UNIVERSIDAD NACIONAL DE COLOMBIA

Multiagent Control of Autonomous Vehicles in Presence of Non-Cooperative Agents using Game Theory

Nestor Ivan Ospina Gaitan

Universidad Nacional de Colombia
Engineering Faculty, Electronic and Electric Engineering Department.
Bogotá, Colombia
Año 2023

Multiagent Control of Autonomous Vehicles in Presence of Non-Cooperative Agents using Game Theory

Nestor Ivan Ospina Gaitan

Thesis presented as partial requirement for apply to the tittle of:
Master in Industrial Automation

Advisor:
Ph.D. Eduardo Alirio Mojica Nava
Co-Advisor:
Ph.D. Duván Andrés Téllez Castro

Research Line:
Control and Robotics
Research Group:
PAAS - Processing, Acquisition and Analysis of Signals

Universidad Nacional de Colombia
Engineering Faculty, Electronic and Electric Engineering Department
Bogotá, Colombia
Año 2023

Acknowledgements

I want to thank Professor Eduardo Alirio Mojica and Co-advisor Duván Andrés Téllez Castro for their unconditional help all these years. To Professor Juan Calderon and Ing. Gustavo Cardona for their knowledge, motivation, and support in achieving this goal. Finally, to my mother and brother for always being emotional and unconditional support.

Titulo en español: Control multiagente de vehículos autónomos en presencia de agentes no cooperativos utilizando teoría de juegos

Resumen

Esta tesis propone una solución al problema de la conducción autónoma de vehículos en un entorno vial, concretamente en presencia de vehículos conducidos por agentes con decisiones egoístas y maniobras agresivas. El controlador trata de resolver el problema de optimización basado en un Control Predictivo de Modelo (MPC). Aprovechando la técnica anterior de predicción de trayectoria, el controlador la utiliza para predecir mejor la posición de los vecinos y planificar su trayectoria. Además, el modelo predictivo puede resolver el problema de control óptimo al cumplir con las restricciones de seguridad, evitar obstáculos y lograr su objetivo principal.

El problema de control óptimo tiene restricciones no convexas debido a las variables enteras mixtas en las que se basa. Mediante la creación de MPC no lineales que puedan lidiar con el problema de las variables híbridas, se busca resolver el problema de conducción de vehículos frente a decisiones agresivas y no cooperativas para la red.

Además, todos los agentes del sistema pueden controlarse mediante la creación de controladores locales basados en *Teoría de Juegos*. Analizamos dos métodos para encontrar una solución óptima: centralizado y descentralizado. El controlador más eficaz y viable se elige después de una investigación objetiva y la comparación de todos los demás. Dado que el MPC proporciona la mejor solución para toda la planta, se utiliza como punto de referencia. El primer algoritmo descompuesto es MPC centralizado, en el que los subsistemas vecinos entregan la información al nodo central, calculan las nuevas rutas y transmiten en cada iteración del controlador por MPC. El segundo enfoque se basa en MPC descentralizado distribuido óptimo. Los coches se basan en la teoría del Juego de Potencial Generalizado en ambos casos. Cada agente resuelve su problema secuencialmente y comparte su próximo movimiento con los vecinos, buscando un equilibrio ϵ-Nash. Ambos conductores pueden calcular su trayectoria de manera factible confiando en restricciones adicionales mientras evitan otros vehículos.

Los controladores distribuidos se evalúan en tres escenarios diferentes, utilizando tres criterios: la eficiencia del controlador global, el tiempo que tarda cada controlador en encontrar una respuesta y la viabilidad del controlador con el aumento de pasos que el controlador debe predecir. El primer escenario da una idea del comportamiento del controlador frente a agentes con maniobras desconocidas; el segundo muestra el comportamiento del controlador frente a mayores restricciones y conexiones con vecinos, y el tercero prueba el controlador reduciendo sus variables ambientales.

Palabras clave: Teoria de juegos potenciales con enteros mixtos, control óptimo, control predictivo de modelo, conducción autónoma, red descentralizada .

Abstract

This thesis proposes a solution to the problem of autonomous vehicle driving in a road environment, specifically in the presence of agent-driven vehicles with selfish decisions and aggressive maneuvers. The controller tries to solve the optimization problem using a *Model Predictive Control*(MPC). Taking advantage of the previous technique for trajectory prediction, the controller uses this to better predict the neighbors' position and plan its trajectory. In addition, the predictive model can solve the *Optimal Control Problem* by complying with security restrictions, avoiding obstacles, and achieving its primary objective.

The *Optimal Control Problem* has non-convex constraints due to its based on mixed-integer variables. By creating non-linear MPC that can deal with the problem of hybrid variables, it is sought to solve the problem of driving vehicles against aggressive and non-cooperative decisions for the network.

Furthermore, all agents in the system can be controlled by creating local controllers based on *Game Theory*. We analyzed two methods to find an optimal solution: centralized and decentralized. The most effective and viable controller is chosen after objective research and comparison of all others. Since the centralized MPC provides the best solution for the entire plant, it is used as a benchmark. The first decomposed algorithm is centralized MPC, in which the neighboring subsystems give the information to the central node, calculate the new routes and transmit in each iteration of the MPC. The second approach is based on optimal distributed decentralized MPC. The cars are based on the *Generalized Potential Game theory* in both cases. Each agent solves its problem sequentially and shares its next move with neighbors, looking for a ϵ-Nash equilibrium. Both drivers can feasibly calculate their trajectory by relying on additional constraints while avoiding other vehicles.

Distributed controllers are evaluated in three different scenarios, using three criteria: the efficiency of the global controller, the time it takes for each controller to find an answer, and the feasibility of the controller with the increase in steps that the controller must predict. The first scenario gives an idea of the controller's behavior against agents with unknown maneuvers; the second shows the controller's behavior against increased constraints and connections with neighbors, and the third tests the controller by reducing its environmental variables.

Keywords: Generalized Mixed-Integer Potential Game, Optimal Control, Model Predictive Control, Autonomous Driving, Decentralized Network

Figure List

Content

Symbol List

Symbol	**Item**
a_i	Game theory i-th action profile
A_i	Vehicles states matrix of agent i-th
$\mathcal{A}$	Overall strategies set
$\mathcal{A}_i$	Strategies set of i-th agent
a_i	Aceleration of the agent i
a_{ij}	Graph adjacency matrix
A_m	Reference state matrix
A_d	Adjacency matrix graph representation
b_i	Input vector
b_m	Reference input vector
C	Cooperative agents set
$\tilde{C}$	Non-Cooperative agents set
c_i	Vehicles output vector
d	Disagreement point
$\mathcal{D}^{\int}$	Safety Distance to avoid collition longitudinal
$\mathcal{D}^{\updownarrow}$	Safety Distance to avoid collition lateral
d_{ij}	Distance between the i'th and j'th vehicle
$\mathcal{D}$	Degree matrix graph representation
$\mathbb{E}$	Set of graph communication edge
e_i	Difference between disagreement point and cost function discrete Bargaining game
E_i	i-th agent ADMM constraints matrix
F_i	i-th agent model predictive control objective function weights matrix
G	Game definition

Symbol	**Item**
H_p	Prediction Horizon
H_i	i-th agent model predictive control hessian matrix
i	Agent or player
J	Cost function for optimization problems
j	Neighbor agent
k	Steptime
k_{mi}	Adaptive constant related to reference state matrix
k_{ri}	Adaptive constant related to reference input vector
k_{mij}	Adaptive constant related to neighbors state matrix
k_{rij}	Adaptive constant related to neighbors input vector
L	Bargaining game collective
M_i	Game theory fitness function
$\hat{M}_i$	Excess payoff of i-th player strategy
$\bar{M}_i$	Average payoff of i-th player strategy
$\mathcal{N}_i$	Group of agents in the highway
N_u	control horizon
P	Riccati equation solution matrix
$\mathcal{P}$	Game Theory population set
Q	Weight variable associated with the states of the system for optimization
Q_uu	Weighting matrix inputs optimization problem
Q_uui	Weighting matrix inputs optimization problem for i-th agent
Q_uux	Weighting matrix input-state optimization problem
Q_xui	Weighting matrix input-state optimization problem for i-th agent
Q_xx	Weighting matrix states optimization problem
R	Weight variable associated with the inpuit of the system for optimization
S	Game decision space
T	Prediction Horizon of MPC
u_ad	Optimal modification auxiliary variable
u_i	i-th control action
U_i	Utility functions set

Symbol	**Item**
v_i	Velocity of agent i
$\mathbb{V}$	Set of graph nodes
W	Weakly pareto optimal subset
$\mathcal{W}$	Weights graph representation
$\mathcal{Z}$	Total amount of lanes
z_i	Current lane of the agent i
$z_{i,j}$	Lane difference of agent i to agent j

1. Introduction

With the coming of the Industrial Revolution, human beings started to use new machines for transport, like vehicles powered by combustion engines. These efficient and versatile machines made more accessible and faster the transportation of people, animals, and commodities. However, more than 200 years have passed since constructing the first automobile powered by a vapor engine. Due to the success of this new machine, it has undergone many modifications and upgrades to increase its efficiency, speed, and security. Nevertheless, the most crucial change in those vehicles is their energy source. The first mobile vehicles were powered by carbon, making the use of wood or carbon necessary to power the engine. It could be a great engine, but the massive load and space needed to store this material make it inefficient and annoying in its use. Later, the combustion engine was created by Étienne Lenoir around 1860. It uses fossil fuels like gasoline, natural gas, or diesel as a power source and fossil oils for its lubrication and maintenance. However the electric vehicle was invented in the 19th century, but its battery autonomy and energy capacity were inefficient. Eventually, the capacity and efficiency of electric engines and batteries have grown, making them more accessible, efficient, and powerful battery electric vehicles than internal combustion ones. Finally, the electric vehicle is manufactured and marketed in mass production by companies like Tesla, Chevrolet, and BMW.

The use of electric energy is versatile because it can be transported and manipulated without significant losses; in addition, its weight is insignificant compared to oil and carbon. Besides, it can be easily transformed into multiple other kinds of energies, such as cinematic, thermic, mechanic, and electromagnetic. Due to this facility, electric energy is the best way to get, keep and use in different applications. It is advantageous to use this kind of energy as a source of power in a vehicle, which can be equipped with many power tools that make it an electric machine with many features. One of the great qualities is the automation system, which allows it to autonomously make acceleration, braking, and steering decisions [10]. It makes a technological leap in safety and efficiency. Self-driving is very important because it will make changes in the actual way of transportation. A car will not need a human driver to go from one point to another. It means that an artificial driver will make decisions instead of a human. As a result, the artificial driver will drive efficiently, safely, cooperatively, and smartly. In conclusion, humanity will enjoy revolutionary transportation with the benefits of a car but without the disadvantages of the unsafely and the brain load of making decisions independently.

The coming of electric vehicles into the world makes it possible to develop, build, and use computer systems that let vehicles be driven autonomously without the supervision of a human being, thus making human transportation safe, efficient, and optimal. However, the advantage of 5G technology is that electric and autonomous vehicles can communicate with others to achieve an individual goal cooperatively [11]. On the whole, self-driving currently requires that vehicles cooperate, considering the presence of human drivers. The human driver performs unknown strategies that hinder interaction on the road and lead to unsafe or inappropriate maneuvers. It is expected that autonomous driving will not need a human supervisor for safe working in the future. Due to this reason, in this thesis, we model a human driver as a selfish and non-cooperative agent who seeks their benefit. Simplifying autonomous driving in a natural environment, we formulate an environment with autonomous agents that share their physical variables (velocity, acceleration, lane number, and position) but ignore the non-cooperative agent (human driver) strategy. We intend to solve this problem using a novel method based on Nash equilibrium, which efficiently moves the cooperative agent to a specific objective. It considers each of the different objectives and the different dynamics of each agent. We also use “Mixed Integer” variables to change the behavior of each autonomous agent in the presence of non-autonomous vehicles and ensure safety on the road. Finally, we use a technique of control MPC to predict the states of each agent on the whole network at a specific time in the future and decide the optimal strategy.

1.1. Automated Driving

Automated driving, also known as a self-driving car, is the action of a vehicle or machine to move through an environment making the right decisions to achieve goal and objective [41], [42]. Automated vehicles combine techniques and hardware tools to capture, process, and control signals in their systems. Commonly, hardware tools allow it to sense the Environment, e.g., GPS, LIDAR, radar, sonar, odometry, 3D vision, and cameras, among others. Moreover, these vehicles can use multiple control hardware such as power systems, electric engines, and actuators to interact with the environment [23], [21].

This technology has multiple applications in personal transportation, package delivery, Robotaxis, or platoons of connected vehicles transporting loads. It is worth mentioning that the autonomous driving theory could be used in any other application that solves the problem of driving through a place, avoiding collisions, and arriving somewhere. Some of those applications could be a waiter, a dispenser pills nurse, a coffee dispenser, a tool mechanic assistant, among others [23],[40].

1.2. Control Strategies

Autonomous driving is a challenging task to solve. Multiple solutions have been developed, like artificial intelligence, adaptive control, machine learning algorithms, and non-linear control [1, 2]. As a result, new strategies make managing and controlling complex systems easier with better accuracy and robustness than years before. Nevertheless, no one has an optimal solution to this challenge [6].

The principal feature to consider to solve this challenge is the data environment. It could give a controller enough information to achieve the desired position. Due to this, a suitable control technique and a very well set of parameters are essential. The right strategy will depend on what strategies are being used and when it could be better than any other. In this thesis, the main focus is on control strategies of automated driving. We will explain some control strategies for a specific job and how those are implemented in the specific problem of automated driving.

1.2.1. Model Predictive Control

One of the most popular control theories implemented in non-linear systems is Model Predictive Control (MPC). This control method used to optimize a cost function while satisfying a set of constraints. In other words, MPC appears to be an optimal solution to predict future states while applying constraints to achieve their primary objective. This optimization requires advanced algorithms that solve quadratic problems with constraints [19].

In particular, MPC minimizes the value of a function representing the control problem's characteristics and objectives. Some could be power consumption, efficiency, precision, and others. Furthermore, it is essential to obtain a dynamic model representing the entire system's behaviour. This model is the principal property needed for an MPC to do the control task efficiently. Depending on the conditions and properties of the problem control, it is also essential to implement some constraints that contain and restrain the entire system drop under incorrect conditions [28].

1.3. Game Theory

Game theory is defined in [31] as "The study of mathematical models of conflict and cooperation between intelligent, rational decision-makers". One way to analyze the control game theory is in cooperative games, where Two or more players try to get the most benefit possible depending on neighbors' decisions. Each agent has to take into account the action of the others to make an optimal decision benefiting the whole group. The primary motivation of game theory is to achieve maximum profit to get close to or achieve the goal. Moreover, a game theory is implemented from a cooperative perspective. There can be many agents with the same objective function, and each cooperative agent must make the optimal decision to achieve the global objective. Game theory is functional in environments with multiple interacting controllers, and the decision of each one depends on the other.

2. Mathematical Background

Frequently multi-agent systems are modeled as a distributed optimization problem. Each agent looks for an optimal solution to a specific task, and it could solve it with the cooperation of its neighbors or without it. Indeed, it is essential to design and implement strategies to manage communication and control the interconnect network optimally. This chapter presents a brief explanation of different techniques used for this purpose. Some relate to modeling, analyzing, and managing the entire network of agents. The control and communications techniques are shown in-depth in Chapter 4.

2.1. Graph Theory

A graph is defined as a mathematical and graphical representation of a network. Some of the most common uses of graph representation are people's social interaction in social networks, interconnected networks of robots, servers connected to the internet, cellphones connected to the telephone service, and much more, **Fig. 2-1**.

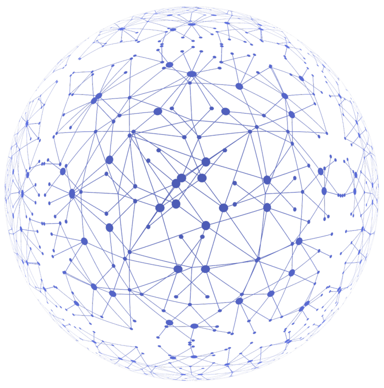


Fig. 2-1.: Social Network.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ consists of the joint of three fundamental parts. A set of vertices (or nodes) $\mathcal{V} = \{v_o, v_1, ...v_n\}$, A set of edges (or links) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} = \{e_{1,2}, e_{2,3}, ..., e_{i,j}\}$, and a weight matrix $\mathcal{W}$ defined as:

$$\mathcal{W} = \mathcal{W}_{i,j} \begin{cases} > 0 & if(i;j) \in \mathcal{E}, \\ = 0 & if(i;j) \notin \mathcal{E}. \end{cases} \tag{2-1}$$

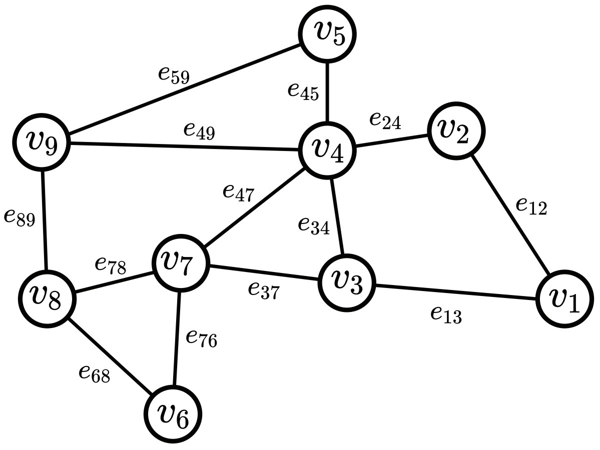


Fig. 2-2.: Graph of Network system.

A set of vertices is a group of agents (or nodes) $v = 1, ..., n$, where each node is an information source. It could be auto-generated or retransmitted by other nodes. In **Fig. 2-2** the set of vertices is $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9\}$. However, the edges (or links) are how the information is shared. It could be directed or undirected depending on how the information is transmitted if just a node can transmit or transmit-receive. In the previous example, the set of edges is $\mathcal{E} = \{e_{12}, e_{13}, e_{24}, e_{34}, e_{37}, e_{45}, e_{47}, e_{49}, e_{59}, e_{68}, e_{76}, e_{78}, e_{89}\}$, and lastly the weight matrix is an algebraic representation of the entire interconnected system. If exist a connection between vertice i and j in the matrix, it will be 1; otherwise, it will be 0. In **Fig. 2-2** the weight matrix is:

$$\mathcal{W} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The neighborhood of node v_i is the set of nodes $\mathcal{N}_i \subseteq \mathcal{V}$ that have a direct interaction with agent i. This set of edges could be directed or undirected depending on their communication. A graph $\mathcal{G}$ is connected if a connection exists between two or more nodes and there is a route between all nodes in the set $\mathcal{V}$. Finally, the set of nodes of a graph has a degree. It depends on the number of neighbors of each node. If all graph nodes have the same degree, the graph is called a regular graph.

A graph could be represented as a joint of multiple matrices. These matrices show information about each node with the other.

Degree and Adjacency Matrix

The degree of a node v_i is represented as $d(v_i)$, which is the name of the cardinality of a node, and it represents the number of agents connected with node i. Therefore, the degree matrix is a diagonal matrix with the degree values of each node:

$$\Delta(\mathcal{G}) = \begin{bmatrix} d(v_1) & 0 & \cdots & 0 \\ 0 & d(v_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d(v_n) \end{bmatrix} \tag{2-2}$$

In addition, an adjacency matrix exists where the relationship of each node with each other is represented. Moreover, the adjacency matrix is defined as:

$$[Y(\mathcal{G})]_{ij} = \begin{cases} 1, & \text{if } v_i v_j \in \mathcal{E}, \\ 0, & \text{Otherwise.} \end{cases} \tag{2-3}$$

Incidence Matrix

There are directional or bidirectional graphics on its edges. If there is a graph with directed edges, it is called a digraph $\mathcal{D}$. Therefore, a digraph is described by its incidence matrix, representing the orientation of each edge of the graph. With the previous information, it is possible to analyze factors such as stability, network controllability, communication times, and the sequence that the information follows to reach each agent. The incidence matrix is defined as:

$$I_{ij}(\mathcal{G}) = \begin{cases} -1, & \text{if } v_i \text{ is the tail of } v_j, \\ 1, & \text{if } v_i \text{ is the head of } v_j, \\ 0, & \text{Otherwise.} \end{cases} \tag{2-4}$$

A graph could also have weights in its connections, representing the importance of the neighbour's information on each node. If a graph has weighted edges, it is called a weight graph. Consequently, in a weight graph, the adjacency matrix is built by the following definition:

$$Ad(\mathcal{G}) = \begin{cases} \omega_{ij}, & \text{if } (v_i v_j) \in \mathcal{E}, \\ 0, & \text{Otherwise.} \end{cases} \tag{2-5}$$

Laplacian

The Laplacian of a graph is another fundamental representation of a graph. The way of representing the Laplacian of a graph $\mathcal{G}$ is by using the joint of adjacency and degree matrix as follows:

$$L(\mathcal{G}) = \Delta(\mathcal{G}) - Y(G) \tag{2-6}$$

Laplacian matrix must be symmetric if it is a digraph or asymmetric otherwise.

2.2. Model Predictive Control

The MPC is a control theory used to manage some variables optimally into a function looking at a specific result. The main idea is to use a discrete model of the entire system to predict future system states. With the capability of predicting future behaviours over a finite-time prediction and control horizon H_u and H_p, depending on different inputs. It is possible to optimize the input control to achieve an optimal solution. The process is to apply an optimal input signal to the system and recalc the next optimal step. This process is repetitive, while the system aims at the principal objective [3].

The first step measures the error of the actual and desired state and the amount of energy implemented, as shown in **Fig. 2-3**. With this information, optimize the states error and control input to achieve its objective **Fig. 2-4**. MPC benefits are achieving the main goal while considering some constraints, the ease of design, simulation, and implementation in different systems, and the powerful way to control linear and nonlinear systems. This section describes the mathematical theory of an MPC and its applications in the described scenario.

2.2.1. Dynamic Prediction Model

Considering a discrete model system (2-7) and the observably output model (2-8), where $x(t) \in \mathbb{R}^n$ represents the state values at time t, the $u(t) \in \mathbb{R}^m$ variable as the discrete control input signal at the time t. The output is represented as $y(t) \in \mathbb{R}^p$.

$$x(t+1) = f(x(t), u(t)), \tag{2-7}$$
$$y(t) = g(x(t), u(t)). \tag{2-8}$$

The dynamic Equation 2-7 express the evolution of the states $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ over the horizon H_u and the output function shows the behavior $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ over the horizon H_p. In order to reduce computation load and time delays, a linearized prediction model can be set up using the state evolution in 2-15. This thesis document linearized some convex models; however, all dynamic models will be explained in-depth in the next chapter. While the controller works, this linearization is processed in each iteration around the actual states. Hence, a matrix E is added to the state matrix A and input matrix B. Future states are predicted using the following iterative substitution:

$$x(t+1) = Ax(t) + Bu(t) + E \tag{2-9}$$
$$x(t+2) = Ax(t+1) + Bu(t+1) + E \tag{2-10}$$
$$= A^2x(t) + ABu(t) + AE + Bu(t+1) + E \tag{2-11}$$
$$\vdots \tag{2-12}$$
$$x(t+H_u) = A^{H_u}x(t) + A^{H_u-1}Bu(t) + \cdots + Bu(t+H_u-1) + A^{H_u-1}E + \cdots + E \tag{2-13}$$
$$\vdots \tag{2-14}$$
$$x(t+H_p) = A^{H_p}x(t) + A^{H_p-1}Bu(t) + \cdots + Bu(t+H_p-1) + A^{H_p-1}E + \cdots + E \tag{2-15}$$

The resulting system is represented by matrices $A(t) \in \mathbb{R}^{n\times n}, B(t) \in \mathbb{R}^{n\times m}, C(t) \in \mathbb{R}^{p\times n}$ and $E(t) \in \mathbb{R}^n$ after linearization. The previous predicted linear model is used when the dynamic model, constraints, and objective function are nonlinear and the computing system has low resources. If this linearization is avoided, the calculation could be slower, and the control system could be unstable.

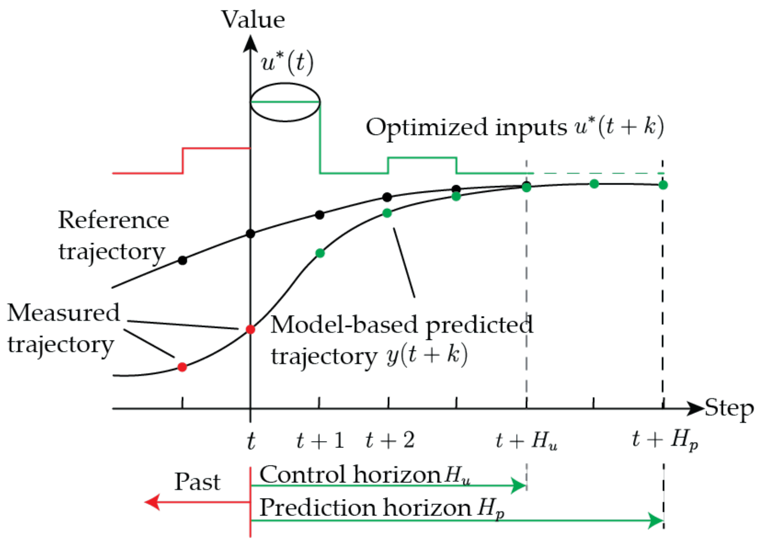


Fig. 2-3.: Control and state signals, First iteration.

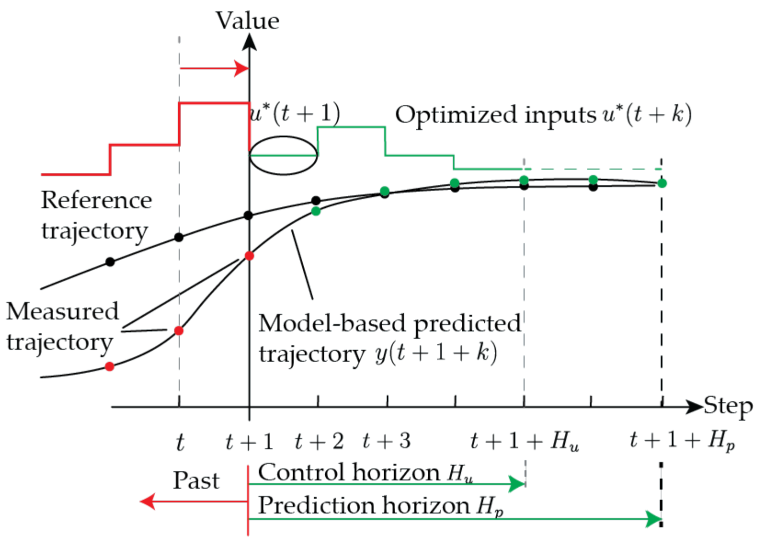


Fig. 2-4.: Control and state signals, Second iteration.

2.2.2. Objective Function

The objective function is the principal function representing the controller's aim that needs to be achieved. This function is also called the cost function. Generally, it is divided into two parts: the *Running cost* and *Terminal costs*. The running cost penalizes the difference between the actual and desired states and control inputs. It allows for tracking a reference and optimizing the consumption of energy. The latter penalizes the weighted difference between the final predicted and the final reference step $\tilde{x}(H_p)$. The objective function is defined as:

$$V_i(\tilde{x}, \tilde{u}) = \underbrace{\sum_{k=1}^{H_p-1} \tilde{x}(k)^T Q_i \tilde{x}(k) + \sum_{k=1}^{H_u-1} \tilde{u}(k)^T R_i \tilde{u}(k)}_{\text{Running cost}} + \underbrace{\tilde{x}(H_p)^T P \tilde{x}(H_p)}_{\text{Terminal cost}}. \tag{2-16}$$

Using 2-15 and replacing in 2-16, the objective function is now:

$$V(x, u) = x^T Q x + u^T R u. \tag{2-17}$$

The terminal cost is augmented in the running cost using the subsequent definitions. The weight matrix Q is represented using $\tilde{Q} = diag(Q_1, \cdots, Q_{H_p-1}, P)$, and $Q_i, P \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^m$. The evolution of the states over the horizon H_p and H_u are denoted by the matrices $\tilde{x}$ and $\tilde{u}$ as follows:

$$\tilde{x} = \begin{bmatrix} x(1) - x_r ef(1) \\ x(2) - x_r ef(2) \\ \vdots \\ x(H_p) - X_{ref}(H_p) \end{bmatrix}, \tilde{u} = \begin{bmatrix} u(1) - u_r ef(1) \\ u(2) - u_r ef(2) \\ \vdots \\ u(H_u) - u_{ref}(H_u) \end{bmatrix}. \tag{2-18}$$

If the MPC objective function is linearized, the number of variables to optimize will decrease, redefining 2-17. This function only will depend on the control input variables. It will convert the convex function into a Quadratic Problem (QP), where the variable to optimize is the sequence of the predicted inputs. The optimization problems become as:

$$\begin{matrix} \text{mín}_{U_{H_u}} & J \\ s.t. & Au \leq b. \end{matrix} := \begin{matrix} \text{mín}_{U_{H_u}} & \frac{1}{2} u_{H_u}^T P u_{H_u} = \frac{1}{2} Q u_{H_u} + r_0 \\ s.t. & Au_{H_u} \leq b. \end{matrix}. \tag{2-19}$$

Each term in 2-19 is defines as follows:

$$u_{H_u} = \begin{bmatrix} u(1) \\ u(2) \\ \vdots \\ u(H_u - 1) \end{bmatrix}, P = S^T \tilde{A} S + R, q = \begin{bmatrix} x(k)^T \tilde{x}_{H_p}^T \tilde{u}_{H_u} \end{bmatrix} \begin{bmatrix} S^T \tilde{Q} T \\ QS \\ R \end{bmatrix}, \tag{2-20}$$

T and S are denoted by:

$$T = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, S = \begin{bmatrix} 0 & 0 & \cdots & \cdots & \cdots & 0 \\ B & \ddots & & & & \\ AB & \ddots & \ddots & & & \\ A^B & \ddots & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & A^2B & AB & B & 0 \end{bmatrix}. \tag{2-21}$$

2.2.3. Constraints

One of the advantages of MPC is the good handly management of constraints. The following are the benefits of the use of an MPC controller with constraints

- States, inputs, and dynamic model systems can implement constraints ensuring safety control.
- The use of constraints can guarantee stability and robustness to the whole system.
- The MPC design could restrict the control input to ensure physical limitations on actuators. However, the output or the system also could be constrained to represent a limitation on the agent.

The input, state, and output variables are constrained, limiting the workspace across their prediction horizon. On the whole, constraints results are represented as:

$$x(t+k) \in \mathcal{X} \subseteq \mathbb{R}^n, k = 1, \cdots, H_p, \tag{2-22}$$
$$y(t+k) \in \mathcal{Y} \subseteq \mathbb{R}^p, \ \ k = 1, \cdots, H_p, \tag{2-23}$$
$$u(t+k) \in \mathcal{U} \subseteq \mathbb{R}^m, k = 0, \cdots, H_u - 1. \tag{2-24}$$

2.3. Game Theory

Fig. 2-5.: Strategy in economics environment.

Game theory is a mathematical method commonly used to analyze and achieve the best decision in a multi-agent environment. It is highly used in cooperative or competitive games. Nobel price John Nash originates it in [33, 32], which establishes the basis of this novel theory.

Nash developed the theory to be implemented in social sciences and economics fields. The general idea in economics is to make decisions in a market, considering that the other agents' decisions can affect the price of some commodity. A company or agent has to take action to achieve the desired goal **Fig. 2-5**. Eventually, this theory has been proven in engineering, computer science, philosophy, and many other fields [31], [24], [43].

From the control theory perspective, it is possible to interpret game theory law as a sort of "intelligent, rational decision-maker" [27]. In other words, game theory is the study of conflict and cooperation between interacting controllers, where success depends on the agents, the preferences, and the settings.

One set is a zero-sum game, where more than one player exists, and the profit of one player is to the detriment of the other. That is, the sum of the profits is zero. It is common to use this setting where the game has limited resources and the objective of each agent is the same.

Another setting is team games. In this setting, a group of agents looks for a cooperative strategy to achieve the global objective. This kind of game's architecture is commonly decentralized, and each agent has the same goal.

The third setting is where agents do not work cooperatively. Each agent tries to achieve a local objective in an environment depending on the neighbors' decisions, but each has a different objective. This kind of setting is usually decentralized and non-cooperative.

This section gives a simple example of a discrete matrix game before explaining the elements of a game theory in a continuous space. Finally, it is explained the third set of potential games specifically. All information explained in this section is obtained from [15], [16], [17], [25], [26], [43].

The discrete matrix form is the most simple and powerful representation of the perspective and profit of each agent. A game written in matrix form shows all possible decisions of an agent and its corresponding utility or cost. Next is explaining one of the most famous examples of matrix games.

Example 2.3.1 (Game Theory: Prisoner's dilemma) *Two criminal of a gang has been arrested and imprisoned. Each prisoner is in a separate room, and no one knows about his couple. Police admit they do not have enough evidence to sentence these criminals to jail. They plan an strategy to get one of them to give enough information to solve the case. The strategy is to inform them that both will be sentenced to a year in prison for the lesser charge. Simultaneously, the police offer each prisoner a bargain. If one testifies against his partner, he will go free, while the other will spend three years in prison on the main charge. However, if both prisoners testify against each other, both will be sentenced to two years of prison. Each criminal (agent) has to make a decision depending on the possible decision of the other one. In matrix form, the prisoner's dilemma game is:*

	B refuses deal	**B testify against his pair**
A refuses deal	1 year, 1 year	3 years, 0 years
A testify against his pair	0 years, 3 years	2 years, 2 years

In the previous example, it is clear that the action of each prisoner would depend on the decision of the other prisoner. In a game, each agent looks at its self-interest. by this, a set of interconnected agents with knowledge of other's decisions, and its self-profit for each possible decision, can make optimal decisions to look at its interest. Game theory helps to formulate this kind of problem mathematically.

2.3.1. Basic Concepts

We begin with a basic overview of game theory. Currently exist many texts explaining this in-depth, including economics ([31], [18], [36]), computer science ([9], [25], [34]), and engineering ([39], [4], [22]).

Game elements

Three elements fundamentally define a game. First is the **set of players**, $\mathcal{P}$. Due to the purpose of this thesis work, we limit the discussion to a finite set of players, i.e.,

$$\mathcal{P} = \{1, 2, ..., N\}.$$

Second, for each player $p \in \mathcal{P}$ there is a **set of strategy**, $\mathcal{S}_p$. The strategy set is

$$\mathcal{S} = \mathcal{S}_1 \times ... \times \mathcal{S}_p,$$

a joint strategy set $s \in \mathcal{S}$ is represented as

$$s = (s_1, s_2, ..., s_p).$$

Also, the notation s_{-p} denotes the set of strategies of players $\mathcal{P} \backslash p$, i.e., players other than player p. Finally, for each player $p \in P$, there is a utility function

$$u_p : \mathcal{S} \to \mathbb{R}.$$

This function expresses the player's preferences over the strategies. For any two joint strategies, $s, \acute{s} \in \mathcal{S}$, player p prefers s to $\acute{s}$ if and only if

$$u_p(s) > u_p(\acute{s}).$$

In utility functions, bigger number is better. In case of $u_p(s) = u_p(\acute{s})$, the player p will be indifferent between which strategy to choose. The **vector utility function** is denoted by u, i.e.,

$$u = (u_1, u_2, ..., u_p) : \mathcal{S} \to \mathbb{R}^P.$$

Sometimes, it is better to express a game in terms of cost functions rather than utility functions. In this case, a small number of the cost function is better. For each player $p \in \mathcal{P}$, there is a cost function

$$c_p : \mathcal{S} \to \mathbb{R}$$

and player p will prefer the joint strategy s over $\acute{s}$ if and only if,

$$c_p(s) < c_p(\acute{s}).$$

A general game can be represented by an optimization problems:

$$\forall i \in P = \{1, 2, ..., N\}\,, \ \min_{s_i \in \mathcal{S}_i} c_p(s_i, \mathbf{s}_{-i}).$$

Where the solution to this optimization problem (OP) is called Nash Equilibrium (NE). We will look at games from each agent's point of view to get more knowledge of Nash equilibrium problems. If an agent has information about the other agents' strategy, his problem becomes simple. Specifically, he would have to solve a problem of a unique agent, where the main task is to choose the best action to get the best utility or minimum cost. However, if agents $-i$ were to commit to playing s_{-1}, the agent i would face the problem of choosing the best response depending on the possible decision of the other agents.

Best Response

The best response of the player i to the set of strategy s_{-i} is a mixed strategy x_i^* such as

$$c_p(s_i^*, s_{-i}) < c_p(s_i, s_{-i}),$$

for all strategies $s_i \in \mathcal{S}$.

Looking at the example of the prisoner dilemma. Suppose one prisoner knows that the other prisoner will refuse the deal. His best response is turning state's evidence. The best response is not unique, and it is not a solution concept. However, it helps to understand arguably the most crucial notion in game theory, the Nash equilibrium. The best response function $BR_p : \mathcal{S}_{-p} \to 2^{\mathcal{S}_p}$ is defined by

$$BR_p(s_{-p}) : \{s_p \in \mathcal{S}_p : u_p(s_p, s_{-p}) \geq u_p(\acute{s_p}, s_{-p}) \text{ for all } \acute{s_p} \in \mathcal{S}_p\}\,.$$

$BR_p(S_{-p})$ is the set of strategies where the maximum utility of the player p in response to the other strategies s_{-p} is achieved. Note that it does not need to be unique and could have multiple best responses.

2.3.2. Nash Equilibrium

At Nash equilibrium, each player's strategy is optimal concerning the other players' strategies. In other words, the set of strategies $(s_1^*, s_1^*, ..., s_P^*) \in \mathcal{S}$ is a Nash equilibrium, if for each p exist a strategy that accomplish

$$s_p^* \in BR_p(s_{-p}^*)$$

Example 2.3.2 (Nash equilibrium) *Two players in a game have one decision variable (x, y) respectively; that means $n_1 = n_2 = 1$. Each decision variable $x \in \mathbb{R}$ and $y \in \mathbb{R}$ denote the player's strategies. The game problem is represented as*

$$\begin{array}{ll} \text{mín}_y \quad y^2 + xy & \text{mín}_x \quad (y-x)^2 \\ s.t.: -1 \leq y \leq 1 & s.t.: 0 \leq x \leq 1 \end{array}, \tag{2-25}$$

solving this optimization problem we get the optimal solution:

$$\mathcal{S}_1(x) = \begin{cases} 1 & : \; x < -2 \\ -x/2 & : \; -2 \leq x \leq 2 \\ -1 & : \; x > 2 \end{cases}, \qquad \mathcal{S}_2(y) = \begin{cases} 0 & : \; y < 0 \\ y & : \; 0 \leq y \leq 1 \\ 1 & : \; y > 1 \end{cases}. \tag{2-26}$$

We can check that exist a unique fixed point of the map $\mathcal{S}_1 \times \mathcal{S}_2$. In words, a pair (x, y) such that $x = S_1(y)$ and $y = S_2(x)$ is $(0, 0)$ which is the Nash Equilibrium of the above game.

A Nash Equilibrium is a stable strategy where any agent is interested in changing its strategy due to it is the best to implement.
In some implementations, the change of strategy could represent a slight change in the cost function. Due to this, players might not change their strategy to the best response, holding in the actual profit. This concept allows to introduce of the idea of an ϵ-NE.

ϵ-Nash Equilibrium

Another concept is that players might not care about changing their strategies to a better one when the amount of utility they could gain is larger than the actual one.

Definition 2.3.1 (ϵ-Nash Equilibrium) *Fix $\epsilon > 0$. A strategy profile $s = (s_1, ..., s_n)$ is an ϵ-Nash equilibrium if, for all agents i and for all strategies $\acute{s}_i \neq s_i$, $u_i(s_i, s_{-i}) \geq u_i(\acute{s}_i, s_{-i}) - \epsilon$.*

This concept has attractive properties. Due to the previous definition, ϵ-Nash always exists. Note that, every Nash equilibrium is inside a ϵ-Nash equilibrium region for any $\epsilon > 0$. The argument that agents are indifferent to small gains is attractive and helpful. Computationally, a better and optimal solution is to search for a ϵ-Nash equilibrium in a finite set of mixed-strategies space rather in an infinite continuous space. Usually, an algorithm starts with common ϵ and continuously adapts the parameter to smaller values until they find an acceptable ϵ-Nash equilibrium. However, ϵ-Nash equilibrium has some difficulties, one of these is a Nash equilibrium is always surrounded by ϵ-Nash equilibrium, but the opposite is not true. In words, a given Nash equilibrium can not be close to any Nash equilibrium. [25]

2.3.3. Generalized Nash Equilibrium Problem

The Generalized Nash Equilibrium Problem (GNEP) concept is an extention of the clasical nash equilibrium. The GNEP assums that each player in a game P has a finite set of estrategies, but this set depends on the others player's strategies.

Definition 2.3.2 (Generalized Nash Equilibrium Problem) *If an agent's feasible set of strategies depends on the other player's strategies, it is defined as $\mathcal{X}_i(x_i) \subseteq \mathbb{R}^{n_i}$. Given the other player's strategies set, the objective of player i is to solve the main problem 2-27, choosing an optimal strategy x_i to minimize it.*

$$\begin{aligned} \text{mín}_{x_i \in \mathcal{X}_i(\bar{x}_{-i})} \quad & V_i(x_i, x_{-i}) \\ s.t. \quad & x_i \in \mathcal{X}_i(x_{-i}). \end{aligned} \tag{2-27}$$

A collective strategy $\bar{X}$ is called GNE if $\forall i$, exist a x_i that solves the next minimization problem, i,e,:

$$V_v(\bar{x}_i, \bar{x}_{-i}) \leq V_v(y_i, \bar{x}_{-i}), \ \forall y_i \in \mathcal{X}_i(\bar{x}_{-i}), \ \forall i \in \mathcal{V}. \tag{2-28}$$

A generalized Nash Equilibrium (GNE) is a point $\bar{x}$ where each player can not decrease his objective function by changing unilaterally $\bar{x}_i$ unless the other player shifts their strategies. An example of a GNEP is the following.

Example 2.3.3 (Generalized Nash Equilibrium) *There is a game with presence of two players. Each player can control one variable. The optimization problem for each players is:*

$$\begin{array}{ll} \text{mín}_{x^1} \left(x^2 - \frac{1}{2}\right)^2 & \text{mín}_{x^2} \quad \left(x^2 - 1\right)^2 \\ s.t. \quad x^1 + x^2 \leq 1 & s.t. \quad x^1 + x^2 \leq 1 \end{array}. \tag{2-29}$$

The optimal solution of the above problem is:

$$\mathcal{S}_1(x^2) = \begin{cases} \frac{1}{2}, & if \quad x^2 \leq \frac{1}{2} \\ 1 - x^1, & if \quad x^2 \geq \frac{1}{2} \end{cases} \quad and \quad \mathcal{S}_2(x^1) = \begin{cases} 1, & if \quad x^1 \leq 0 \\ 1 - x^1, & if \quad x^1 \geq 0 \end{cases}. \tag{2-30}$$

Note that the problem has infinite solutions. It is easy to check that the solution of this problem is a generalized Nash Equilibrium given by $(\alpha, 1-\alpha), \quad \forall \alpha \in [\frac{1}{2}, 1]$. Finally, Generalized Potential game (GPG) as an instance of the GNEP class. In a GNEP, all players are minimizing the same function and where the feasible set is the product space R, [[17], [38]].

Definition 2.3.3 (Generalized Potential Game) *a GNEP is a generalized potential game if exist two main conditions;*

- *Exist a continuous function $P(x) : \mathbb{R}^n \to \mathbb{R}$ such that for all x_i, for all i and for all $y_i, z_i \in \mathcal{X}_i(x_{-i})$,*

  $$V_i(y_i, x_{-i}) - V_i(z_i, x_{-i}) > 0, \tag{2-31}$$

  implies

  $$P(y_i, x_{-i}) - P(z_i, x_{-i}) \geq \sigma(V_i(y_i, x_{-i}) - V_i(z_i, x_{-i})). \tag{2-32}$$

  let $\sigma : \mathbb{R}_+ \to \mathbb{R}_+$ be a forcing function $\text{lím}_{k\to\infty} \sigma(t_k) = 0 \Rightarrow \text{lím}_{k\to\infty} t_k = 0$.

- *Exist a closet and nonempty set $\mathcal{X} \subseteq \mathbb{R}^n$ such that, for all $i = 1, ..., N$, exist a function*

  $$\mathcal{X}_i(x_{-i}) \equiv \{x_i \in \mathcal{X}_i : (x_i, x_{-i})\}, \tag{2-33}$$

  where $\mathcal{X} \subseteq \mathbb{R}^{n_i}$ are closed and nonempty set such that $\prod_{i=1}^N \mathcal{X}_i \cap \mathcal{X} \neq \emptyset$.

3. Problem Statement

From the beginning of civilization, humans have been looking to transport from one place to another more efficiently than walking. First, they started with horses and floats, but horses had a brain and could make their own decisions; sometimes, those decisions were based on an emotional reaction rather than intelligent choices. Moreover, it could not continuously work for a long duty cycle, and its power was limited. Later in 1769, Nicolas-Joseph Cugnot invented the first steam-powered vehicle. This incredible machine could load an extended weight, and drivers could make their own driving decisions. Until now, this transportation system is still working, unless the advantage of efficiency, power consumption, pollution, and safety is not enough to be an optimal way to transport people and loads.

Research areas have made many efforts to achieve an optimal way of transportation to be accessible, optimal, and safe for everybody. Some ideas have been proposed, like hydrogen, diesel, and natural gas engines. Also, some nobel research about control safety systems like better and more powerful breaks, automated airbags, and good chassis material tries to save more lives, but it is not enough. Unfortunately, all past advantages are focused on the driver and vehicle users. Currently, no control system guarantees the security and integrity of the other road authors. The human decision in a driving context frequently represents a danger because it is based on emotion and natural reactions making no optimal driving. Even though these options could improve the current vehicles, they may find a better and cheapest solution to the main problem.

The research areas try to solve the safety driving problem with deep-learning, Recurrent Neural Networks, machine learning, and other theories. However, using these theories only guarantees stability and controllability in some scenarios. To get a closer solution to this problem, we study one way of achieving security and efficiency in a particular method as a highway area in this thesis. Currently, the experts on optimal control system areas rely on automated driving. This technology could solve all of the problems mentioned above.

Some safety decision criteria that we take into account were the following:

- Avoid obstacles that may be on the road.
- Avoid a collision with other cars on the road regardless of whether it is a cooperative or non-cooperative agent.

- Make decisions in the presence of aggressive maneuvers by selfish or non-cooperative agents on the road.

Therefore, designing and implementing different controllers with access to specific data sets was necessary. It depends on each controller's necessity. These decisions are based on logical and optimal decision-maker control to achieve the desired goal. These decisions are based on safety rules established for the users of any particular road.

3.1. Automated driving

The idea of automated driving is to safely and efficiently coordinate and synchronize several autonomous vehicles (AV) safely and efficiently.

Managing many AVs requires an extensive network with high computational loads and advanced control techniques. Usually, in the research literature is implemented one control algorithm is for an agent to achieve the objective. However, automated driving is an area that must consider different scenarios and situations. Besides, it is essential to prove all possibilities to warrant security and entire controllability. The following are the control theories used in this document.

- **Mixed-Integer Potential Game:** uses integer and continuous variables in a cost function to control the AV's position and velocity on a highway optimally.
- **Model Predictive Control:** uses information about the environment and local variables to get the optimal decision avoiding near obstacles.
- **Alternating Direction Method of Multipliers:** utilizes a linear model of the AV and inequality constraints as possible to get a faster and more efficient solution to the problem.

All the above controllers are explained in detail in chapter 4. In addition, the following section describes how AV and collision avoidance are modeled.

3.2. Vehicle Model

The automated driving is complex task, and one model is not enough to represent the behaviour of a vehicle around the environment. However, multiple controllers require multiple models that give the correct information. We use three different controllers that have a specific task. Each of them needs a particular model of vehicle. The high-level controller takes the network's information. For that reason, we implement a linear model. The middle-level controller uses just the knowledge of the environment acquired by its sensors. Due to this reason, we implemented a unicycle model. Besides, the low-level control gets information about the car and its decision-making behavior. Thus, we use a differential model in this control section.

3.2.1. Linear Model System

For controllers based on game theory, convexity is the main prerequisite for convergence. Therefore, the robot model and its constraints must be linear and convex. Due to this, the system is based on a Mixed-Logical-Dynamical system [5]. We consider a set of vehicles $\mathcal{I} := \{1, ..., N\}$ where N is the total number of vehicles interacting with the local agent in a multi-lane environment. The set $\mathcal{L} := \{1, ..., L\}$ represents the set of current lines on the highway, and the L variable is the maximum number of lines on the way. We assume that each vehicle i can control its longitudinal speed $v_i \in \mathcal{V}_i \subset \mathbb{R}$ and can change lane $z_i \in \mathcal{L} \subset \mathbb{N}$. Over the prediction horizon $\mathcal{T} := \{0, ..., T\}, T \geq 1$. We use two different decision variables to better represent an AV on a highway. Mixed logical variables can simplify, make linear, and convex the entire system.

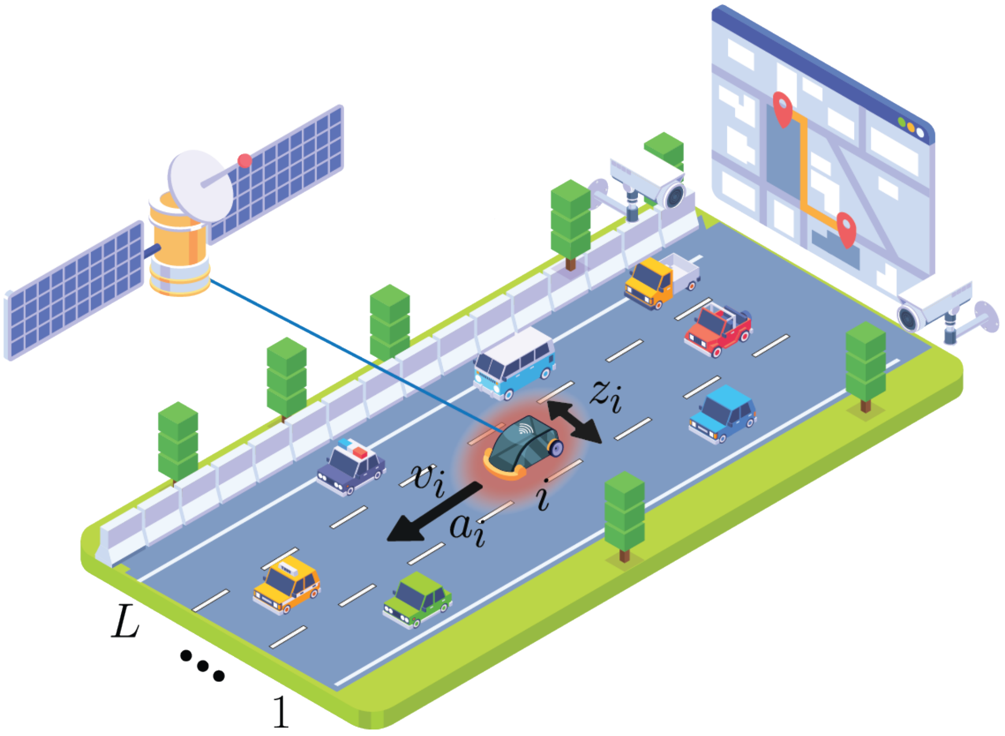


Fig. 3-1.: Linear model's variables.

Continuous Decision Variables

Each vehicle model has continuous variables that represent some physical features. Longitudinal acceleration $a_i \in A_i := \left[\underline{a_i}, \overline{a_i}\right] \subset \mathbb{R}$, the variables $\left\{\underline{a_i}, \overline{a_i}\right\}$ represent the minimum and maximum acceleration allowed, with $\underline{a_i} < 0 < \overline{a_i}$, assuming that the $\underline{a_i}$ is negative and is due the break job of the i vehicle, as illustrated in **Fig.** . The longitudinal speed $v_i \in \mathcal{V}_i \subset \mathbb{R}$ is determined by a standard Forward-Euler Scheme, i.e.,

$$v_i(t+1) = v_i(t) + \tau a_i(t), \tag{3-1}$$

where $\tau > 0$ denotes the time interval of each simulation step. Hence, the acceleration and velocity longitudinal over the horizon $\mathcal{T}$ are:

$$a_i := [a_i(0); ...; a_i(t-1)] \in \mathcal{A}_i^T,$$
$$v_i := [v_i(0); ...; v_i(t-1)] \in \mathcal{V}_i^T$$

Discrete Decision Variables

The lanes in a highway are predefined internationally as a space where vehicles can and should travel. Thanks to this design, the agent will drive inside a lane unless the vehicle needs to change to another one. That means the set of vehicles $\mathcal{V}$ needs to be modelled as an integer variable that represents the actual and future travelling lane $z_i \in \mathcal{L}$. The discrete decision variables over the horizon T are:

$$z_i := [z_i(0); ...;\ z_i(t-1)] \in \mathcal{Z}_i^T.$$

Coupling Variables

The control system requires different sense variables that allow knowing the actual and future states of the agent i with its neighbours $\mathcal{I}_{-i}$. Therefore, we denote by $d_{i,j} \in \mathbb{R}$ as the longitudinal distance between the connected vehicles i and j at the time $t \in \mathcal{L}$ as shown **Fig. 3-2**.

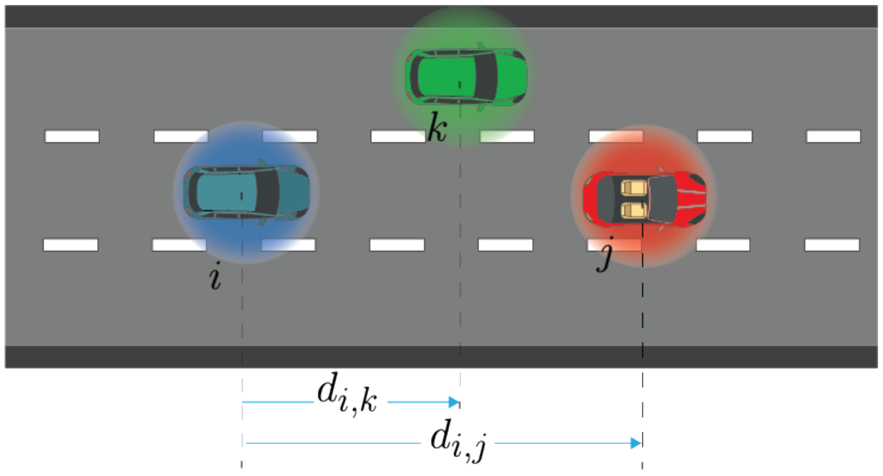


Fig. 3-2.: Relative distance between agents.

Over time, the equation of $d_{i,j}$ evolves as a Forward-Euler function. In 3.2.1 the longitudinal distance is over the axis x in the working environment. The difference in velocities is over the current time t, and the τ is the controller's step time at each iteration.

$$d_{i,j}(t+1) = d_{i,j}(t) + \tau(v_j(t) - v_i(t)). \tag{3-2}$$

It allows us to introduce the set of vehicles in the neighbourhood of $\mathcal{N}_i := \left\{j \in \mathcal{I} | \, |d_{i,j}| \leq \overline{d}, t \in \mathcal{T}\right\}$; this data can be measured locally or globally, i.e., on the on-board sensors or with communication of a driving network. From now we refer to the variable j as a generic vehicle in the set $\mathcal{N}_i$. According to , each agent knowing the velocity of its neighbour $v_j(t)$, can estimate the relative distance between each other. In addition, we implement a relative lane distance $z_{i,j}$. It represents the lateral distance of each vehicle from its neighbours. The difference of lane between agent i with its neighbors $\mathcal{N}_i$ and is calculated by the following equation:

$$z_{i,j}(t) = z_j(t) - z_i(t). \tag{3-3}$$

Lateral distance is the difference between integer values. It means the space working is $\mathbf{N} \times \mathbf{N} \to \mathbf{N}$. The distance is calculated over the y axis.

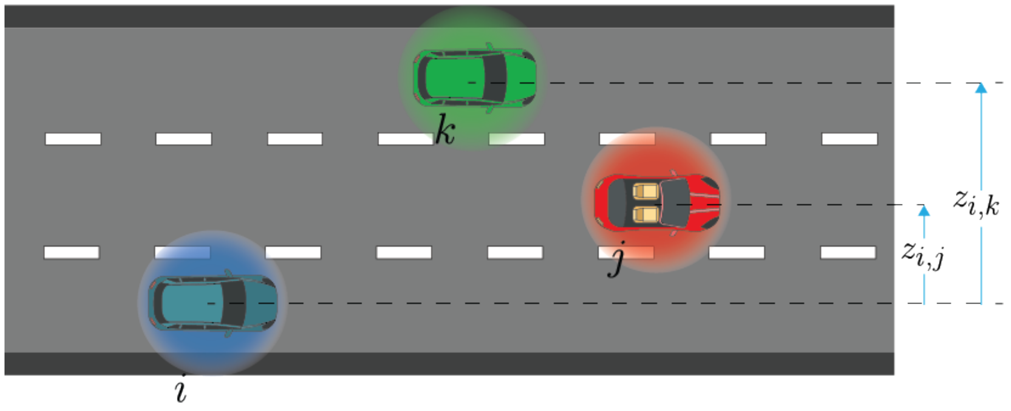


Fig. 3-3.: Lateral Distance.

We assume that each agent makes decisions for their selfish interest, e.g., drive through desired speed profile $v_i^d \in \mathcal{V}_i^T$ or get into a lane $z_i^d \in \mathcal{L}_i^T$, **Fig. 3-3**. Each agent's control system considers its neighbors' current and future states. Still, it does not provide information about the targets of the other agents. As a result of this behavior, each agent takes a decision seeking its individual goals through a sequence of hybrid decisions. With the previous model, we formulate as a first step an MPC motion planning with mixed-integer variables:

$$\begin{cases} \min\limits_{v_i,a_i,z_i} J_i(v_i, a_i, z_i) \\ \text{s.t.} \quad v_i(t+1) = v_i(t) + \tau\ a_i(t), \quad \forall t \in \mathcal{T} \\ \qquad a_i(t) \in \mathcal{A}_i, \\ \qquad v_i(t+1) \in \mathcal{V}_i, \\ \qquad z_i(t+1) \in \mathcal{L}_i, \end{cases} \tag{3-4}$$

The J_i cost function is a linear and convex objective function for each vehicle i where $J_i : \mathcal{V}_i^T \times \mathcal{A}_i^T \times \mathcal{L}_i^T \longrightarrow \mathbb{R}$. The sets $\mathcal{V}_i$ and $\mathcal{L}_i \subset \mathcal{L}$ shall be defined to restrict them to possible values in the real environment. Given maximum and minimum acceleration $\left[\underline{a_i}, \overline{a_i}\right]$ as also maximum and minimum velocity $\left[\underline{v_i}, \overline{v_i}\right]$, we can limit the sets as:

$$\mathcal{V}_i(t) := \left[0, \overline{v_i(t)}\right] \cap \left[v_i(t) + \tau\underline{a_i}, \quad v_i(t) + \tau\overline{a_i}\right] \tag{3-5}$$

$$\mathcal{L}_i(t) := \mathcal{L} \cap \left[z_i(t) - 1, \quad z_i(t) + 1\right] \tag{3-6}$$

From 3-5, it is possible to limit the speed to legally possible values, and from 3-6 limits the change of lane to just one per step. This restriction makes agents' movement smoother through lanes and allows lateral safety rules. The following section explains the rules to make safer automated driving with selfish agents.

Collision Avoidance Rules

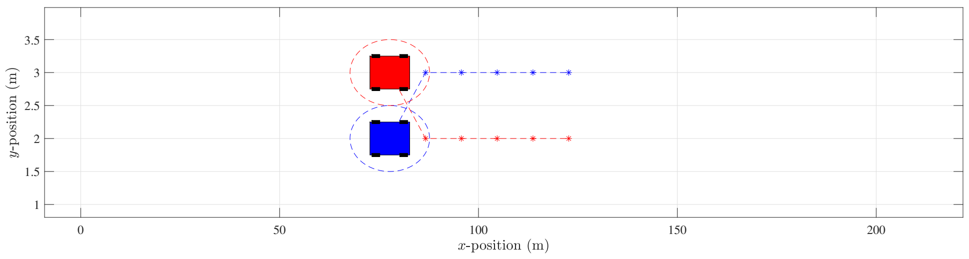
Human driving is usually unsafe; exist some rules established in each country, like speed limits, preferential lanes, and safety distance. In this section, we postulate some safety rules to avoid the collisions of the interconnected agents with the presence of non-cooperative agents.

Longitudinal rules

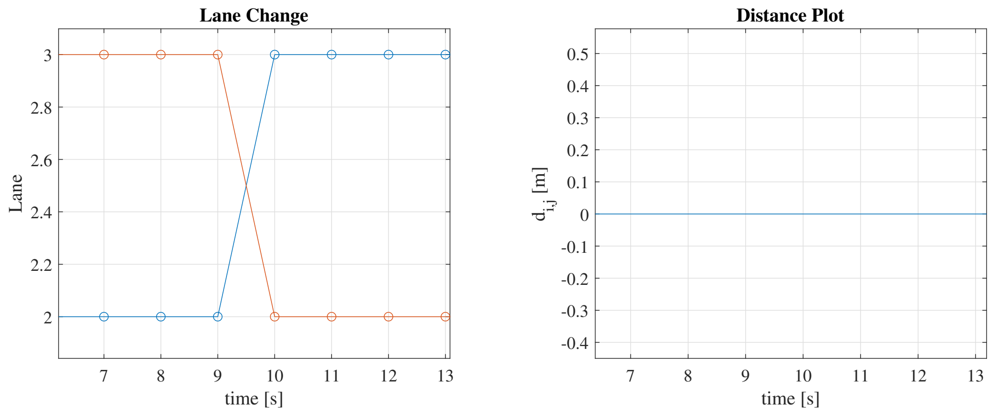
The main reason for a collision of two or more vehicles is that one of the vehicles violates the safety distance. Due to this is implemented the following safety rule. If the lateral distance between two agents is equal to zero $z_j(t) = z_i(t)$ and the longitudinal distance is greater or less than a safety distance $|d_{i,j}(t)| > D_s$, and the agents will continue in the same lane $z_i(t+1) = z_i(t), z_j(t+1) = z_j(t)$. The control system must ensure that the behind vehicle maintains a safety distance $|d_{i,j}(t)| > D_s$ how is shown in Algorithm 1, where D_s is a predefined secure distance.

Algorithm 1 Algorithm of longitudinal distance.

```
if $z_{i,j}(t) = 0$ and $z_{i,j}(t+1) = 0$ then
    if $d_{i,j}(t) > 0$ then
        $d_j(v_j(t)) - d_i(v_i(t)) > D_s$
    else if $d_{i,j}(t) < 0$ then
        $d_j(v_j(t)) - d_i(v_i(t)) < -D_s$
    else
        Continue normal driving
    end if
end if
```



(a) Position of vehicles during the example.



(b) Profiles of velocity and lanes over highway.

Fig. 3-4.: Example of lateral collision.

Lateral rules

There are other common collisions in a regular lane; those are lateral collisions, usually generated by occlusion problems. This model does not have this issue thanks to interconnected and full neighbor position knowledge. However, the controller solver could be in a position where each vehicle decide to change at the same time the lane as **Fig. 3-4** shows. Owing to this, we implement the lateral rule for each vehicle i, j driving over the horizon $\mathcal{T}$, if the longitudinal distance $d_{i,j}$ is less than a safe distance and the difference of lane position $z_{i,j}$ is one $|z_{i,j}(t)| = 1$, and the pair of agents will change the lane $z_i(t+1) \neq z_i(t)$, $z_j(t+1) \neq z_j(t)$, the controller must constraint the change of lane as is explained in the next algorithm.

Algorithm 2 Algorithm of lateral distance.

```
if $|z_{i,j}(t)| = 1$ and $(z_i(t+1) \neq z_i(t)$ or $z_j(t+1) \neq z_j(t))$ then
    if $d_{i,j}(t) > 0$ then
        $d_j(v_j(t)) - d_i(v_i(t)) > D_s$
    else if $d_{i,j}(t) < 0$ then
        $d_j(v_j(t)) - d_i(v_i(t)) < -D_s$
    else
        Continue normal driving
    end if
end if
```

Finally, the previous safety rules are executed in a mixed-integer decision-making framework explained in-depth way in Chapter 3.2.2. Nevertheless, this thesis work does not focus on the issue of communication among vehicles. Consequently, we assume that:

- Vehicles can share data of their position, speed, current lane and future states planned.
- Each vehicle is autonomous, therefore it has the capability to change its position and velocity without the presence or intervention of a human being

3.2.2. Non-Linear Model System

Currently, there are several kinds of wheel robots used for mobile applications. Some of them are unicycle, Ackerman, differential, and omnidirectional, among others. In [7] was compared the two main ways to represent a dynamic system and the kinetic and kinematic models were compared in the vehicle collision avoidance framework. Although the kinetic model has more accuracy than the kinematic, the entire system implemented in an MPC is faster with the kinematic model than with the kinetic model. Due to better prediction accuracy and computational time in the established prediction horizon, we decided to use the robot's kinematic model.

The nonlinear kinematic unicycle model is one of the most popular models for automated driving in multiagent systems [8] and mobile robotics. Therefore, we designed, fine-tuned, manufactured and implemented a linear unicycle model for the medium level of control. It makes the following assumptions:

- Into the prediction horizon of the MPC, the velocity V could change.
- Forces are applied only in the lateral wheels, neither in the caster wheels.
- The robot is not equipped with a steering wheel.
- Aerodynamic drag and rolling resistance are ignored.
- The position of the front and rear caster wheel does not matter.
- The Center of Gravity (CG) position is assumed in the middle of the lateral wheels; the height is assumed to be zero since the vehicle's motion is planar.
- Pitch and roll dynamics are neglected.
- The electrical energy and forces applied to the traction wheels are neglected.

In order to specify the actual position of the robot within two-dimensional space, we use a relationship between the robot's global reference and the robot's local reference, as shown in **Fig. 3-5**. The Axis X and Y define the position related to an origin $O : \{X, Y\}$. The robot position is specified by three coordinates, x, y, and θ, where θ is the angular position of the robot referred to as the global reference. It is also possible to describe the robot position as a vector of dimension three as follows.

$$P_i = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3-7}$$

Kinematic model of a differential robot

Kinematics is one of the mechanic's branches that describes a particle's behavior in an environment by the action of multiple forces regardless of its mass. Kinematic equations are used to express the motion of a point after the act of a force. The inputs interacting with the robot are the linear velocity V[cm/s] and the angular velocity w[rad/s]. The rate of change of position of the vehicle in the x-direction and the y-direction are x and y, respectively and are given by:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{3-8}$$

Let x_i, y_i be the lateral and longitudinal position of agent i in the working space, and θ_i be the agent orientation. The v_i is the linear velocity, and w_i is the angular velocity. The control inputs are v and w and the output variables are the global position of the robot x_i, y_i and θ_i. **Fig. 3-5** shows the representation of the mentioned variables.

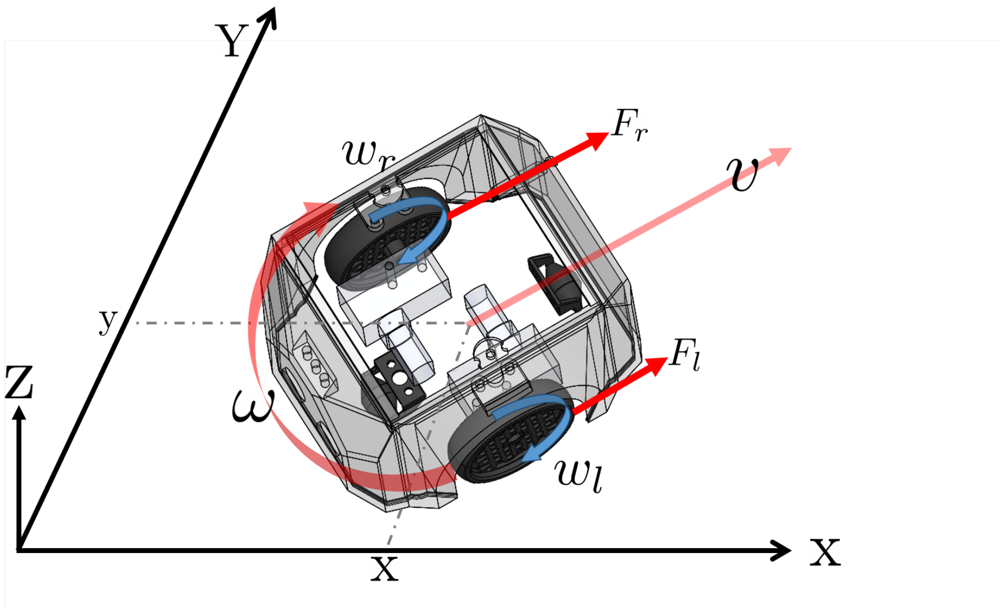


Fig. 3-5.: Non-linear model of a differential robot.

3.2.3. Robot Model

A unicycle model is unfeasible in the real world due to geometric constraints. However, it is good to model and control a particle object in an environment. Due to this, the robot uses a differential model in the low-level control architecture. This model allows calculating the control action in each wheel with only the information of linear and angular velocity. Equation (3-9) is the low-level model used to control each wheel's speed. Where w_r and w_l are the angular velocity of the right and left wheels, L is the distance in cm between the two wheels, and R is the radius of both wheels.

$$\begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} = \frac{1}{2R} \begin{bmatrix} 2 & -L \\ 2 & L \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{3-9}$$

In pursuit of faster computation time and easy processing, we use the above model for the internal processing of the robot. The differential model is one of the most popular models used to simulate autonomous vehicles used in robotics. It was selected for its simplicity and the best results obtained. The actual robot uses this model to predict the position of the robot, the microprocessor calculates the speed required for each wheel to take the robot to a specific position.

3.3. Platform Design

A test system was designed, built, and implemented to test the control algorithms used in this document in a real environment. The Section 3.2.3 explains the model that the test robot uses in its control algorithm to find an optimal solution. This section will explain how the complete system is built and how it works. The [37] document explains this test system in deep. All the test simulations shown in Chapter 6 are implemented in this testbed and are evidenced in the Annex 7.2.

3.3.1. Robot Processing System

The robot processing system was equipped with an Auriga board. This developing board contains an Atmega 2560 microcontroller running at 16 MHZ speed clock. In addition, the developing board has gyroscope, thermistor, and a sound sensor for perception applications. Besides, it has a Bluetooth shield that allows microprocessor communication and programming, but it can be replaced for RF or WiFi modules. Moreover, it has 9 I2C ports, 1 UART port, and one port for “smart motor” applications. The mainboard controls the encoder motors with a TB6612 chip. With the integration of this chip, the microprocessor can control with high precision the speed, direction and position of each wheel with an internal control closed loop.

3.3.2. Communication System

The communication system considers the number of nodes and the bandwidth of the shared information. As a result, the ESP 8266 module was chosen due to its easy programming, powerful processing, economical and affordable. This communication system has a novel protocol to share information with the other agents without needing an external router. It can connect, transmit, and receive up to a 250-byte payload by WiFi network. This new protocol supports unencrypted peers. However, their total number should be less than 20, including encrypted peers. This protocol allows sharing information without requiring a WiFi station or router as a principal interconnection module. Besides, with this protocol, if one of the boards suddenly loses power or resets, it will automatically connect to its peer to continue the communication when it restarts.

3.3.3. Global Position System

The robot swarm are equipped with a fiducial marker system for camera pose estimation [20]. This system uses a web camera and system image processing to detect and estimate the position of each robot. It is currently implemented using the OpenCV library in python language. The system can calculate x, y, z, and θ position of each agent. The platform uses as a visual sensor the Logitech C930 at a resolution of 1920x1080 pixels and a 90-degree field-of-view. It can achieve frame rates of up to 30 fps. This library can detect up to 1000 Aruco tags in the same frame.

3.3.4. Power System

Each robot has a lithium-ion battery of 3000 mAh. It provides energy for about ten working hours. Moreover, the charging system recharges the battery in about 2 hours. This battery can provide up to 3 amps in a barrel and USB ports. The previous advantages make the power system robust and valuable to implement with different development boards.

3.3.5. The Interconnect System

The testbed was built by gathering multiple communication, tracking, processing, and sensing systems. All of them can be replaced in the future with better technologies or another system with better results.

The process of operation of the testbed is the following. The tracking system uses the OpenCV library and Aruco tags. The camera takes multiple frames to process the information, estimating the positions x, y, and orientation θ. It gives a vector $p \in \mathbb{R}^{2,nr}$, where n_r is the number of robots in each experiment.

With this information, the central processing unit computes the value of the robot's control variables using the selected technique. Then it broadcasts this information to each mobile robot via WiFi protocol. If the test needs feedback information, it will wait for it. Eventually, each robot gets data from the server, processes it, and makes a decision.
The server can record, graph, and save all the information taken in each test. Additionally, it could record videos and process the previous information to analyze the data accurately. **Fig. A-1** shows the connection of the entire testbed and how each section interacts.

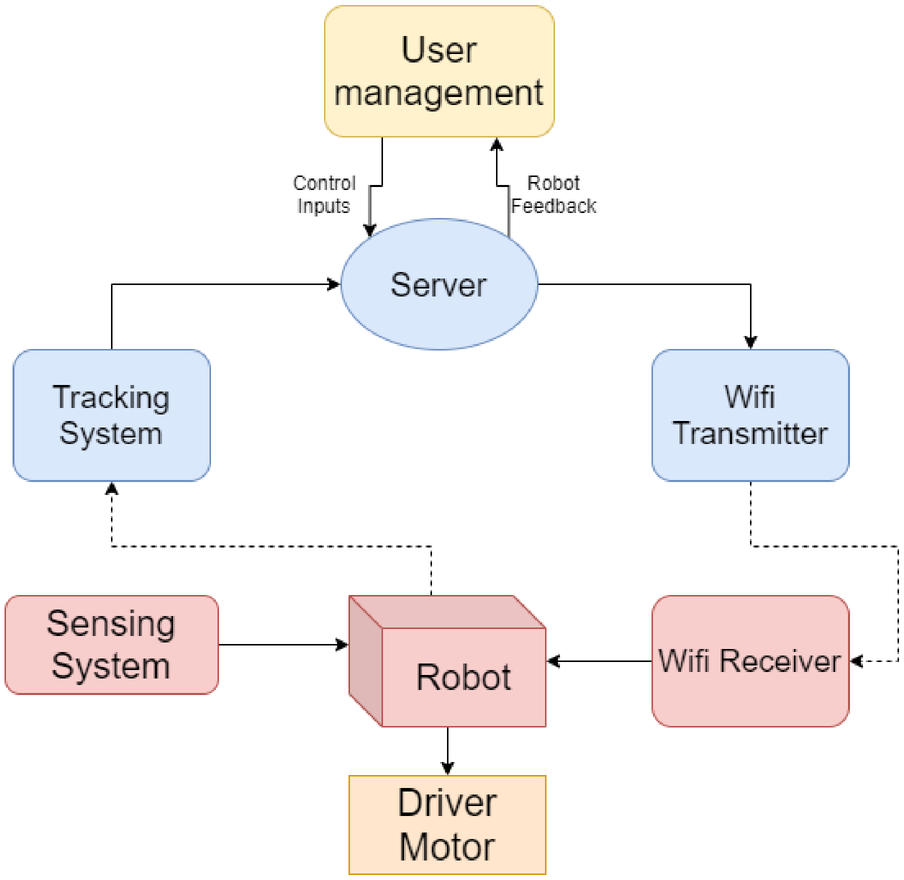


Fig. 3-6.: System Architecture.

4. Controller Architecture

In this chapter, we consider a hierarchical control architecture. First, each vehicle has an embedded High-Level Controller (HLC). It has access to the internet and different kind of services. Here, the goal is to manage the position of the vehicle on the highway, its velocity, and the lane position it should use. Then using the previous information, a Mid-Level Controller (MLC) takes as a reference and moves the vehicle avoiding obstacles that could appear in automated driving. Finally, a Low-Level controller (LLC) is implemented to generate the fair inputs of the vehicle's actuators. It will calculate the power units required to move the vehicle to the desired position. This control layer is simple, and we will not focus on it. **Fig. 4-1** shows the complete architecture.

4.0.1. High-Level Controller

A MPC controller with a Generalized Mixed-Integer Potential Game (GMIPG) framework was designed and implemented for high-level control to guarantee that the vehicles follow the basic traffic rules. MPC is helpful because it could prevent future issues and pre-correcting signal inputs from achieving the main objective. The GMIPG is necessary to control the vehicle in specific parameters. Each vehicle has to be in an integer number of the lane (1,2,3,..., N) and use binary variables to activate some logical constraints; therefore, we implement the Mixed Integer variables. Furthermore, finally, it is a multi-agent game where each agent aims for a selfish objective. We use a generalized potential game structure to achieve a global and fair solution.
Each vehicle has a solid and large number of variables of control and sense. The HLC uses a set of network variables such as velocity, acceleration, lane position, and future states. Some of them are part of the information shared by the other agents, and the users give the other one. However, we assume that:

- Each road user aims to pursue their selfish objective.
- Each control system has a partial connection to the entire network, and can access to the entire network if it needs.
- The communication does not have lags or problems in sharing the desired information.
- The selfish objective of each vehicle could be a result of a personal decision o a result of an intelligent algorithm., e.g., maps, Waze, or some traffic and routing algorithm.

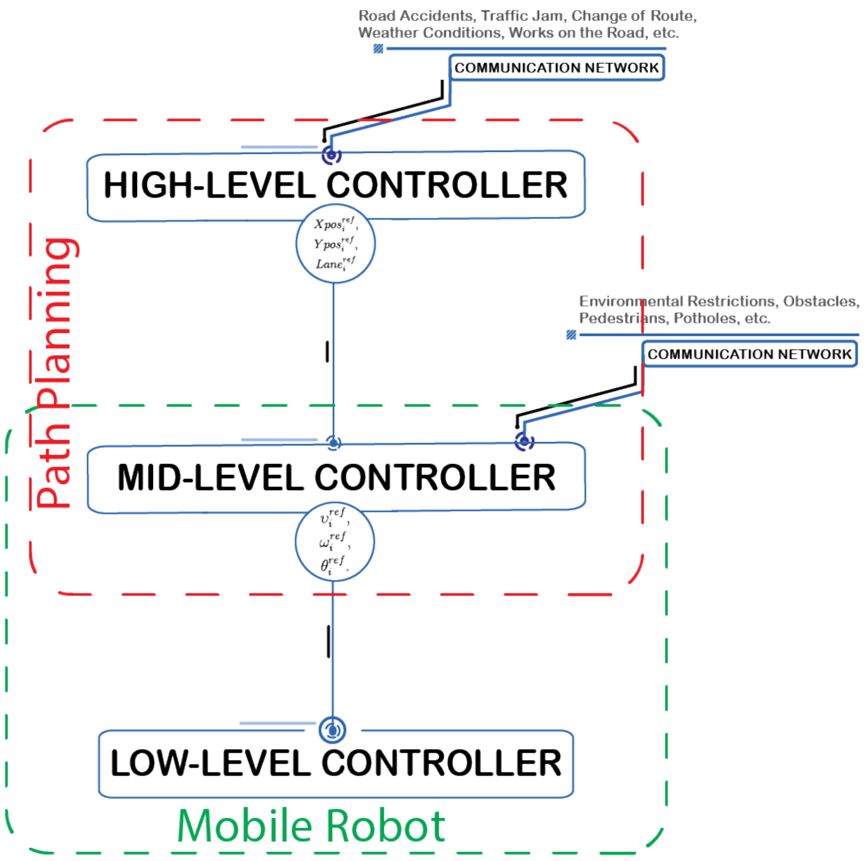

Road Accidents, Traffic Jam, Change of Route,
Weather Conditions, Works on the Road, etc.
COMMUNICATION NETWORK
HIGH-LEVEL CONTROLLER
$Xpos_i^{ref}$, $Ypos_i^{ref}$, $Lane_i^{ref}$
Environmental Restrictions, Obstacles,
Pedestrians, Potholes, etc.
COMMUNICATION NETWORK
Path Planning
MID-LEVEL CONTROLLER
υ_i^{ref}, ω_i^{ref}, θ_i^{ref}.
LOW-LEVEL CONTROLLER
Mobile Robot

Fig. 4-1.: Control architecture.

- Crossroads, traffic lights, or intersections are not part of the experimental environment.

This thesis document aims to manage, control and guarantee safe automated driving in the most common situation a vehicle can face on a highway against agents who make selfish decisions. Specially, we focus on the mixed-integer decision-maker layer for motion planning of the vehicles.

4.0.2. Mixed-Integer Linear Constraints

Chapter 3.2.1 explains the autonomous driving rules to make driving safe without affecting the objective of each agent. In this section, we will convert the above rules into linear logical constraints. Due to the previous, the control algorithm can constrain the solution space to one that is safe for the environment. Two different types of constraints will be discussed: lateral and longitudinal.

Longitudinal Constraints

Let us consider the safety rules in Section 3.2.1 where it should be fulfilled that if a pair of vehicles circulated in the same lane and with a distance greater than zero, a safety distance should be observed:

$$[z_{i,j}(t) = 0] \wedge [|d_{ij}(t)| \geq 0] \Rightarrow [|d_{ij}(t)| \geq D_i^s] \tag{4-2}$$

We introduce two logical and binary variables, $\alpha, \beta \in \mathbb{B} := \{0, 1\}$. The variable α discriminates between vehicles travelling in the same lane at the same time ($l_{i,j} = 0$), and β discriminates between vehicles ahead ($\beta = 1$) or behind ($\beta = 0$), no matter what lane they are on

$$[\alpha_{i,j}(t) = 1] \Leftrightarrow [z_{i,j}(t) \leq 0] \wedge [z_{i,j}(t) \geq 0] \tag{4-3}$$

$$[\beta_{i,j}(t) = 1] \Leftrightarrow [d_{i,j}(t) \geq 0] \tag{4-4}$$

Therefore, the equations 4-3 and 4-4 can be written as nonlinear inequalities:

$$\alpha_{i,j}(t)[\beta_{i,j}(t)(d_i^s(t) - d_{i,j}(t)) + (1 - \beta_{i,j}(t))(d_i^s(t) + d_{i,j}(t))] \leq 0, \tag{4-5}$$

Relying on the pattern of inequalities summarized in **Table 4-1**, we will use it to handle the logical implications and nonlinear constraints. Therefore, let us consider the right-hand side of the Equation 4-3. Introducing η and $\theta \in \mathbb{B}$, where if $z_{i,j} \leq 0$ implies $\eta_{i,j} = 1$:

$[\eta_{i,j}(t) = 1] \Leftrightarrow [z_{i,j}(t) \leq 0]$

and it is translated into $\mathcal{S}_{\leq}(\eta_{i,j}(t), z_{i,j}(t), 0)$. Same way if $\theta_{i,j} = 1$ implies $z_{i,j} \geq 0$:

$[\theta_{i,j}(t) = 1] \Leftrightarrow [z_{i,j}(t) \geq 0]$

and is translates into $\mathcal{S}_{\geq}(\theta_{i,j}(t), z_{i,j}(t), 0)$. Finally 4-3 is represented as follows:

$$(\mathbf{4\text{-}3}) \Rightarrow \begin{cases} \mathcal{S}_{\leq}(\eta_{i,j}(t), z_{i,j}(t), 0), \\ \mathcal{S}_{\geq}(\theta_{i,j}(t), z_{i,j}(t), 0), \\ \mathcal{S}_{\wedge}(\alpha_{i,j}(t), \eta_{i,j}(t), \theta_{i,j}(t)). \end{cases} \tag{4-6}$$

The same procedure is used to transform 4-4 in a mixed-integer linear constraint:

$$(\mathbf{4\text{-}4}) \Rightarrow \mathcal{S}_{\geq}(\beta_{i,j}(t), d_{i,j}(t), 0). \tag{4-7}$$

Now let me factor 4-5 into a reduced non-linear equation.

$$-2 \underbrace{\alpha_{i,j}(t)\beta_{i,j}(t)}_{\xi_{i,j}(t)} \ d_{i,j}(t) + \alpha_{i,j}(t) \ d_i^s(t) + \alpha \ d_{i,j}(t) \leq 0. \tag{4-8}$$

As we can see, the equation is nonlinear and is composed of binary and integer variables. In 4-8 we transform $\alpha_{i,j}(t)\beta_{i,j}(t)$ into $\xi_{i,j}(t)$ to linearize the equation.

$\mathcal{S}_{\wedge}(\xi_{i,j}(t), \alpha_{i,j}(t), \beta_{i,j}(t))$.

The next and last step to linearize the equation is to transform the multiplication of a binary variable and integer variable into real auxiliary variables. We define $f_{i,j}(t) := \xi_{i,j}(t)d_{i,j}(t)$, $g_{i,j}(t) := \alpha_{i,j}(t)d_i^s(t)$, and $h_{i,j}(t) := \alpha_{i,j}(t), d_{i,j}(t)$ that shall satisfy $\mathcal{S}_{\Rightarrow}$ in the **Table 4-1** as follows:

$$\mathcal{S}_{\Rightarrow}(f_{i,j}(t), d_{i,j}(t), \xi_{i,j}(t)). \tag{4-9}$$
$$\mathcal{S}_{\Rightarrow}(g_{i,j}(t), d_i^s(t), \alpha_{i,j}(t)). \tag{4-10}$$
$$\mathcal{S}_{\Rightarrow}(h_{i,j}(t), d_{i,j}(t), \alpha_{i,j}(t)). \tag{4-11}$$

Finally, 4-5 is expressed as a linear mixed integer constraint equation:

$$-2f_{i,j}(t) + g_{i,j}(t) + h_{i,j}(t) \leq 0. \tag{4-12}$$

Lateral Constraints

Let us consider the scenario in **Fig. 3-4** where two vehicles are side by side. The second Mixed-Integer coupling constraint we introduce is about avoiding collision between vehicles driving next to each other on a highway. It was explained in the **Algorithm** 2 in Section 3.2.1 how each agent's restrictions must avoid a lateral collision. Therefore, we implement the same transformation as in the previous section. The main idea is to linearize a logical constraint equation into a set of Mixed-Integer Linear Constraints that can avoid lateral vehicles regardless of the situation. **Algorithm** 2 can be transformed first as a logical implication form

$$[|z_{i,j}(t)| = 1] \wedge [|d_{ij}(t)| \leq \hat{d}] \wedge \left\{[l_i^r(t) = 1] \vee [l_i^l(t) = 1]\right\} \Rightarrow [z_{i,j}(t+1) - z_{i,j}(t) = 0]. \quad (4\text{-}13)$$

It represents the behaviour that each agent should have in a collision risk situation. However, the logical constraints model cannot be solved by MPC due to non-linearities and non-convexity. Therefore, we transform the previous equation into multiple linear inequalities that will allow the controller to find an optimal answer. As in the previous transformations, let us consider three auxiliary variables γ^l, γ^r and $\zeta \in \mathbb{B}$:

$$\begin{aligned}
[\gamma_{i,j}^l(t) = 1] &\Leftrightarrow [z_{i,j}(t) \leq 1] \wedge [z_{i,j}(t) \geq 1] \\
[\gamma_{i,j}^r(t) = 1] &\Leftrightarrow [z_{i,j}(t) \leq -1] \wedge [z_{i,j}(t) \geq -1] \\
[\zeta_{i,j}(t) = 1] &\Leftrightarrow [d_{i,j}(t) \leq \hat{d}] \wedge [d_{i,j}(t) \geq -\hat{d}]
\end{aligned}$$

then, 4-13 can be reformulated as:

$$\zeta_{i,j}(t)[l_i^l(t)\gamma_{i,j}^l(t) + l_i^r(t)\gamma_{i,j}^r(t)](z_{i,j}(t+1) - z_{i,j}(t)) = 0. \quad (4\text{-}14)$$

Hence, it is rewritten in reduced form :

$$\begin{aligned}
&\zeta_{i,j}(t)\ l_i^l(t)\ \gamma_{i,j}^l(t)\ z_{i,j}(t+1) + \zeta_{i,j}(t)\ l_i^r(t)\ \gamma_{i,j}^r(t)\ z_{i,j}(t+1) \\
&\qquad - \zeta_{i,j}(t)\ l_i^l(t)\ \gamma_{i,j}^l(t)\ z_{i,j}(t) - \zeta_{i,j}(t)\ l_i^r(t)\ \gamma_{i,j}^r(t)\ z_{i,j}(t) = 0 \quad (4\text{-}15)
\end{aligned}$$

Next, we add four auxiliar variables to remove nonlinearities $\varpi_{i,j}, \lambda_{i,j}, \rho_{i,j}$ and $\varrho_{i,j} \in \mathbb{B}$. In 4-15 are reformulated into a linear form using both real and binary auxiliary variables following the linearization steps above. We define $\varpi_{i,j} := \zeta_{i,j}(t)\ l_i^l(t)$, $\lambda_{i,j} := \zeta_{i,j}(t)\ l_i^r(t)$, $\rho_{i,j} := \zeta_{i,j}(t)\ l_i^l(t)$, and $\varrho_{i,j} := \zeta_{i,j}(t)\ l_i^r(t)$ as binary variables which satisfy the next system of inequalities

$$\mathcal{S}_{\wedge}(\varpi_{i,j}(t), \zeta_{i,j}(t), l^l_{i,j}(t)), \tag{4-16}$$
$$\mathcal{S}_{\wedge}(\lambda_{i,j}(t), \zeta_{i,j}(t), l^r_{i,j}(t)), \tag{4-17}$$
$$\mathcal{S}_{\wedge}(\rho_{i,j}(t), \varpi_{i,j}(t), \gamma^l_{i,j}(t)), \tag{4-18}$$
$$\mathcal{S}_{\wedge}(\varrho_{i,j}(t), \lambda_{i,j}(t), \gamma^r_{i,j}(t)). \tag{4-19}$$

We also define real variables $\psi_{i,j} := z(t+1), \phi_{i,j} := z(t+1), \varphi_{i,j} := z(t)$,and $\iota_{i,j} := z(t)$ that must satisfy $\mathcal{S}_{\Rightarrow}$ in Table **4-1** as follows:

$$\mathcal{S}_{\Rightarrow}(\psi_{i,j}(t), z_{i,j}(t+1), \rho_{i,j}(t)). \tag{4-20}$$
$$\mathcal{S}_{\Rightarrow}(\phi_{i,j}(t), z_{i,j}(t+1), \varrho_{i,j}(t)). \tag{4-21}$$
$$\mathcal{S}_{\Rightarrow}(\varphi_{i,j}(t), z_{i,j}(t), \rho_{i,j}(t)). \tag{4-22}$$
$$\mathcal{S}_{\Rightarrow}(\iota_{i,j}(t), z_{i,j}(t), \varrho_{i,j}(t)). \tag{4-23}$$

Finally, 4-13 was reduced to a system of equations where the following pair of inequalities must be fulfilled.

$$\begin{cases} \psi_{i,j}(t) - \varphi_{i,j}(t) + \phi_{i,j}(t) - \iota_{i,j}(t) \leq 0 \\ \psi_{i,j}(t) - \varphi_{i,j}(t) + \phi_{i,j}(t) - \iota_{i,j}(t) \geq 0 \end{cases} \tag{4-24}$$

All the previous Mixed-Integer linear inequalities are organized to obtain the final reference frame for each agent.

$$\begin{cases} \underset{v,a,z,l^l_i,l^r_i}{\text{mín}} & J_i\left(v_i, a_i, z_i\right) \\ \text{s.t.} & v_i(t+1) = v_i(t) + \tau a_i(t), \qquad \forall t \in \mathcal{T} \\ & z_i(t+1) = z_i(t) + l^l_i(t) - l^r_i(t), \\ & a_i(t) \in \mathcal{A}, \\ & v_i(t) \in \mathcal{V}_i, \\ & l^l_i(t), l^r_i(t) \in \mathbb{B}, \\ & l^l_i(t) + l^r_i(t) \leq 1, \\ & \text{(4-6)} - \text{(4-24)} \qquad \forall j \in \mathcal{N}_i, \forall t \in \mathcal{T} \end{cases} \tag{4-25}$$

Each agent has $c_i := (75\,\mathcal{N}_i{+}7)$ constraints, whereas the entire network has $c := (\sum_{j\in\mathcal{N}_i} c_j) + c_i$ constraints. Note that the strategies of the neighbours are contained in the coupling constraints in (4-6)–(4-24) as affine, provided terms. Let $x_{-i} \in \mathbb{R}^{n-n_i}$ stacks the variables of the neighbours, we define $x_i := [v_i; a_i; ...; s_i] \in R^{n_i}$, $n_i := (28|N_i| + 5)T$ as the i^{th} vector of decision variables and the vector that represent all the decision variables in the neighbourhood is $n := (\sum_{j\in\mathcal{N}_i} n_j) + n_i$. Finally, the portable hybrid motion planner is

$$\forall i \in \mathcal{I} : \begin{cases} \text{mín}_{x_i} & J_i(x_i) \\ s.t. & Ax + b, \end{cases} \tag{4-26}$$

where $A \in \mathbb{R}^{c \times n}, b \in \mathbb{R}^c$.

Name	Logical Implication	System Inequalities
$\mathcal{S}_{\geq}(\eta, f(x), c)$	$[\eta = 1] \Leftrightarrow [f(x) \geq c]$	$\begin{cases} (c-m)\eta \leq f(x) - m \\ (M - c + \epsilon)\eta \geq f(x) - c + \epsilon \end{cases}$
$\mathcal{S}_{\leq}(\eta, f(x), c)$	$[\eta = 1] \Leftrightarrow [f(x) \leq c]$	$\begin{cases} (M-c)\eta \leq M - f(x) \\ (c + \epsilon - m)\eta \geq \epsilon + c - f(x) \end{cases}$
$\mathcal{S}_{\wedge}(\eta, \alpha, \gamma)$	$[\eta = 1] \Leftrightarrow [\alpha = 1] \wedge [\gamma = 1]$	$\begin{cases} -\alpha + \eta \leq 0 \\ -\gamma + \eta \leq 0 \\ \alpha + \gamma - \eta \leq 1 \end{cases}$
$\mathcal{S}_{\vee}(\eta, \alpha, \gamma)$	$[\eta = 1] \Leftrightarrow [\alpha = 1] \vee [\gamma = 1]$	$\begin{cases} \alpha - \eta \leq 0 \\ \gamma - \eta \leq 0 \\ -\alpha - \gamma + \eta \leq 0 \end{cases}$
$\mathcal{S}_{\Rightarrow}(g, f(x), \eta)$	$[\eta = 0] \Rightarrow [g = 0], [\eta = 1] \Rightarrow [g = f(x)]$	$\begin{cases} m\eta \leq g \leq M\eta \\ -M(1-\eta) \leq g - f(x) \leq -m(1-\eta) \end{cases}$

Table 4-1.: Basic Logical Implications and associated system of inequalities. ($f : \mathbb{R} \longrightarrow \mathbb{R}$ Linear Function, $M := Max_{x \in X} f(x)$,$m := Min_{x \in X} f(x)$, $\mathcal{X} CompactSet; c \in \mathbb{R}, \eta, \gamma, \alpha \in \mathbb{B}$).

4.0.3. Generalized Mixed-Integer Potential Games

In theory, every selfish road user $i \in \mathcal{I}$ can compute the solution of optimization problem 4-25. However, the linear constraints that were added earlier couple the dynamics of the vehicles, making the solutions strategies interdependent. Furthermore, each controller takes into account the strategies of the other agents. Therefore, if an agent does not share her strategy, it can make non-optimal or unsafe decisions. In addition, this interconnected problem cannot be solved by traditional optimization methods. Thus, we focus on designing a robust and interconnected controller to deal with decision-making and communication problems with other agents. To achieve the objective, we propose to formalize the autonomous driving problem as a GNEP [15].

First, we define the feasible set of each agent (i.e., automated vehicles) $\mathcal{X}_i(x_{-i}) := \{x_i \in \mathbb{R}^{n_i} | A(x_i, x_{-i} \leq b)\}$, where $\mathcal{X} := \{x \in \mathbb{R}^n | Ax \leq b\}$. However, each $J_i(x_i)$ depends only to the local states x_i, due this we introduce the function $P : \mathbb{R}^n \to \mathbb{R}$ defined as $P(x) := \sum_{i\in\mathcal{I}} J_i(x_i)$ satisfying:

$$P(x_i, x_{-i}) - P(y_i, x_{-i}) = J_i(x_i) - J_i(y_i)$$

for all $i \in \mathcal{I}$, for all x_{-i} and for all x_i, $y_i \in \mathcal{X}_i(x_{-i})$.
P is an exact potential function in the Multi Vehicle Automated Driven framework [17]. Let me introduce the mixed-integer response for each player i taking into account the strategies of its neighbors x_i:

$$x_i^*(x_{-i}) \leq \begin{cases} \arg \text{mín}_{x_i} J_i(x_i) \\ \text{s.t.}(x_i, x_{-i}) \in \mathcal{X}. \end{cases} \tag{4-27}$$

4.0.4. Centralize MPC

Centralized control has been used for decades as a solution to multi-agent problems. A node is responsible for calculating the decisions to be made by other agents taking into account all the information on the network. In **Fig. 4-2** is possible to see how the central computing node (CMPC) receives the information from each agent and gives a control signal as a response. This control is mostly used in networks with communication problems or slave nodes that do not have computing capacity. Centralized systems have some benefits over interconnected networks, although they are not the most used for this purpose. This section will introduce the Centralized Model Predictive Control (C-MPC), which leads to an optimal solution for the entire network.

The dynamics of the vehicles are represented by a monocycle model **3-5**. The vehicles' dynamics are combined to create a dynamic of the entire network. For simplicity, we represent the network model as a discrete, non-linear model of the form:

$$x(t+1) = \begin{bmatrix} f_1(x_1(t), u_1(t)) \\ \vdots \\ f_1(x_N(t), u_N(t)) \end{bmatrix} \tag{4-28}$$

A standard quadratic cost function MPC is defined. The objective is to follow a reference $r_i(t+k)$. The controller avoids collisions with other vehicles and at the same time achieves its safety constraints. The resulting MPC is:

$$V^*(x(\cdot),u(\cdot)) = \underset{x(\cdot),u(\cdot)}{\text{mín}} \sum_{i=1}^{N} \left(\sum_{k=1}^{Hp-1} \ell_{x_i}(x_i(t+k), r_i(t+k)) + V_{if}(x_i(Hp)) + \sum_{k=1}^{Hu-1} \ell_{u_i}(u_i(t+k)) \right), \tag{4-29}$$

s.t. ($\forall_{i,j} \in \mathcal{V}$):

$$x_i(t+1+k) = f_i(x_i(t+k), u_i(t+k)), \quad k = 0, ..., H_p, \tag{4-30}$$
$$x_i(t+k) \in X_i, \quad k = 0, ..., H_p, \tag{4-31}$$
$$u_i(t+H_p) \in \mathcal{U}_i, \quad k = 0, ..., H_u - 1, \tag{4-32}$$
$$c_{c.a.}^{(i,j)}(x_i(t+k), x_j(t+j)) \leq 0, \quad k = 1, .., H_p. \tag{4-33}$$

The objective function is defined as follows:

- $\ell_{x_i} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}$ is a function that represents the error between the desired trajectory $r_i(t+k)$ of the vehicle i and its current state $x_i(t+k)$. The following quadratic function represents the tracking error, where $Q_i(k)$ is the weighted matrix

$$\ell_{x_i}(x_i(t+k), r_i(t+k)) = \|x_i(t+k) - r_i(t+k)\|_{Q_i}^2 . \tag{4-34}$$

- $\ell_{u_i} : \mathbb{R}^{m_i} \to \mathbb{R}$ is the control input at each step for each vehicle i over the prediction horizon H_p. Using $R_i(k)$ as weighted matrix, ℓ_{u_i} is defined as:

$$\ell_{u_i}(u_i(t+k)) = \|u_i(t+k)\|_{R_i}^2 . \tag{4-35}$$

Finally, the definition of the restrictions of each vehicle are:

- $x_i(t+k) \in \mathcal{X}_i \subseteq \mathbb{R}^{n_i}$ represents the states of each vehicle i for $k = 1, ..., H_p$.
- $u_i(t+k) \in \mathcal{U}_i \subseteq \mathbb{R}^{m_i}$ represents the control inputs of each vehicle i. $\mathcal{U}_i$ represents the set of restrictions that each vehicle i has in its control signal.
- $f_i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$ describes the dynamic model of each vehicle i connected to the MVAD network.
- $c_{c.a.}^{i,j} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$ represents the coupling Mixed-Integer linear constraints between agent i and j (from 4-6 to 4-24).

In our implementation, a solution algorithm called Branch and bound was used. The Optimal Control Problem OCP is solved with the Gurobi framework [35] with a student license in MatLab software.

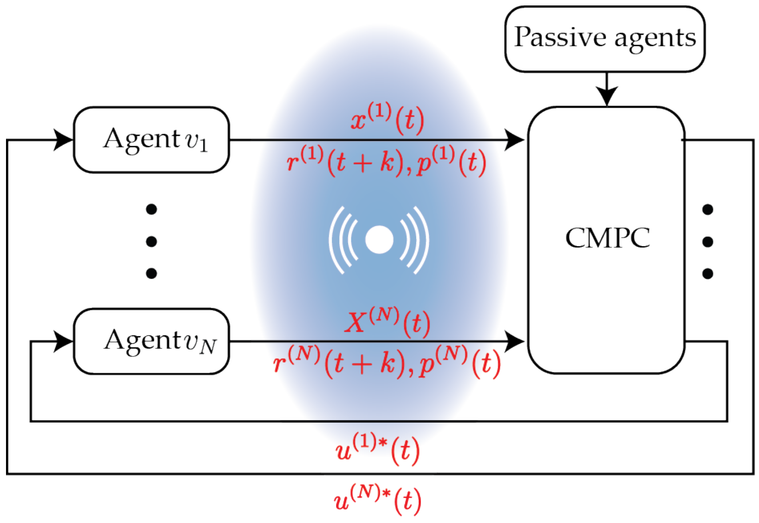


Fig. 4-2.: Control architecture.

4.0.5. Decentralized MPC

In centralized systems, the entire network relies on a single control node. However, as the network grows, the computation and synchronization of the agents become exponentially more difficult. However, suppose we decompose the whole network optimization problem into subproblems and rely on the communication between the neighbours. In that case, each subsystem will be able to obtain its own control input and achieve its goal even when some of the nodes are disconnected from the network. The architecture of the complete network can be seen in **Fig. 4-3**. This method is adopted from [13, 29].

Neighbors $\mathcal{N}_i$

Two or more vehicles are considered neighbors if:

$$\|p_i(t) - p_j(t)\| \le \alpha d_{covered} \tag{4-36}$$

where $d_{covered} = v_i\ h$ means the distance that each vehicle covers in order to take into account other agents around it. $p_i(t) = \sqrt{x_i^2(t) + y_i^2(t)}$ and $\alpha \in (1, 2)$. A pair of vehicles are considered neighbors if:

$$|x_i(t+k) - x_j(t+k)| \le d_{covered} \quad \& \quad |y_i(t+k) - y_j(t+k)| \le \mu. \tag{4-37}$$

where μ means the road-width of a single lane.

In the literature, the previous method is referred to as nonlinear distributed cooperative control [12, 13, 29, 45]. However, this method is not based on distributed optimization. Therefore, to avoid confusion, it will be called decentralized MPC (Dec. MPC).

All subsystems solve their optimization problem. Therefore, each vehicle $i \in \mathcal{V}$ must know the trajectories predicted by its neighbours $j \in \mathcal{N}_i$, which are considered optimal and obtained in the previous iteration. In the first step $t = 0$, the Optimal control problem OCP has not been resolved; therefore it has to be initialized. This starts with choosing a feasible value for the estimated control variable $\hat{u}_i$.

Initialization of estimated control

At every step of time the instance τ in interval $[t, t + H_p]$, the estimated control $\hat{u}_i$ for each vehicle is defined as:

$$\begin{cases} \hat{u}_i(\tau, t) & = u_i^*(\tau; (t-1)) \text{ if } \tau \in [\tau \ , \ (t-1) + H_p] \\ \hat{u}_i(\tau, t) & = 0, \text{ if } x_i(t) = r_i(H_p) \textbf{ or } \text{ if } \tau \in [(t-1) + H_p \ , \ t + H_p] \end{cases} \tag{4-38}$$

$u_i^*(\tau; (t-1))$ denotes the optimal solution to the OCP of the previous MPC iteration with initial state $x_i(t-1)$. However, sometimes for the first step t_o the OCP has not been solved, and it is required to have an optimal solution already established. We propose the following algorithm where a solution to the problem is defined in the interval $[(t_0 - 1), (t_0 - 1) + H_p]$

Algorithm 3 OCP initialization.

```
if $\tau \in (t_0 - 1)$ then
    for $\forall \tau \in [(t_0 - 1), (t_0 - 1) + H_p]$ do
        solve 4-40 with the initial state $x_i(t_0 - 1)$, $x_j(t_0 - 1)$ and $\hat{u}_j(\tau; x_j(t_0 - 1)) = 0$.
    end for
end if
```

The result obtained by **Algorithm** 3 is the optimal solution given the previously named initial conditions in the interval $[t_0, t_0 + H_p]$.

During each MPC iteration, the states and control signal obtained at $(t_o - 1)$ over the interval $\tau \in [(t_0 - 1), (t_0 - 1) + H_p]$ are defined by $x_i^*(\tau; x_i(t_0 - 1))$ as current states and the control signal as $u_i^*(\tau; x_i(t_0 - 1)$. u_i^* is applied to each vehicle i over time $[(t_0 - 1), (t_0 - 1) + H_p]$.

Again, the Decentralized Nonlinear MPC for Vehicle Collision Avoidance Problem use a nonlinear bicycle model 3-8 to represent the dynamics of the vehicles $i \in \mathcal{V}$ at time t in the decentralized network:

$$V_i^*(x(0), \hat{x}_j(\cdot), u_i(\cdot)) = \underset{x(\cdot),u(\cdot)}{\text{mín}} \left(\sum_{k=1}^{Hp-1} \ell_{x_i}(x_i(t+k), r_i(t+k)) + V_{if}(x_i(Hp)) + \right.$$

$$\left. \sum_{k=1}^{Hu-1} \ell_{u_i}(u_i(t+k)) + \sum_{j \in \mathcal{N}_{i,j} \neq i} \sum_{k=1}^{H_p-1} \gamma \left\| L_{col}(\mu_{i,j}(x_i, \hat{x}_j)) \right\|^2 \right), \quad (4\text{-}39)$$

s.t. $(\forall_{i,j} \in \mathcal{N}_i)$:

$$\begin{aligned}
&x_i(t+1+k) = f_i(x_i(t+k), u_i(t+k)), && k = 0, ..., H_p, && (4\text{-}40)\\
&\hat{x}_j(t+1+k) \in X_i, && k = 0, ..., H_p, && (4\text{-}41)\\
&x_i(t+k) \in \mathcal{X}_i, && k = 1, ..., H_p, && (4\text{-}42)\\
&u_i(t+H_p) \in \mathcal{U}_i, && k = 0, ..., H_u - 1, && (4\text{-}43)\\
&\|x_i(t+k) - \hat{x}_i(t+k)\| \leq h^2\mathcal{K} && && (4\text{-}44)
\end{aligned}$$

The objective function in 4-44 is equivalent to the Equation 4-33. However, many constraints have been modified for the decentralized controller. Each of them is explained in more detail below.

- $L_{col}(\mu_{i,j}(x_i, \hat{x}_j)) : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$ describes the cost it would have on the OCP for vehicle i to have a collision with vehicle $j \in \mathcal{N}_i$:

$$\sum_{j \in \mathcal{N}, j \neq i} \sum_{k=1}^{H_p-1} \gamma \left\| L_{col}(\mu_{i,j}(t+k)) \right\|^2 \text{ Where, } L_{col}(t) = \begin{cases} \frac{1}{\|x_i(t) - x_j(t)\| - 2D_i} & : \text{ if } j \in \mathcal{N}_i \\ 0 & : \text{if } j \notin \mathcal{N}_i \end{cases}$$

- $f_j : \mathbb{R}^{n_j} \times \mathbb{R}^{m_j} \to \mathbb{R}^{n_j}$ describes the prediction model of the other vehicles j in the neighborhood $\mathcal{N}_i$.

- $\hat{x}_j(t+k) : \mathbb{R}^{n_j}$ represents the states that were communicated from the previous iteration of the MPC.

- $\|x_i(t+k) - \hat{x}_i(t+k)\| \leq h^2\mathcal{K} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$ denotes the compatibility of agent i constraints with its previous results. Here, $h^2\mathcal{K}$ is close to zero, meaning feasibility for each computed control input.

In the same way as in C-MPC, the OCP in 4-44 is solved with the Gurobi optimizer.

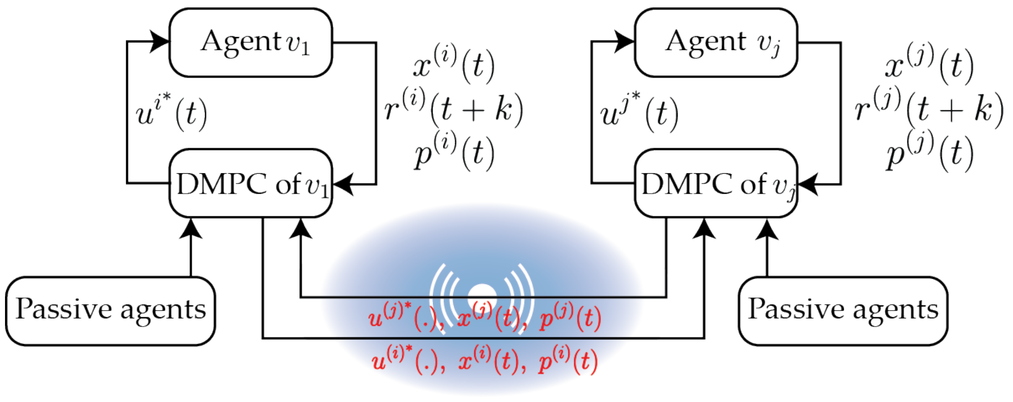


Fig. 4-3.: Control architecture for decentralized scheme.

4.0.6. Solution Method

Solving a GNEP is challenging even when only continuous variables are used. Our Optimization problem is a GNEP with mixed integer variables, and although it is used in both centralized and decentralized networks, its solution demands a large amount of computation. Therefore, we use the Gauss-seidel method to calculate the ϵ-MINE of the MVAD problem through iterations [15]. This method solves the GNEP despite the mixed nature of 4-25.

We will refer to the algorithmic step as an iteration to compute the solution; and the time step as a sample step of the decision variables. Let $x_i(k)$ denotes the decision vector of the agent i at the k-th step of the algorithm. Moreover, the $x_i(t|k)$ as the k-th step at time t of the i-th agent in the solution algorithm. Finally, the cost variance will be $\Delta J_i(k) := J_i(x_i(k)) - J_i(x_i^*(k))$, where $x_i^*(k) \in x_i^*(x_{-i}(k))$ will be the optimal decision vector of the i-th agent concerning its neighbours.

Gauss-Seidel Algorithm

We use a gauss seidel method from [14] for solution of the problem 4-25 we use a gauss seidel method of [14]. The algorithm follows a consecutive and ordered sequence to solve the optimization problem for each agent in the decentralized network and the entire network simultaneously in the centralized network. In **Algorithm** 4 the steps that the controller follows to compute the equilibrium of the GMIPG are detailed.

Algorithm 4 Gauss-Seidel Method for Centralized Networks.

```
for all $i \in \mathcal{I}$ do
    The controller choose a feasible state $x \in \hat{\mathcal{X}}_t$ for the whole network at time $t = 0$ and set $k := 0$
    while $x(k)$ is not an $\epsilon$-MINE do
        Load $x(t+1|k)$
        for all $i \in \mathcal{O}$ do
            $x_i(k+1) := \begin{cases} x_i^*(k) & \text{if } \Delta J_i(k) \leq \epsilon \\ x_i(k) & \text{Otherwise} \end{cases}$
            Load $x_i(t+1|k+1)$ to be use in the next step.

        Set $k := k+1$
```

Thus, for each time step $t \in \mathcal{T}$ we define an ordered and closed set of neighbors. In the centralized network, the set of neighbours is complete and is always of the same size.

Algorithm 5 Gauss-Seidel Method for Decentralized Networks.

```
for all $i \in \mathcal{I}$ do
    Each agent $i$ choose a feasible state $x(0) \in \hat{\mathcal{X}}_t$ and set $k := 0$
    while $x(k)$ is not an $\epsilon$-MINE do
        Broadcast $x_i(t+1|k)$
        for all $i \in \mathcal{O}$ do
            $x_i(k+1) := \begin{cases} x_i^*(k) & \text{if} \Delta J_i(k) \leq \epsilon \\ x_i(k) & \text{Otherwise} \end{cases}$
            Broadcast $x_i(t+1|k+1)$ to all $j \succ_t i$

        Set $k := k+1$
```

In the decentralized network, the ordered neighbor group differs from the previous method. Given a pair of vehicles (i, j), the algorithm considers them neighbors if $d_{i,j} \leq D_covered$ ordering and selecting the closest ones. The Algorithm 5 gives an initial condition to each subsystem. Then each subsystem shares its states and predicted trajectories with its neighbors $\mathcal{O}_t$. Each controller, with the information obtained from its neighbors and its current conditions, calculates a solution to 4-25. The last step is to share the new trajectories $x(t+1|k+1)$ to the neighbors.

4.1. Mid-Level control

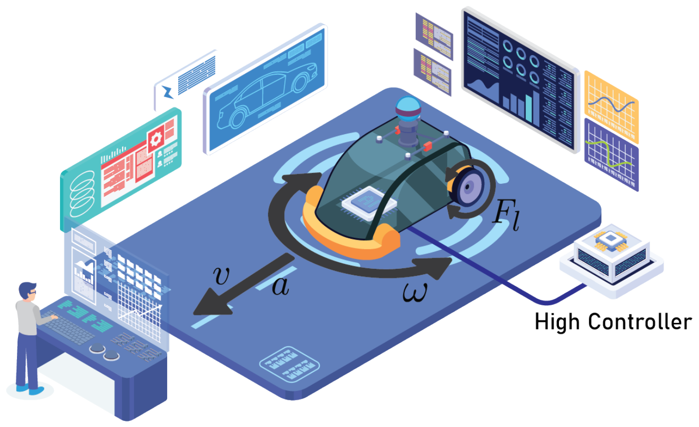


Fig. 4-4.: Mid-Level model.

A non-interconnected MPC was chosen for the Mid-Level controller. This method was chosen for its ease and robustness with nonlinear models. To set up this controller, we took into account the following features:

- The mid-level controller will not have a connection with the other agents, only with the high and low-level controller of each vehicle.
- The reference trajectory is given as the predicted trajectories of the upper high-level controller.
- The result chosen by the mid-level controller is the linear speed v and angular speed ω of the vehicle in its environment.
- The controller will have sensors that will detect if another obstacle, such as vehicles, people, or obstructions, is in the way.

Fig. 4-4 shows a brief idea of how the mid-level works, the variables it controls, and how it moves in the environment. Continuously will be given more information in detail about the mid-level model. We will not go deep into detail because the main contribution of this thesis is found in the high-level controller.

4.1.1. Overview

The main goal of any MPC is to solve an optimization problem with some constraints by minimizing certain control variables. Introducing the cost function $J(x_p, y_p)$, which we want to minimize, being $f(x, y)$ the nonlinear model of the Section 3.2.2 and $G(x, y)$ the constraint function, within the range of $\{g_{lb}, g_{ub}\}$.

$$\begin{cases} \underset{v,\omega}{\text{mín}} & J\left(x_p, y_p\right) \\ \text{s.t.} & l_{lb} \leq f(x,y) \leq l_{up} \\ & g_{lb} \leq G(x,y) \leq g_{up} \end{cases} \tag{4-45}$$

Since the solution of the algorithm is discrete, it is necessary to discretize the nonlinear model. From [44] we discretize 3.2.2 as shown in 4-46.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \overset{\text{Euler Discretization}}{\longrightarrow} \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} + \Delta T \begin{bmatrix} v(k)\cos\theta(k) \\ v(k)\sin\theta(k) \\ \omega(k) \end{bmatrix} \tag{4-46}$$

The objective function as a standard MPC is defined as the square of the difference between the desired position and the vehicle's current position, adding the magnitude of the control variables. The first term punishes the error between the desired state and the current state. The second term seeks to use the least number of control units to reach the objective. The following equation describes the objective function to minimize:

$$f(x,u) = \left\| x_o - x^{ref} \right\|_Q^2 + \|u\|_R^2 \tag{4-47}$$

Finally, the set of constraints is composed of the discretized dynamic model $f(x(k), u(k))$, the constraints of the control variable $u(k) \forall k \in [0, H_p - 1]$ and the spatial constraints of the highway $x(k) \forall k \in [0, H_p]$
The objective OCP that each controller must solve at each iteration does not have dynamics or restrictions coupled with its other neighbours.

$$\begin{cases} \underset{v,\omega}{\text{mín}} & J\left(x,u\right) = \sum_{k=0}^{H_p-1} f(x(k), u(k)) \\ \text{s.t.} & x(k+1) = f(x(k), u(k)), \\ & x(0) = x_0, \\ & u(k) \in U, \forall k \in [0, H_p - 1], \\ & x(k) \in X, \forall k \in [0, H_p]. \end{cases} \tag{4-48}$$

In the implementation, the Python programming language was used together with the ARGroHBotS [37] test lab. The GUROBI library was used to be able to provide faster solutions when it was implemented in real-time.

4.2. Low-Level control

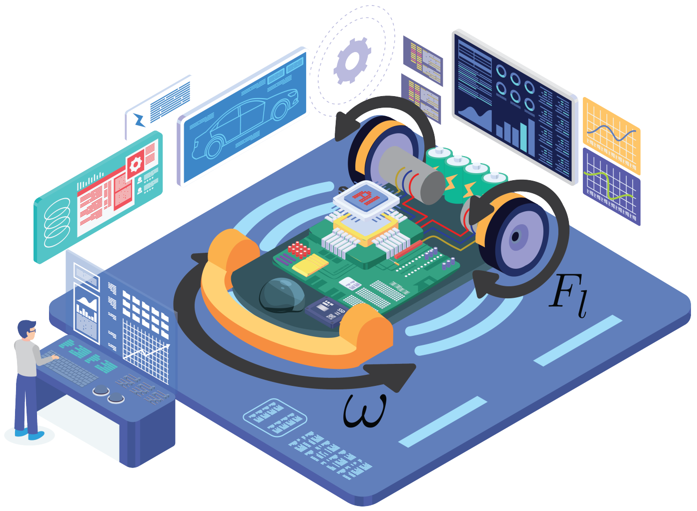


Fig. 4-5.: Low-level model.

A Proportional Integral Differential controller (PID) was implemented for the controller integrated into the robot. This was chosen for its ease, low computation level, and easy adjustment. Each agent has a microchip with a low level of processing, so it is necessary to have solution algorithms with low computational consumption, see subsection 3.3.1. However, it is enough to give a sufficiently optimal solution to the problem. The controller receives the reference position delivered by the mid-level controller and transforms it into the angular velocity of each wheel. **Fig. 4-5** shows how the low-level model works in the vehicle. The low level is controlled by a microcontroller embedded inside each car. Moreover, with some actuators, it can move each motor to give movement to the vehicle finally.

We define the control signal as the solution to the following mathematical equation:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de(t)}{dt}, \tag{4-49}$$

where the proportional gain $K_p = 30$, the integral gain $k_i = 0{,}1$ and the differential gain $k_d = 1$.

In **Fig. 4-6**, each controller delivers a response at a different time. This is due to the complexity, high computational load and the communication level that each level handles. The simulation results will be discussed in the next chapter.

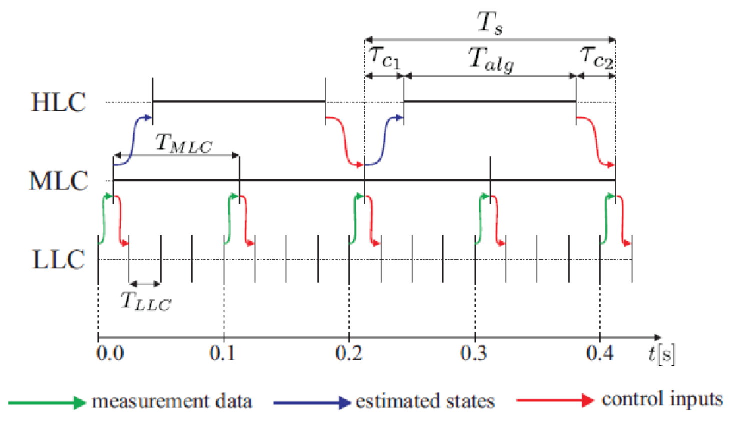


Fig. 4-6.: Illustration of the interaction between the hierarchical levels.

5. Controller Evaluation

This chapter presents the three main scenarios to test the algorithms. The primary purpose is to evaluate the efficiency and viability of the system in adverse conditions. In addition, in a simulated way, it seeks to demonstrate that the algorithms work. Later in the Annex 7.2, the same tests will be shown in a real test environment. Each scenario studied and the evaluation criteria used to evaluate each controller are explained below.

5.1. Testing Scenarios

We implement three main scenarios; the first one is selected to see how each agent works in a selfish environment to achieve its goal. The second one tests how well the whole group can avoid an obstacle. They have to synchronize with each other and avoid obstacles without changing their goals. Moreover, the third one is selected to see how the controller works by changing the environment variables restricting the scenario and seeing how it will deal with this situation.

5.1.1. Unrestricted Highway Scenario

In the first scenario, we consider a highway with L lanes and N vehicles driven parallel. They can be in front, behind, or to the sides. Each vehicle follows its own goal. In this section there are no restrictions or obstructions.

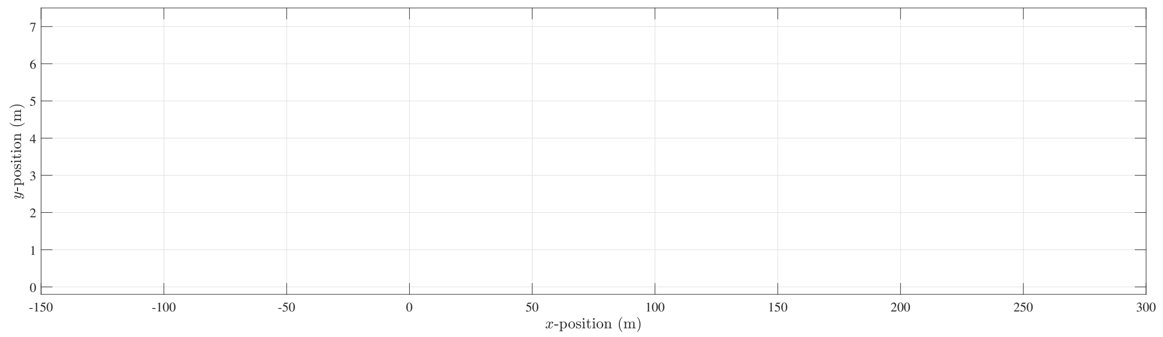


Fig. 5-1.: Unrestricted scenario.

5.1.2. Obstacle Avoidance Scenario

In the Obstacle Avoidance Scenario, we consider the previous scenario with the same number of lanes and vehicles. Moreover, it also has an obstacle represented as a broken-down car in the middle of the road. The high-level system can detect the position of vehicles and variables in the environment. Thus a possible obstacle is a slow-moving or broken-down car.

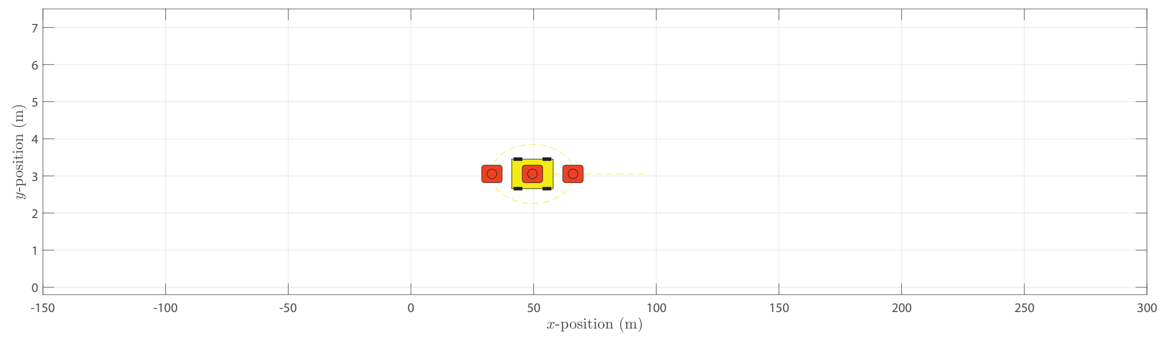


Fig. 5-2.: Obstacle avoidance scenario.

5.1.3. Reduced Lane Scenario

In the restricted scenario, we consider a restriction of the environment variables. Due to work on the road, the highway is reduced to the third lane, going from 5 lanes to 1. Each vehicle will have to slow down and synchronize to be able to share a single lane. The main idea of this scenario is to check how the system behaves in the face of an aggressive variation of the environment variables.

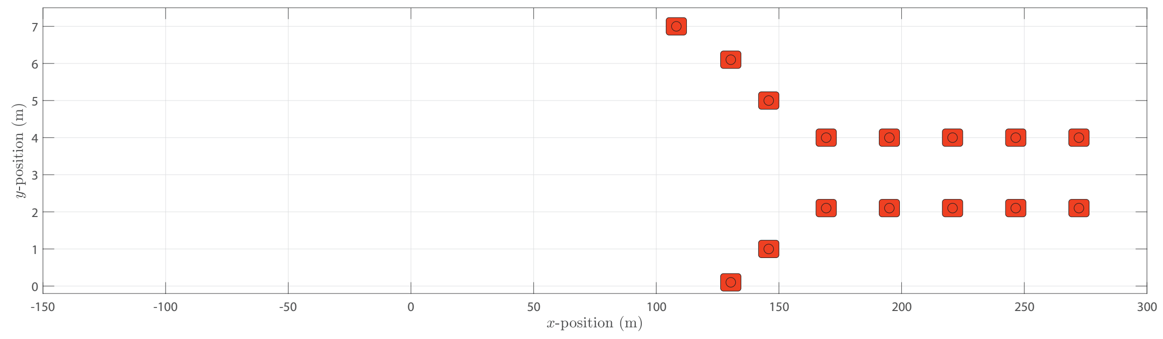


Fig. 5-3.: Reduced lane scenario.

5.2. Evaluation Criteria

The controller's effectiveness and the possible strategies' feasibility established the evaluation criterion. Effectiveness focuses on the quality of the trajectory and how each agent achieves its goal. Further, feasibility focuses on the computation time and the algorithm's synchronization with the other agent's controllers.

5.2.1. Feasibility

Feasibility is one of the most essential qualities in an interconnected vehicle control system. The controller must find a control response within the sampling time. Security will be compromised if the response time is greater. We seek to obtain an algorithm with a response as fast as possible.

Computation Time

Computation time is crucial for the feasibility analysis of a control algorithm. A critical aspect of implementing the MPC algorithm is the sample time. The most important requirement is that the control variable's value is found within the sampling time. In centralized systems, the increase in control agents increases computing time exponentially. In decentralized systems, the computation time is held regardless of the number of agents; therefore, the following equation must be maintained: $T_{sol} \leq h$, where T_{sol} is the solution time it takes for the entire system to solve, and h is the sampling time.
The computation time is evaluated by computing the mean, maximum, and minimum over the entire simulation time.

Robustness to Hardware Failure

Robustness is the second most important aspect of autonomous driving vehicle. A central node makes control decisions for all agents in centralized control networks. When there are connection problems, the agents cannot act due to the lack of a control signal. Each agent calculates its control response in distributed networks with information from its environment. If there are connection or communication problems, agents can still make decisions without having information about their neighbours.

The proposed system is decentralized; each agent receives information from its environment, finds the optimal solution in the solution space, and calculates the best control signal given the present conditions. It is sought through the simulations to see the system's robustness in the face of disconnections, network variations, and aggressive and selfish maneuvers of its neighbours.

6. Simulation

This chapter presents the results obtained in simulation for the MPC problem of path planning according to the parameters and conditions mentioned in Chapter 5. In an unrestricted scenario with no change in environmental conditions or road conditions.

For the simulation, the following assumptions are taken:

- All vehicles have the same dimensions.
- All vehicles have the same dynamic model, and there is no difference between them.
- The highway environment has no curves or intersections.
- The target speed and the desired lane are adjusted according to the interest of each driver.

The figures shown below are from different times. The initial time step shows the desired location and trajectories when $t_0 = 0$. In the time step, it is possible to see the most difficult maneuver to perform for some of the network agents. Finally, it is possible to see that everyone achieves their goal at the end of the control solution.

Colored squares represent vehicles, and circles represent possible crash risk areas. Colored dotted lines represent the predicted trajectories according to the agent to which they correspond. A final plot will be shown with as much information as possible in order to interpret the movement that the vehicles would have in a real scenario.

The main simulations were carried out with $N = 12$ vehicles. However, simulations were made to analyze computation time with $N = \{2, 10\}$ vehicles. First, the three most essential times in the simulation are shown, and then the quality and feasibility of each of the controllers will be discussed. In decentralized and centralized controllers, the trajectories are similar because the intention is to have the same working conditions. After a centralized and decentralized controller simulation, it is presented an analytic analysis. The analysis is based on the computation time as a function of the number of vehicles, the time per iteration of each controller, the increment of the computation time as a function of the prediction horizon, and the total simulation time as a function of the number of vehicles connected to the network at the end of each section, a conclusion of each scenario will be given.

The solution algorithm is solved in MATLAB [30]. OCPs are solved using the GUROBI mathematical optimization engine with the YALMIP programming interface. GUROBI optimization engine is chosen to solve most problems due to its powerful branch and bound algorithm. With this, it is possible to solve Mixed-Integer Quadratic Programming (MIQP) problems in the presence of non-linear models. This section of the document seeks to evaluate the behavior of vehicles in a selfish environment dealing with selfish agents and trying to reach their goal.

6.1. Unrestricted Scenario

The unrestricted scenario is designed with 12 vehicles on the same road. It also has six lanes in line, and the road is one-way. There are no intersections, traffic lights, obstacles, crossings, or any obstructions in the simulation environment.

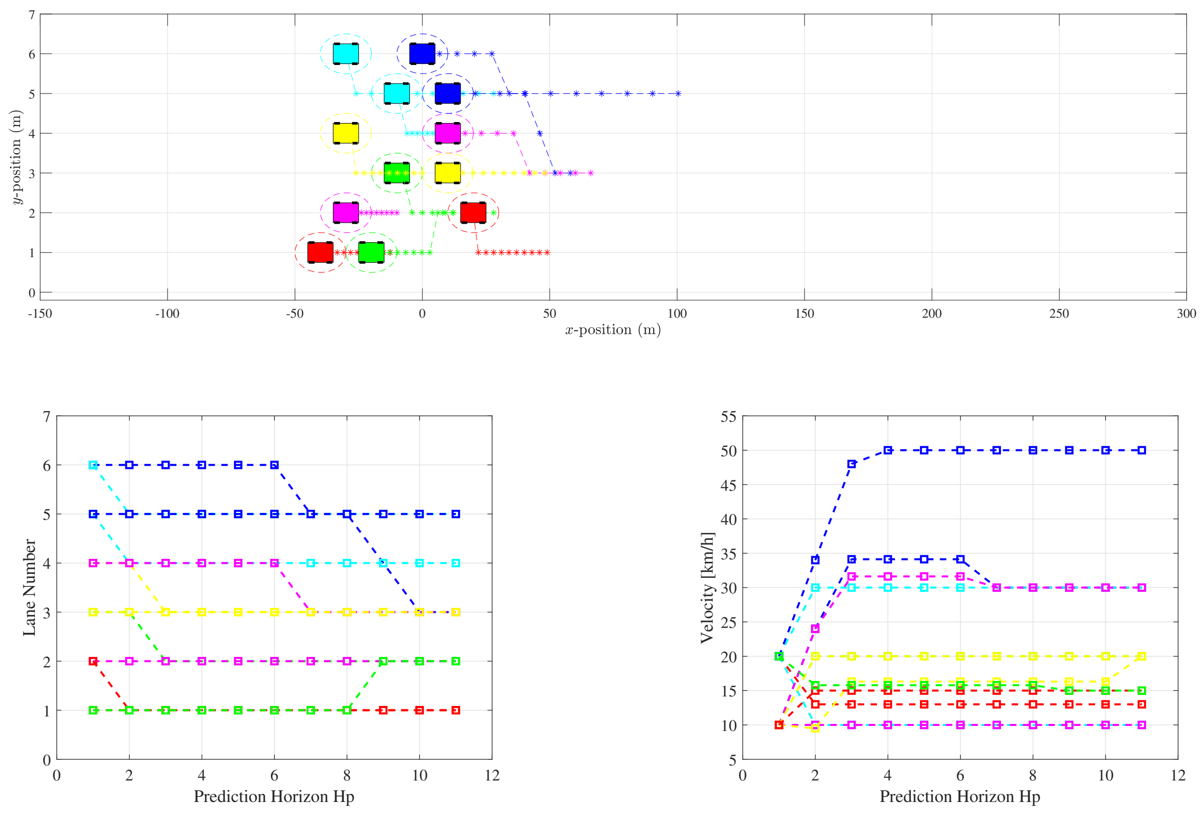
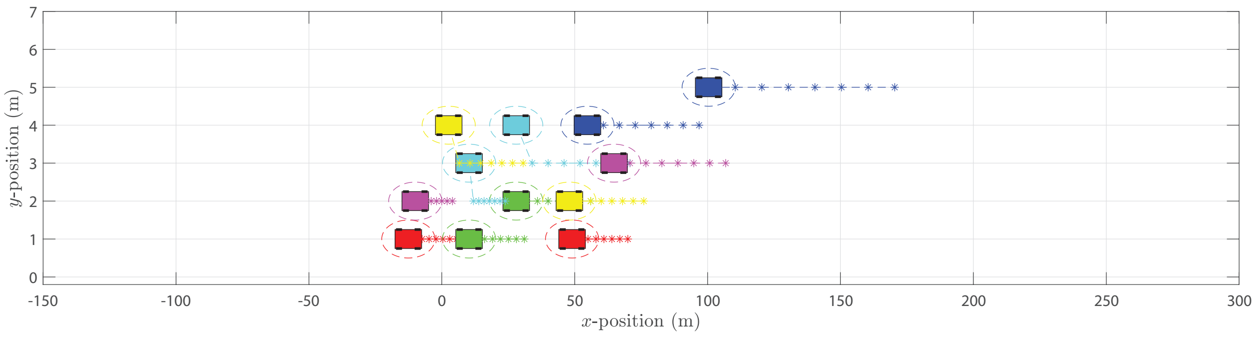
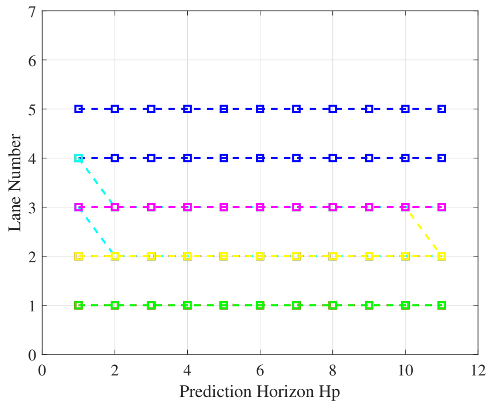


Fig. 6-1.: MPC Iteration = 0. Initial conditions.

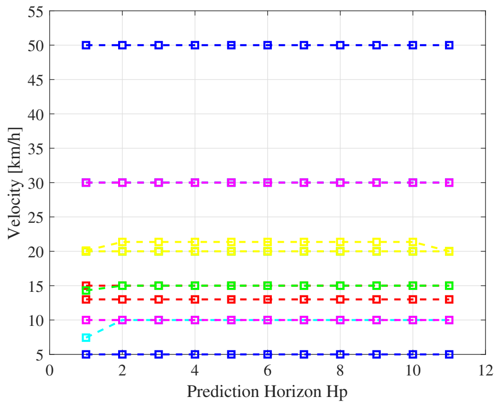
At the beginning of the simulation, the vehicles are adjusted to an initial position and speed, as explained in more detail in Chapter 4.0.5. As seen in **Fig. 6-1** in the first step, each of the agents has a predicted trajectory taking into account the predicted trajectories of its neighbors.



(a) Predicted position.



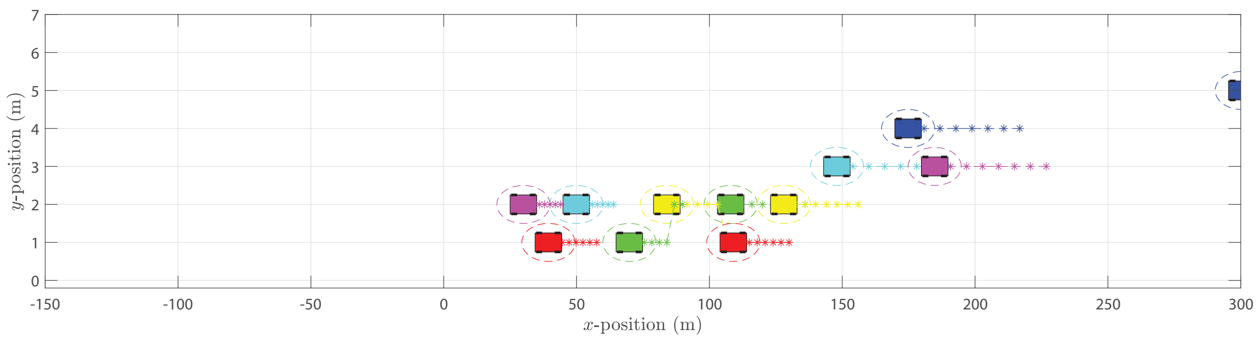
(b) Predicted lane positions.



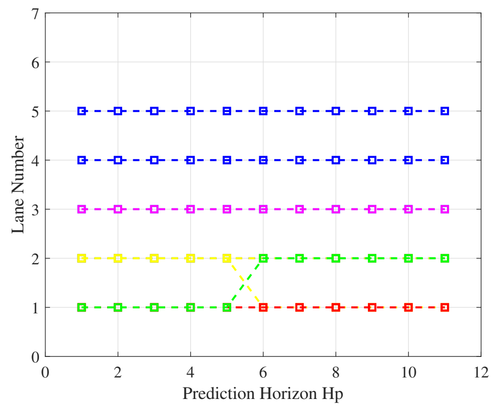
(c) Predicted velocity profiles.

Fig. 6-2.: MPC Iteration = 10. Vh 2,3,4,6 change lanes.

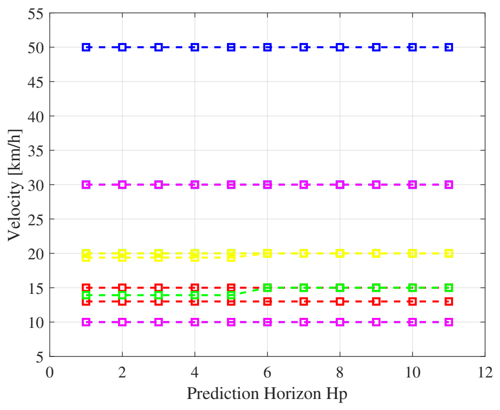
In the first scenario, the centralized and decentralized controllers achieve the main objective of each of the agents. The trajectories travelled during the simulation time in both cases are the same because the initial conditions and the main objective are the same. Therefore, the paths shown are equivalent for both network architectures, **Fig. 6-2**.



(a) Predicted position at first position.



(b) Predicted lane positions.



(c) Predicted velocity profiles.

Fig. 6-3.: MPC Iteration = 30. The overall goal of the network is achieved.

In the **Fig. 6-3** we can see how the entire network can achieve its main objective without any collision with their neighbors. The trajectory traveled by each of the vehicles can be seen in **Fig. 6-4**

The computation time of D-MPC during the entire simulation is computed with the mean of each step sample. In **Fig. 6-5**, it can see the average time that the network takes to solve each step k of the solution algorithm. In addition, the time taken by the fastest controller (lower barrier) and the slower controller (upper barrier) is shown. The time it takes for each agent to resolve its OCP depends on the environmental conditions and the main objective it is looking for.

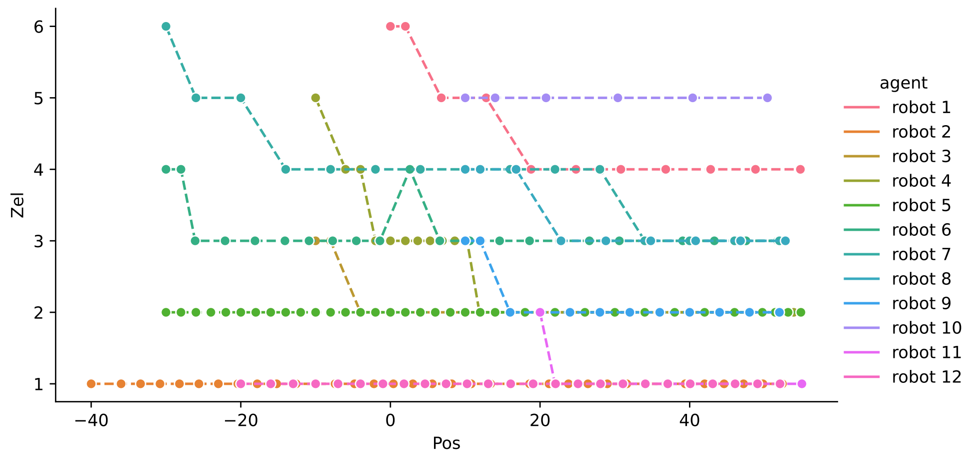


Fig. 6-4.: Trajectories of the entire network during the simulation time.

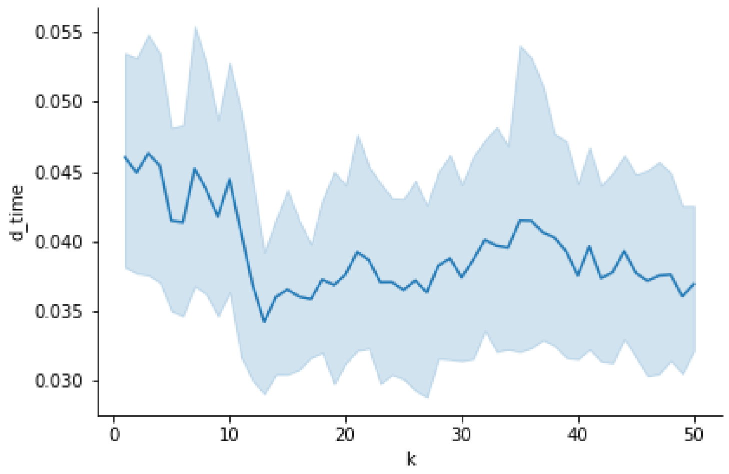


Fig. 6-5.: Step time.

Additionally, an analysis was made of how the prediction horizon affected the response time of each controller. We choose of having a prediction horizon of 10. Although this is large, it is not enough to affect the reliability of the system **Fig. 6-6**. Finally, an analysis of the time difference in the architectures is made. It is remarkable to see the improvement of the D-MPC networks. However, for real-time driving, the solution time is not enough **6-7**.

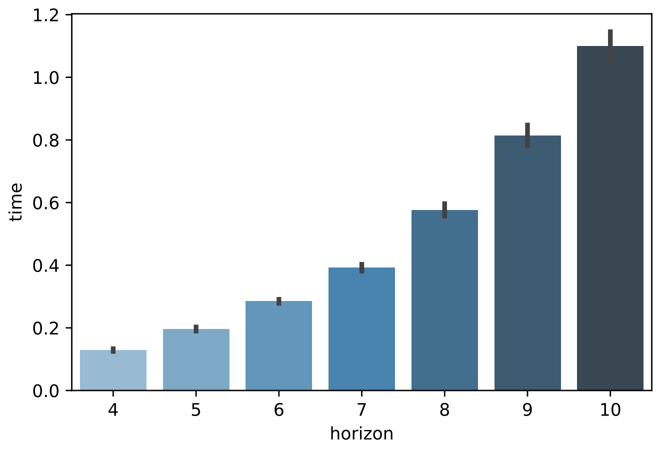


Fig. 6-6.: Computation time vs Horizon

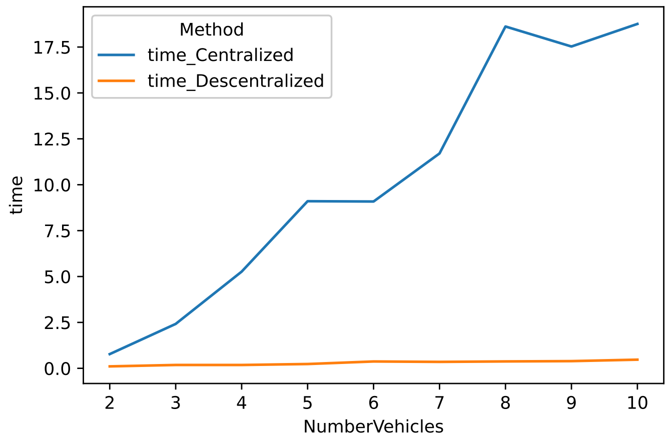


Fig. 6-7.: Computation time [s] vs Number of vehicles.

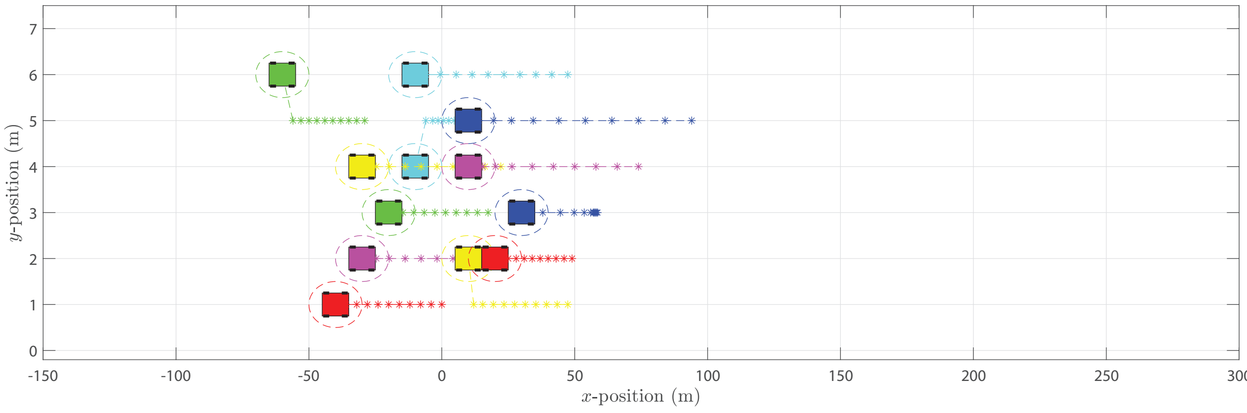
Conclusion Unrestricted Scenario

Both control structures (C-MPC and D-MPC) show that autonomous driving is possible in a highway environment. The potential game's framework allows the controller to take into account the conditions of others and make the optimal decisions in selfish scenarios.

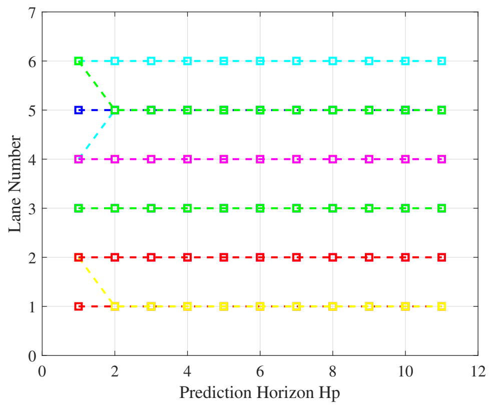
Given the time results, it can be concluded that in a highway scenario, it is not possible to drive a vehicle with the centralized technique due to the lack of computing power that we currently have to be able to solve this problem in a specific time. Solution times in D-MPC are much lower than what is seen in C-MPC. The C-MPC manages to give an optimal solution, but its solution takes a long time. Also, sometimes the controller falls into local minimums but fails to reach a global minimum.

6.2. Obstacle Avoidance Scenario

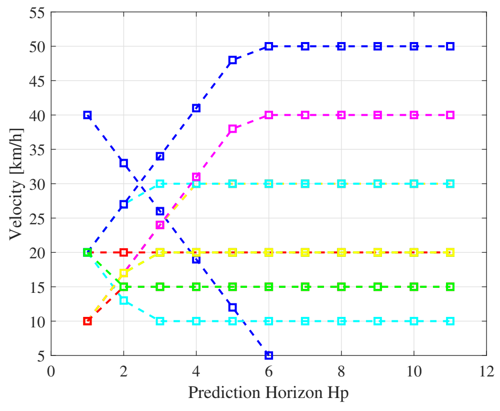
The obstacle avoidance scenario was designed with 12 vehicles on the same road. It also has six lanes in line, and the road is one-way. There is one vehicle in the lane $L = 3$ with no movement, **Fig. 6-8**. It represents an obstacle that the controller's vision system could recognize. The main objective is to prevent the damaged vehicle from achieving its objective speed and lane.



(a) Predicted positions.



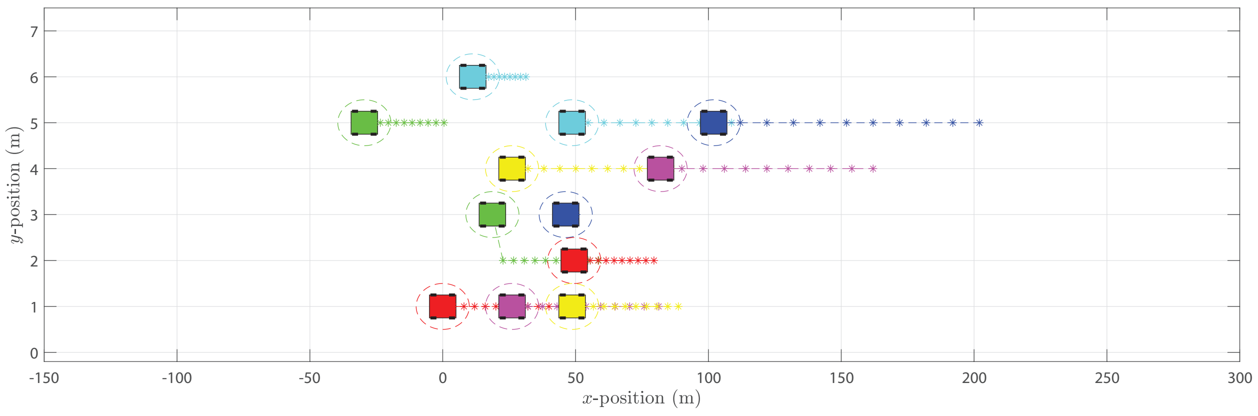
(b) Predicted lane profiles.



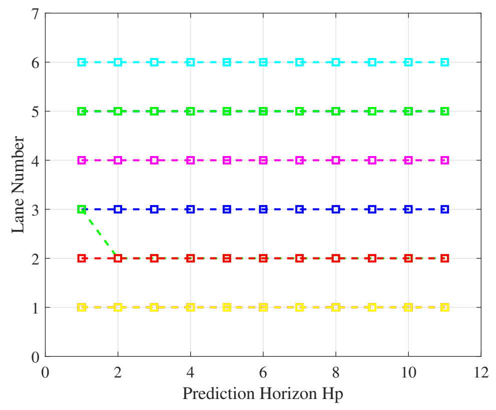
(c) Predicted velocity profiles.

Fig. 6-8.: MPC Iteration = 0.

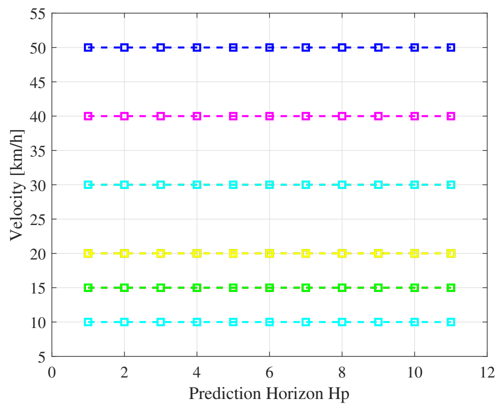
In iteration 0, the controller adjusts the initial positions of all the vehicles. The scenario poses a vehicle blocked on the road so that the vehicle detection system can detect this new obstacle. It is essential to mention that in the first step, three vehicles are planning in their path plans to change lanes. Those are the only vehicles that can change lanes. In addition, the rest of the neighbors do not plan to change lanes because they have vehicles around them.



(a) Predicted position.



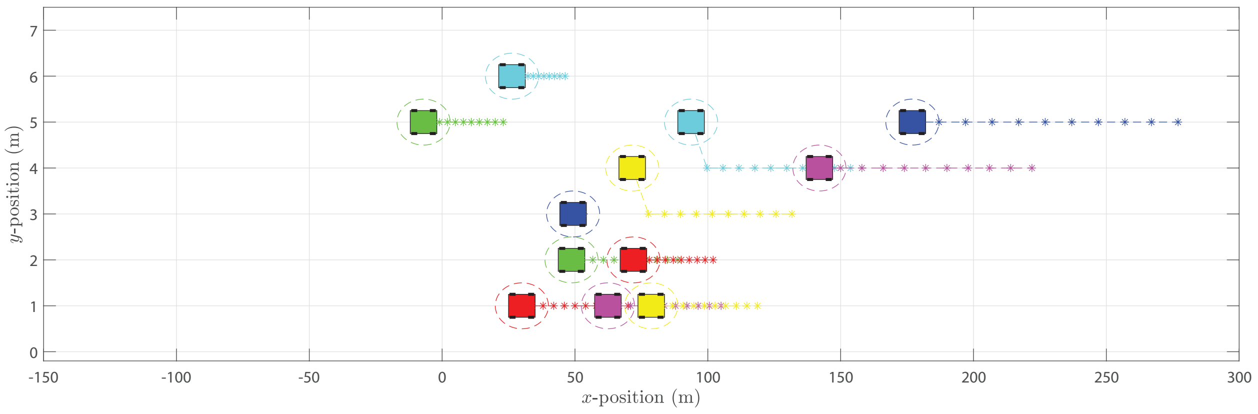
(b) Predicted lane positions.



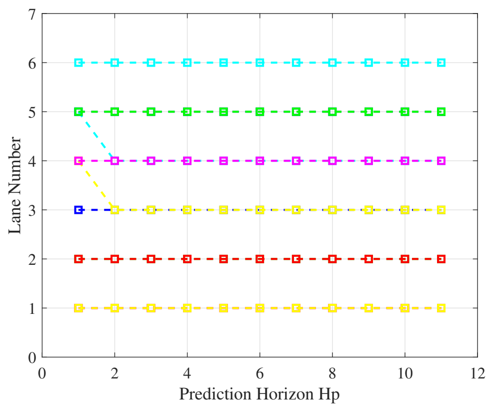
(c) Predicted velocity profiles.

Fig. 6-9.: MPC Iteration = 10.

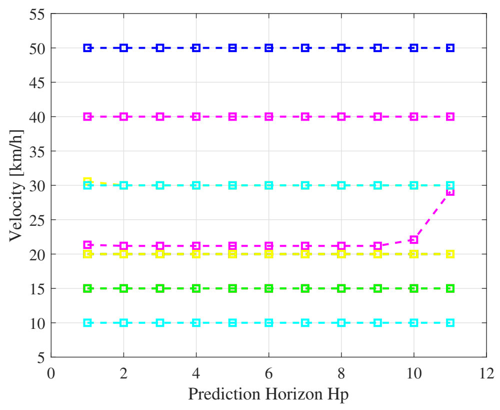
In **Fig. 6-9**a, it is possible to notice how vehicle number 3 (green color) has a trajectory prediction avoiding the obstacle. In addition, the other vehicles have changed lanes avoiding the obstacle. The vehicles $\mathcal{V} = \{4, 2, 5\}$ (color red, purple, yellow) have already reached their desired lane without colliding with the others. In **Fig. 6-9**b, the vehicles are not planning to change lanes because the majority have achieved their lane goal.



(a) Predicted position at first position.



(b) Predicted lane positions.



(c) Predicted velocity profiles.

Fig. 6-10.: MPC Iteration = 30.

In the **Fig. 6-10** we can see how the entire network can achieve its main objective without any collision with their neighbors. The trajectory traveled by each of the vehicles can be seen in **Fig. 6-11**

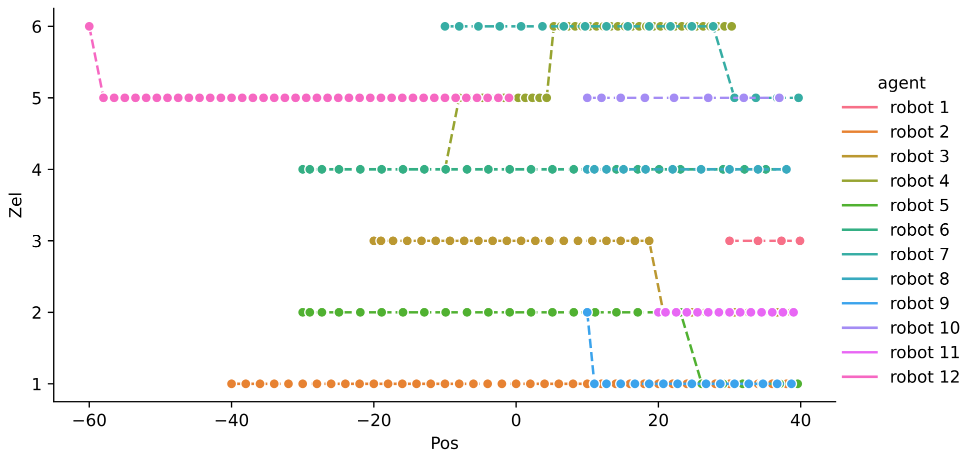


Fig. 6-11.: Trajectories of the entire network during the simulation time.

The simulation results show that the controller can provide a solution to obstacles that may arise **Fig. 6-12**. Also, the solution time of each of the sample steps is less in this scenario compared to the unconstrained scenario. This is due to the few interactions they have to do between vehicles after avoiding the obstacle vehicle. In addition, near step 10, it is possible to analyze the increase in solution time because in this step, the vehicles are dealing with the obstacle on the road.

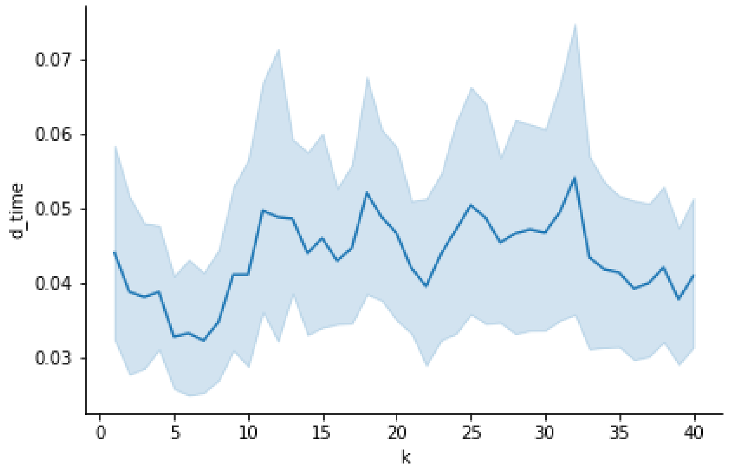


Fig. 6-12.: Step Time.

In the time analysis, depending on the prediction horizon, it can be seen that despite reaching 12 prediction steps, the system can provide a reasonable solution with a good time. In any case, the controller sometimes fails to find a minimum due to the big number of constraints.

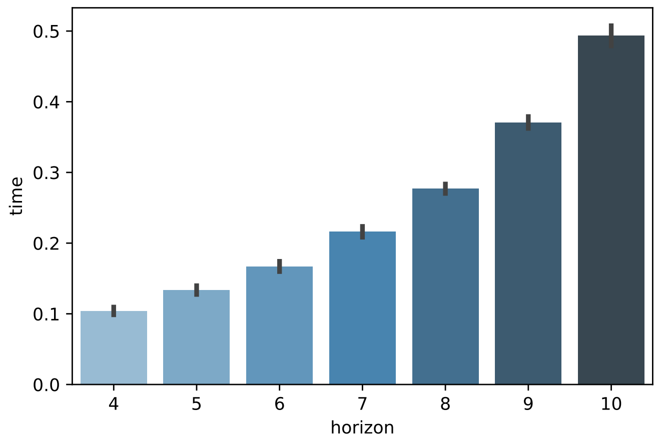


Fig. 6-13.: Computation time vs Horizon

Additionally, an analysis was made of how the prediction horizon affected the response time of each controller. We choose of having a prediction horizon of 10. Although this is large, it is not enough to affect the reliability of the system **Fig. 6-13**. Finally, as in the previous scenario, the difference in solving time is appreciated as the number of agents in the network increases. However, unlike the previous scenario, the D-MPC increases its duration with more vehicles, **Fig. 6-14**.

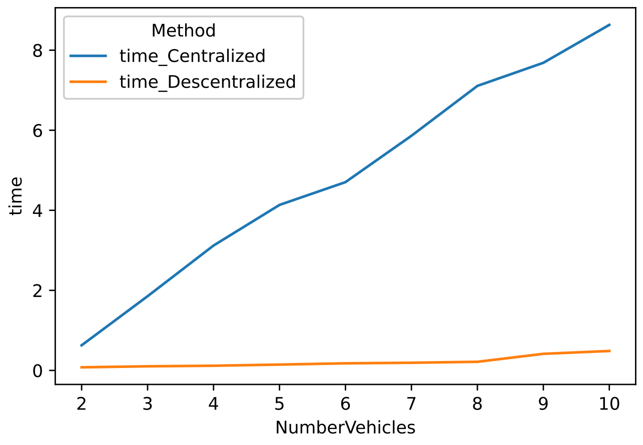


Fig. 6-14.: Computation time [s] vs Number of vehicles.

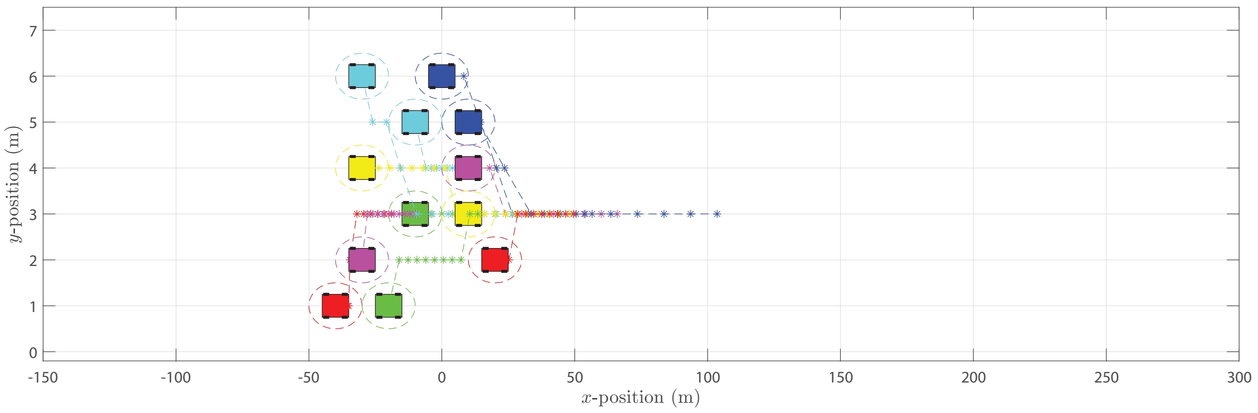
6.2.1. Conclusion Obstacle Avoidance Scenario

In a real highway environment, if a vehicle or other obstacle appears on the road, the decisions made by other vehicles are crucial. For this reason, this scenario of obstacles on the road was analyzed. With this new scenario, the number of constraints and the number of communications that each of the vehicles must deal with is greater than usual.

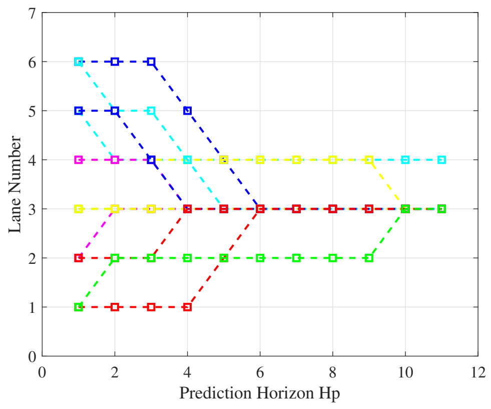
The obstacle avoidance scenario simulation shows that both C-MPC and D-MPC manage to solve the autonomous driving problem despite the increase in its constraints and a significant increase in communications between vehicles for the D-MPC case. However, the C-MPC continues to have difficulties providing an optimal solution in many evaluation cases. The system fell into an unfeasible position causing the entire network to crash. This is because the algorithm, that is used by the optimization engine (branch and bound), solves the problem by trying multiple possible solutions. While this is powerful, it can also be marginally unstable. The D-MPC manages to deal with the increase in constraints properly. However, more than the solution time is needed to be able to be implemented in real-time.

6.3. Reduced Scenario

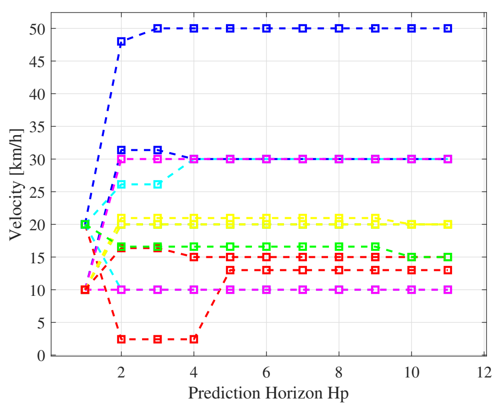
The reduced avoidance scenario was designed with 12 vehicles on the same road. It also has six lanes in line, and the road is one-way. There is a reduction in the number of lanes from 6 to 1. At a distance of 150 cm, the change starts. The main objective is to emulate a scenario where construction or change of the highway could happen. Moreover, it is an excellent opportunity to test the controller's behavior in the presence of a drastic change in its solution space, **Fig. 6-15**.



(a) Predicted positions.



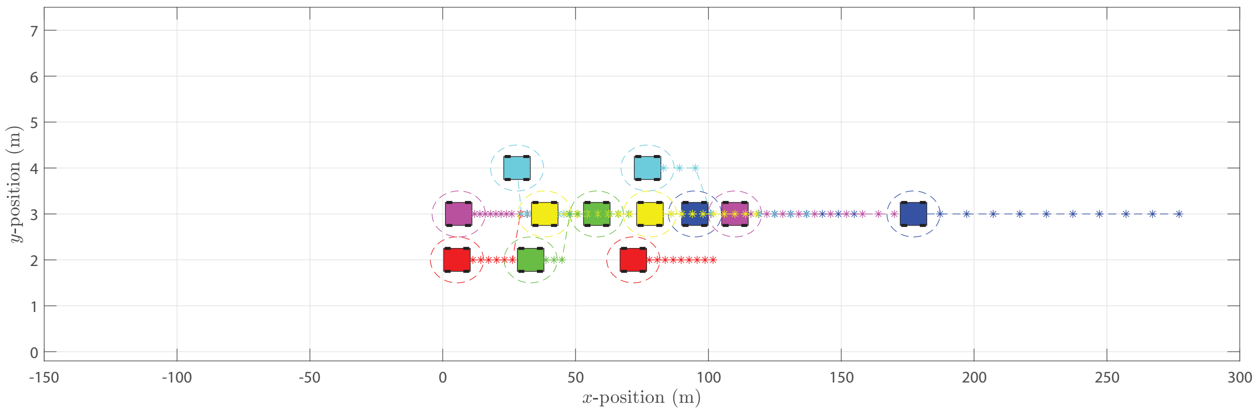
(b) Predicted lane profiles.



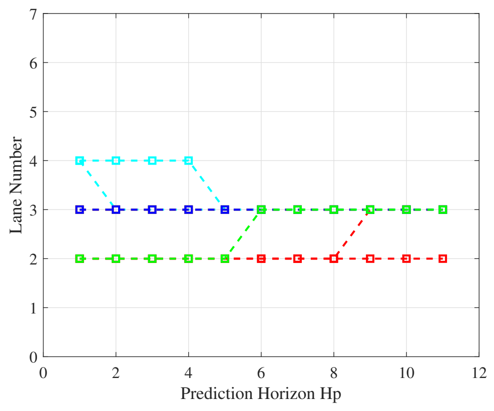
(c) Predicted velocity profiles.

Fig. 6-15.: MPC Iteration = 0. Lane reduction scenario.

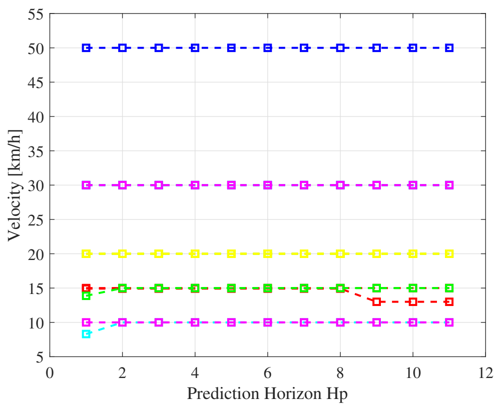
Step 10 of the simulation in the reduced scenario shows how the vehicles have managed to occupy three lanes without colliding. In addition, **Fig. 6-16**a shows seven vehicles that have managed to occupy the central lane and five planning to occupy it. Moreover, it shows how they consider the speed, current position, and current lane to predict the future where they can enter lane three without colliding with the others.



(a) Predicted position.



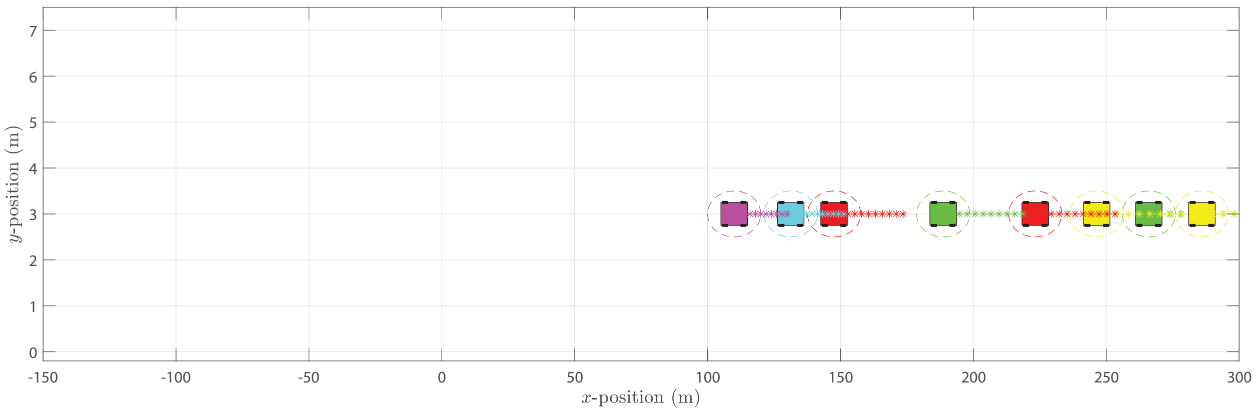
(b) Predicted lane positions.



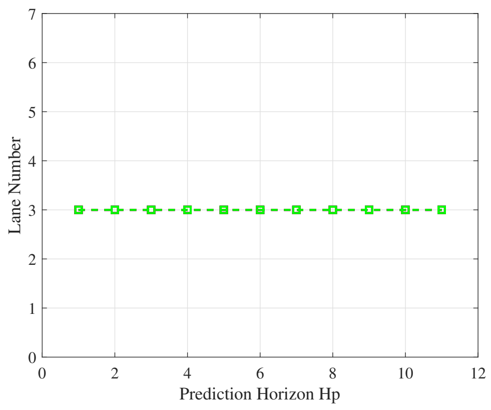
(c) Predicted velocity profiles.

Fig. 6-16.: MPC Iteration = 10. Lane reduction scenario.

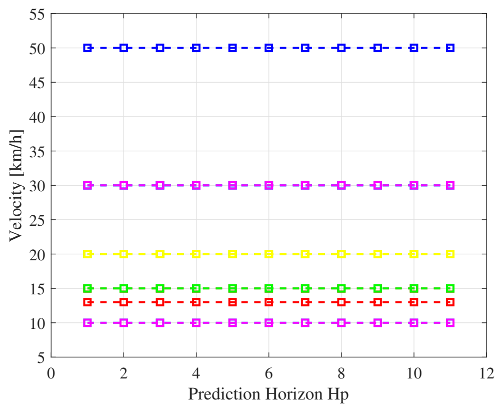
In **Fig. 6-16**c, it is seen how most do not change their speed even though they may have different target speeds. This consensus is defined by the Game Theory Controller, where it is of greater importance to comply with the security regulations than to achieve the main objective for the controller.



(a) Predicted position at first position.



(b) Predicted lane positions.



(c) Predicted velocity profiles.

Fig. 6-17.: MPC Iteration = 30. Lane reduction scenario.

In **Fig. 6-17** show how the entire network can achieve lane 3 without collision and fulfilling the restriction of the number of lanes. **Fig. 6-17**b and **Fig. 6-17**c are constant. It shows how the vehicles do not need to accelerate or make a lane change.

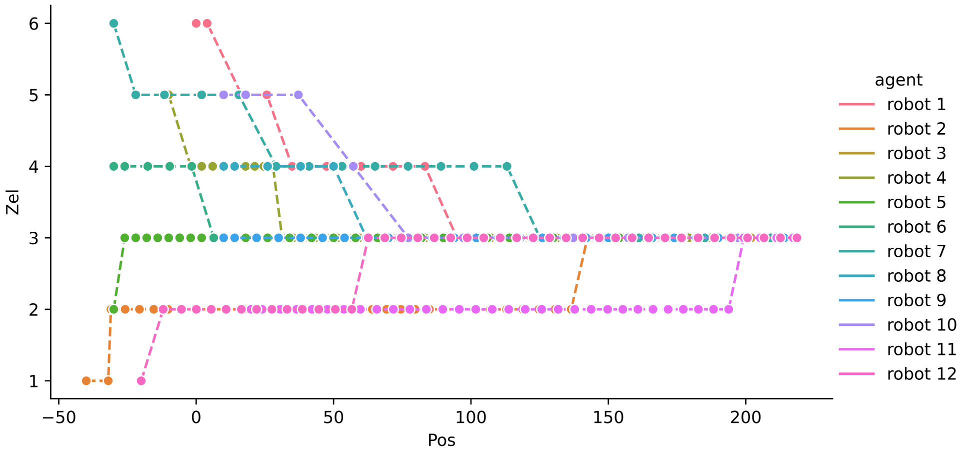


Fig. 6-18.: Trajectories of the entire network during the simulation time.

Fig. 6-18 shows the trajectories travelled by vehicles as a result of a Generalized Potential Game control repeatedly solved by D-MPC for 12 vehicles. It is interesting to see how, despite the short space and time, 12 vehicles manage to synchronize to be able to enter the same lane without colliding and taking into account the decisions of the others. Also, the last vehicle $i = 12$, is the last one that manages to join the lane. This is due to the initial conditions.

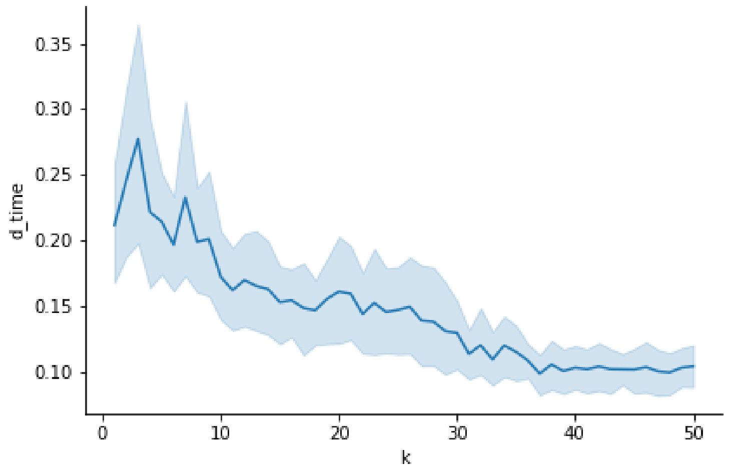


Fig. 6-19.: Step Time.

Fig. 6-19 shows how the average solution time of the D-MPC is reduced by the reduction of restrictions and by the limited solution space. The network is at its maximum computation level at the simulation's beginning. This is due to the beginning, exist 12 vehicles trying to synchronize in order to be all on the same line.

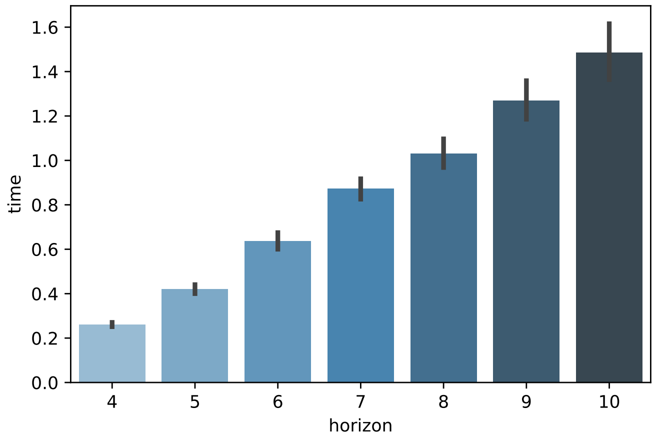


Fig. 6-20.: Computation time vs Horizon

The same feasibility analysis was performed by increasing the prediction horizon, and the D-MPC was able to solve the OCP without problems, **Fig. 6-20**. This system does not represent an effort to the Generalized Potential games theorem due to its low level of interactions between vehicles at the end of the simulation.

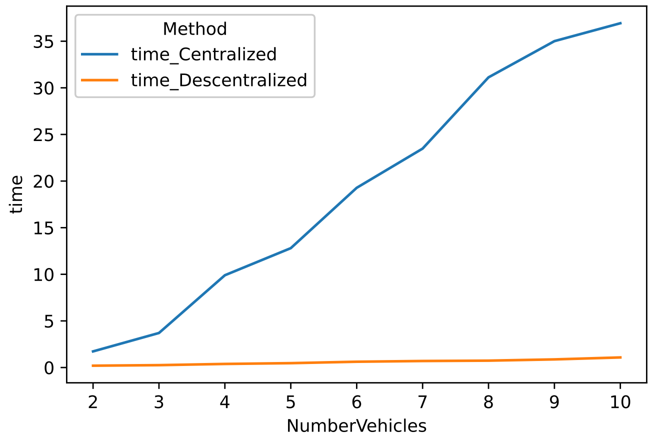


Fig. 6-21.: Computation time [s] vs Number of vehicles.

Finally, in the comparison of the times of each of the architectures, the C-MPC has an exponential increase in its solution time. The reduction of interactions does not significantly favor the centralized architecture **Fig. 6-21**. However, the D-MPC manages to maintain the average time of the complete simulation regardless of the increase in vehicles. This is due to the fact that the ϵ-Nash equilibrium can be solved more quickly, and the OCP problem is reduced to the interaction of each vehicle with the vehicle in front and behind.

6.3.1. Conclusion Reduced Scenario

One of the scenarios that a vehicle could face on a highway would be that of construction on the road. This new change in the road conditions would be reflected in the set of spatial variables $\mathcal{L}$, which would be reduced from 6 to 1. We trust that the GMIPG in a D-MPC architecture shown in this thesis document can deal with this problem.

In C-MPC, when vehicles are merging into a single lane, it is difficult for them to find an optimal decision to avoid colliding with each other. With an increase in the prediction horizon, this inconvenience could be solved. In D-MPC, the solution was always possible regardless of the prediction horizon used. In addition, the solution time could be reduced as each of the vehicles adhered to the single lane. This is because there are fewer connections that each agent must take into account to provide an optimal solution.

In short, the C-MPC fails to give an adequate response to the problem of driving on a freeway with a lane reduction and in front of selfish agents trying to join the same lane. The D-MPC controller manages to deal with this task with improved solution time and does not have problems with the solutions. See **Fig. 6-19** . The reduced scenario was chosen to analyze the behavior of both drivers in the face of a sudden reduction, such as the lanes of a highway. See **Fig. 6-18**. All vehicles manage to synchronize to be able to stay in a single lane.

7. Conclusions and Recommendations

7.1. Conclusions

This thesis document compares the results obtained by a centralized system versus a decentralized system. In both cases, the complete network solves the problem of autonomous driving of multiple vehicles as a generalized mixed-integer potential game problem. Centralized networks solved with a C-MPC can give enough results to solve the problem. Moreover, they depend on a single central node, besides being slow and requiring large amounts of computational power to solve the problem. Decentralized networks solved with a D-MPC provide a faster and more feasible solution to the same problem without requiring a central node to solve the main problem. Those are more robust because the solution is not entrusted to a single node. It means communication problems will not break the network like they would on the C-MPC. Despite all these advantages, this method relies completely on efficient communication. The network will only be the most appropriate if the latter exists.

The generalized mixed-integer potential games framework offers excellent results in solving problems with mixed-integer variables and dynamic graph topology. The use of game theory substantially improves the solution to the problem, by taking into account the decisions of others, make your own. It allows each vehicle to anticipate an unsafe maneuver for the driver or his neighbors on the highway.

The use of mixed integer variables converts the model to a simple system. The previous theory is an advantage of making the problem solvable and improveing safety. Unfortunately, it is currently impossible to implement this kind of algorithm in production vehicles due to technological shortcomings in communication and computing power.

The solution proposed in this thesis is restricted only to road scenarios with vehicle detection systems. It does not consider the other scenarios and conditions a real driver may face. In addition, it assumes that the communication and identification of the vehicles around it are optimal and that there will be no delays, miscommunications, or communication problems between drivers. The calculation time of each of the controllers is different. The high-level controller was simulated in a virtual environment, and the result was used as a reference for the medium and low-level controllers.

In **Appendix** 7.2, the images collected from the implementation of the controller proposed in this thesis in a real environment of vehicles used in mobile robotics were added. The environment was designed, programmed, and built to implement this control system efficiently. However, that doesn't limit you to other possible controllers.

7.2. Future Work

For future studies, it is recommended to investigate further solutions to the problem of low computational power that an electric vehicle can have. since large amounts of computation are required that translate to large hardware and high energy consumption. Finally, it is important to investigate in depth the use of better sensors to identify any reactive mobile agent that may appear.

A. Annex: Implementation Images

In addition to simulations, the complete system was implemented in a testbed system [37]. Below are the images taken in the implementation of each of the scenarios analyzed in this document.

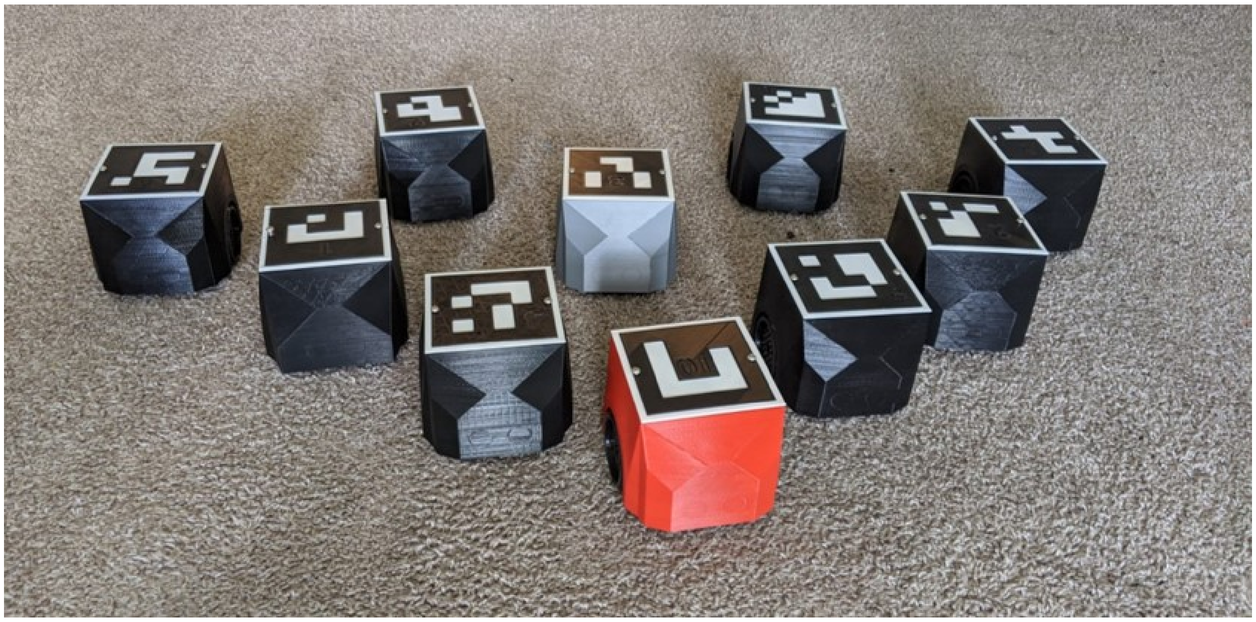
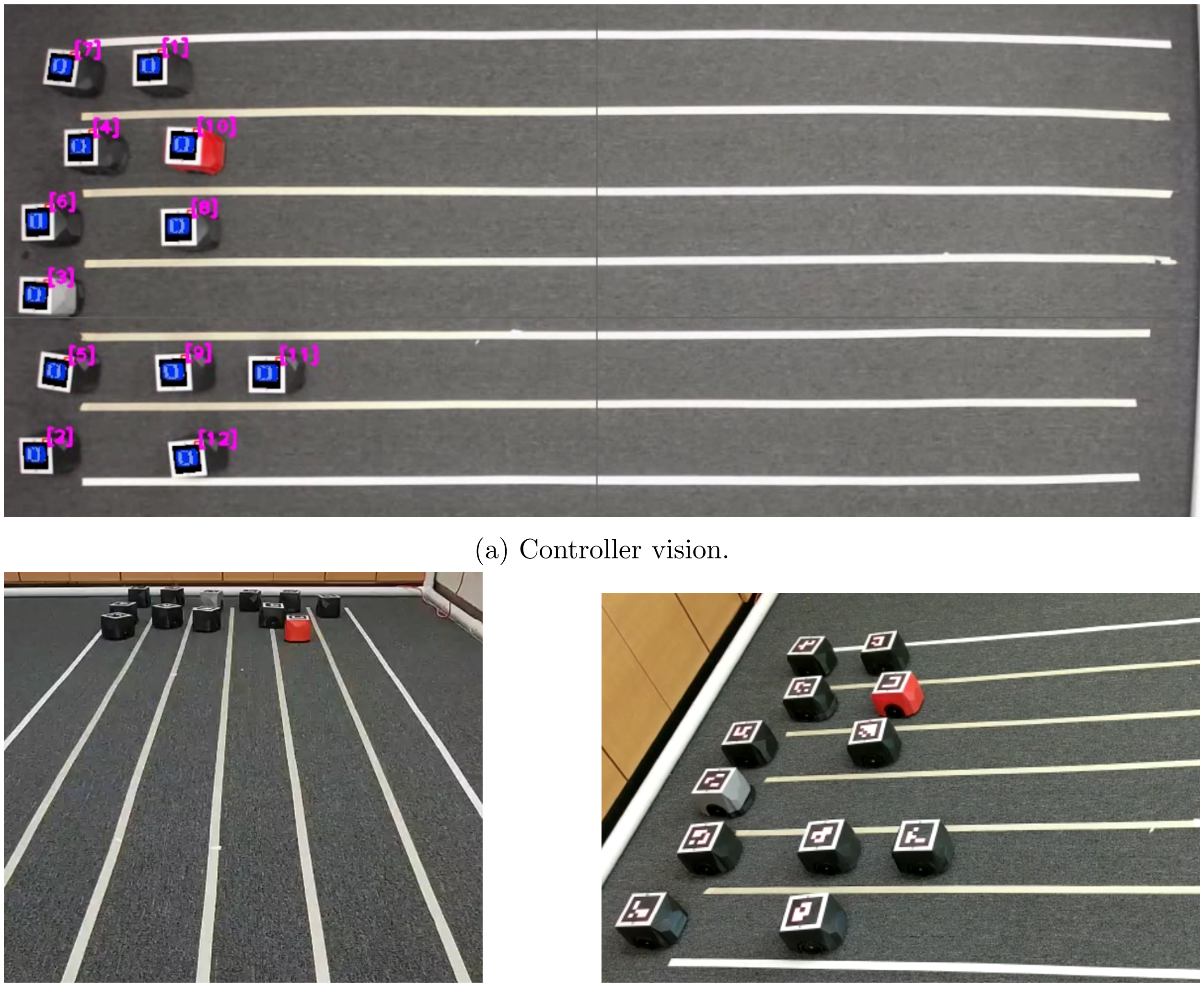


Fig. A-1.: Vehicles used in the implementation.

A.1. Unrestricted Scenario

For the unrestricted scenario, we use 12 vehicles in the laboratory. It was necessary to mark off the lanes on the floor. Like the simulation, we use a scenario with six lanes. **Fig. A-2**a shows the video used for the pc to calculate the position of each robot in the environment. **Fig. A-2**b shows a frontal vision of the scenary. Finally the Fig. **A-2**c shows the vision of the vehicles horizontally.



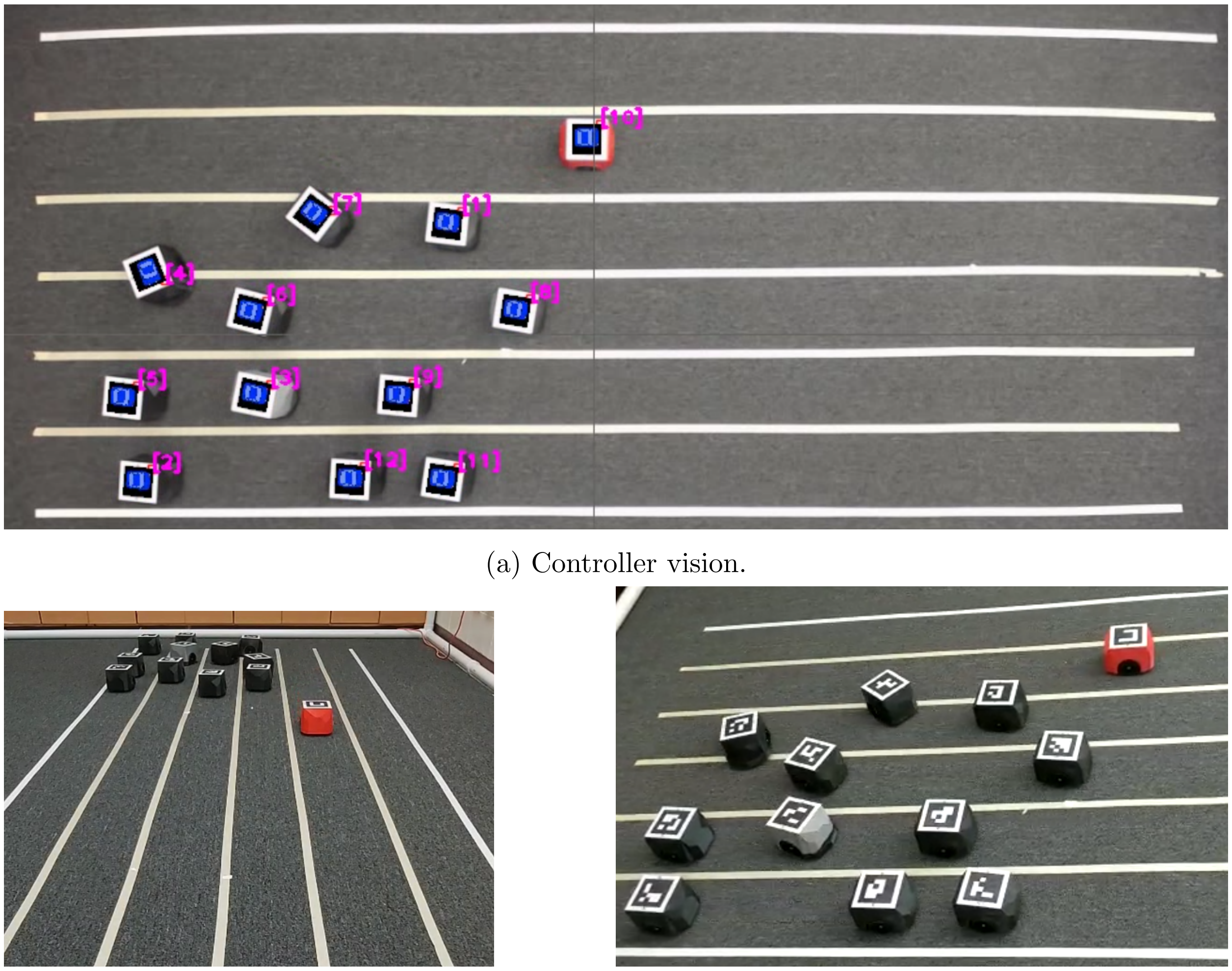
(a) Controller vision.

(b) Front vision.

(c) Lateral vision.

Fig. A-2.: Implementation at iteration 0.

The first part for implementing unrestricted scenario vehicles is positioned in the initial conditions established in the simulation already carried out in Chapter 6. Each vehicle has a different goal, and the whole group will try to fulfill its objective avoiding the agents.



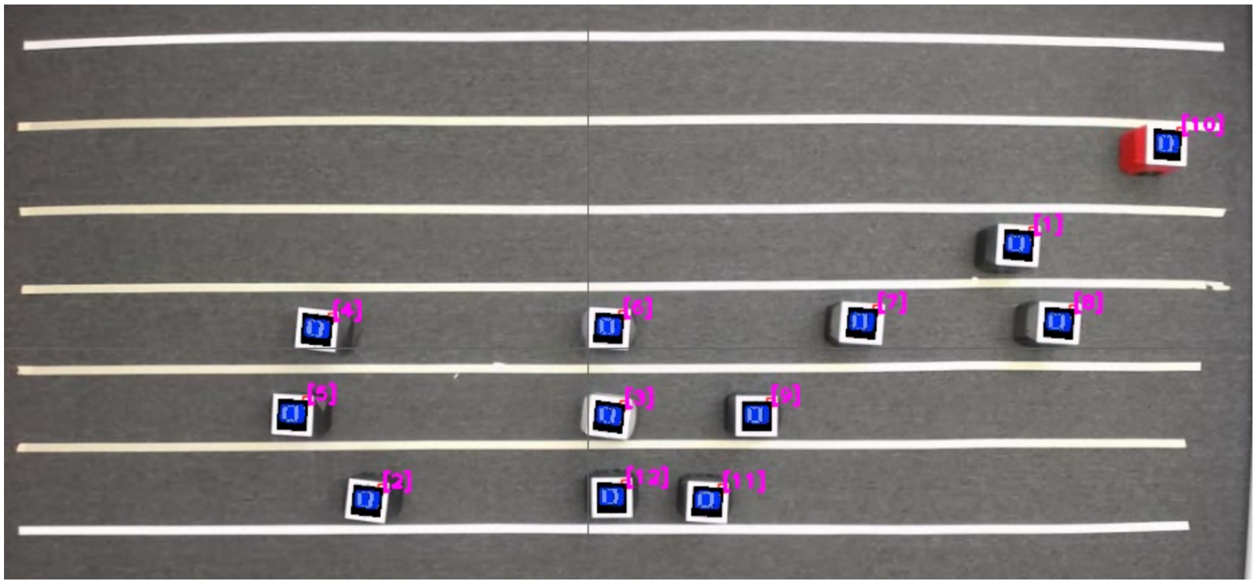
(a) Controller vision.

(b) Front vision.

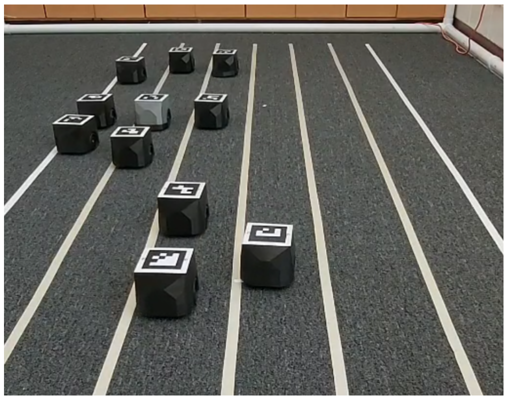
(c) Lateral vision.

Fig. A-3.: Implementation at iteration 10.

In iteration 10, we can see how each car tries to change lanes, but some of them keep its lane due to safety restrictions, **Fig. A-3**. For example, the white car keeps going in the same lane $z = 2$ because a change of lane could be represented in a collision.



(a) Controller vision.


(b) Front vision.

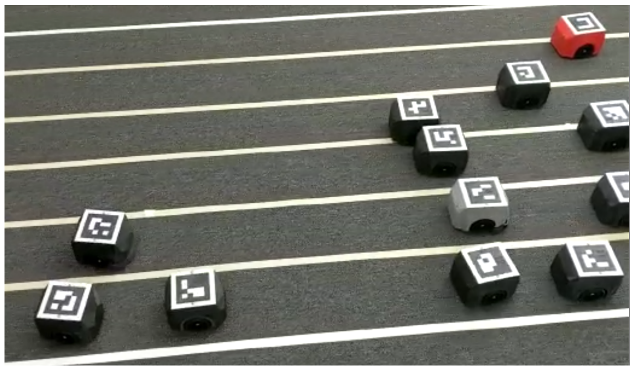
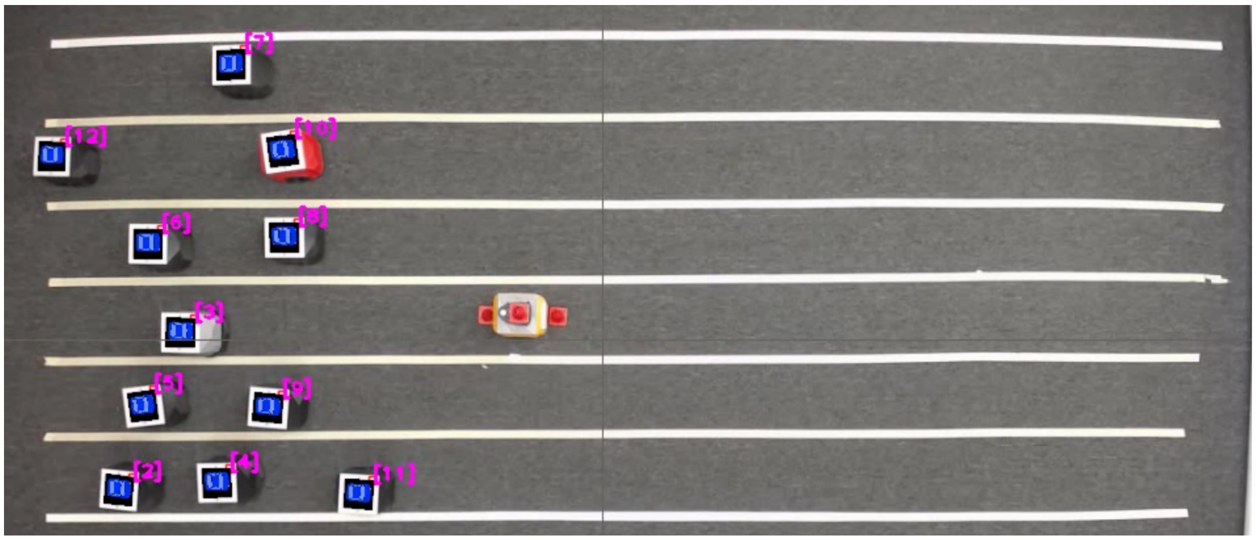

(c) Lateral vision.

Fig. A-4.: Implementation at iteration 30.

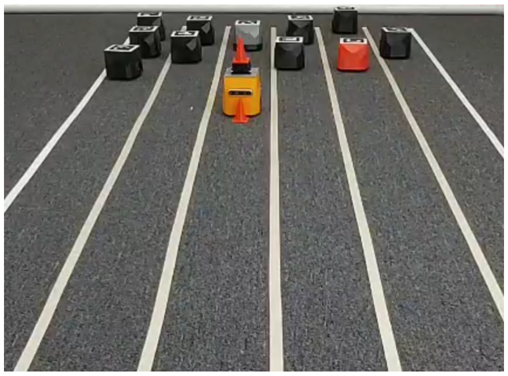
How it was predicted and simulated, the entire network can achieve its primary goal without a critical collision, **Fig. A-4**. The vehicles fulfil the lane at an speed desired previously.

A.2. Obstacle Avoidance

In Obstacle avoidance implementation, we use a vehicle with a cone on top as an obstacle. The objective was to simulate what could happen if an obstacle appeared on the highway and how the network could deal with it.



(a) Controller vision.



(b) Front vision.

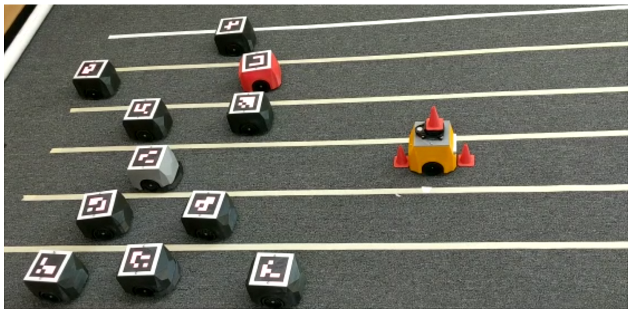
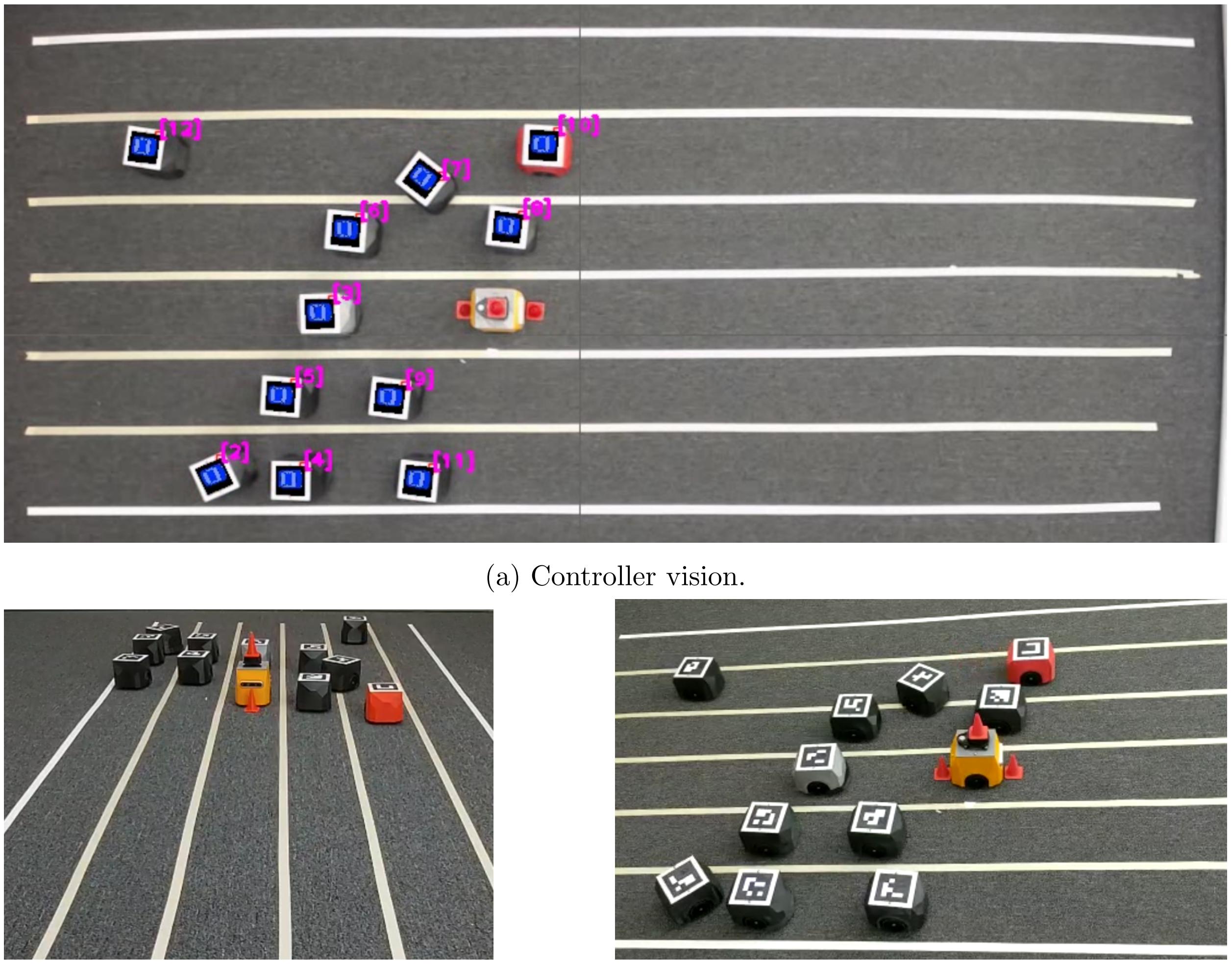


Fig. A-5.: Implementation at iteration 0.

We placed 12 vehicles in the initial condition, and we also used some cones as a sign of a wrecked car, **Fig. A-5**. All vehicles will try to avoid it but simultaneously achieve their primary goal.



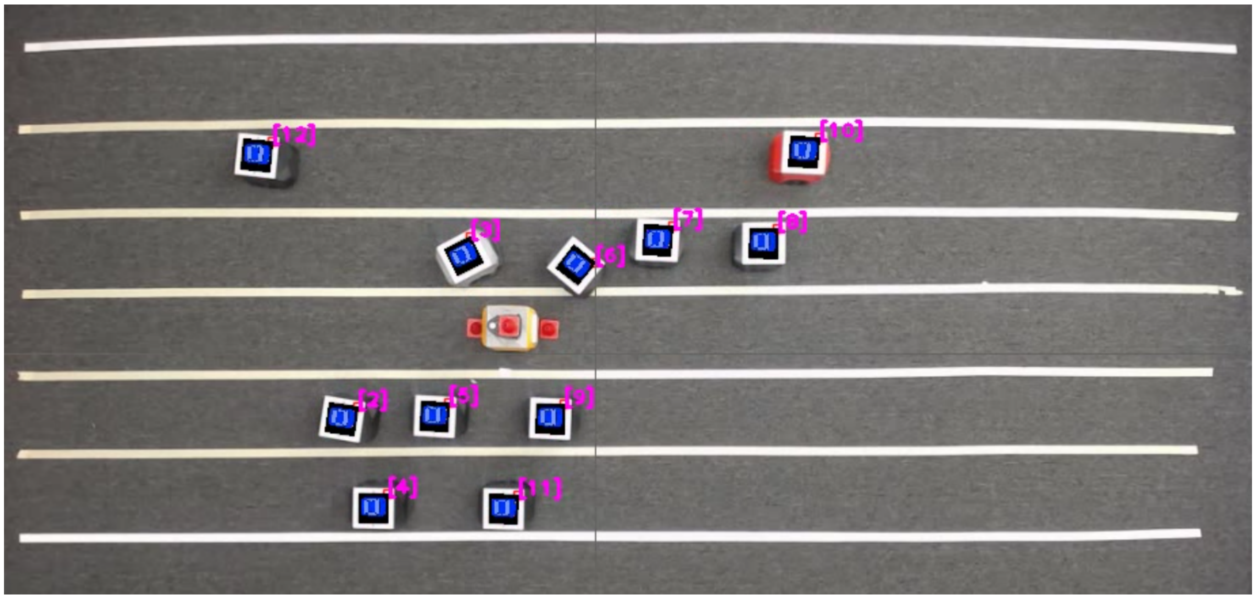
(a) Controller vision.

(b) Front vision.

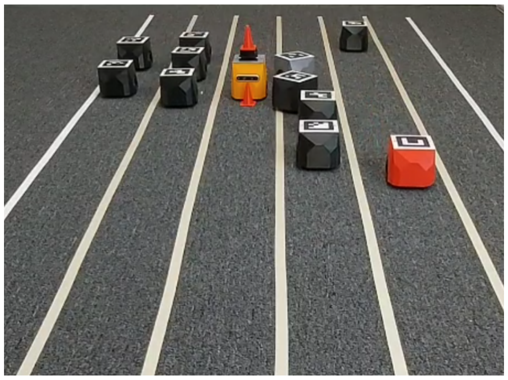
(c) Lateral vision.

Fig. A-6.: Implementation at iteration 10.

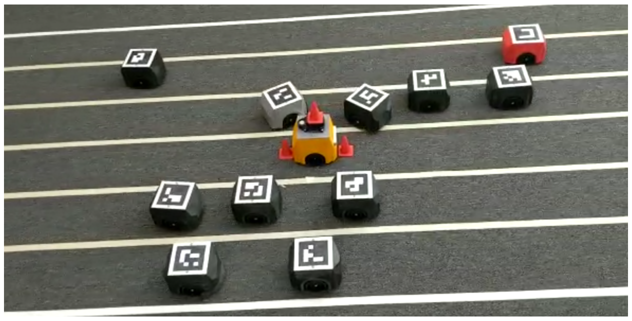
In **Fig. A-6** is possible to see that all cars are changing lanes, but the white vehicle $i = 3$ is dealing with the obstacle in front. In the implementation, we can see how the white vehicle reduces its speed, waiting for the other lanes to change and then pursuing its goal.



(a) Controller vision.



(b) Front vision.



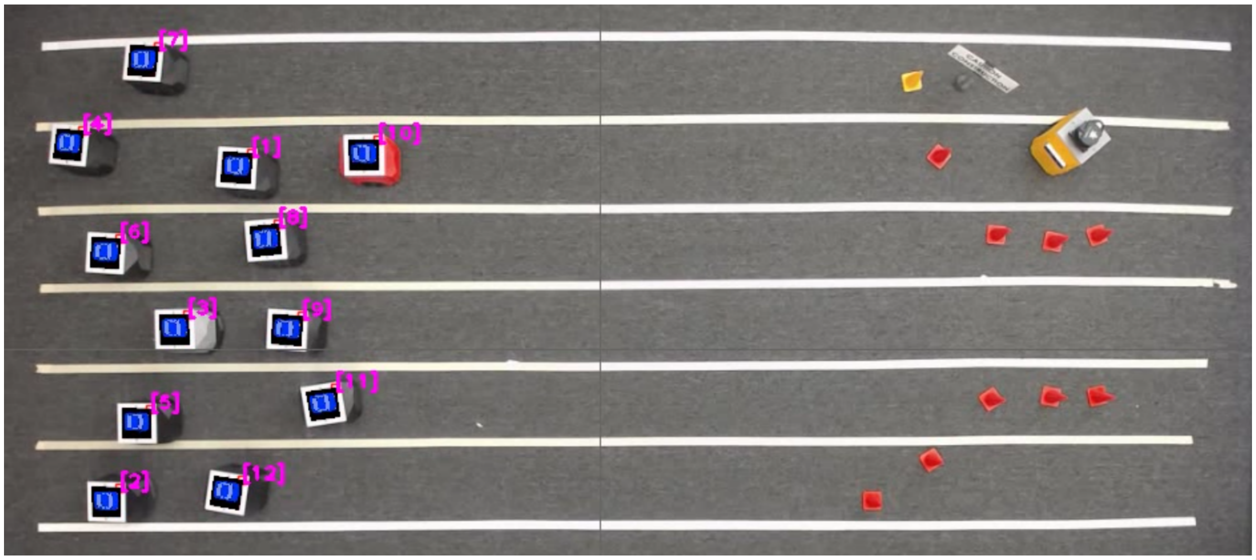
(c) Lateral vision.

Fig. A-7.: Implementation at iteration 40.

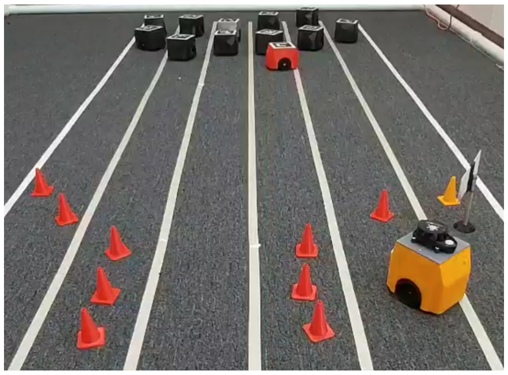
Finally, the agents can cross the obstacle and then look for its objective but deal with it beforehand. The white car could avoid the wrecked car and now is achieving its objective, **Fig. A-7**. The implementation shows the expected results in the simulations, and it is possible to be replicated in a laboratory.

A.3. Reduced Scenario

For the reduced scenario, we place some cones to represent a highway where any kind of construction or accident reduces the lanes. The six lanes are reduced just to one looking at an extreme change in its environment. This reduction represents a change in the possible solution to each agent, reducing $L = 6$ lanes to a $L = 1$, **Fig. A-8**. The main objective is to observe the behavior of the whole network in a challenging environment and how they deal with it. The yellow vehicle does not interact with each other and represents a construction vehicle in the scenario. It is not taken into account by the control system.



(a) Controller vision.


(b) Front vision.

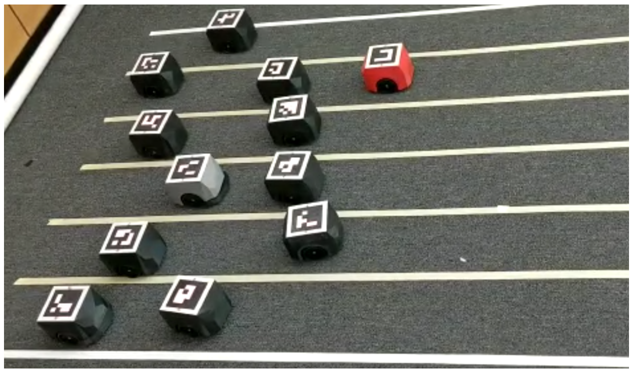
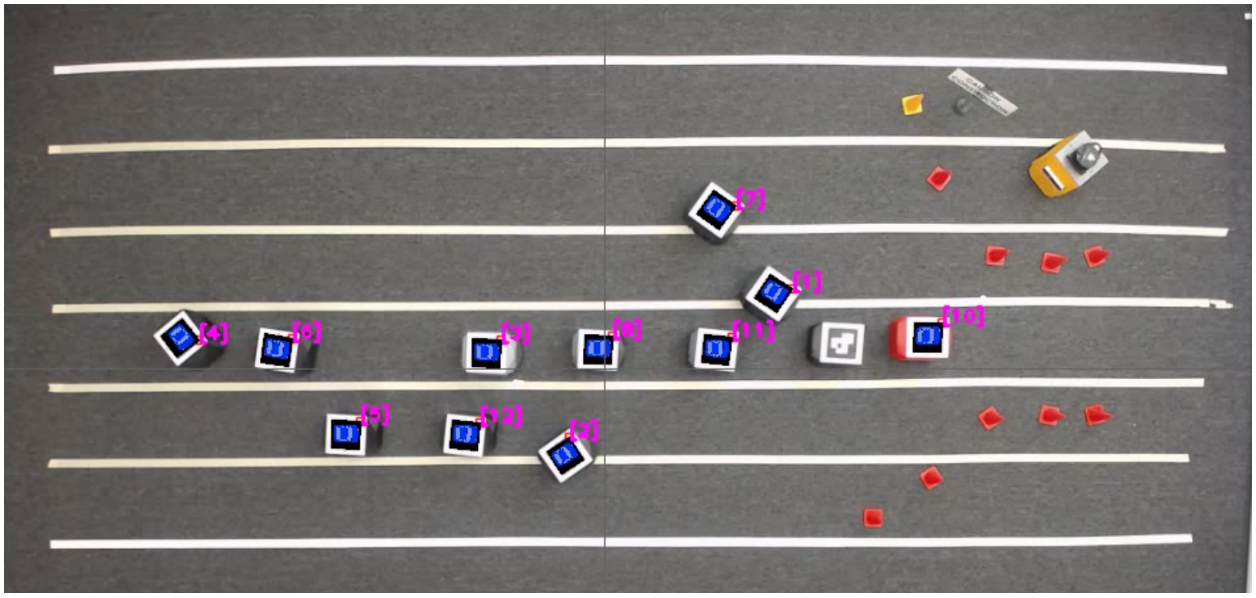
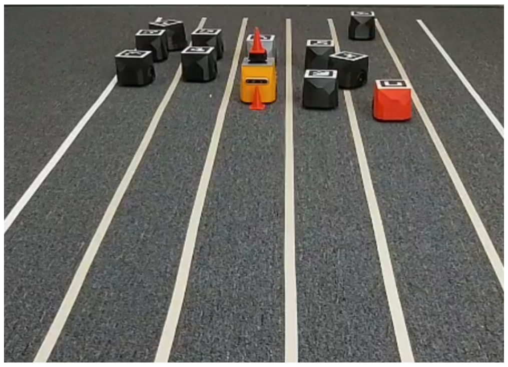


Fig. A-8.: Implementation at iteration 0.

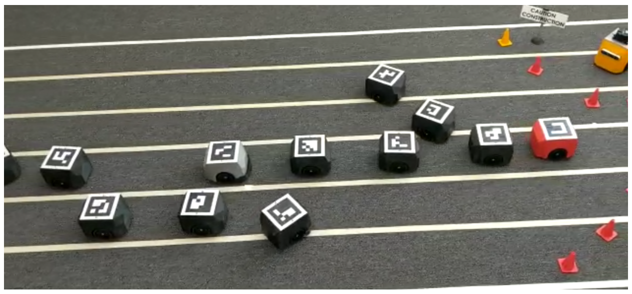
In the beginning, the vehicles are placed in the initial condition established in Chapter 6. The whole network will try to change to a lane $L = 3$, avoiding collisions and fulfilling the lane objective before crashing with the cones.



(a) Controller vision.



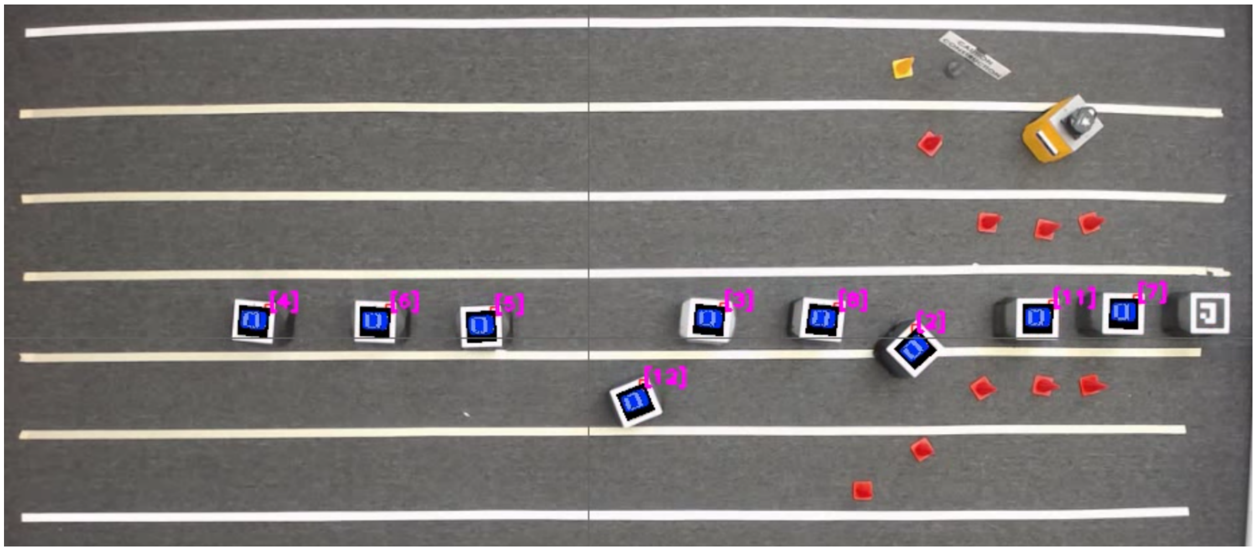
(b) Front vision.



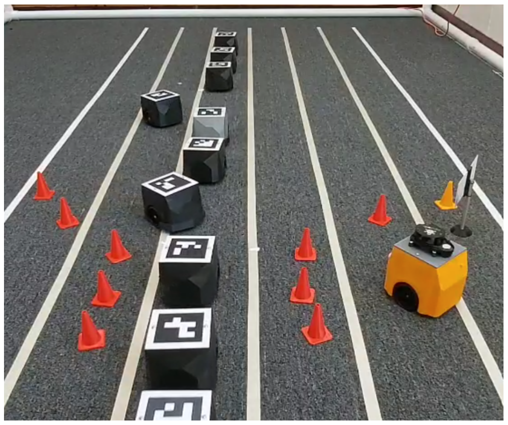
(c) Lateral vision.

Fig. A-9.: Implementation at iteration 10.

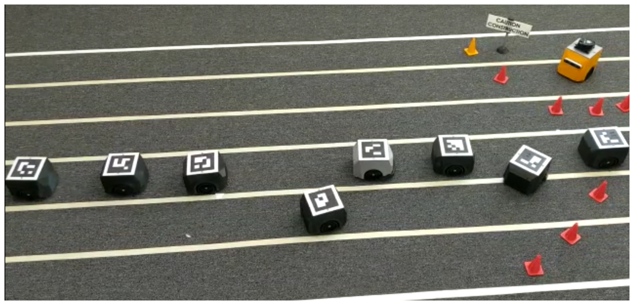
In iteration 10, some agents have managed to reach lane 3, **Fig. A-9**. However, some are already trying to get into it. Vehicle 1 is entering lane 3 thanks to vehicle 11 reducing its velocity allowing it to enter.



(a) Controller vision.



(b) Front vision.



(c) Lateral vision.

Fig. A-10.: Implementation at iteration 40.

Finally, the entire network achieves to get into lane three without collision. Although the scenario was challenging, the implementation showed a good result, **Fig. A-10**. The implementation had some problems with the vision system, but it could be solved, and the implementation was successful.

Bibliography

[1] Alrifaee, Bassam: *Networked Model Predictive Control for Vehicle Collision Avoidance*, Tesis de Grado, 05 2017

[2] Alrifaee, Bassam ; Ghanbarpour, Masoumeh ; Abel, Dirk: Centralized Non-Convex Model Predictive Control for Cooperative Collision Avoidance of Networked Vehicles, 2014

[3] Alrifaee, Bassam ; Maczijewski, Janis ; Abel, Dirk: Sequential Convex Programming MPC for Dynamic Vehicle Collision Avoidance, 2017

[4] Bauso, Dario: Game Theory: Models, Numerical Methods and Applications. En: *Foundations and Trends in Systems and Control* 1 (2014), 10, p. 379–522

[5] Bemporad, Alberto ; Morari, Manfred: Control of systems integrating logic, dynamics, and constraints. En: *Automatica* 35 (1999), Nr. 3, p. 407–427. – ISSN 0005–1098

[6] Bennett, S.: A brief history of automatic control. En: *IEEE Control Systems Magazine* 16 (1996), Nr. 3, p. 17–25

[7] Brock, Oliver ; Khatib, Oussama: High-Speed Navigation Using the Global Dynamic Window Approach. (2000), 01

[8] Carona, Ricardo ; Aguiar, A P. ; Gaspar, Jose: Control of unicycle type robots tracking, path following and point stabilization. (2008)

[9] Cesa-Bianchi, Nicolò ; Lugosi, Gábor: *Prediction, Learning, and Games.* 2006. – ISBN 978–0–521–84108–5

[10] Cheng, Shuo ; Li, Liang ; Chen, Xiang ; nan Fang, Sheng ; yu Wang, Xiang ; heng Wu, Xiu ; bing Li, Wei: Longitudinal autonomous driving based on game theory for intelligent hybrid electric vehicles with connectivity. En: *Applied Energy* 268 (2020), p. 115030. – ISSN 0306–2619

[11] Dogra, Anutusha ; Jha, Rakesh K. ; Jain, Shubha: A Survey on Beyond 5G Network With the Advent of 6G: Architecture and Emerging Technologies. En: *IEEE Access* 9 (2021), p. 67512–67547

[12] Dunbar, William: Distributed Receding Horizon Control of Multiagent Systems. (2004), 01

[13] Dunbar, William ; Murray, Richard: Distributed receding horizon control for multi-vehicle formation stabilization. En: *Automatica* 42 (2006), 04, p. 549–558

[14] Fabiani, Filippo ; Grammatico, Sergio: Multi-Vehicle Automated Driving as a Generalized Mixed-Integer Potential Game. En: *IEEE Transactions on Intelligent Transportation Systems* PP (2019), 03, p. 1–10

[15] Facchinei, F. ; Kanzow, Christian: Generalized Nash equilibrium problems. En: *Annals of Operations Research* 175 (2010), 02, p. 177–211

[16] Facchinei, Francisco ; Pang, Jong-Shi: Nash equilibria: The variational approach. En: *Convex Optimization in Signal Processing and Communications* (2009), 01. ISBN 9780511804458

[17] Facchinei, Francisco ; Piccialli, Veronica: Decomposition algorithms for generalized potential games. En: *Computational Optimization and Applications* 50 (2011), 10, p. 237–262

[18] Fudenberg, Drew ; Tirole, Jean: *Game Theory*. Cambridge, MA : MIT Press, 1991. – Translated into Chinesse by Renin University Press, Bejing: China.

[19] García, Carlos E. ; Prett, David M. ; Morari, Manfred: Model predictive control: Theory and practice—A survey. En: *Automatica* 25 (1989), Nr. 3, p. 335–348. – ISSN 0005–1098

[20] Garrido-Jurado, Sergio ; Muñoz-Salinas, Rafael ; Madrid-Cuevas, Francisco J. ; Marín-Jiménez, Manuel J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. En: *Pattern Recognition* 47 (2014), Nr. 6, p. 2280–2292

[21] Gehrig, S. K. ; Stein, F. J.: Dead reckoning and cartography using stereo vision for an autonomous car. En: *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)* Vol. 3, 1999, p. 1507–1512 vol.3

[22] Hespanha, Joao: *Noncooperative Game Theory: An Introduction for Engineers and Computer Scientists*. 2017. – ISBN 9781400885442

[23] Hu, J. ; Bhowmick, P. ; Arvin, F. ; Lanzon, A. ; Lennox, B.: Cooperative Control of Heterogeneous Connected Vehicle Platoons: An Adaptive Leader-Following Approach. En: *IEEE Robotics and Automation Letters* 5 (2020), Nr. 2, p. 977–984

[24] Jr, John ; Nash, John: Two-Person Cooperative Game. En: *Econometrica* 21 (1953), 02, p. 128–140

[25] Leyton-Brown, Kevin ; Shoham, Yoav: *Essentials of Game Theory: A Concise Multidisciplinary Introduction*. Vol. 2. 2008. – ISBN 978–1598295931

[26] Maestre, J.M.: *Distributed Model Predictive Control Based on Game Theory*, Tesis de Grado, 10 2010

[27] Marden, Jason R. ; Shamma, Jeff S.: Game Theory and Control. En: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), Nr. 1, p. 105–134. – ISSN 2573–5144

[28] Mayne, D.Q. ; Rawlings, J.B. ; Rao, C.V. ; Scokaert, P.O.M.: Constrained model predictive control: Stability and optimality. En: *Automatica* 36 (2000), Nr. 6, p. 789–814. – ISSN 0005–1098

[29] Mohseni, Fatemeh ; Frisk, Erik ; Åslund, Jan ; Nielsen, Lars: Distributed Model Predictive Control for Highway Maneuvers. En: *IFAC-PapersOnLine* 50 (2017), 07, p. 8531–8536

[30] Moler., Clever: Matlab Optimizacion Toolbox,. (2022)

[31] MYERSON, ROGER B.: *Game Theory: Analysis of Conflict*. Harvard University Press, 1991. – ISBN 9780674341166

[32] Nash, John: Equilibrium Points in N-Person Games. En: *Proceedings of the National Academy of Sciences of the United States of America* 36 (1950), 02, p. 48–9

[33] In: Nash, John: *THE BARGAINING PROBLEM*. 2020, p. 5–13. – ISBN 9780691011929

[34] Nisan, Noam ; Roughgarden, Tim ; Tardos, Éva ; Vazirani, Vijay: *Algorithmic Game Theory*. 2007

[35] Optimization., LLC G.: gurobi optimizer reference manual,. (2021)

[36] Osborne, Martin ; Rubinstein, Ariel: *A course in Game Theory*. Vol. 63. 1994

[37] Ospina Gaitan, Nestor ; Mojica-Nava, Eduardo ; Jaimes, L.G. ; Calderon, Juan: ARGroHBotS: An Affordable and Replicable Ground Homogeneous Robot Swarm Testbed. En: *IFAC-PapersOnLine* 54 (2021), 01, p. 256–261

[38] Sagratella, Simone: Algorithms for generalized potential games with mixed-integer variables. En: *Computational Optimization and Applications* 68 (2017), 12

[39] SAR, Tamer ; OLSDER, G.J.: *Dynamic Noncooperative Game Theory.* 1995

[40] SHAKEY, Peter H.: *the world's first mobile, intelligent robot.* 2015

[41] TAEIHAGH, Araz ; LIM, Hazel Si M.: Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks. En: *Transport Reviews* 39 (2018), Jul, Nr. 1, p. 103–128. – ISSN 1464–5327

[42] THRUN, S.: Toward robotic cars. En: *Commun. ACM* 53 (2010), p. 99–106

[43] VALENCIA, Felipe ; PATIÑO, Julian: Game Theory Based Distributed Model Predictive Control for a Hydro-Power Valley Control, 2013. – ISBN 9783902823397, p. 538–544

[44] WORTHMANN, Karl ; MEHREZ, Mohamed ; ZANON, Mario ; MANN, G.K.I. ; GOSINE, Ray ; DIEHL, Moritz: Model Predictive Control of Nonholonomic Mobile Robots Without Stabilizing Constraints and Costs. En: *IEEE Transactions on Control Systems Technology* 24 (2015), 10, p. 1–13

[45] YANG, Xue ; LIU, Jie ; ZHAO, Feng ; VAIDYA, Nitin: A Vehicle-to-Vehicle Communication Protocol for Cooperative Collision Warning., 2004, p. 114–123