrams during the execution of the code.

6.2 Recommendations

In this part, we list some recommendations based on the problems and limitations found in our proposal.

- In addition to the previous knowledge possessed, we recommend reviewing the technical support resources when deciding the tools to use, such as documentation, communities, forums, repositories, and usability.
- Be aware of changes between software versions used in this work with the most recent versions. There may be performance improvements with the new versions, but at the same time, it is possible that the functionalities used in this work no longer exist or have substantial changes.
- Consider the type numbers precision of the selected programming language and limitations of libraries concerning the exactitude of the generated values previously. According to this, difficulties related to the generation of numerical solutions can be anticipated.
- Establish an incremental style for designing simulation and algorithm implementations. In this way, it is possible to construct from the most simple structures, find bugs and limitations, and deal with them as the program becomes complex.

- In the design stage, we recommend setting a rank of values for each parameter. Then, we suggest testing their behavior for selecting the parameters with the best results. Of course, the setting of these values will have to be related to the computational resources of the machine where the implementation will be executed.

- We recommend applying coding best practices, such as the modularization of the algorithm. This programming technique will allow reusing parts of the code even in other CA algorithms and understanding the information flow in a better way. Also, document it in the code and allow a verbose mode.

- Test heuristics along with proposal implementation to improve algorithm performance in solution convergence, especially in deadlock situations. The set of heuristics can represent a substantial change in the implementation, becoming a new proposal increasingly within research.

- In the method comparison stage, we recommend using some already implemented algorithms instead of self-implementations. These could give us a better idea of the performance of our proposal.

- It is not too much to suggest that both written notes and code implementations be saved on cloud platforms. For the present work, we use Overleaf and GitHub, respectively.

6.3 Future Work

In this part, we propose some future research and implementation, which could be considered in future works.

- **High-Performance Computing with more Massive Data Sets.** The performance values calculated using many robots can predict results on a grand scale. However, this scalability quality has been partially studied in the present work because the computational limitations and some variants have shown unfinished results yet. For this reason, we propose to develop more experiments with more robots to study the scalability performance of the algorithm. In this way, it would be necessary

to continue implementing the algorithm in Python or another programming language that supports high-performance computers such as C or C++.

- **Concurrency of Collision Avoidance Algorithms.** The current project proposed some variants which implement sequential algorithms where the robots perform a movement one by one. However, it is not necessarily the basic idea of the Collision Avoidance Algorithms because they can be executed concurrently; therefore, the algorithm could be concurrent. This concurrency can give us some advantages over other algorithms in terms of execution time. For this reason, another proposal is designing some variants of concurrent algorithms that can deal with the collision avoidance problem.

- **Use better Visualization Frameworks.** For better observation of the algorithm's behavior during its execution, the use of more sophisticated visualization systems is recommended. These systems should allow the code to run and, from the resulting data, produce the respective movement in the robot. It is recommended to give Gazebo a try as it is an open-source platform.

- **Extension to 3D Space and Higher-Order Dynamics.** The algorithmic and geometric nature of the project makes it easy to extend it to higher-order space. Therefore, it is recommended that test simulations be performed with vehicles that fly and can be suspended in the air.

- **Use of others Optimization Solvers.** Test the efficiency of running new optimization problem solvers and compare them. For example, one of them is CVXPY which can be embedded in the code and is written in C so that it could take a faster execution time.

- **Try others Initial and Ending Configurations:** In the current project, we use circular configurations, but this is not a limitation to approach the analysis of the algorithm. So it is recommended to use other spatial configurations, such as matrix shapes where the robots have to move between columns and rows.

Bibliography

[1] U. Hamid, Y. Saito, H. Zamzuri, M. Rahman, and P. Raksincharoensak, "A review on threat assessment, path planning and path tracking strategies for collision avoidance systems of autonomous vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 14, no. 2, pp. 134–169, 2018.

[2] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Optimal reciprocal collision avoidance for multi-agent navigation," in *Proc. of the IEEE International Conference on Robotics and Automation, Anchorage (AK), USA*, 2010.

[3] P. Xuan and V. Lesser, "Multi-agent policies: from centralized ones to decentralized ones," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, 2002, pp. 1098–1105.

[4] J. Dahl, G. R. de Campos, C. Olsson, and J. Fredriksson, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 101–113, 2018.

[5] S. Lefevre, A. Carvalho, and F. Borrelli, "A learning-based framework for velocity control in autonomous driving," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 32–42, 2015.

[6] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *Dynamic systems and control conference*, vol. 44175, 2010, pp. 265–272.

[7] A. Simon and J. C. Becker, "Vehicle guidance for an autonomous vehicle," in *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383)*. IEEE, 1999, pp. 429–434.

[8] L. Perko, *Differential Equations and Dynamical Systems*, ser. Texts in Applied Mathematics. Springer New York, 2008. [Online]. Available: https://books.google.com.ec/books?id=A7fvvz9Puf8C

[9] J. Hale, H. Buttanri, and H. Kocak, *Dynamics and Bifurcations*, ser. Texts in Applied Mathematics. Springer New York, 1996. [Online]. Available: https://books.google.com.ec/books?id=h2gq4NULS_4C

[10] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annual Reviews in Control*, vol. 48, pp. 147–164, 2019.

[11] A. Vagale, R. Oucheikh, R. T. Bye, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles i: a review," *Journal of Marine Science and Technology*, pp. 1–15, 2021.

[12] D. Agarwal and P. S. Bharti, "A review on comparative analysis of path planning and collision avoidance algorithms," *International Journal of Mechanical and Mechatronics Engineering*, vol. 12, no. 6, pp. 608–624, 2018.

[13] P. Raksincharoensak, T. Hasegawa, and M. Nagai, "Motion planning and control of autonomous driving intelligence system based on risk potential optimization framework," *International Journal of Automotive Engineering*, vol. 7, no. AVEC14, pp. 53–60, 2016.

[14] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International journal of physical sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[15] M. Yao and M. Zhao, "Unmanned aerial vehicle dynamic path planning in an uncertain environment," *Robotica*, vol. 33, no. 3, pp. 611–621, 2015.

[16] M. Zakaria, H. Zamzuri, R. Mamat, and S. Mazlan, "A path tracking algorithm using future prediction control with spike detection for an autonomous vehicle robot," *International Journal of Advanced Robotic Systems*, vol. 10, no. 8, p. 309, 2013.

[17] F. Yakub and Y. Mori, "Model predictive control for car vehicle dynamics system-comparative study," in *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*. IEEE, 2013, pp. 172–177.

[18] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[19] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52–65, 2011.

[20] P. Whittle, *Optimization over time*. John Wiley & Sons, Inc., 1982.

[21] S. Keerthi and E. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *Journal of optimization theory and applications*, vol. 57, no. 2, pp. 265–293, 1988.

[22] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic swarm guidance using optimal transport," in *2014 IEEE Conference on Control Applications (CCA)*. IEEE, 2014, pp. 498–505.

[23] R. Mao, H. Gao, and L. Guo, "A novel collision-free navigation approach for multiple nonholonomic robots based on orca and linear mpc," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[24] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[25] A. Van der Horst, "A time-based analysis of road user behaviour in normal and critical encounters," Ph.D. dissertation, Technische Universiteit Delft, 1991.

[26] J. Hayward, "Near miss determination through use of a scale of danger," Pennsylvania Transportation and Traffic Safety Center, Tech. Rep., 1972.

[27] A. Van der Horst and J. Hogema, "Time-to-collision and collision avoidance systems," *Verkeersgedrag in Onderzoek*, 1994.

[28] J. Jansson, "Collision avoidance theory: With application to automotive collision mitigation," Ph.D. dissertation, Linköping University Electronic Press, 2005.

[29] P. Falcone, M. Ali, and J. Sjoberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, 2011.

[30] R. Kiefer, M. Cassar, C. Flannagan, D. LeBlanc, M. Palmer, R. Deering, and M. Shulman, "Refining the camp crash alert timing approach by examining last second braking and lane change maneuvers under various kinematic conditions," *Accident Investigation Quarterly*, no. 39, 2004.

[31] R. Kiefer, D. LeBlanc, and C. Flannagan, "Developing an inverse time-to-collision crash alert timing approach based on drivers' last-second braking and steering judgments," *Accident Analysis & Prevention*, vol. 37, no. 2, pp. 295–303, 2005.

[32] S. Noh and W.-Y. Han, "Collision avoidance in on-road environment for autonomous driving," in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*. IEEE, 2014, pp. 884–889.

[33] S. Sontges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1891–1897.

[34] A. Tamke, T. Dang, and G. Breuel, "A flexible method for criticality assessment in driver assistance systems," in *2011 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 697–702.

[35] E. Sørbø, "Vehicle collision avoidance system," Master's thesis, Institutt for teknisk kybernetikk, 2013.

[36] M. Ali, A. Gray, Y. Gao, J. Hedrick, and F. Borrelli, "Multi-objective collision avoidance," in *Dynamic Systems and Control Conference*, vol. 56147. American Society of Mechanical Engineers, 2013, p. V003T47A004.

[37] A. Gray, M. Ali, Y. Gao, J. Hedrick, and F. Borrelli, "A unified approach to threat assessment and control for automotive active safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1490–1499, 2013.

[38] S. Brunson, E. Kyle, N. Phamdo, and G. Preziotti, "Alert algorithm development program: Nhtsa rear-end collision alert algorithm," National Highway Traffic Safety Administration, Tech. Rep., 2002.

[39] Y. Zhang, E. Antonsson, and K. Grote, "A new threat assessment measure for collision avoidance systems," in *2006 IEEE Intelligent Transportation Systems Conference*. IEEE, 2006, pp. 968–975.

[40] E. Bauer, F. Lotz, M. Pfromm, M. Schreier, S. Cieler, A. Eckert, A. Hohm, S. Lüke, P. Rieth, B. Abendroth, V. Willert, J. Adamy, R. Bruder, U. Konigorski, and H. Winner, "Proreta 3: An integrated approach to collision avoidance and vehicle automation," *at - Automatisierungstechnik*, vol. 60, 12 2012.

[41] A. Balachandran, M. Brown, S. Erlien, and J. Gerdes, "Predictive haptic feedback for obstacle avoidance based on model predictive control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 26–31, 2015.

[42] M. Secanell, J. Wishart, and P. Dobson, "Computational design and optimization of fuel cells and fuel cell systems: a review," *Journal of Power Sources*, vol. 196, no. 8, pp. 3690–3704, 2011.

[43] A. Bazmi and G. Zahedi, "Sustainable energy systems: Role of optimization modeling techniques in power generation and supply—a review," *Renewable and sustainable energy reviews*, vol. 15, no. 8, pp. 3480–3500, 2011.

[44] H. Rahimian and S. Mehrotra, "Distributionally robust optimization: A review," *arXiv preprint arXiv:1908.05659*, 2019.

[45] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control Theory and Design*. Nob Hill Publishing, Madison, WI, 1999.

[46] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[47] S. Anderson, S. Peters, T. Pilutti, and K. Iagnemma, "Design and development of an optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," in *Robotics Research*. Springer, 2011, pp. 39–54.

[48] S. Anderson *et al.*, "A unified framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[49] A. Gray, Y. Gao, T. Lin, J. Hedrick, and F. Borrelli, "Stochastic predictive control for semi-autonomous vehicles with an uncertain driver model," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 2329–2334.

[50] B. Gutjahr and M. Werling, "Automatic collision avoidance during parking and maneuvering—an optimal control approach," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, 2014, pp. 636–641.

[51] R. Hult, G. Campos, P. Falcone, and H. Wymeersch, "An approximate solution to the optimal coordination problem for autonomous vehicles at intersections," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 763–768.

[52] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Primal decomposition of the optimal coordination of vehicles at traffic intersections," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 2567–2573.

[53] M. Zanon, S. Gros, H. Wymeersch, and P. Falcone, "An asynchronous algorithm for optimal vehicle coordination at traffic intersections," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 12 008–12 014, 2017.

[54] S. Shladover, C. Desoer, J. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle con-

trol developments in the path program," *IEEE Transactions on vehicular technology*, vol. 40, no. 1, pp. 114–130, 1991.

[55] R. Behringer and N. Muller, "Autonomous road vehicle guidance from autobahnen to narrow curves," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 810–815, 1998.

[56] V. Kunchev, L. Jain, V. Ivancevic, and A. Finn, "Path planning and obstacle avoidance for autonomous mobile robots: A review," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2006, pp. 537–544.

[57] Y.-K. Hwang and N. Ahuja, "Gross motion planning—a survey," *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 219–291, 1992.

[58] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[59] C. Sammut and G. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[60] M. Gökçe, E. Öner, and G. Işık, "Traffic signal optimization with particle swarm optimization for signalized roundabouts," *simulation*, vol. 91, no. 5, pp. 456–466, 2015.

[61] M. Saska, M. Macas, L. Preucil, and L. Lhotska, "Robot path planning using particle swarm optimization of ferguson splines," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*. IEEE, 2006, pp. 833–839.

[62] S. Rotter, "Swarm behaviour for path planning," *Master's thesis, Freie Universität Berlin*, 2014.

[63] F. Ulbrich, S. S. Rotter, and R. Rojas, "Adapting to the traffic swarm: swarm behaviour for autonomous cars," in *Robotic Systems: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2020, pp. 1391–1414.

[64] S. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

[65] J. Nasir, F. Islam, and Y. Ayaz, "Adaptive rapidly-exploring-random-tree-star (rrt*)-smart: algorithm characteristics and behavior analysis in complex environments," *Asia-Pacific J. of Inf. Tech. and Multimedia*, vol. 2, p. 39, 2013.

[66] J.-H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the rrt," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 3276–3282.

[67] D.-W. Gong, J. Zhang, and Y. Zhang, "Multi-objective particle swarm optimization for robot path planning in environment with danger sources." *J. Comput.*, vol. 6, no. 8, pp. 1554–1561, 2011.

[68] A. Atyabi and D. Powers, "Review of classical and heuristic-based navigation and path planning approaches," *International Journal of Advancements in Computing Technology*, vol. 5, no. 14, p. 1, 2013.

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

[70] A. Rashid, A. Ali, M. Frasca, and L. Fortuna, "Path planning with obstacle avoidance based on visibility binary tree algorithm," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1440–1449, 2013.

[71] M. Reiter and D. Abel, "Two and a half carrots-a versatile and intuitive optimisation-based path-following approach for road vehicles," in *2015 23rd Mediterranean Conference on Control and Automation (MED)*. IEEE, 2015, pp. 364–370.

[72] C. Carbone, U. Ciniglio, F. Corraro, and S. Luongo, "A novel 3d geometric algorithm for aircraft autonomous collision avoidance," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 1580–1585.

[73] P. Bhattacharya and M. Gavrilova, "Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path," *IEEE Robotics & Automation Magazine*, vol. 15, no. 2, pp. 58–66, 2008.

[74] M. Jankovic and M. Santillo, "Collision avoidance and liveness of multi-agent systems with cbf-based controllers," *arXiv preprint arXiv:2012.10261*, 2020.

[75] S. Dergachev and K. Yakovlev, "Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path finding," *arXiv preprint arXiv:2107.00246*, 2021.

[76] P. Velagapudi, K. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4603–4609.

[77] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[78] L. Cuenca, "Collision avoidance algorithms in 2d using voronoi diagrams," 2021. [Online]. Available: https://github.com/leduin/Collision-Avoidance-Algorithms-in-2D-using-Voronoi-Diagrams

[79] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha, "Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation," 2016. [Online]. Available: http://gamma.cs.unc.edu/RVO2/

[80] L. Liu, "Python bindings for optimal reciprocal collision avoidance," 2017. [Online]. Available: https://github.com/Taospirit/Python-RVO2