FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Dynamic Obstacle Avoidance in Agricultural Robotics

Bárbara Freitas Fernandes



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Luís Carlos Feliz dos Santos

Second Supervisor: André Silva Aguiar

October 14, 2025

© Bárbara Freitas Fernandes, 2025

Resumo

Os robôs agrícolas modernos requerem navegação autónoma em ambientes dinâmicos, de modo a evitar obstáculos tais como máquinas em movimento, pessoas e animais. A prevenção eficaz de obstáculos dinâmicos é essencial para garantir a segurança, a continuidade da operação e a autonomia nessas configurações. Além dos perigos dinâmicos, os ambientes agrícolas muitas vezes apresentam terrenos irregulares e inclinados, o que aumenta a complexidade do planeamento e controlo do percurso.

Esta dissertação centra-se no desenvolvimento de um sistema robusto e adaptável para a prevenção de obstáculos dinâmicos, especificamente concebido para ser utilizado em cenários agrícolas. Estes ambientes são imprevisíveis e muitas vezes desestruturados, exigindo estratégias de navegação responsivas, capazes de se adaptar ao movimento de agentes próximos. O sistema proposto amplia a estrutura AgRobPP-CA, que foi originalmente desenvolvida para evitar obstáculos estáticos em terrenos com declives acentuados. Ao integrar o rastreamento dinâmico de obstáculos, a classificação de movimentos e o replaneamento de trajetórias com base em curvas Bézier, o sistema proposto permite que os robôs agrícolas reajam com segurança a perigos em movimento.

Esta investigação visa abordar lacunas críticas no domínio da robótica agrícola, fornecendo um sistema capaz de reagir em tempo real a obstáculos dinâmicos em terrenos agrícolas. Espera-se que os resultados contribuam para reduzir a intervenção humana, melhorar a fiabilidade operacional e apoiar o objetivo mais amplo de uma agricultura sustentável e orientada para a tecnologia. Foi realizado um processo de validação, envolvendo um ambiente simulado, para avaliar a robustez, precisão e eficácia das soluções propostas em condições agrícolas.

Abstract

Modern agricultural robots require autonomous navigation in dynamic environments, in order to avoid obstacles such as moving machinery, people, and animals. Effective dynamic obstacle avoidance is essential for ensuring safety, continuity of operation, and autonomy in these settings. In addition to dynamic hazards, agricultural environments often present irregular and sloped terrain, which adds complexity to path planning and control.

This dissertation focuses on the development of a robust and adaptive system for dynamic obstacle avoidance, specifically designed for deployment in agricultural scenarios. These environments are unpredictable and often unstructured, demanding responsive navigation strategies capable of adapting to the motion of nearby agents. The proposed system extends the AgRobPP-CA framework, which was originally developed for static obstacle avoidance in steep-slope terrains. By integrating dynamic obstacle tracking, motion classification, and Bézier curve-based path replanning, the proposed system enables agricultural robots to safely react to moving hazards.

This research aims to address critical gaps in the field of agricultural robotics by delivering a system capable of real-time responsiveness to dynamic obstacles in agricultural terrains. The outcomes are expected to contribute to reducing human intervention, improving operational reliability, and supporting the broader goal of sustainable, technology-driven agriculture. A validation process, involving a simulated environment, was conducted to assess the robustness, accuracy, and efficiency of the proposed solutions under agricultural conditions.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice. The specific Sustainable Development Goals mentioned have the following names:

SDG 2 End hunger, achieve food security and improved nutrition and promote sustainable agriculture

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all.

The Targets associated with the referred SDGs have the following names:

Target 2.4 By 2030, ensure sustainable food production systems and implement resilient agricultural practices that increase productivity and production, that help maintain ecosystems, that strengthen capacity for adaptation to climate change, extreme weather, drought, flooding and other disasters and that progressively improve land and soil quality.

Target 8.2 Achieve higher levels of economic productivity through diversification, technological upgrading and innovation, including through a focus on high-value added and labour-intensive sectors.

SDG	Target	Contribution	Performance Indicators and Metrics
2	2.4	Promotion of resilient agricultural practices using advanced robotic technologies.	Reduction in crop loss due to environmental factors and improved operational adaptability.
8	8.2	Advancement of technological innovation for sustainable farming operations.	Increase in the adoption rate of automation technologies by farmers.

Acknowledgements

I would like to thank my supervisor, Luís Santos, and my second supervisor, André Aguiar, for their guidance and help throughout the development of this dissertation. Their support was fundamental in developing this work. I am also thankful to my family and friends for their support during this time. A very special thank you goes to Ana Luísa, for being by my side every day, working, supporting, and keeping me company through it all. Another special thank you goes to Francisco for keeping me calm during stressful moments. Lastly, thank you to Milú, for keeping me company during long hours of work at home, always nearby.

Bárbara Freitas Fernandes

"The most effective way to do it, is to do it."

Amelia Earhart

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 2
- 1.3 Objectives . . . 2
- 1.4 Dissertation Structure . . . 3

2 State of the Art **5**
- 2.1 Collision Avoidance . . . 7
  - 2.1.1 Static Obstacle Avoidance . . . 7
  - 2.1.2 Dynamic Obstacle Avoidance . . . 9
- 2.2 Critical Review . . . 12

3 AgRobPP-CA **15**
- 3.1 Limitations . . . 19

4 Dynamic Obstacle Avoidance in AgRobPP-CA **21**
- 4.1 Software Structure . . . 21
- 4.2 Updated State Machine . . . 23
- 4.3 Dynamic Obstacle Avoidance Implementation . . . 24
  - 4.3.1 Dynamic Obstacle Mapping . . . 24
  - 4.3.2 Path Replanning . . . 27
- 4.4 Testing Environment and Simulation Tools . . . 28

5 Results **31**
- 5.1 Single Horizontal Obstacle from the Left . . . 31
  - 5.1.1 Robot Speed 0.4 m/s . . . 31
  - 5.1.2 Robot Speed 0.2 m/s . . . 34
  - 5.1.3 Robot Speed 0.8 m/s . . . 35
- 5.2 Single Horizontal Obstacle from the Right . . . 36
  - 5.2.1 Robot Speed 0.4 m/s . . . 36
  - 5.2.2 Robot speed 0.2 m/s . . . 38
  - 5.2.3 Robot Speed 0.8 m/s . . . 39
- 5.3 Vertical Obstacle Crossing Path . . . 40
  - 5.3.1 Robot speed 0.4 m/s . . . 40
  - 5.3.2 Robot speed 0.2 m/s . . . 42
  - 5.3.3 Robot speed 0.8 m/s . . . 43
- 5.4 Diagonal Obstacle Crossing Path . . . 44
  - 5.4.1 Robot speed 0.4 m/s . . . 44

5.4.2 Robot speed 0.2 m/s 46
5.4.3 Robot speed 0.8 m/s 47
5.5 Two Crossing Obstacles from Different Directions 48
5.5.1 Robot speed 0.4 m/s 49
5.5.2 Robot speed 0.2 m/s 52
5.5.3 Robot speed 0.8 m/s 53
5.6 Result Analysis 54

6 Conclusions and Future Work **57**

References **59**

List of Figures

2.1 Collision Avoidance for static and dynamic obstacles [26]. 9

3.1 AgRobPP-CA diagram [28]. 15
3.2 AgRobv16 robot referential frames configuration [28]. 16
3.3 Collision avoidance state machine diagram [28]. 16
3.4 Generation of alternative Bézier curve (green) for collision avoidance [28]. . . . 17
3.5 Gazebo simulation result with one obstacle at steep slope vineyard. [28]. 18
3.6 AgRobPP-CA result with one obstacle at steep slope vineyard. [28]. 18

4.1 UML Class Diagram. 22
4.2 Updated collision avoidance state machine diagram. 23
4.3 Local safety regions map. 25
4.4 Dynamic obstacle map. 26
4.5 Gazebo environment. 28
4.6 RViz environment. 29

5.1 Gazebo (left) and RViz (right) views for a single horizontal obstacle from the left. 32
5.2 Dynamic Obstacle Map for a single horizontal obstacle from the left. 32
5.3 Commanded linear and angular velocities during avoidance of horizontal obstacle from the left. (robot speed = 0.4 m/s). 33
5.4 Commanded linear and angular velocities during avoidance of horizontal obstacle from the left (robot speed = 0.2 m/s). 34
5.5 Commanded linear and angular velocities during avoidance of horizontal obstacle from the left (robot speed = 0.8 m/s). 35
5.6 Gazebo (left) and RViz (right) views for a single horizontal obstacle from the right. 36
5.7 Dynamic Obstacle Map for a single horizontal obstacle from the right. 36
5.8 Commanded linear and angular velocities during avoidance of horizontal obstacle from the right (robot speed = 0.4 m/s). 37
5.9 Commanded linear and angular velocities during avoidance of horizontal obstacle from the right (robot speed = 0.2 m/s). 38
5.10 Commanded linear and angular velocities during avoidance of horizontal obstacle from the right (robot speed = 0.8 m/s). 39
5.11 Gazebo (left) and RViz (right) views for a vertical obstacle crossing the robot's path. 40
5.12 Dynamic Obstacle Map for a vertical obstacle crossing the robot's path. 40
5.13 Commanded linear and angular velocities during avoidance of vertical obstacle (robot speed = 0.4 m/s). 41
5.14 Commanded linear and angular velocities during avoidance of vertical obstacle (robot speed = 0.2 m/s). 43

5.15 Commanded linear and angular velocities during avoidance of vertical obstacle (robot speed = 0.8 m/s). 44
5.16 Gazebo (left) and RViz (right) views for a diagonal obstacle crossing the robot's path. 44
5.17 Dynamic Obstacle Map for a diagonal obstacle crossing the robot's path. 45
5.18 Commanded linear and angular velocities during avoidance of diagonal obstacle (robot speed = 0.4 m/s). 46
5.19 Commanded linear and angular velocities during avoidance of diagonal obstacle (robot speed = 0.2 m/s). 47
5.20 Commanded linear and angular velocities during avoidance of diagonal obstacle (robot speed = 0.8 m/s). 48
5.21 Gazebo (left) and RViz (right) views for first obstacle avoidance. 49
5.22 Gazebo (left) and RViz (right) views for second obstacle avoidance. 49
5.23 Dynamic Obstacle Map showing avoidance of first obstacle. 50
5.24 Dynamic Obstacle Map showing avoidance of second obstacle from opposite direction. 50
5.25 Commanded linear and angular velocities during avoidance of two horizontal obstacles from opposite directions (robot speed = 0.4 m/s). 51
5.26 Commanded linear and angular velocities during avoidance of two horizontal obstacles from opposite directions (robot speed = 0.2 m/s). 52
5.27 Commanded linear and angular velocities during avoidance of two horizontal obstacles from opposite directions (robot speed = 0.8 m/s). 54

List of Tables

2.1 Comparison of Global Path Planning Algorithms [22] 8
2.2 Point-to-Point path planning applications in agriculture. 11

5.1 Test results for obstacle approaching horizontally from the left (robot speed = 0.4 m/s). 33
5.2 Test results for obstacle approaching horizontally from the left (robot speed = 0.2 m/s). 34
5.3 Test results for obstacle approaching horizontally from the left (robot speed = 0.8 m/s). 35
5.4 Test results for obstacle approaching horizontally from the right (robot speed = 0.4 m/s) 37
5.5 Test results for obstacle approaching horizontally from the right (robot speed = 0.2 m/s) 38
5.6 Test results for obstacle approaching horizontally from the right (robot speed = 0.8 m/s) 39
5.7 Test results for vertical obstacle crossing path (robot speed = 0.4 m/s) 41
5.8 Test results for vertical obstacle crossing path (robot speed = 0.2 m/s). 42
5.9 Test results for vertical obstacle crossing path (robot speed = 0.8 m/s). 43
5.10 Test results for diagonal obstacle crossing path (robot speed = 0.4 m/s) 45
5.11 Test results for diagonal obstacle crossing path (robot speed = 0.2 m/s) 46
5.12 Test results for diagonal obstacle crossing path (robot speed = 0.8 m/s) 47
5.13 Test results for two horizontal obstacles from opposite directions (robot speed = 0.4 m/s) 51
5.14 Test results for two horizontal obstacles from opposite directions (robot speed = 0.2 m/s) 52
5.15 Test results for two horizontal obstacles from opposite directions (robot speed = 0.8 m/s) 53

List of Acronyms

A*	A-Star
ACO	Ant Colony Optimization
AgRobPP	Agricultural Robotics Path Planning
AgRobPP-CA	Agricultural Robotics Path Planning - Collision Avoidance
CA	Collision Avoidance
CPP	Coverage Path Planning
DA-OCSORT	Depth Aware Observation Centric Simple Online and Realtime Tracking
DWA	Dynamic Window Approach
GA	Genetic Algorithm
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
ID	Identifier
IMU	Inertial Measurement Unit
INESC TEC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
LiDAR	Light Detection And Ranging
PRM	Probabilistic RoadMap
PSO	Particle Swarm Optimization
P2P	Point-to-Point
RGB	Red Green Blue
RGB-D	Red Green Blue - Depth
R-SAC	Residual-like Soft Actor Critic
RNN	Recurrent Neural Networks
RRT	Rapidly-exploring Random Tree
RRT*	Rapidly-exploring Random Tree Star
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization And Mapping
TTC	Time-To-Collision
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UML	Unified Modeling Language
2D	Two-Dimensional
3D	Three-Dimensional

Chapter 1

Introduction

1.1 Context

Agriculture is essential for the economic growth and development of any country. It is crucial to human survival, as it is the primary source of food production. However, the agricultural sector is currently undergoing a major transformation driven by a combination of demographic, economic, and environmental challenges. Among these challenges are the decreasing availability of agricultural labor [1], increasing demand for higher productivity, and the urgent need for sustainable practices to reduce environmental impact.

One of the most promising avenues for addressing these issues is the introduction of automation and robotics in agricultural operations. In recent years, agricultural robotics has evolved from simple mechanization tools to complex, intelligent systems capable of performing a wide range of tasks, including planting, monitoring, harvesting, and transporting goods [2]. Autonomous navigation is at the core of these capabilities, enabling robots to move through fields, orchards, or vineyards without constant human supervision.

Despite this progress, the deployment of autonomous systems in real-world agricultural environments remains highly challenging. Unlike structured industrial settings, agricultural fields are inherently unstructured and dynamic. The presence of moving machinery, animals, workers, and shifting environmental conditions (such as weather or terrain erosion) creates a highly unpredictable environment. In such conditions, static path planning methods become insufficient, and the need for dynamic obstacle avoidance becomes crucial.

Additionally, the physical characteristics of agricultural terrains vary widely depending on the type of crop and geographic location. Navigating the terrain safely and efficiently requires not only robust mechanical design but also intelligent software that can adapt in real time to changing conditions and obstacles.

Dynamic obstacle avoidance in this context involves more than simply detecting and steering around objects. It requires the ability to interpret environmental changes and react accordingly in real time. The robot must make quick and context-sensitive decisions, adjusting its path, speed, and orientation while continuing to follow mission objectives such as reaching a target location or

staying on a designated route. This is particularly important on sloped terrain, where maintaining stability is essential to prevent tipping or slipping.

1.2 Motivation

This dissertation is motivated by the need to adapt existing robotic navigation systems to meet the complex demands of dynamic and challenging agricultural environments. While considerable advancements have been made in path planning and control algorithms for autonomous robots, many of these systems are designed for relatively flat, predictable environments. Agricultural fields are far from predictable.

Dynamic obstacle avoidance is one of the most critical capabilities that an agricultural robot must possess to function autonomously and safely. In many practical scenarios, obstacles such as moving vehicles, workers, or animals may suddenly appear within the robot's operational path. These obstacles are not always stationary or easily predictable in their movements, which requires the robot to continuously monitor its environment. This reactive capability is essential to prevent collisions, ensure the safety of humans and animals nearby and protect the integrity of the robotic system itself.

In steep-slope environments, the challenge is further compounded. The robot must not only detect and avoid dynamic obstacles, but it must also take into account the risk of tipping, slipping, or getting stuck. This requires a tight integration of perception, planning, and control, where sensor data must be processed in real time to produce smooth and feasible trajectories that consider both the location of dynamic obstacles and the geometry of the terrain.

In regions with labor shortages, such systems can help maintain or even increase productivity without requiring large numbers of human workers. In hazardous environments, they can reduce the risk of injury by taking over tasks that are physically demanding or dangerous.

1.3 Objectives

The primary goal of this dissertation is to develop, implement, and validate a robust system for dynamic obstacle avoidance in agricultural robotics. This work aims to enhance the autonomy and safety of agricultural robots by enabling them to perceive, interpret, and respond in real time to dynamic elements in their environment, such as moving workers, animals, and agricultural machinery.

To achieve this goal, the following specific objectives have been defined:

- **Literature Review and Problem Characterization:** Conduct an in-depth review of existing research and technologies related to static and dynamic obstacle avoidance in mobile robotics, with a special focus on agricultural applications. This includes identifying the limitations of current solutions in dynamic environments.

- **Development of Dynamic Obstacle Avoidance Algorithm:** Design and implement an algorithm capable of detecting and avoiding dynamic obstacles in real time. The algorithm should integrate perception (e.g., LiDAR or camera-based), path planning, and control strategies that are adaptable to dynamic changes in the environment.
- **Terrain-Aware Navigation and Control:** Ensure that the obstacle avoidance strategy is adapted to the characteristics of steep and uneven terrain. This includes maintaining robot stability, considering terrain inclination, and integrating motion constraints into the planning system.
- **Simulation Testing:** Validate the developed algorithm in a simulated environment that accurately represents an agricultural terrain. Use ROS2 and simulation tools like Gazebo to test various dynamic scenarios.

By following these structured objectives, this dissertation aims to develop a practical and deployable solution to a pressing problem in agricultural robotics. The expected outcomes include a robust navigation system capable of operating safely and autonomously in dynamic environments, contributing to the broader goal of advancing sustainable and intelligent agricultural practices.

1.4 Dissertation Structure

In addition to the introduction, this dissertation is divided into five chapters. Chapter 2 reviews relevant literature on path planning and obstacle avoidance, with a focus on agricultural robotics. Chapter 3 presents the AgRobPP-CA framework, describing its architecture and approach to static obstacle avoidance using Bézier curves. Chapter 4 outlines the modifications introduced to enable dynamic obstacle avoidance, including real-time tracking, path replanning and simulation tools. Chapter 5 presents and analyzes the results obtained from tests carried out in simulated dynamic scenarios. Chapter 6 summarizes the contributions of this work and proposes future improvements.

Chapter 2

State of the Art

Path planning is essential for autonomous robot navigation, allowing robots to calculate and follow paths that ensure safe, efficient movement without collisions. To successfully implement dynamic obstacle avoidance, it is necessary to understand the fundamental concepts of path planning.

In terms of the environment knowledge, path planning can be divided into two main types [3]: global path planning, which relies on a complete map for navigating known environments, and local path planning, which adapts to partially or fully unknown, dynamic environments. Global path planning involves a predefined map and is often used for environments where the layout is static and known, whereas local path planning adapts in real-time using sensors, making it suitable for unpredictable and dynamic environments.

Agricultural environments may be unpredictable and dynamic, with challenges like irregular terrain, steep slopes and unexpected obstacles. These factors require specialized path planning techniques capable of adapting to continuously changing landscapes. A literature review under the subject of path planning for ground robots in agriculture [4] provides a comprehensive review of various path planning techniques.

Path planning approaches can be classified into two main categories: Point-to-Point path planning, which focuses on finding a path between a starting point and a goal point, and Coverage path planning, which aims to cover a defined area. Depending on the tasks that the robot needs to accomplish, there are various algorithms that can be used.

In Point-to-Point (P2P) path planning, the Potential Field Method creates an artificial field where obstacles exert repulsive forces, and the goal applies an attractive force. It is useful for obstacle avoidance in real-time navigation through fields, although a common problem is the existence of local minimums, which happens when the repulsive potential becomes stronger than the attractive potential, preventing the robot to reach its goal [5]. The A* (A-Star) graph-based search algorithm guarantees an optimal path, by evaluating the actual cost from the start to a node and an estimated cost from that node to the goal. The Dijkstra's Algorithm is a graph-based search method that finds the shortest path from a given source node to every other node. Even though the A* and Dijkstra algorithms are similar in their goal of finding the shortest path, A* introduces a heuristic to guide its search, making it more efficient in many cases [6]. These two algorithms

can be adapted for agricultural row navigation in orchards or vineyards, where the robot must efficiently move from one row to another while avoiding obstacles. The Rapidly-exploring Random Tree (RRT) sampling-based algorithm quickly explores large spaces [7], which is suitable for path planning in large, irregular fields where robots may encounter unexpected terrain features. The Genetic Algorithm (GA) is a nature-inspired optimization technique that iteratively "evolves" a path by selecting and refining potential routes to improve overall performance. It is often used to optimize paths across irregular terrains or to navigate through specific waypoints, points in space that guide the movement of an autonomous robot [8]. The Particle Swarm Optimization (PSO) algorithm is inspired by the movement of swarms and adjusts potential paths based on collective "swarm" intelligence to find efficient solutions. The Ant Colony Optimization (ACO) algorithm mimics the behavior of ants in finding the shortest route. These two algorithms can be modified to find paths in uneven terrain or navigate complex field layouts by simulating natural movement [4].

In Coverage path planning (CPP) [4], the Exact Cellular Decomposition [9] breaks the space into cells, each representing a portion of the area to be covered. The Boustrophedon Cellular Decomposition algorithm [9] is a variation of cellular decomposition that generates zig-zag patterns. These two algorithms are commonly used in agricultural fields for tasks like spraying or seeding, where each "cell" represents a segment of the crop field or row [9]. Greedy Algorithms partition the area into smaller, manageable sections, aiming to minimize overlap and maximize efficiency by covering each sub-region in sequence. The Spiral Algorithm is commonly used in simpler coverage tasks, such as fields with uniform layouts, to achieve full coverage with minimal complexity. These two algorithms can be applied to cover large fields or greenhouses systematically. Some algorithms, like A*, GA, and ACO, can serve both Point-to-Point and Coverage path planning but are adapted based on the task's objectives and constraints [10].

Selecting an appropriate path planning algorithm in agriculture depends significantly on the specific task to be performed, whether it is moving efficiently between points or ensuring full coverage of an area. For dynamic environments, such as those involving moving workers or machinery, P2P planning approaches are generally better suited. Algorithms like A*, Dijkstra, and Rapidly-exploring Random Trees (RRT) are capable of computing efficient paths from a starting point to a goal location while adjusting dynamically to new obstacles detected through sensors like LiDAR or ultrasonic systems. These capabilities are particularly valuable in unstructured agricultural settings, where robots must respond in real time to changing conditions and dynamic obstacles.

In contrast, CPP algorithms, such as Boustrophedon Cellular Decomposition or the Spiral Algorithm, are designed for systematic coverage of areas and are more appropriate for tasks like spraying, seeding, or mowing. While effective in structured fields, these approaches are less flexible in environments that demand immediate trajectory adjustments. In agricultural settings where the priority is navigating from one location to another, rather than covering every square meter of land, P2P algorithms provide a more practical and effective solution. Real-world testing remains crucial for validating and optimizing these planning strategies in the context of agricultural

applications.

Table 2.2 presents the main ideas for each work mentioned in this chapter. While significant progress has been made in static obstacle avoidance, dynamic obstacle avoidance in agricultural robotics remains under explored. Most existing systems are designed for static environments, where obstacles do not change position during navigation. In contrast, dynamic obstacles require real-time perception, prediction, and path re-adjustment, which imposes additional computational and sensor accuracy challenges.

2.1 Collision Avoidance

While autonomous robots are well explored in other fields, their application in agriculture still remains challenging. Irregular terrains and unpredictable obstacles make it difficult to develop effective obstacle avoidance algorithms that work in these conditions. As a result, research and findings in this specific area are still limited, with only a few studies focusing on collision avoidance in agricultural robotics.

The main issue is creating safe, real-time path adjustments for autonomous robots navigating uneven, sloped areas, which often disrupt Global Navigation Satellite Systems (GNSS) and make traditional path planning approaches difficult.

To address the complexity of navigating in such environments, obstacle avoidance strategies are typically divided into two categories: static and dynamic. Static obstacle avoidance deals with fixed elements in the environment, whose positions do not change over time. In contrast, dynamic obstacle avoidance requires predicting and responding to moving obstacles. The following subsections detail both approaches and their relevance to agricultural robotics.

2.1.1 Static Obstacle Avoidance

In static environments, global path planning is commonly performed using pre-defined GNSS waypoints [11] [12] [13] [14] [15] [16]. GNSS provides precise global localization data, allowing autonomous robots to accurately follow these planned trajectories in open-field agricultural settings, where line-of-sight to satellites is unobstructed and environmental conditions remain stable. These systems allow for pre-mapped trajectories to be executed without requiring frequent recalculations, reducing computational demands and enabling efficient large-scale field coverage. Additionally, GNSS integration supports repeatable and predictable navigation, which is essential for tasks such as seeding, spraying, and harvesting, where precision and consistency across paths are critical. However, their effectiveness diminishes in occluded environments, such as greenhouses or beneath thick vegetation, where signal accuracy and reliability can be compromised.

To overcome these limitations in environments where a prior map is unavailable or GNSS signals are unreliable, advanced techniques such as Simultaneous Localization and Mapping (SLAM) can be employed. SLAM systems enable the robot to build a map of the environment while simultaneously tracking its position within it, making them especially useful in GNSS-denied areas.

A notable example of SLAM tailored for agricultural use is VineSLAM [17], which produces reliable 3D maps of vineyard rows for navigation and planning purposes. These maps serve as a foundation for subsequent planning tasks, where global path planning algorithms like RRT* [18], [19], A* [19], [20], and Dijkstra [21] can be used to establish clear, obstacle-free trajectories. These algorithms are particularly effective in determining optimal or near-optimal paths between a starting location and a target, taking into account both terrain features and static obstacles. Table 2.1 summarizes the most commonly used global path planning algorithms, highlighting their strengths and weaknesses.

Table 2.1: Comparison of Global Path Planning Algorithms [22]

Algorithm	**Advantage**	**Disadvantage**
Dijkstra	• Visits all nodes • Optimal solution	• Take a long time
A*	• Visits some nodes • Fast navigation speeds • Infinite space exploration • Optimal solution	• Possibility of path navigation failure
RRT	• Small amount of calculation • Navigates complex environments	• Sample dependencies • Unpredictable
RRT*	• Optimality • Uniformity • Navigates complex environments	• Sample dependencies • Slow convergence speed • Unpredictable

In agricultural environments, where global path planning alone is insufficient, local path planning algorithms play a crucial role in ensuring precise and adaptive navigation. These algorithms are primarily responsible for real-time decision making, allowing robots to respond dynamically to immediate obstacles and terrain variations that may not have been accounted for in the global plan. A predominant approach in agricultural robotics is row-following navigation, where robots are designed to autonomously track and navigate along crop rows or predefined pathways. Robots collect data from image-based sensors, LiDAR, or depth-based cameras.

Image-based row-following [23] [12] [24] [25] relies on camera systems to detect and follow visual markers, crop row edges, or distinctive patterns in the environment. These systems use computer vision algorithms and, in many cases, deep learning models to process the captured images, extract relevant features, and guide the robot along the desired path.

LiDAR-Based Row Following [15] [16] uses laser beams to create precise 3D maps of the robot's surroundings. In row-following applications, LiDAR detects row boundaries, obstacles, and terrain variations, enabling precise alignment and navigation.

Depth-Based Row Following [14] uses stereo cameras, ToF or RGB-D cameras to capture depth information for perception of spatial details in the environment. These systems help robots detect obstacles, measure row spacing, and ensure accurate navigation along predefined pathways.

For non row following tasks, alternative algorithms are used in order to achieve the robot's objectives effectively.

2.1.2 Dynamic Obstacle Avoidance

Dynamic obstacle avoidance is crucial for autonomous robots to safely navigate around moving objects in real time. Unlike static obstacle avoidance, this approach must account for moving objects, such as workers, animals, or other machinery, which are common in dynamic agricultural settings. However, in agriculture, dynamic obstacle avoidance is less widely adopted, partly due to the complexity and unpredictability of open field. In these dynamic environments, sensors are essential for enabling autonomous robots to detect, track, and respond to moving obstacles in real time, ensuring safe navigation and avoiding collisions despite the challenges posed by uneven and dynamic environments.

Figure 2.1 illustrates the architecture of a system for collision avoidance, which integrates static and dynamic obstacle avoidance.

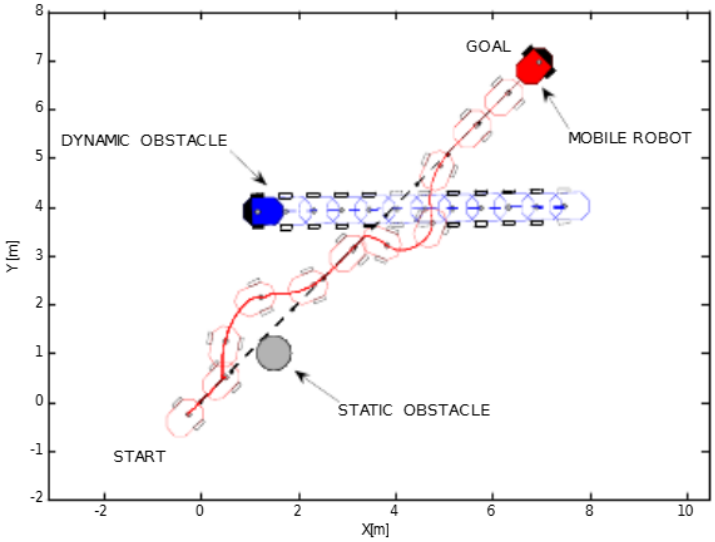


Figure 2.1: Collision Avoidance for static and dynamic obstacles [26].

Dynamic obstacle avoidance has not been as extensively studied in agricultural robotics as static obstacle avoidance. This is because real-time path planning in dynamic and unpredictable agricultural environments presents significant challenges.

For the specific application of autonomous navigation in steep-slope terrains, the challenges of dynamic obstacle avoidance become even more pronounced. Navigation in steep slopes presents more difficulties in maintaining stability and collision-free movement.

To address obstacle avoidance and waypoint navigation challenges, trajectory planning algorithms based on Bézier curves, Dubins curves, and Reeds-Shepp curves have been widely applied [27]:

- **Bézier Curves**: Known for generating smooth trajectories, Bézier curves are essential for stable navigation in agricultural settings. However, they are limited by their unidirectional movement and the complexity involved in accounting for the minimum turning radius.

- **Dubins Curves**: These curves consider the minimum turning radius, making them suitable for navigating constrained spaces. Nevertheless, they are unable to support reverse motion.
- **Reeds-Shepp Curves**: By enabling reverse motion, Reeds-Shepp curves help minimize path length when necessary. However, their disadvantages include high computational complexity and the potential to create longer overall paths.

Curve-based trajectory planning algorithms are particularly well-suited for dynamic obstacle avoidance because they allow the generation of adaptable paths.

In this context, the iterative Bézier curve-based approach [28] offers a promising solution. Bézier curve-based trajectory planning algorithms are particularly well-suited for dynamic obstacle avoidance in steep-slope environments because of their ability to generate smooth and continuous paths. In steep terrains, smooth trajectories are critical to ensuring the robot's stability and minimizing abrupt changes in movement, which could lead to instability or collisions, especially when navigating irregular surfaces.

Table 2.2: Point-to-Point path planning applications in agriculture.

Agricultural Application	Path Planning Approach		Dynamic or Static Obstacles	Real-World Testing	Sensors Used	Limitations	Ref.
	Global	Local					
Navigation in constrained headland spaces	Classical turning patterns or hybrid A* algorithm	Gradient-based trajectory optimization	Static	Yes	RTK-GNSS, IMU and wheel odometer	Limited to rigidly connected implements	[20]
Tractor navigation across fields	Waypoint-based global trajectory planning	Minimum snap algorithm	Static	Yes	GNSS, IMU	N/A	[11]
Greenhouse cucumber-picking	N/A	Row following (image based)	Static	Yes	Industrial camera, micro-controller	Uneven lighting can cause inconsistencies in image segmentation	[23]
Navigation in vineyard rows	Waypoint-based global trajectory planning	Row following (image based)	Static	Yes	RGB-D Camera, CPU-based inference, IMU	Limited to 5 Hz for control outputs and extreme lighting differences can affect image segmentation accuracy	[12]
Cooperative use of UAVs and UGVs in vineyards on sloped terrain	RRT*	DWA	Static	Yes	UAV: Multispectral Camera, RTK-GPS, IMU; UGV: Ultrasonic sensors, IMU	Limited to small-scale scenarios and may require adaptation for larger commercial applications	[18]
Precision agriculture	Waypoint-based global trajectory planning	Depth-based obstacle avoidance	Static	Yes	Depth camera, tracking camera, LiDAR, GPS	Cannot fully replace the expensive commercial solutions, but can be a supplementary	[13]
Navigation in paddy fields	N/A	Row following (image based)	Static	Yes	Camera, industrial onboard computer	Model was trained on only 350 images and restriction of the tractor's speed to 0.5 m/s	[24]
Navigation in narrow pathways	RRT* and A*	E-band and Reeds-Sheppcurve	Static	Yes	LiDAR, depth camera, onboard controller	May struggle in zigzag narrow spaces	[19]
Under-canopy navigation	RTK-GPS	Row following (depth based)	Static	Yes	ToF Camera, RTK-GPS, IMUs, encoders	ToF camera has a short measurement range and narrow horizontal field of view	[14]
Spraying robot for precision agriculture	N/A	Row following (LiDAR based)	Static	No	Horizontal LiDAR, vertical LiDAR	Design requires two separate LiDAR systems	[29]
Navigation in greenhouses	Dijkstra	DWA	Static	Yes	3D LiDAR, 2D LiDAR, IMU, encoders	Obstacle avoidance relies on preset parameters, which might not adapt well to unexpected dynamic changes	[21]
Navigation in vineyards	RTK-GNSS	Row following (LiDAR based)	Static	No	2D LiDAR, odometry	Requires a minimum number of LiDAR-detected points to begin trajectory estimation	[15]
Navigation for cotton crops	N/A	Row following (image based)	Static	Yes	Camera, software environment	Limited applicability to unstructured or heavily weeded fields	[25]
Navigation in orchards	Waypoint-based global trajectory planning	Row following (LiDAR based)	Static	Yes	RTK-GNSS, LiDAR, odometer, communication systems	The Wi-Fi-based system may face challenges in larger areas	[16]

Table 2.2 (continued)

Navigation in steep slope terrains	A*	Bézier curve-based approach	Static	No	LiDAR	Tested only in simulation	[28]
Navigation in orchards	N/A	DA-OCSORT algorithm	Dynamic	Yes	Stereo cameras, IMU	Rapid camera movements in uneven terrains can lead to tracking inaccuracies	[30]
Navigation in unstructured environments	PRM and Dijkstra	RNN based obstacle avoidance	Dynamic	Yes	LiDAR, RGB cameras, GNSS, IMU	Limited field of view creates blind spots and real-time fusion of 2D and 3D data incurs computational overhead	[31]
Navigation in smart farms	Dijkstra, RRT, A*, RRT*	DWA	Dynamic	Yes	LiDAR, IMU	N/A	[22]
Navigation in farmland	N/A	R-SAC	Dynamic	No	RGB-D cameras, multi-sensor fusion	Tested only in simulated environments	[32]
Navigation in orchards and vineyard-like settings	N/A	DWA enhanced with TD3 reinforcement learning	Dynamic	Yes	Stereo camera, RTK-GPS	Assumes linear motion of obstacles	[33]
Navigation in structured farmland	Predefined navigation path	Double Deep Q-Network reinforcement learning	Dynamic	Yes	Stereo camera, RTK-GPS, GPU-based laptop	Limited to straight paths, only velocity is controlled, not steering	[34]

2.2 Critical Review

The field of agricultural robotics has seen significant advancements in recent years, particularly in the development of autonomous systems capable of performing repetitive or hazardous tasks. Despite this progress, dynamic obstacle avoidance in steep-slope agricultural settings remains a relatively underexplored area. Most systems documented in the literature focus primarily on static obstacle avoidance, leveraging techniques such as GNSS-based navigation, LiDAR-based mapping, and predefined trajectory planning. While effective in controlled or relatively predictable environments, these approaches often fall short when applied to more dynamic scenarios commonly encountered in real-world agriculture, as they lack the adaptability required to react to moving obstacles and terrain variability.

One notable contribution in this area is the AgRobPP-CA framework [28], which builds upon the AgRobPP path planning system by introducing local, real-time obstacle avoidance using Bézier curves for adaptable path generation in response to local sensor observations. The architecture includes several key components, such as a dynamic safety zone, elevation mapping, and iterative path replanning based on parametric curves. These features make it highly suitable for dealing with irregular, sloped agricultural environments. However, while AgRobPP-CA provides a solid foundation for dealing with static obstacles, it exhibits several limitations when applied to dynamic contexts.

The system's reactive nature and its reliance on instantaneous sensor data limit its effectiveness in dealing with moving obstacles, such as workers, animals, or machinery. These dynamic entities require not only detection but also tracking and prediction to ensure safe navigation decisions. AgRobPP-CA currently lacks the ability to anticipate obstacle motion, which can result in delayed or suboptimal avoidance maneuvers. Moreover, steep inclines or terrain irregularities are occasionally misclassified as obstacles, leading to unnecessary trajectory changes or even failures to proceed. This limitation highlights the need for more sophisticated terrain classification methods that can distinguish between legitimate hazards and navigable terrain features. This dissertation builds upon the AgRobPP-CA framework. While the original framework, described in Chapter 3, provides a solid foundation for local obstacle avoidance, it was primarily designed for static environments and lacks mechanisms for handling dynamic obstacles effectively. The proposed work, detailed in Chapter 4, extends the system to support robust dynamic obstacle avoidance in real-time.

Chapter 3

AgRobPP-CA

This chapter presents the AgRobPP-CA framework [28], an independent collision avoidance framework based on real-time local observations and parametric Bézier curves, that define a curve based on several control points. It builds on the capabilities of AgRobPP by introducing real-time local obstacle avoidance using sensor data and Bézier curve-based trajectory generation.

While AgRobPP-CA addresses real-time collision avoidance, AgRobPP operates as a path planning tool that generates safe trajectories based on a complete map of the environment. It uses an A* algorithm to calculate paths, taking into account the robot's center of mass and the terrain's elevation.

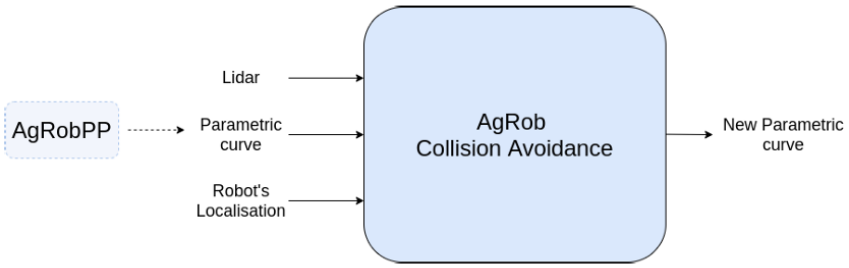


Figure 3.1: AgRobPP-CA diagram [28].

Figure 3.1 shows that AgRobPP-CA has three inputs: the LiDAR observations, the parametric curve that the robot is following, and the robots localization. Aditionally, AgRobPP-CA is composed by the following steps: definition of local safety zone, prior obstacle detection, construction of local elevation map, detection of dangerous slopes and an alternative path for collision avoidance.

The referential frame configuration of the AgRobv16 robot is represented in Figure 3.2, where M is the map frame, S is the sensor frame, B the base frame and F the footprint frame.

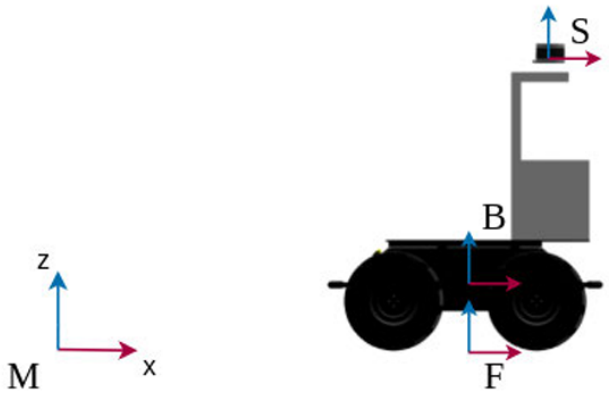


Figure 3.2: AgRobv16 robot referential frames configuration [28].

The collision avoidance process operates as a state machine with four states, as represented in Figure 3.3:

- **Waiting for Path:** The robot halts and awaits the reception of a parametric curve.
- **Following Original Path:** The robot navigates along the received path until an obstacle is detected on the robot path.
- **Recalculate Path:** If an obstacle is detected, an alternative trajectory is generated using Bézier curves. If recalculation fails, the robot returns to the "Waiting for Path" state.
- **Following Alternative Path:** If a valid trajectory is found, the robot switches to the alternative path and resumes operation.

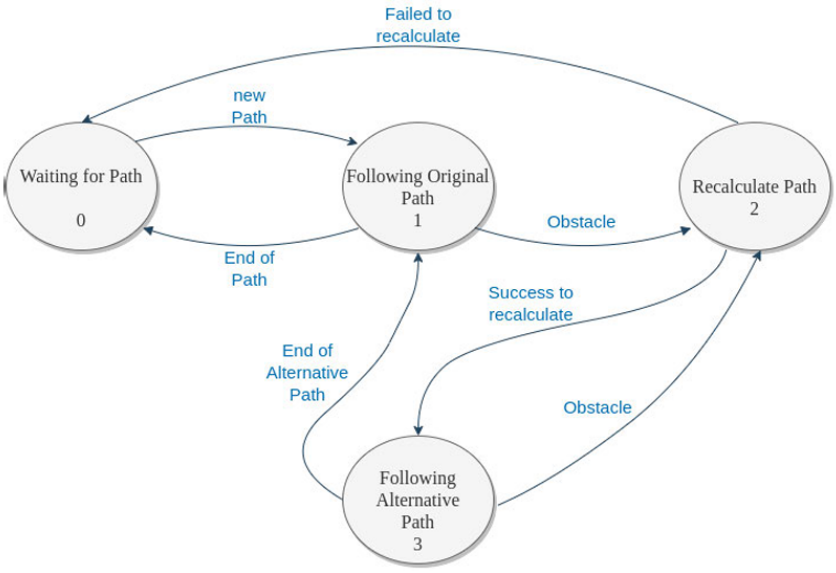


Figure 3.3: Collision avoidance state machine diagram [28].

AgRobPP-CA generates a 4th-degree Bézier curve as the alternative trajectory. Figure 3.4 illustrates the five control points used to construct the curve: P_0, P_1, P_c, P_3, and P_4. P_0 represents the robot's current position (origin of the base frame), while P_c denotes the predicted collision point. P_4 marks the end of the avoidance maneuver and is placed at a distance equal to the planned trajectory length beyond P_c. The intermediate control points P_1 and P_3 ensure smooth transitions and are calculated based on trajectory angles and distances. The midpoint P_m defines the deviation and serves as a reference for positioning the central control point P_2, which influences the curve's shape.

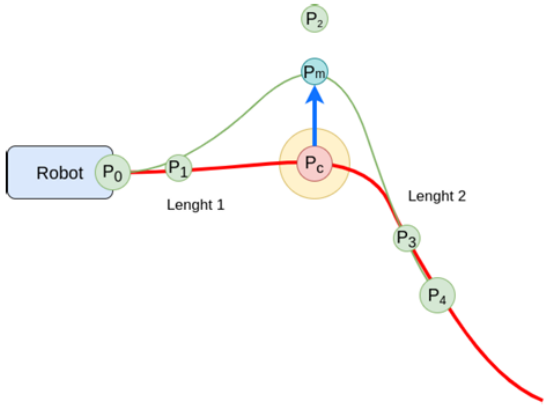


Figure 3.4: Generation of alternative Bézier curve (green) for collision avoidance [28].

The midpoint P_m plays a critical role in shaping the avoidance trajectory and is determined iteratively as it follows:

1. Generate a perpendicular vector to the robot path at P_c with an initial length equal to the robot's width.
2. Place P_m at the vector's extremity and use it to calculate P_2 (the midpoint control point).
3. Generate a new Bézier curve and verify the curve's validity within the safety region.
4. If invalid, rotate the perpendicular vector 180° and reattempt.
5. If still invalid, increment the vector's length and repeat until a valid curve is found or the distance between P_m and P_c exceeds 2 meters.

Once the final control points are defined, the Bézier curve is converted to a polynomial form suitable for the robot's trajectory controller. This parametric curve is computed using Equation 3.1:

$$\begin{aligned} P(t) = {} & t^4(P_0 - 4P_1 + 6P_2 - 4P_3) \\ & + t^3(-4P_0 + 12P_1 - 12P_2 + 4P_3) \\ & + t^2(6P_0 - 12P_1 + 6P_2) \\ & + t(4P_1 - 4P_0) + P_0, \quad t \in [0,1] \end{aligned} \tag{3.1}$$

where t is the curve parameter that varies from 0 to 1, representing the normalized progression along the trajectory from the start point P_0 to the endpoint P_4.

To better understand the baseline functionality of the original AgRobPP-CA framework, Figures 3.5 and 3.6 illustrate simulation results from prior work [28]. These results demonstrate the system's ability to avoid static obstacles using Bézier-based trajectory generation in challenging environments, such as steep-slope vineyards.

Figure 3.5 presents a Gazebo simulation scenario in which the robot must navigate a vineyard with one static obstacle placed in its path. The robot detects the obstacle and deviates accordingly using the static local path replanning strategy.

Figure 3.6 shows the RViz visualization of this avoidance behavior. While effective for static obstacles, this system does not account for moving objects, such as workers or animals, which is the key limitation addressed in this dissertation.



Figure 3.5: Gazebo simulation result with one obstacle at steep slope vineyard. [28].

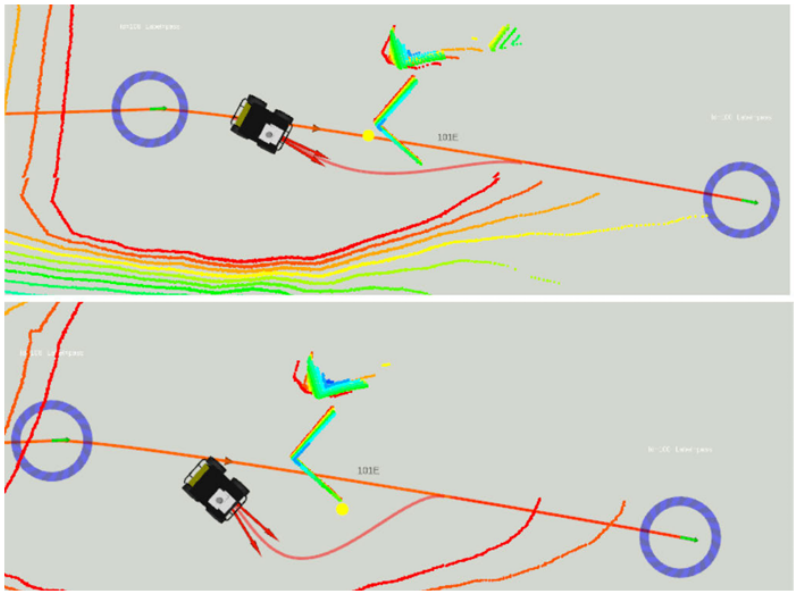


Figure 3.6: AgRobPP-CA result with one obstacle at steep slope vineyard. [28].

3.1 Limitations

Although AgRobPP-CA provides a foundational framework for real-time local obstacle avoidance in agricultural environments, it presents several limitations that motivated the work developed in this dissertation.

One significant architectural limitation is that the collision avoidance system was originally implemented as a single monolithic class. This class encapsulated all core functionalities: sensor data processing, obstacle detection, local elevation mapping, slope analysis, and Bézier-based trajectory generation. While functional, this design reduced code readability and modularity, making it challenging to isolate or extend individual components for future development.

Functionally, AgRobPP-CA is well-suited for static obstacle avoidance but lacks mechanisms for effectively dealing with dynamic obstacles, such as humans or animals moving within the robot's path. The absence of time-aware obstacle modeling, prediction, and motion-based filtering limits its responsiveness to real-world scenarios where the environment changes rapidly.

Additionally, the system's ability to detect and interpret steep slopes is constrained. Though it attempts to identify dangerous inclinations based on center-of-mass projections, its robustness in challenging terrain remains an open issue.

These limitations formed the basis for this dissertation's contributions. The work focuses on improving the system's architecture by modularizing it into distinct, reusable classes and extending its capability to support dynamic obstacle avoidance.

Chapter 4

Dynamic Obstacle Avoidance in AgRobPP-CA

This chapter presents the modifications made to the AgRobPP-CA framework [28] to enable dynamic obstacle avoidance. While the original system focused on static obstacles, it lacked the ability to detect and react to moving obstacles. The following sections describe the redesigned software structure, the updated state machine logic, the avoidance algorithm implemented in code, and the testing environment used to validate the avoidance algorithm.

4.1 Software Structure

The original implementation of AgRobPP-CA was structured as a single monolithic class, which made it difficult to extend or modify specific features. In this work, AgRobPP-CA was refactored into different classes according to the functionalities of the system. Figure 4.1 illustrates the UML class diagram, showing the classes and the relationships between them. Each class has a specific functionality, as described below:

- **collision_avoidance_ros_node:** This is the main ROS 2 node. It initializes the system, creates and manages all other classes, and contains the state machine in which collision avoidance logic is described. It also handles ROS-related operations such as subscriptions, publishers and parameters.

- **collision_avoidance:** Implements the core logic for collision avoidance. It manages the decision-making state machine and computes avoidance maneuvers. It uses functions from the *CV*, *parametric*, and *aux* classes.

- **map:** Manages the dynamic obstacle map and terrain information. It is responsible for the detection and tracking of dynamic obstacles. It processes LiDAR data to identify potential collisions and assess the environment for static and dynamic objects. It uses functions from the *collision_avoidance*, *CV*, *parametric*, and *aux* classes.

- **parametric:** Responsible for generating or manipulating the Bézier curves that represent the robot's planned path. It evaluates and samples those curves. It uses functions from the *CV* and *aux* classes.
- **CV:** Transforms the robot's environmental data into a 2D visual map format for analysis and navigation, using OpenCV[1]. It uses functions from the *aux* class.
- **aux:** Provides utility functions used across the other classes.

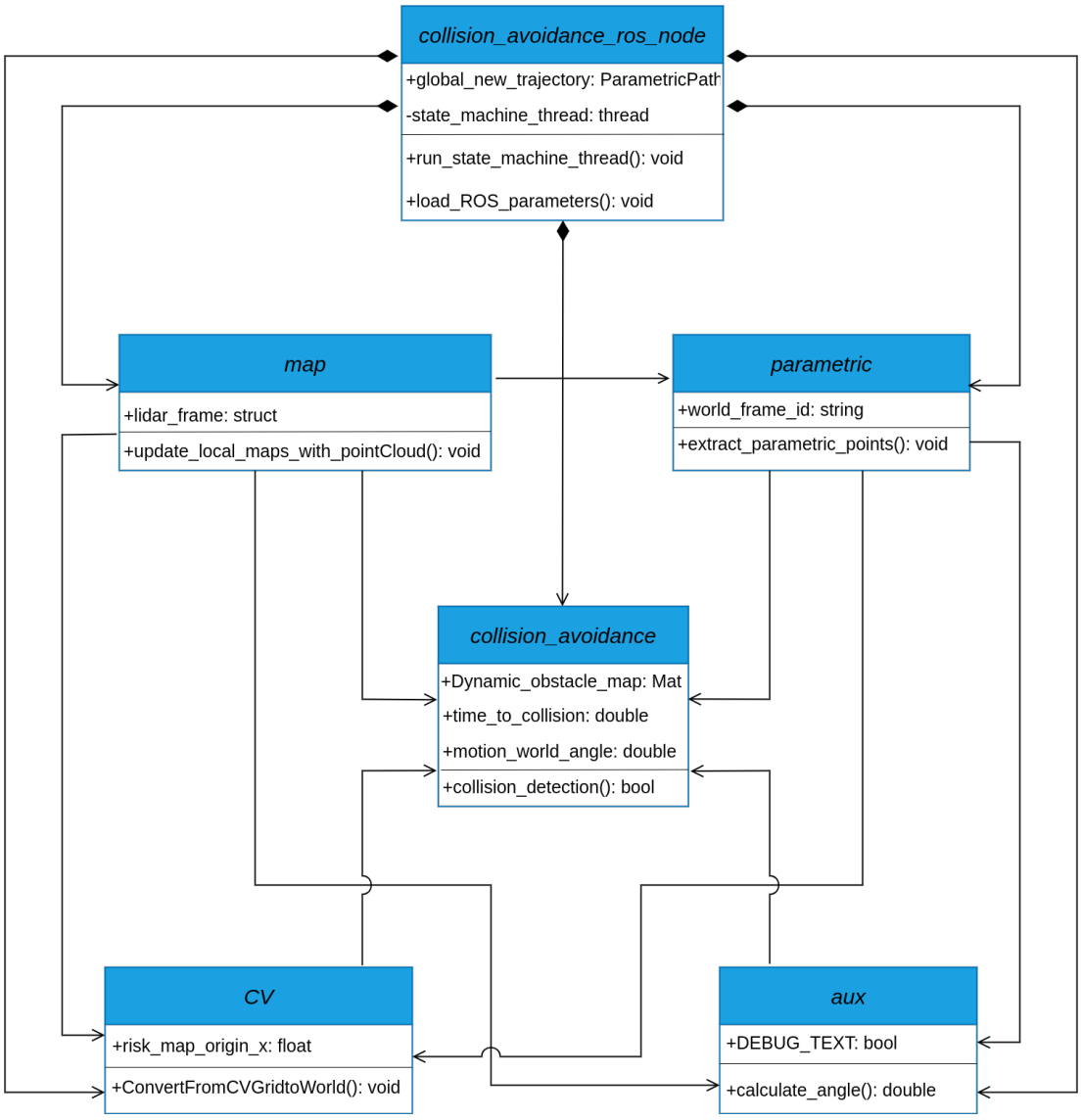


Figure 4.1: UML Class Diagram.

The full UML class diagram, detailing all attributes and methods, is available at the following link[2].

[1] https://opencv.org/ (Accessed on 27/07/2025)
[2] https://drive.google.com/

4.2 Updated State Machine

To support dynamic obstacle avoidance, the AgRobPP-CA framework required updates to its internal decision logic. The state machine was modified to monitor time-to-collision values and respond appropriately by triggering emergency stops or path recalculations based on real-time motion predictions.

This section presents the new state transitions, including the integration of dynamic safety checks and resume conditions. Figure 4.2 represents the updated state machine that illustrates how the robot now reacts, not only to static hazards, but also to moving agents in the environment.

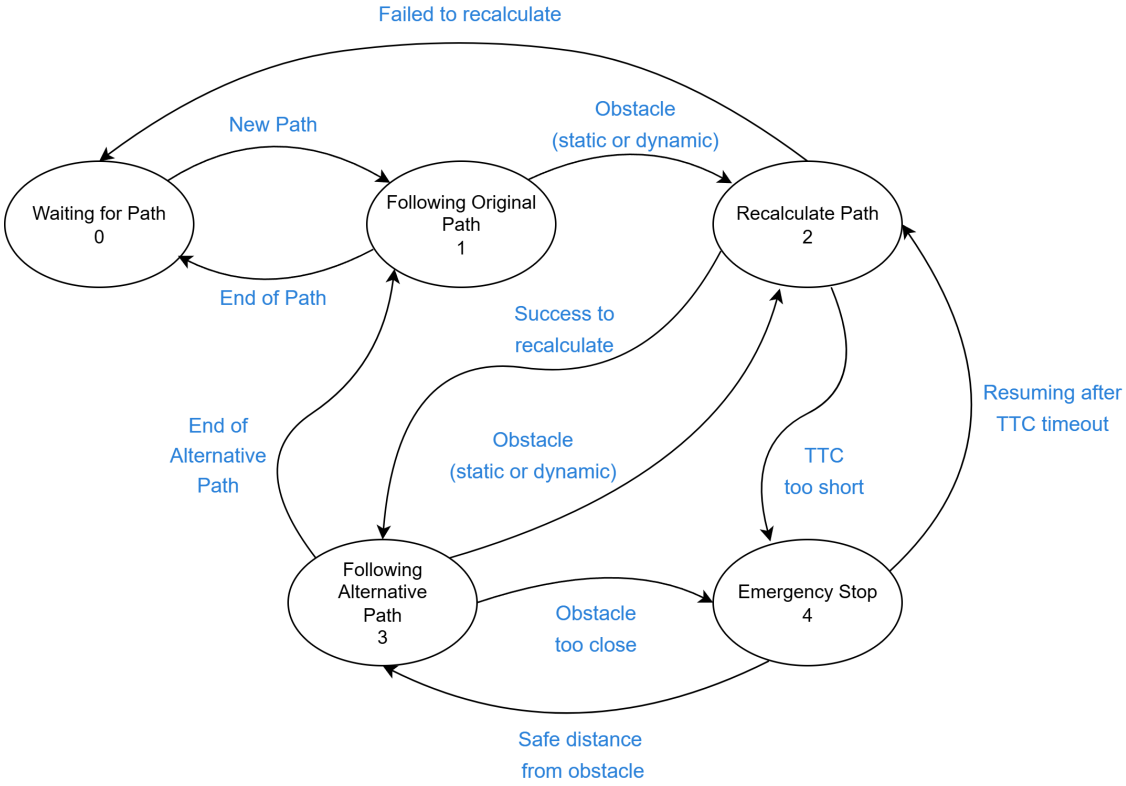


Figure 4.2: Updated collision avoidance state machine diagram.

The collision avoidance process now operates as a finite state machine with five states, that can be described as the following:

- **Waiting for Path:** The robot stops and waits to receive a parametric curve.
- **Following Original Path:** The robot navigates along the received path until an obstacle is detected. Obstacle will be classified as dynamic or static.
- **Recalculate Path:** If an obstacle is detected within a predefined safety distance, an alternative trajectory is generated using Bézier curves. The Bézier curve will depend in the nature

of the obstacle. If recalculation fails, the robot returns to the "Waiting for Path" state. Time-to-collision (TTC) represents the time in which the robot and the obstacle will collide. If TTC is too short, the robot will go to the "Emergency Stop" state.

- **Following Alternative Path:** If a valid trajectory is found, the robot switches to the alternative path and resumes operation. If an obstacle is detected while the robot is navigating the alternative path, the robot will return to "Recalculate Path" state. While the robot is following the alternative path, if an obstacle appears too close, the robot will switch to the "Emergency Stop" state. If the robot ends the alternative path, it will resume the "Following Original Path" state.

- **Emergency Stop:** In this state the robot stops the trajectory that it was following. If the emergency stop is triggered by the TTC being too short, the robot will go to the "Recalculate Path" state after the TTC timeout. If the emergency stop was triggered because of an obstacle being too close to the robot, the robot will return to "Following Alternative Path", once there is a safety distance.

4.3 Dynamic Obstacle Avoidance Implementation

The dynamic obstacle avoidance system in the AgrobPP-CA framework enables the robot to safely navigate environments with moving obstacles. This section explains how dynamic obstacles are perceived, tracked, and how the robot replans its path in real time to avoid potential collisions.

4.3.1 Dynamic Obstacle Mapping

Using 3D LiDAR data, a 2D risk map called Local Safety Regions is constructed. This map visually represents the obstacles around the robot. Each point is colored according to its distance to the robot. Importantly, this map is always expressed in the robot's referential frame, meaning the robot is consistently positioned at the center of the image, and all obstacles are represented relative to it. Color assignment is based on predefined distance thresholds that divide the surrounding area into safety zones. Green indicates the safe zone, where obstacles are not considered for avoidance. In contrast, the remaining colors are used to represent potential threats and are included in the dynamic obstacle map: yellow marks the warning zone, orange indicates a danger zone, and red represents a critical zone requiring immediate attention. Light green is used for obstacles that are farther away but still pose a potential collision risk.

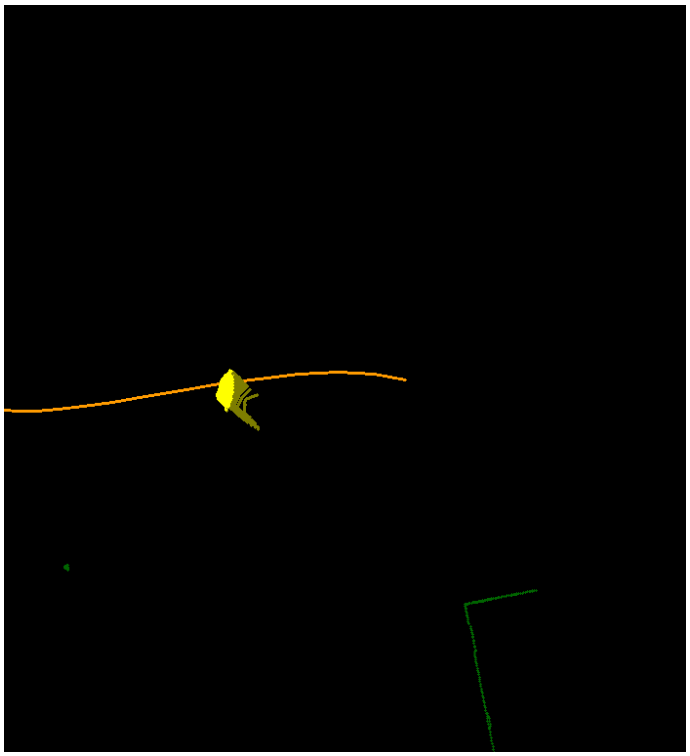


Figure 4.3: Local safety regions map.

Figure 4.3 illustrates this map, where the robot's current trajectory is shown in orange, and surrounding obstacles are visualized with varying color intensities according to their risk level. This visual representation allows for intuitive monitoring of nearby threats and forms the basis for dynamic avoidance logic.

In addition, another 2D map called the Dynamic Obstacle Map was generated, in order to focus exclusively on the obstacles that can be a potential threat to the robot. While all points from the Local Safety Regions map are included, only those outside the safe zone are considered for tracking. To identify and isolate obstacle regions, the system applies color-based thresholding. Each type of risk color is converted into a binary mask, while points belonging to the safe zone are excluded. These masks are combined into one, to represent all potential problematic obstacles. A dilation operation is applied to the combined mask to merge nearby points into connected regions. Contours are extracted and each contour is bounded by a rectangle (bounding box), then filtered to remove very small areas, which are likely noise, and extremely thin or elongated boxes, for example from slope edges.

Valid bounding obstacles represent obstacle candidates, and are shown in the Dynamic Obstacle Map in a red bounding box with its centroid in red. The centroid is the middle point of the rectangle and it is transformed into the appropriate frame, to be used as the obstacle position.

After detecting potential obstacle bounding boxes, the boxes are associated with previously tracked obstacles. Each tracked obstacle has its unique ID. If there is no match found, a new trajectory is created with a new ID. Old unmatched obstacles are deleted when they are not updated in a long time. If there is a match, append the new position and update the timestamp. If enough

data exists, the velocity of each tracked obstacle is estimated using the difference in its relative position with respect to the robot over consecutive frames. An obstacle is then classified as dynamic if its estimated velocity exceeds a predefined threshold, otherwise, it is considered static.

If the obstacle is confirmed dynamic, Kalman filter [35] is initialized, if it was not already. The filter is used to refine position and velocity estimates. Using the tracked dynamic obstacle velocity to estimate future positions, the obstacle's future trajectory is represented by a blue line in the Dynamic Obstacle Map. Assuming that the robot continues moving forward at its current speed and heading, the robot's future trajectory is predicted and represented by a green line. Furthermore, each robot prediction is compared with each obstacle prediction, to find if a potential collision is detected. The robot's future trajectory prediction is used because the current trajectory represented in orange can be too small to do an accurate prediction of the intersection point. The intersection point is saved and visualized as a red circle in the Dynamic Obstacle Map.

Figure 4.4 shows the result of thresholding and contour detection, with bounding boxes drawn around potential dynamic obstacles and their centroids marked for trajectory estimation. The ID and trajectory of each dynamic obstacle are tracked and displayed in white. The robot's current trajectory is shown in orange and its prediction in green, while the predicted motion of the obstacle is represented in blue. The intersection point between the robot and obstacle paths is marked with a red circle.

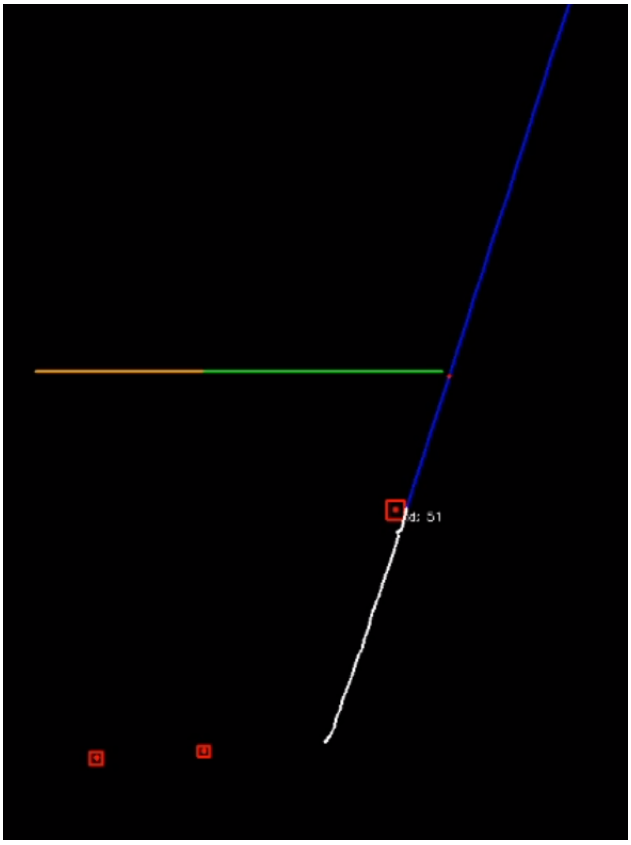


Figure 4.4: Dynamic obstacle map.

4.3.2 Path Replanning

The same Bézier curve generation method used for static obstacle avoidance is extended for dynamic obstacles, with additional parameters driven by the obstacle's predicted motion. When a dynamic obstacle is detected intersecting the robot's path, the AgRobPP-CA system attempts to generate an alternative path that proactively avoids collision based on predicted future positions and relative timing.

The process begins similarly to the static case: a 4th-degree Bézier curve is created with five control points (P_0 to P_4), where P_0 is the robot's current position, P_c is the predicted collision point, and P_4 is the rejoining point after deviation. The main difference lies in how the control point P_2 (the curve's midpoint) is calculated.

Unlike in static scenarios, the placement of P_2 for dynamic avoidance is guided by the obstacle's predicted motion vector, TTC, and lateral direction of approach. First, the algorithm estimates the obstacle's velocity vector by predicting its future positions, then calculates the lateral direction of motion relative to the robot's trajectory. The robot then biases its curve in the opposite lateral direction to maximize spatial clearance. For example, if the obstacle is moving left relative to the robot's trajectory, the robot curves right.

Additionally, the system calculates the estimated time before the robot and obstacle would reach the intersection point, known as TTC. Equation 4.1 defines the TTC, used to evaluate the urgency of an avoidance maneuver. In this context, d represents the Euclidean distance between the obstacle's centroid (treated as a point) and the predicted intersection point, while v_o and v_r are the speeds of the obstacle and the robot, respectively. This TTC value dictates how aggressive the avoidance maneuver must be: the lower the TTC, the greater the curvature applied to the trajectory. This is achieved using a non-linear urgency scaling function that adjusts both the angle and lateral displacement of the control point P_2.

$$\text{TTC} = \frac{d}{v_o + v_r} \tag{4.1}$$

The angle of the obstacle's motion with respect to the global frame, denoted as θ_{motion} and calculated using Equation 4.2, also plays a critical role. If the obstacle moves mostly horizontally or vertically, the robot continues with avoidance planning. However, if the obstacle's motion is too diagonal, falling outside of defined horizontal or vertical thresholds, the robot considers the situation too unpredictable. In such cases, rather than risking an avoidance maneuver, the robot enters a paused state and waits for the obstacle to clear or for conditions to change. In this equation v_x and v_y represent the obstacle's velocity components along the x and y axes of the global coordinate frame, respectively.

$$\theta_{\text{motion}} = |\arctan 2(v_y, v_x)| \cdot \frac{180}{\pi} \tag{4.2}$$

An iterative loop tests the generated curve for safety. At each iteration, the midpoint control point P_2 is recalculated to increase lateral deviation. A new Bézier curve is constructed and

sampled into discrete points. Each point is checked against the predicted future positions of the obstacle.

If any predicted collision is found within a safety threshold, the curve is rejected, and the offset is increased further. This continues until a safe curve is found or a maximum number of iterations is reached. If no valid curve is possible, the robot may remain in a paused state until the situation clears.

Finally, once a safe Bézier curve is validated, it is transformed into the global frame, merged with the remaining original trajectory, and sent as the updated command path for the robot.

4.4 Testing Environment and Simulation Tools

To validate the dynamic obstacle avoidance algorithm within the AgRobPP-CA framework, the system was tested in a ROS2 based simulation environment built using Gazebo[3] and RViz[4]. This setup allows for realistic modeling of both the robot and its surrounding environment, while offering tools for visualization, logging and debugging.

The simulation environment was based on a custom Gazebo world created for the FIRA 2025 Hackathon, available in a public repository[5]. The robot is modeled as a ground vehicle equipped with an Ackermann steering configuration and an agricultural implement. The simulation environment was designed specifically for the competition, to replicate real-world agricultural conditions.

To simulate dynamic obstacles, a human actor was added to the Gazebo environment. This actor mimics a pedestrian walking back and forth across the robot's trajectory. The actor was configured by modifying the simulation world file according to the official Gazebo tutorial on animated actors[6].

This setup enabled repeatable and controlled testing of the robot's response to moving obstacles under varying conditions. The actor's predictable path ensured consistent test cases across multiple simulation runs. Figure 4.5 shows the Gazebo environment used for testing.



Figure 4.5: Gazebo environment.

[3] https://gazebosim.org/home (Accessed on 27/07/2025)
[4] https://wiki.ros.org/rviz (Accessed on 27/07/2025)
[5] https://github.com/FiraHackathon/hackathon2025_ws (Accessed on 27/07/2025)
[6] https://classic.gazebosim.org/tutorials?tut=actor (Accessed on 27/07/2025)

The robot was equipped with a 3D LiDAR sensor and its movement was controlled using the ROS2 navigation stack provided by the TRIBE Laboratory of INESC TEC . This stack includes key navigation components such as localization, path planning, and motion control. The AgRobPP-CA node was integrated into this stack as an additional layer for local dynamic obstacle avoidance.

RViz was used for real-time visualization of the robot's behavior. It provided live feedback of LiDAR scans, the planned robot path and alternative paths generated for collision avoidance. In Figure 4.6, the orange dots and blue line connecting the dots represent the original robot path provided by the ROS2 navigation stack. Due to a detected obstacle, the robot generates a new trajectory shown as a red Bézier curve.

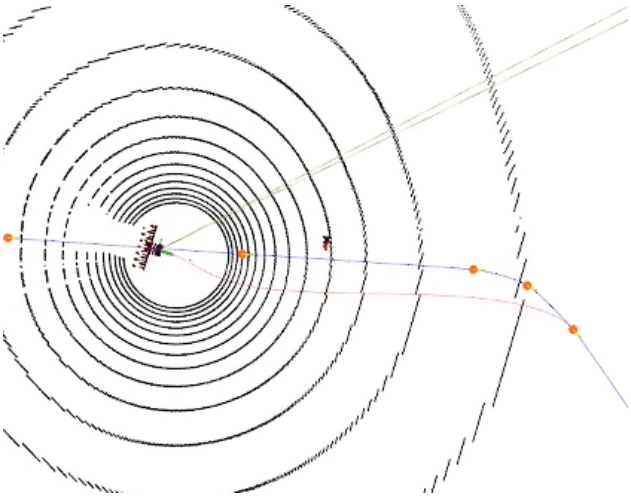


Figure 4.6: RViz environment.

This testing setup provides the basis for the results and performance evaluation presented in the next chapter.

Chapter 5

Results

This chapter presents the experimental results obtained during simulation-based testing of the dynamic obstacle avoidance system implemented in the AgRobPP-CA framework. The primary goal is to evaluate the system's ability to detect, assess, and safely avoid moving obstacles in a range of scenarios while maintaining feasible navigation behavior.

Tests were carried out in a controlled Gazebo environment using the ROS2-based navigation stack and real-time visualizations in RViz. The obstacles are marked by a red circle for better visualization. The robot was tested at a nominal speed of $0.4\,\mathrm{m/s}$, with additional experiments conducted at slower and faster velocities ($0.2\,\mathrm{m/s}$ and $0.8\,\mathrm{m/s}$, respectively) to assess how speed affects avoidance performance and decision making.

Five scenarios were tested, in which each test scenario involved dynamic obstacles with varying trajectories and directions relative to the robot's path. The performance was analyzed in terms of trajectory generation, timing and collision avoidance success.

5.1 Single Horizontal Obstacle from the Left

A dynamic obstacle with a constant speed of $1.4\,\mathrm{m/s}$ enters from the left side of the robot's path, moving horizontally across the planned trajectory. This test evaluates the system's ability to predict lateral motion and trigger early left-curving avoidance.

5.1.1 Robot Speed $0.4\,\mathrm{m/s}$

Figure 5.1 illustrates the test scenario as visualized in the Gazebo and RViz environments, while Figure 5.2 presents the corresponding Dynamic Obstacle Map highlighting the avoidance maneuver.

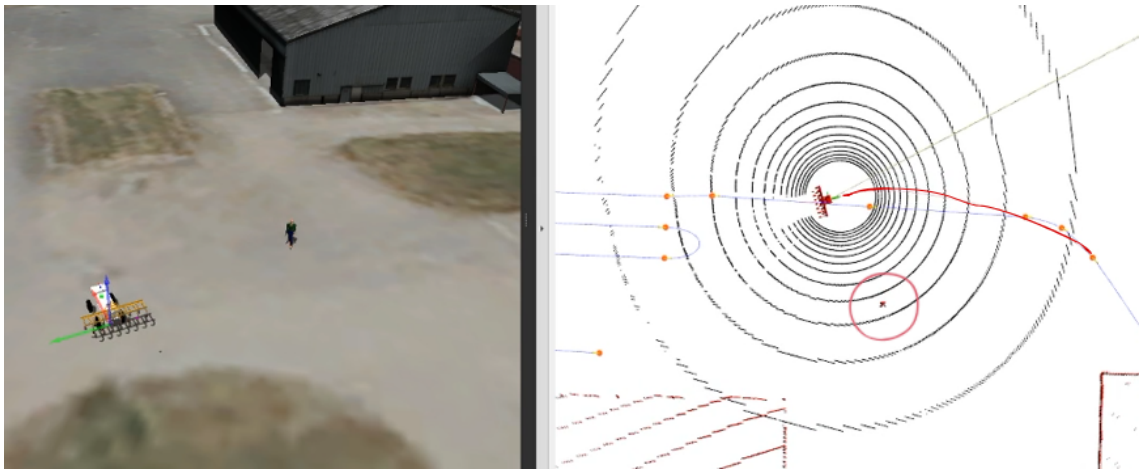


Figure 5.1: Gazebo (left) and RViz (right) views for a single horizontal obstacle from the left.

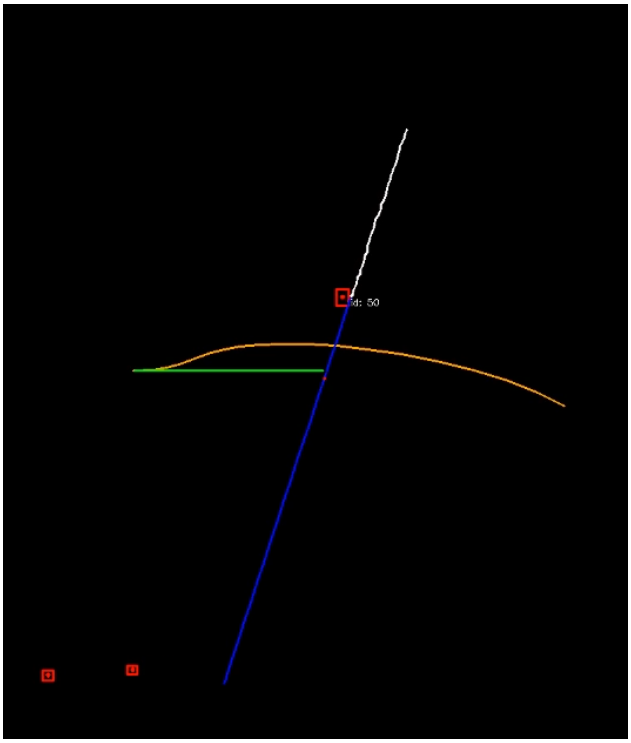


Figure 5.2: Dynamic Obstacle Map for a single horizontal obstacle from the left.

The system successfully detected the intersection point. The predicted distance from the obstacle to the intersection point was approximately 6.13 m, and the obstacle's velocity was estimated at 2.67 m/s. The robot's velocity remained at 0.4 m/s, resulting in a TTC of 2.00 s, as computed by Equation 4.1.

Given the short TTC, the system classified the situation as requiring early avoidance and triggered a proactive maneuver. The motion angle of the obstacle was computed to be 90.21°, indicating horizontal motion. The obstacle's relative direction of movement with respect to the robot's path was classified as going to the right.

Consequently, the system generated a leftward deviation using a 4th-degree Bézier curve. The avoidance curve was validated as safe, and the robot transitioned to the new path. The final trajectory was generated and the robot resumed motion along the new curve. Table 5.1 shows the key parameters and outcomes of the test scenario.

Table 5.1: Test results for obstacle approaching horizontally from the left (robot speed = 0.4 m/s).

Parameter	**Value**
Robot speed	0.4 m/s
Predicted obstacle speed	2.67 m/s
Time-to-collision (TTC)	2.00 s
Predicted intersection distance	6.13 m
Motion angle relative to global frame	90.21°
Obstacle motion classification	Horizontal
System response	Robot curves left

Figure 5.3 shows the command velocities issued to the robot during the test. The linear velocity (top plot) remains constant at $0.4\,\mathrm{m/s}$, while the angular velocity (bottom plot) increases and then decreases smoothly. The angular velocity is initially positive due to the leftward avoidance maneuver, then becomes negative to realign with the original trajectory.

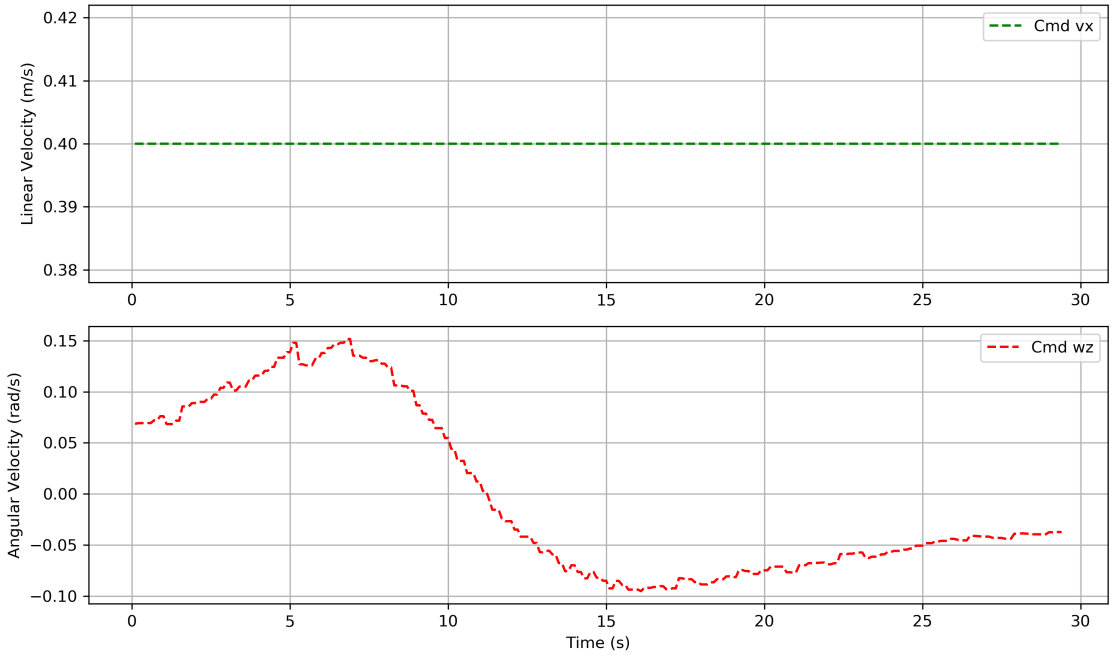


Figure 5.3: Commanded linear and angular velocities during avoidance of horizontal obstacle from the left. (robot speed = 0.4 m/s).

5.1.2 **Robot Speed** 0.2 m/s

With the robot's linear velocity reduced to 0.2 m/s, the same scenario was evaluated.

The system accurately detected the intersection point and calculated a TTC of approximately 3.08 s, using the estimated obstacle speed of 2.22 m/s and a predicted intersection distance of 7.46 m. The motion angle was computed as 90.17°, confirming the obstacle's horizontal movement across the robot's path.

Table 5.2 summarizes the key parameters and outcomes of the test.

Table 5.2: Test results for obstacle approaching horizontally from the left (robot speed = 0.2 m/s).

Parameter	**Value**
Robot speed	0.2 m/s
Predicted obstacle speed	2.22 m/s
Time-to-collision (TTC)	3.08 s
Predicted intersection distance	7.46 m
Motion angle relative to global frame	90.17°
Obstacle motion classification	Horizontal
System response	Robot curves left

Figure 5.4 shows the commanded linear and angular velocities during the test. The linear velocity remains mostly constant at 0.2 m/s, while the angular velocity increases positively as the robot curves left to avoid the obstacle, followed by a negative correction to realign with the original trajectory. The lower velocity of the robot enables a more controlled and smooth maneuver, comparing to higher speeds.

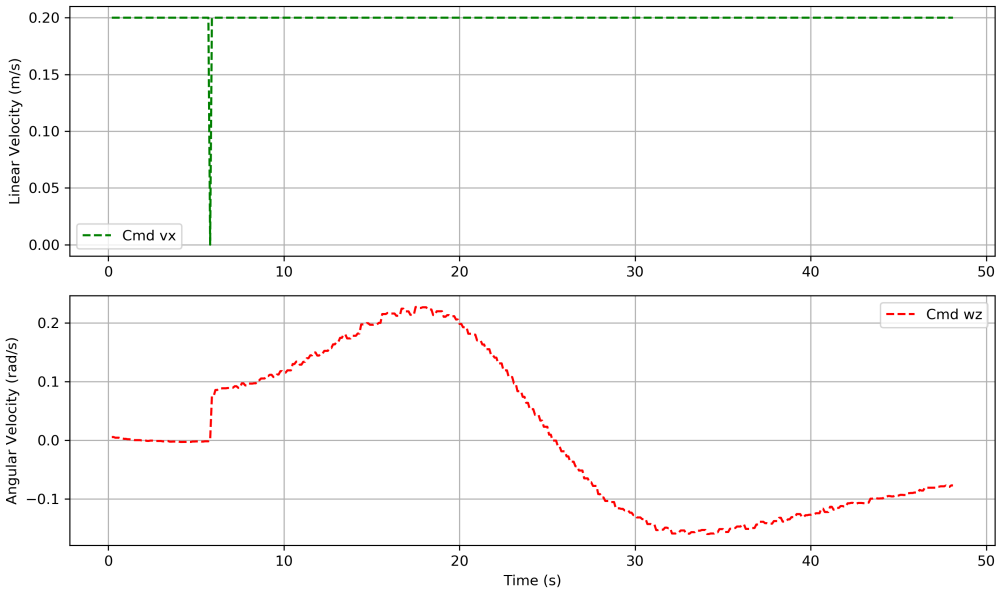


Figure 5.4: Commanded linear and angular velocities during avoidance of horizontal obstacle from the left (robot speed = 0.2 m/s).

5.1.3 **Robot Speed** $0.8\,\mathrm{m/s}$

In this scenario, the robot operated at a higher constant speed of $0.8\,\mathrm{m/s}$, while the same horizontal obstacle crossed its path.

The system correctly detected the potential intersection and computed a TTC of approximately 6.65 s, using the obstacle's speed of $1.11\,\mathrm{m/s}$ and a predicted intersection distance of 12.72 m. The motion angle was calculated as 91.93°, confirming the obstacle was moving horizontally from left to right. As indicated in Table 5.3, the robot curved left to safely avoid the obstacle.

Table 5.3: Test results for obstacle approaching horizontally from the left (robot speed = 0.8 m/s).

Parameter	**Value**
Robot speed	0.8 m/s
Predicted obstacle speed	1.11 m/s
Time-to-collision (TTC)	6.65 s
Predicted intersection distance	12.72 m
Motion angle relative to global frame	91.93°
Obstacle motion classification	Horizontal
System response	Robot curves left

Figure 5.5 shows the command velocities during the test. The linear velocity remains stable at $0.8\,\mathrm{m/s}$, while the angular velocity curve shows a sharp rise and fall, reflecting the faster execution of the avoidance maneuver. Compared to lower speed cases, the higher velocity causes the system to perform a more aggressive curve, with steeper angular changes, in order to complete the avoidance faster.

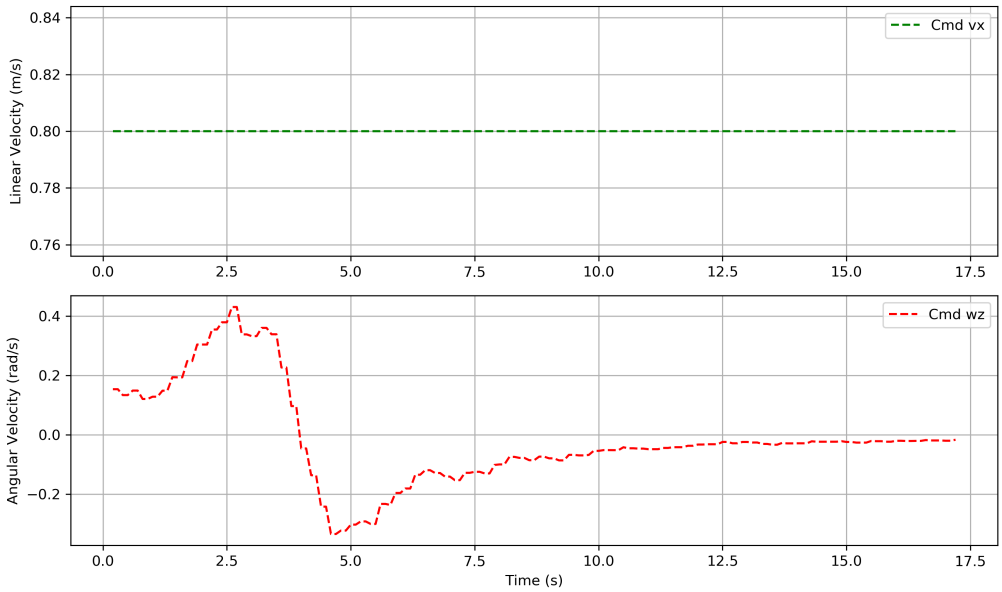


Figure 5.5: Commanded linear and angular velocities during avoidance of horizontal obstacle from the left (robot speed = 0.8 m/s).

5.2 Single Horizontal Obstacle from the Right

A dynamic obstacle with a constant speed of $1.4\,\mathrm{m/s}$ enters from the right side of the robot's path, moving horizontally across the planned trajectory. The following tests evaluate the system's ability to classify right-to-left motion and trigger an avoidance maneuver that curves the robot to the right.

5.2.1 Robot Speed $0.4\,\mathrm{m/s}$

Figure 5.6 shows the test environment in Gazebo and RViz, while Figure 5.7 illustrates the dynamic obstacle's interaction and resulting path planning in the Dynamic Obstacle Map.

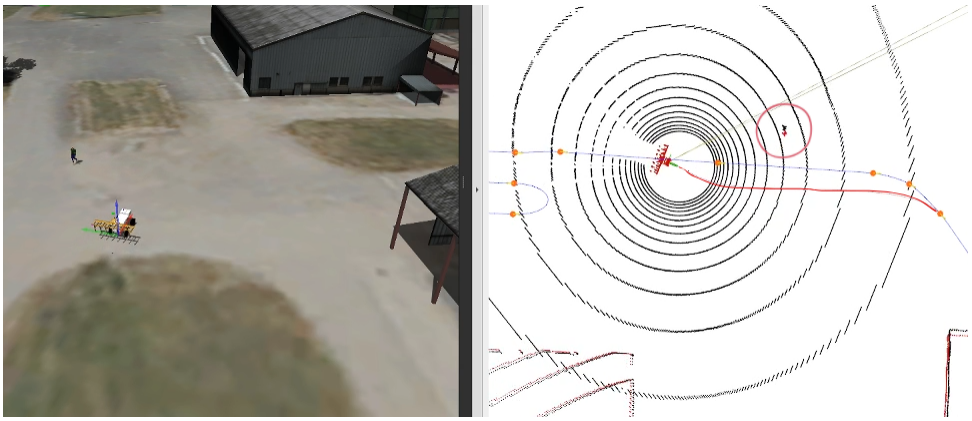


Figure 5.6: Gazebo (left) and RViz (right) views for a single horizontal obstacle from the right.

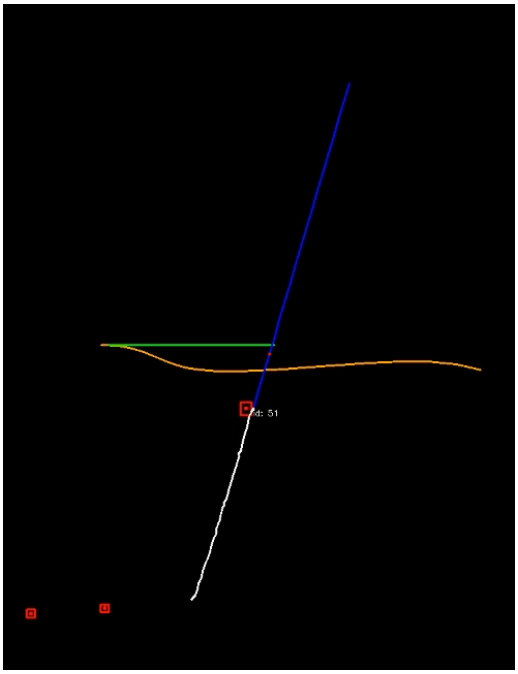


Figure 5.7: Dynamic Obstacle Map for a single horizontal obstacle from the right.

The system detected a valid intersection point, with a predicted distance from obstacle to collision point of 7.07 m. The obstacle's estimated velocity was 2.80 m/s, and with the robot moving at 0.4 m/s, the TTC was computed as 2.21 s.

The motion angle of the obstacle was calculated as 89.93°, confirming horizontal motion. The obstacle was identified as moving to the left relative to the robot, so the curve was generated to the right.

A new Bézier curve was successfully generated and validated. The replanning produced a new trajectory that was published and followed by the robot as the updated path. Table 5.4 shows the key parameters and outcomes of the test scenario.

Table 5.4: Test results for obstacle approaching horizontally from the right (robot speed = 0.4 m/s)

Parameter	**Value**
Robot speed	0.4 m/s
Predicted obstacle speed	2.80 m/s
Time-to-collision (TTC)	2.21 s
Predicted intersection distance	7.07 m
Motion angle relative to global frame	89.93°
Obstacle motion classification	Horizontal
System response	Robot curves right

Figure 5.8 presents the velocity commands issued to the robot during the avoidance. The linear velocity remains constant at 0.4 m/s, while the angular velocity shows a clear transition, starting negative as the robot curves right to avoid the obstacle, and then gradually becoming positive as the robot returns to its original path.

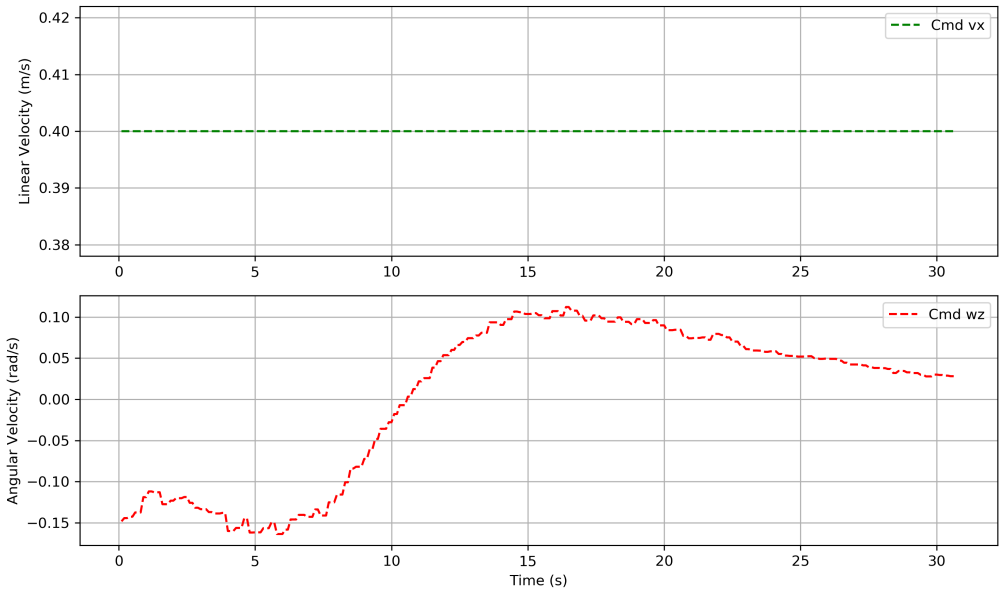


Figure 5.8: Commanded linear and angular velocities during avoidance of horizontal obstacle from the right (robot speed = 0.4 m/s).

5.2.2 **Robot speed** $0.2\,\mathrm{m/s}$

The robot's linear velocity was reduced to $0.2\,\mathrm{m/s}$ while facing a dynamic obstacle entering from the right and moving horizontally across the trajectory.

The system correctly detected the intersection point at a distance of approximately $13.07\,\mathrm{m}$. The obstacle was moving at $1.65\,\mathrm{m/s}$, and the TTC was estimated to be $7.08\,\mathrm{s}$. The obstacle's motion angle was calculated as 91.56°, consistent with horizontal movement. Since the obstacle was approaching from the right, the robot initiated a rightward deviation using a Bézier curve, which was successfully validated and followed.

Table 5.5 summarizes the key parameters for this scenario.

Table 5.5: Test results for obstacle approaching horizontally from the right (robot speed = 0.2 m/s)

Parameter	**Value**
Robot speed	0.2 m/s
Predicted obstacle speed	1.65 m/s
Time-to-collision (TTC)	7.08 s
Predicted intersection distance	13.07 m
Motion angle relative to global frame	91.56°
Obstacle motion classification	Horizontal
System response	Robot curves right
Path replanning outcome	Success

The command velocities for this test are shown in Figure 5.9. The linear velocity remains constant throughout the test, while the angular velocity shows a smoother curve compared to higher speed scenarios.

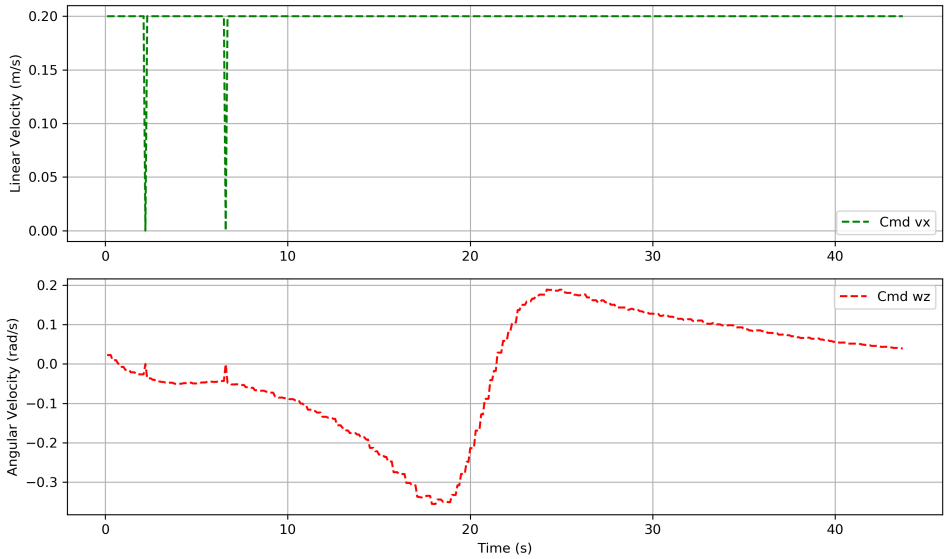


Figure 5.9: Commanded linear and angular velocities during avoidance of horizontal obstacle from the right (robot speed = 0.2 m/s).

5.2.3 **Robot Speed** 0.8 m/s

In this test, the robot's linear speed was increased to 0.8 m/s to evaluate the system's performance under faster motion conditions.

The intersection was correctly detected at a distance of 7.54 m, and the estimated obstacle velocity was 1.37 m/s. With the robot moving at a higher speed, the TTC calculated was 3.48 s. The obstacle's motion angle was computed as 90.14°, confirming its horizontal trajectory across the robot's path from right to left.

The robot responded by generating a Bézier curve that curved right, consistent with the obstacle's relative direction. The trajectory was validated as safe, and the system published the new path, transitioning the robot into avoidance mode.

Table 5.6 presents a summary of the test parameters and results.

Table 5.6: Test results for obstacle approaching horizontally from the right (robot speed = 0.8 m/s)

Parameter	**Value**
Robot speed	0.8 m/s
Predicted obstacle speed	1.37 m/s
Time-to-collision (TTC)	3.48 s
Predicted intersection distance	7.54 m
Motion angle relative to global frame	90.14°
Obstacle motion classification	Horizontal
System response	Robot curves right

Figure 5.10 illustrates the velocity profile during the maneuver. The angular velocity shows a pronounced right-turning motion followed by a gradual return to align with the original path. This behavior reflects the robot's more abrupt response at higher speeds.

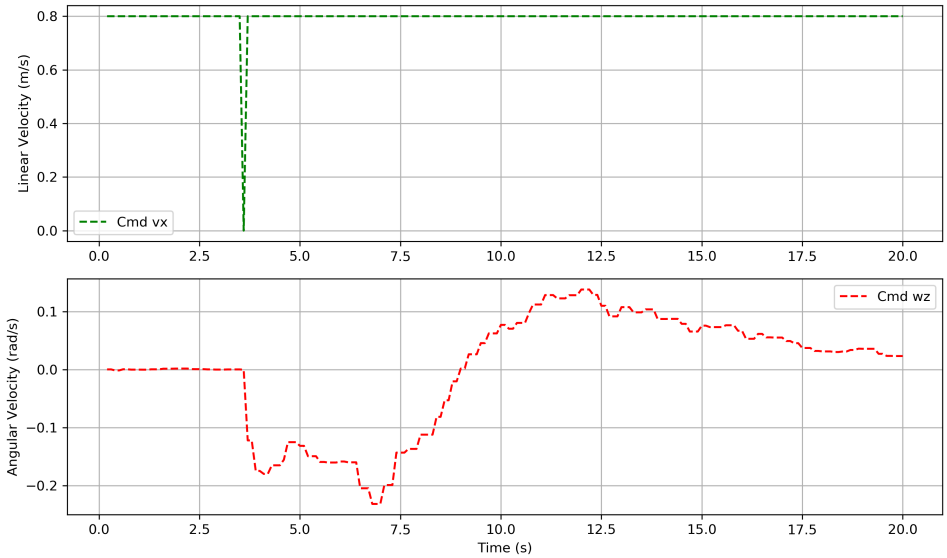


Figure 5.10: Commanded linear and angular velocities during avoidance of horizontal obstacle from the right (robot speed = 0.8 m/s).

5.3 Vertical Obstacle Crossing Path

This test introduces a dynamic obstacle moving vertically at speed $0.86\,\mathrm{m/s}$ across the robot's path, simulating a top-down crossing scenario. The objective is to verify the system's ability to correctly identify vertical motion and initiate a lateral avoidance in the opposite direction of the obstacle's relative path.

5.3.1 Robot speed $0.4\,\mathrm{m/s}$

The obstacle enters from the robot's right and moves in a nearly vertical trajectory. The robot is expected to curve left to avoid a potential collision. Figure 5.11 shows the Gazebo and RViz views, while Figure 5.12 presents the Dynamic Obstacle Map illustrating the calculated avoidance maneuver.

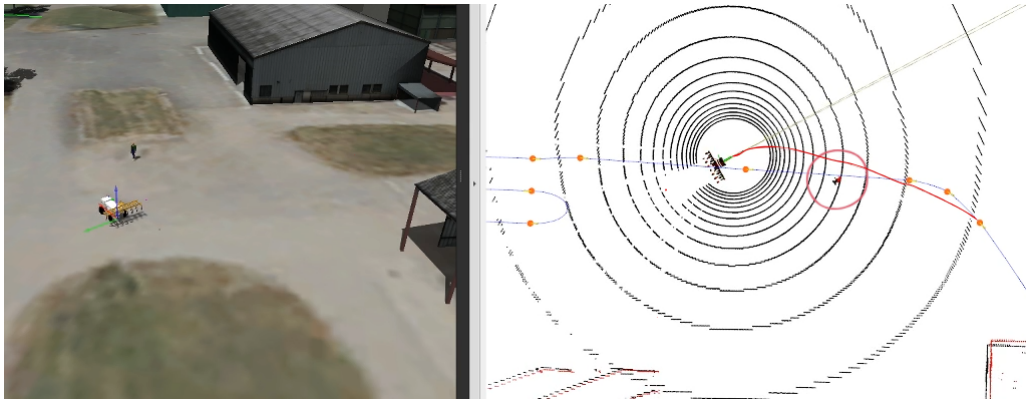


Figure 5.11: Gazebo (left) and RViz (right) views for a vertical obstacle crossing the robot's path.

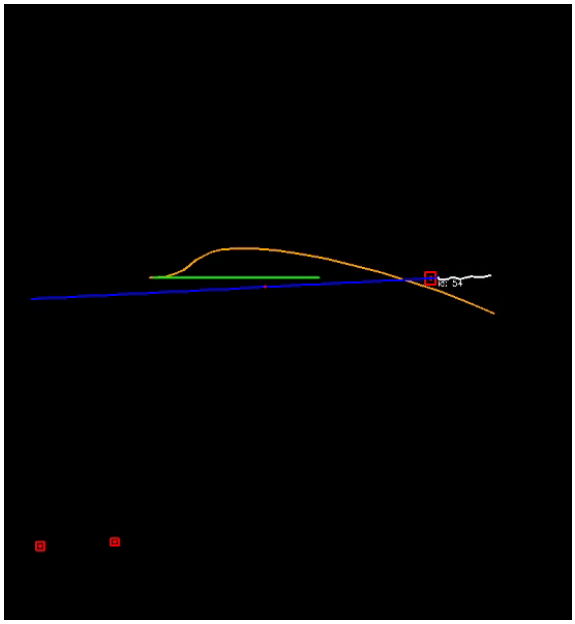


Figure 5.12: Dynamic Obstacle Map for a vertical obstacle crossing the robot's path.

The system detected a valid intersection point, with a predicted distance from the obstacle to the intersection point of 9.84 m. The obstacle's estimated velocity was 1.33 m/s, while the robot maintained a constant speed of 0.4 m/s. This led to a TTC of 5.70 s. The motion angle of the obstacle was computed to be approximately 20.97°, indicating a predominantly vertical trajectory. Relative to the robot's path, the obstacle was moving in direction to the right of the robot, and the robot curved to the left accordingly.

A valid 4th-degree Bézier curve was generated and passed the safety checks. The robot followed the new trajectory, avoiding the obstacle successfully. However, during the execution of the avoidance, the obstacle crossed very close to the robot's position. As a result, an emergency stop was triggered temporarily to prevent a potential collision, entering the `EMERGENCY_STOP` state. The robot remained paused until the path was verified as safe, after which it resumed operation by transitioning back to the `FOLLOWING_ALTERNATIVE_PATH` state. Table 5.7 shows the key parameters and outcomes of the test scenario.

Table 5.7: Test results for vertical obstacle crossing path (robot speed = 0.4 m/s)

Parameter	**Value**
Robot speed	0.4 m/s
Predicted obstacle speed	1.33 m/s
Time-to-collision (TTC)	5.70 s
Predicted intersection distance	9.84 m
Motion angle relative to global frame	20.97°
Obstacle motion classification	Vertical
System response	Robot curves left

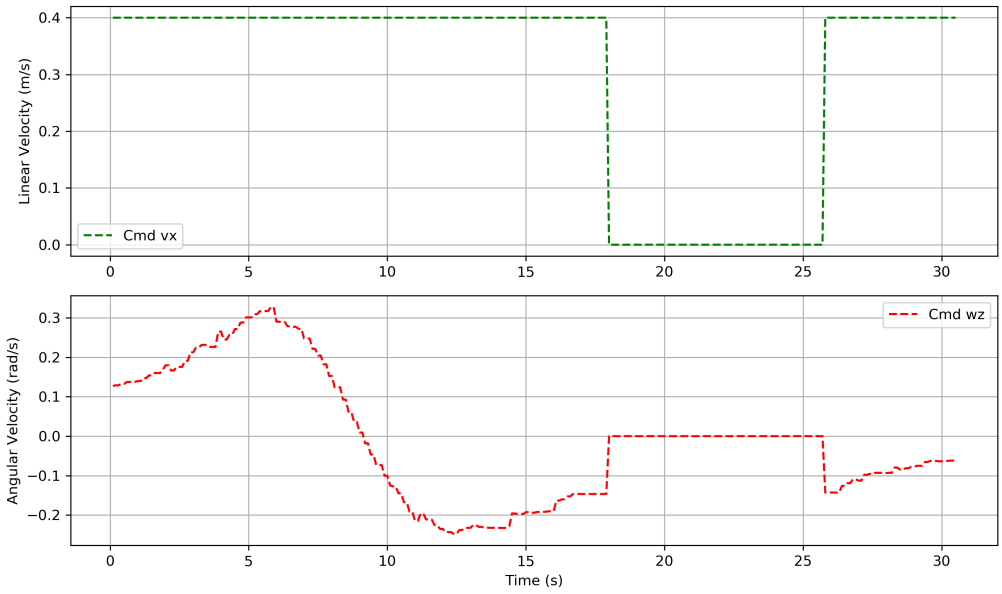


Figure 5.13: Commanded linear and angular velocities during avoidance of vertical obstacle (robot speed = 0.4 m/s).

Figure 5.13 shows the linear and angular velocity commands during the vertical obstacle crossing. The robot initially maintains its linear velocity at 0.4 m/s while executing a curved maneuver, as indicated by the varying angular velocity. Around 18 seconds, the linear velocity drops sharply to zero, signaling an emergency stop. This was triggered when the obstacle passed in close proximity to the robot. The robot resumes motion once the path is safe to follow.

5.3.2 Robot speed 0.2 m/s

When the robot's velocity was set to 0.2 m/s, the algorithm accurately identified the vertical obstacle with a motion angle of 21.47°, and classified the relative direction of the obstacle as being to the right of the robot. Consequently, the robot initiated a leftward deviation.

The predicted intersection distance was 11.26 m, and the TTC was 8.18 s, prompting an early avoidance strategy. A new Bézier-based trajectory was computed, validated, and published to the controller. However, during the execution of the new path, the obstacle passed too close to the robot, triggering the emergency stop mechanism.

Table 5.8 summarizes the scenario outcomes and parameters.

Table 5.8: Test results for vertical obstacle crossing path (robot speed = 0.2 m/s).

Parameter	**Value**
Robot speed	0.2 m/s
Predicted obstacle speed	1.18 m/s
Time-to-collision (TTC)	8.18 s
Predicted intersection distance	11.26 m
Motion angle relative to global frame	21.47°
Obstacle motion classification	Vertical
System response	Robot curves left

Figure 5.14 shows the velocity profile of this scenario. The linear velocity drops abruptly when the emergency stop is triggered. The angular velocity curve displays a smooth increase as the robot starts the avoidance curve, followed by an immediate cut to zero once the emergency condition is met.

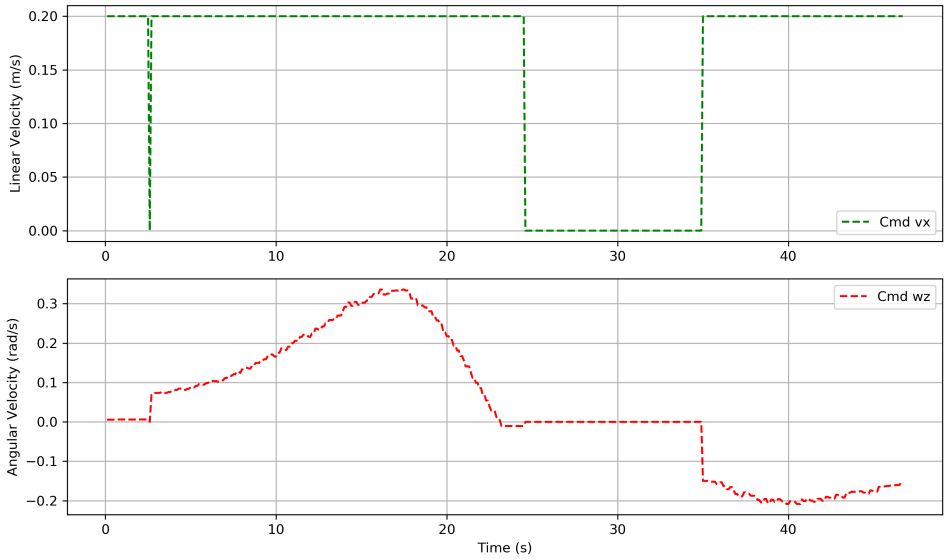


Figure 5.14: Commanded linear and angular velocities during avoidance of vertical obstacle (robot speed = 0.2 m/s).

5.3.3 Robot speed $0.8\,\mathrm{m/s}$

At a higher linear velocity of $0.8\,\mathrm{m/s}$, the same vertical crossing scenario was tested. The system correctly identified a dynamic obstacle moving vertically across the robot's path and calculated a valid intersection point. The obstacle's velocity was estimated at $0.93\,\mathrm{m/s}$, and the predicted TTC was approximately $10.19\,\mathrm{s}$, with a predicted intersection distance of $17.60\,\mathrm{m}$. The motion angle of $20.17°$ confirmed vertical motion, and the system triggered an early avoidance by generating a curve to the left. A valid trajectory was successfully generated using a Bézier curve and published as the new path.

Table 5.9 presents the key metrics for this test.

Table 5.9: Test results for vertical obstacle crossing path (robot speed = 0.8 m/s).

Parameter	**Value**
Robot speed	0.8 m/s
Predicted obstacle speed	0.93 m/s
Time-to-collision (TTC)	10.19 s
Predicted intersection distance	17.60 m
Motion angle relative to global frame	20.17°
Obstacle motion classification	Vertical
System response	Robot curves left

Figure 5.15 presents the commanded velocities for the test conducted at higher speed. In contrast to the lower speed cases, the plots reveal sharper velocity changes, reflecting the system's need to respond more rapidly to potential collisions. No emergency stop was triggered, as the

robot successfully executed the avoidance maneuver in time to maintain a safe distance from the obstacle.

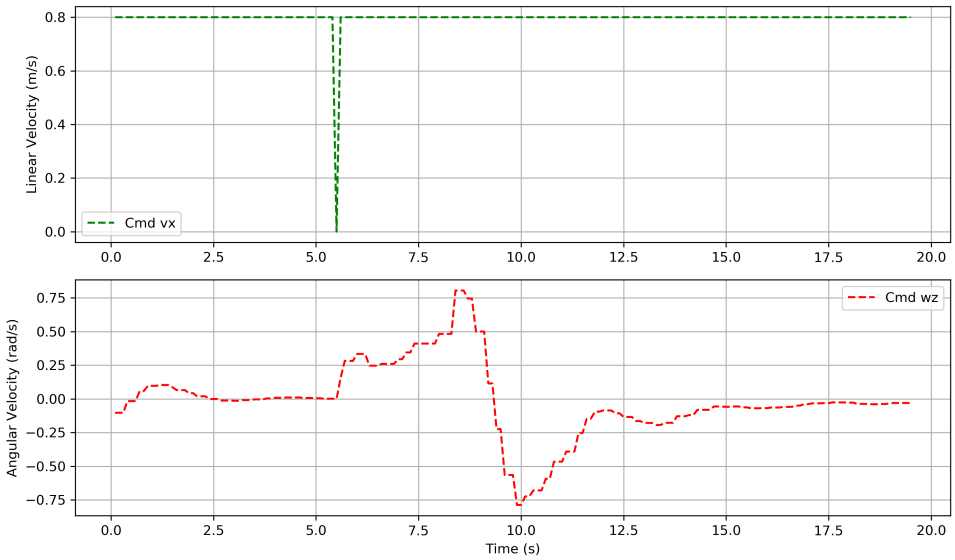


Figure 5.15: Commanded linear and angular velocities during avoidance of vertical obstacle (robot speed = 0.8 m/s).

5.4 Diagonal Obstacle Crossing Path

An obstacle with speed of $1.61\,\mathrm{m/s}$ approaches diagonally at an angle that is not identified as horizontal or vertical. This scenario tests the system's ability to classify ambiguous motion angles and the decision to pause in order to avoid dangerous avoidance trajectories.

5.4.1 Robot speed $0.4\,\mathrm{m/s}$

Figure 5.16 shows the Gazebo and Rviz environment, while Figure 5.17 presents the Dynamic Obstacle Map for this test.

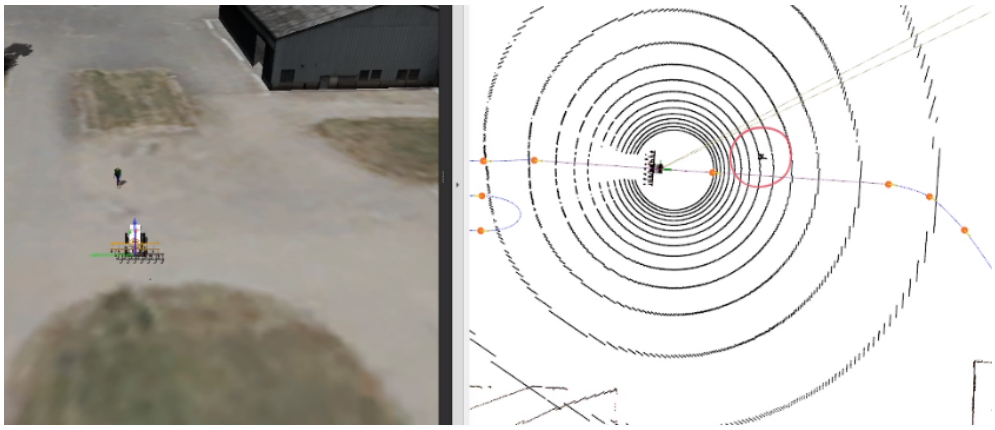


Figure 5.16: Gazebo (left) and RViz (right) views for a diagonal obstacle crossing the robot's path.

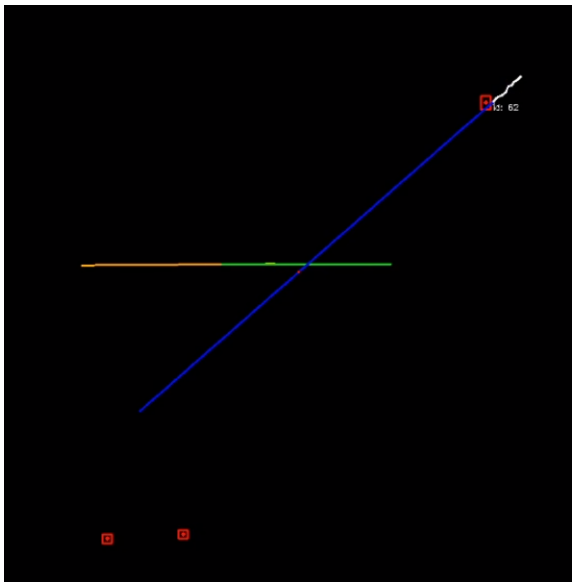


Figure 5.17: Dynamic Obstacle Map for a diagonal obstacle crossing the robot's path.

The system detected an intersection point. The obstacle's estimated speed was 1.14 m/s and the predicted distance to the intersection point was 14.88 m. The TTC was calculated as 9.67 s.

The obstacle's motion angle was computed to be 61.88°, which did not meet the criteria for horizontal or vertical classification. Consequently, the system flagged the obstacle as diagonal and triggered a safe pause, entering the `EMERGENCY_STOP` state.

To ensure the obstacle no longer posed a threat, the robot remained stationary for the full duration of the predicted TTC plus an additional threshold margin. Once the pause timer ended, the system resumed motion. Table 5.10 shows the key parameters and outcomes of the test scenario.

Table 5.10: Test results for diagonal obstacle crossing path (robot speed = 0.4 m/s)

Parameter	**Value**
Robot speed	0.4 m/s
Predicted obstacle speed	1.14 m/s
Time-to-collision (TTC)	9.67 s
Predicted intersection distance	14.88 m
Motion angle relative to global frame	61.88°
Obstacle motion classification	Diagonal
System response	Emergency stop (pause)

The velocity plots in Figure 5.18 reflect the robot's response to the diagonally approaching obstacle. As shown in the upper plot, the linear velocity drops sharply to zero at around 7.5 s, indicating the triggered emergency stop. The robot remains stationary for approximately 15 seconds, aligning with the predicted TTC plus a safety threshold. After the pause, the robot resumes motion along the path.

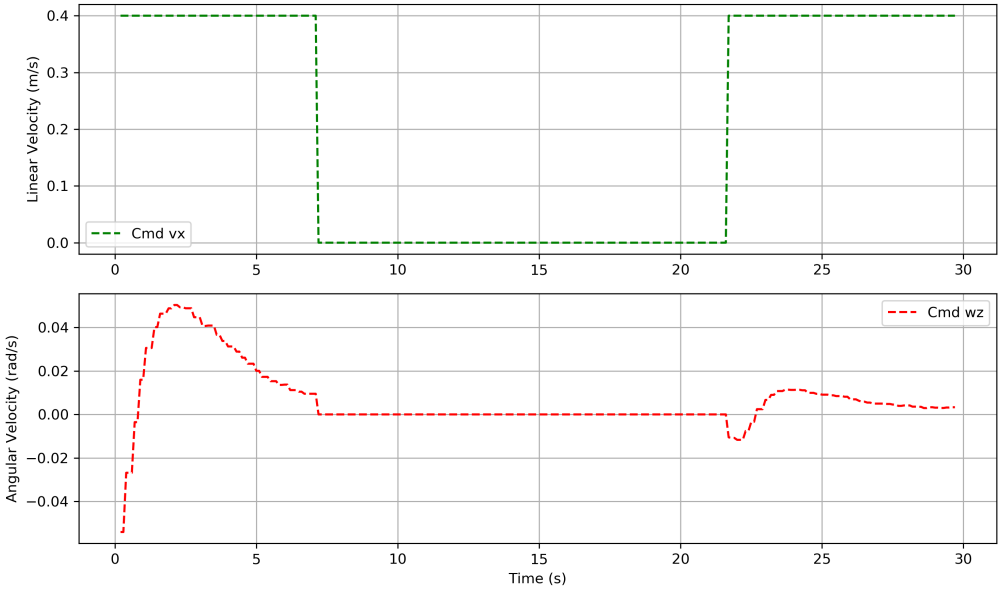


Figure 5.18: Commanded linear and angular velocities during avoidance of diagonal obstacle (robot speed = 0.4 m/s).

5.4.2 **Robot speed** 0.2 m/s

In this scenario, the robot moves at a reduced speed of 0.2 m/s while facing a diagonally approaching obstacle. As shown in Table 5.11, the obstacle has a velocity of 1.58 m/s and the system predicted a collision point at a distance of 16.83 m. The calculated TTC was 9.43 s.

The obstacle's motion angle was determined to be 60.29°, which does not fit within the classification ranges for horizontal or vertical motion. Consequently, the system flagged the obstacle as moving diagonally and initiated a safety pause. The robot entered the `EMERGENCY_STOP` state to prevent an unsafe avoidance maneuver.

Table 5.11: Test results for diagonal obstacle crossing path (robot speed = 0.2 m/s)

Parameter	**Value**
Robot speed	0.2 m/s
Predicted obstacle speed	1.58 m/s
Time-to-collision (TTC)	9.43 s
Predicted intersection distance	16.83 m
Motion angle relative to global frame	60.29°
Obstacle motion classification	Diagonal
System response	Emergency stop (pause)

As seen in Figure 5.19, the robot's linear velocity drops to zero and remains stationary throughout the perceived threat duration. Once the obstacle was no longer seen as a threat, after waiting the predicted TTC plus a threshold, the robot resumed its trajectory.

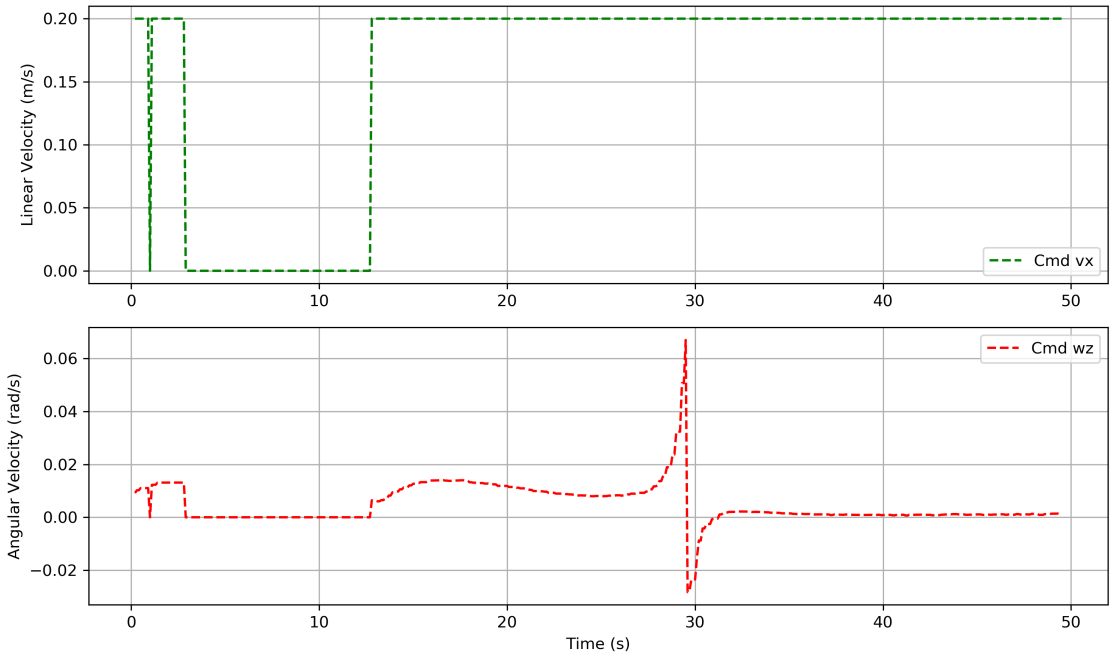


Figure 5.19: Commanded linear and angular velocities during avoidance of diagonal obstacle (robot speed = 0.2 m/s).

5.4.3 Robot speed $0.8\,\mathrm{m/s}$

In this test, the robot moves at a linear velocity of $0.8\,\mathrm{m/s}$ while facing an obstacle approaching diagonally at an angle of 62.32° and a speed of approximately $1.42\,\mathrm{m/s}$. The system correctly identified the obstacle as moving diagonally, which does not conform to either horizontal or vertical classification. As a result, the system triggered a pause maneuver, prioritizing safety.

The predicted intersection distance was $13.78\,\mathrm{m}$, with a computed TTC of $6.20\,\mathrm{s}$. Since this trajectory could pose a high-risk collision if an inaccurate curve were attempted, the robot entered the `EMERGENCY_STOP` state and came to a complete stop. The robot remained stationary throughout the expected collision window. After the pause duration elapsed, it resumed movement. Table 5.12 presents the key parameters for this test.

Table 5.12: Test results for diagonal obstacle crossing path (robot speed = 0.8 m/s)

Parameter	**Value**
Robot speed	0.8 m/s
Predicted obstacle speed	1.42 m/s
Time-to-collision (TTC)	6.20 s
Predicted intersection distance	13.78 m
Motion angle relative to global frame	62.32°
Obstacle motion classification	Diagonal
System response	Emergency stop (pause)

As shown in Figure 5.20, the linear velocity drops abruptly around 3.5 s, marking the start of the emergency pause. The robot stays idle until approximately 14 s, after which it resumes movement. The angular velocity remains low throughout the pause, confirming that no trajectory was executed during the critical collision window.

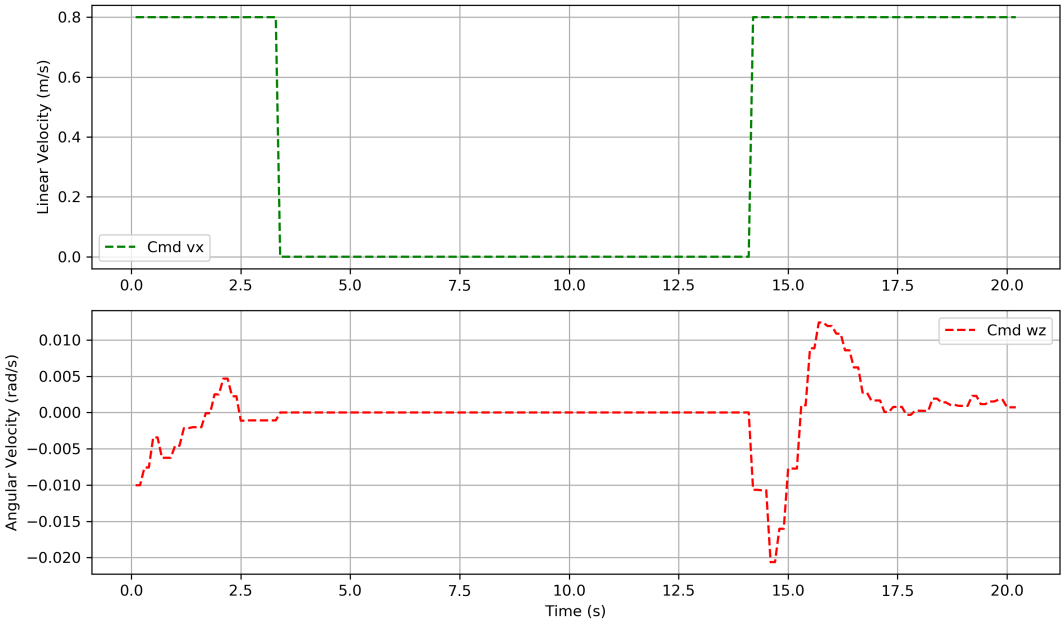


Figure 5.20: Commanded linear and angular velocities during avoidance of diagonal obstacle (robot speed = 0.8 m/s).

5.5 Two Crossing Obstacles from Different Directions

This scenario evaluates the system's capability to handle multiple dynamic obstacles crossing the robot's path. The first obstacle enters horizontally from the left with a constant speed of 1.4 m/s, while a second obstacle approaches shortly after from the right side, also in a horizontal trajectory with a constant speed of 1.4 m/s. The goal is to assess whether the system can detect consecutive threats, replan multiple times, and maintain safety.

5.5.1 **Robot speed** $0.4\,\mathrm{m/s}$

Figures 5.21 and 5.22 present the robot's behavior in Gazebo and RViz. Figures 5.23 and 5.24 show the Dynamic Obstacle Map in each avoidance. These views illustrate how the system reacts to consecutive dynamic threats and generates multiple valid avoidance curves.

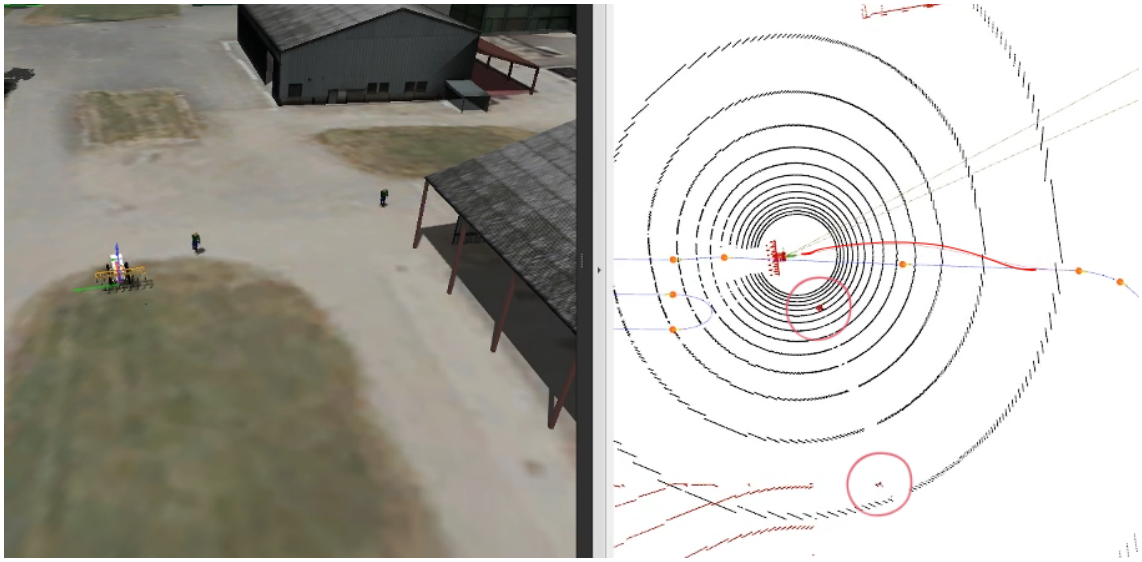


Figure 5.21: Gazebo (left) and RViz (right) views for first obstacle avoidance.

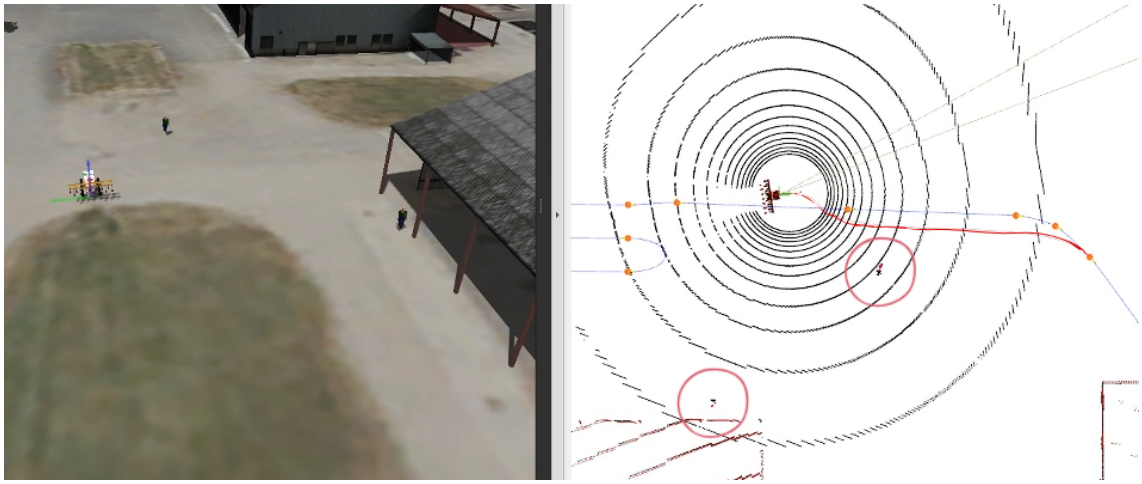


Figure 5.22: Gazebo (left) and RViz (right) views for second obstacle avoidance.

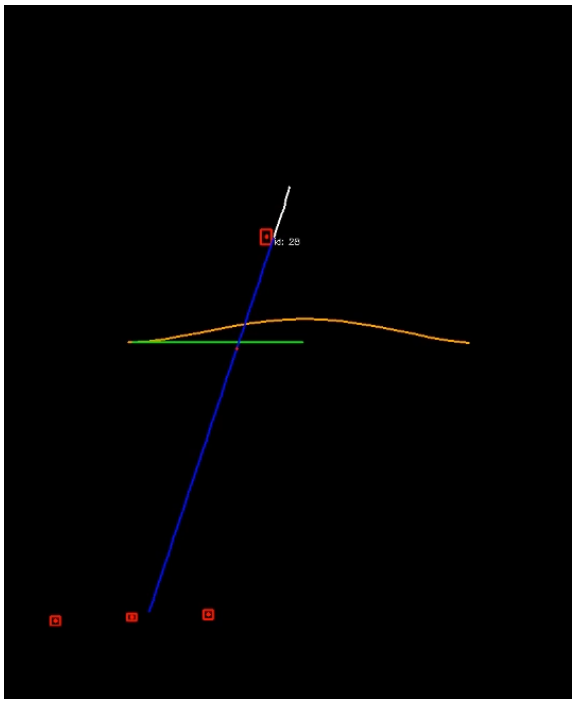


Figure 5.23: Dynamic Obstacle Map showing avoidance of first obstacle.

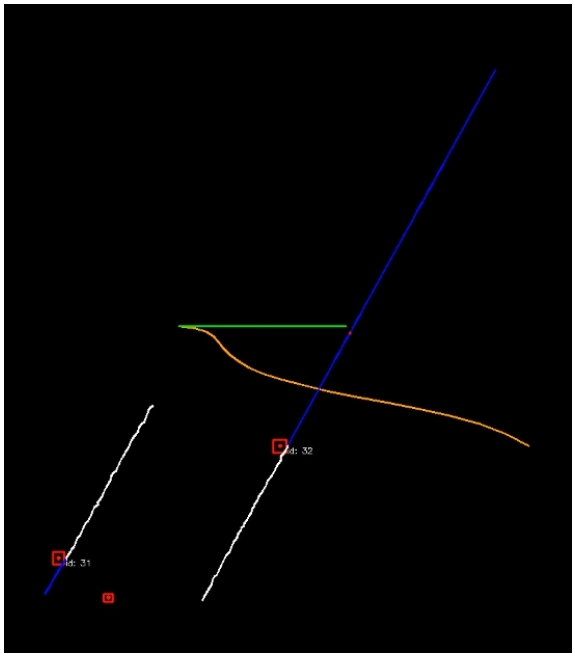


Figure 5.24: Dynamic Obstacle Map showing avoidance of second obstacle from opposite direction.

The first obstacle intersected the robot's path with a predicted speed of $1.17\,\mathrm{m/s}$. The system predicted a collision distance of $13.66\,\mathrm{m}$, resulting in a TTC of $8.73\,\mathrm{s}$. The motion angle was computed as $93.86°$, confirming horizontal movement. The obstacle's relative direction was classified

as going the right, so the robot curved to the left using a Bézier trajectory. The avoidance was validated and executed successfully.

Shortly after, a second dynamic obstacle was detected, with a predicted speed of 1.50 m/s. The collision distance was 9.91 m and the TTC was 5.20 s. The system identified this motion as coming from the left, with a motion angle of 89.86°. The robot curved to the right in response. Notably, a replan was triggered again during the first avoidance curve due to a new intersection being detected. A second Bézier curve was generated, validated and applied. Table 5.13 shows the key parameters and outcomes of the test scenario.

Table 5.13: Test results for two horizontal obstacles from opposite directions (robot speed = 0.4 m/s)

Parameter	**First Obstacle**	**Second Obstacle**
Predicted obstacle speed	1.17 m/s	1.50 m/s
Time-to-collision (TTC)	8.73 s	5.20 s
Predicted intersection distance	13.66 m	9.91 m
Motion angle relative to global frame	93.86°	89.86°
Obstacle motion classification	Horizontal	Horizontal
System response	Robot curves left	Robot curves right
Path replanning outcome	Success	Success

Figure 5.25 shows the velocity commands during the test. Two distinct velocity drops in the linear velocity plot correspond to the system pausing the robot in response to each detected obstacle. Meanwhile, the angular velocity changes indicate the robot curving first to the left and then to the right, validating the dual avoidance maneuvers generated for each obstacle.

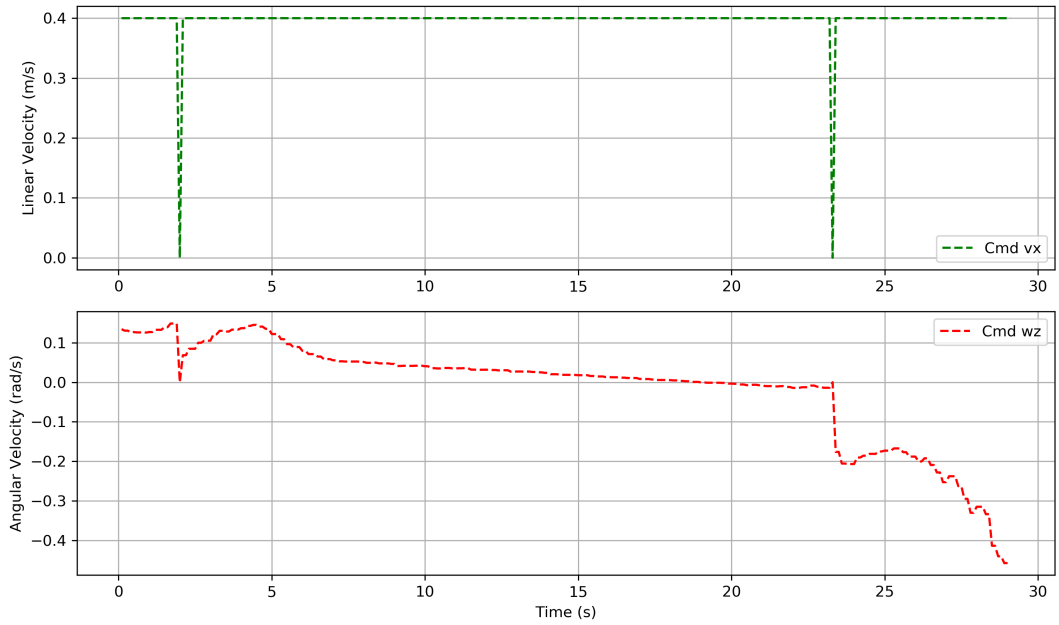


Figure 5.25: Commanded linear and angular velocities during avoidance of two horizontal obstacles from opposite directions (robot speed = 0.4 m/s).

5.5.2 **Robot speed** $0.2\,\mathrm{m/s}$

At a reduced robot speed of $0.2\,\mathrm{m/s}$, the system was tested against the same scenario to assess how slower motion affects threat evaluation and trajectory planning. The first dynamic obstacle was a detected with a speed of $1.14\,\mathrm{m/s}$, a predicted intersection distance of $11.74\,\mathrm{m}$, and a calculated TTC of $8.76\,\mathrm{s}$. The system identified the obstacle's motion angle as $81.86°$, confirming horizontal motion, and classified it as moving to the right relative to the robot. As a result, a valid Bézier curve was generated, curving the robot to the left to safely avoid the obstacle.

However, due to the robot's lower speed, the second obstacle detected with a speed of $1.65\,\mathrm{m/s}$, arriving shortly after, was not classified as an imminent threat. No intersection requiring replanning was identified. Therefore, no second avoidance maneuver was triggered.

Table 5.14: Test results for two horizontal obstacles from opposite directions (robot speed = 0.2 m/s)

Parameter	**First Obstacle**	**Second Obstacle**
Robot speed	0.2 m/s	0.2 m/s
Predicted obstacle speed	1.14 m/s	1.65 m/s
Time-to-collision (TTC)	8.76 s	–
Predicted intersection distance	11.74 m	–
Motion angle relative to global frame	81.86°	–
Obstacle motion classification	Horizontal	–
System response	Robot curves left	No action taken
Path replanning outcome	Success	–

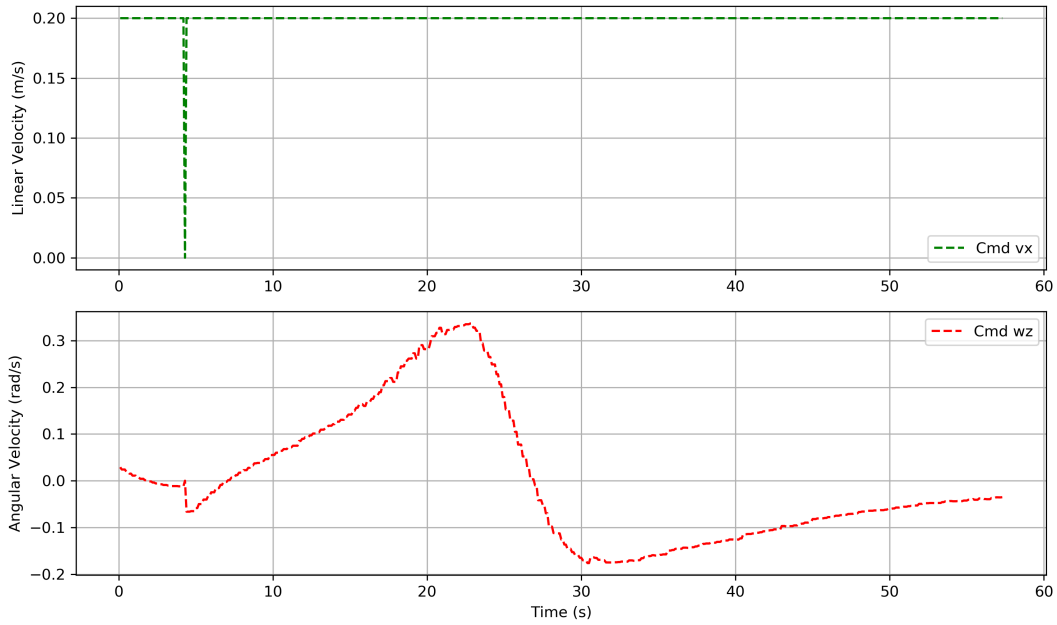


Figure 5.26: Commanded linear and angular velocities during avoidance of two horizontal obstacles from opposite directions (robot speed = 0.2 m/s).

As shown in Figure 5.26, the angular velocity increases during the leftward deviation and smoothly returns to near zero as the robot realigns with its original path.

5.5.3 Robot speed $0.8\,\mathrm{m/s}$

At a higher speed of $0.8\,\mathrm{m/s}$, the robot faced the same scenario of two dynamic obstacles crossing from opposite directions. The first obstacle was detected moving to the right with an estimated speed of $1.53\,\mathrm{m/s}$, at a predicted intersection distance of 12.39 m, leading to a TTC of 5.31 s. The motion angle was measured at 93.17°, indicating horizontal movement. The system responded by curving the robot to the left.

As the robot was executing this avoidance, a second obstacle moving to the left appeared, moving horizontally at $1.31\,\mathrm{m/s}$. The TTC was predicted as 8.25 s, with an intersection distance of 17.41 m. The obstacle's motion angle of 90.51° also classified it as horizontal. In response, the system triggered a second avoidance, this time curving the robot to the right.

The system handled both threats by generating two consecutive Bézier trajectories and successfully replanning in real time. Compared to the $0.4\,\mathrm{m/s}$ scenario, the higher velocity led to shorter TTC values, requiring quicker decision-making and sharper transitions. Nevertheless, the avoidance strategy remained successful and stable.

Table 5.15: Test results for two horizontal obstacles from opposite directions (robot speed = 0.8 m/s)

Parameter	**First Obstacle**	**Second Obstacle**
Predicted obstacle speed	1.53 m/s	1.31 m/s
Time-to-collision (TTC)	5.31 s	8.25 s
Predicted intersection distance	12.39 m	17.41 m
Motion angle relative to global frame	93.17°	90.51°
Obstacle motion classification	Horizontal	Horizontal
System response	Robot curves left	Robot curves right
Path replanning outcome	Success	Success

Figure 5.27 shows the velocity commands issued to the robot. Two drops in linear velocity indicate replanning phases. The angular velocity transitions reflect the robot first curving to the left and later to the right, validating the proper execution of both planned trajectories.

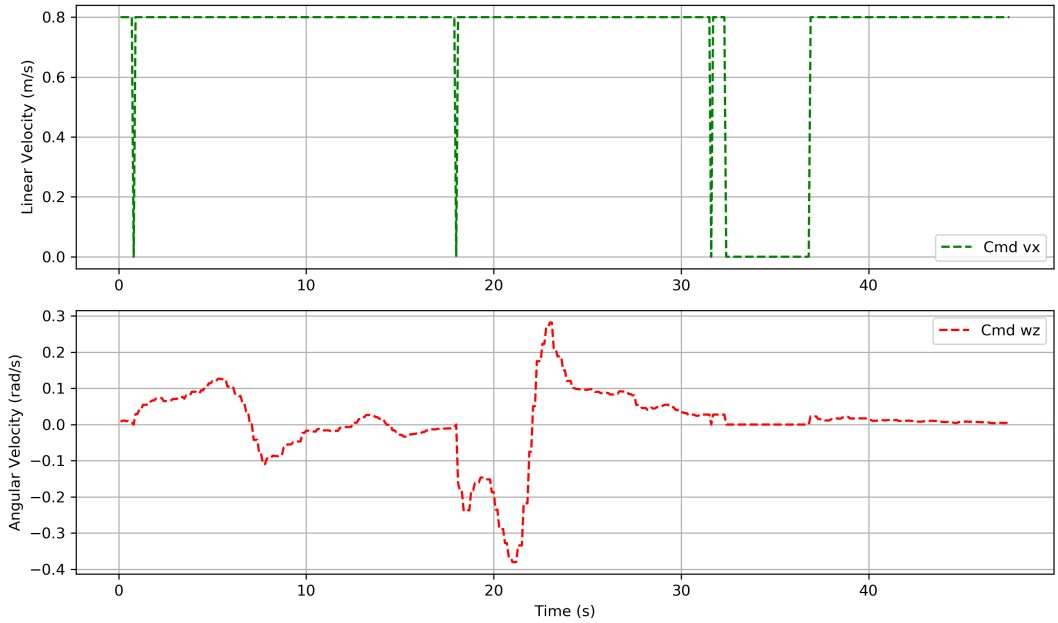


Figure 5.27: Commanded linear and angular velocities during avoidance of two horizontal obstacles from opposite directions (robot speed = 0.8 m/s).

5.6 Result Analysis

Each of the five test cases was conducted using three different robot speeds: 0.2 m/s, 0.4 m/s, and 0.8 m/s, allowing a comprehensive evaluation of the system's performance under varying conditions. The results highlight the flexibility of the proposed approach in adapting its avoidance behavior to different robot velocities and threat configurations.

In the horizontal obstacle scenarios (left and right), the system consistently detected motion angles near 90°, accurately classified them as horizontal, and generated avoidance curves in the direction opposite to the obstacle's trajectory. At higher speeds, the system responded more aggressively, leading to sharper changes in angular velocity. In contrast, at lower speeds, the robot had more time to respond, producing smoother, more gradual maneuvers.

For vertical obstacles, where the motion angle was close to 20°, the robot was instructed to curve laterally. In all speed conditions, the motion was correctly classified, and the system responded with a valid Bézier curve. However, at lower speeds, the robot performed brief stops, in order to maintain safety when the obstacle was close to the robot, as the robot could not avoid the obstacle quickly enough.

In the diagonal obstacle scenario, the system correctly identified ambiguous motion angles (between horizontal and vertical thresholds) and triggered a safe pause instead of initiating avoidance. This behavior was consistent across all speeds, confirming the robustness of the diagonal classification logic.

The scenario involving two consecutive obstacles approaching from opposite directions tested the system's ability to manage complex, time-sensitive situations. At 0.4 m/s and 0.8 m/s, the

robot detected and avoided both obstacles with two separate Bézier curves, validating its capacity for successive replanning. In contrast, at $0.2\,\mathrm{m/s}$, the system completed only one avoidance, as the second obstacle no longer posed a threat by the time it approached, demonstrating that reduced speeds can mitigate compound risks by increasing spatial and temporal separation.

One notable limitation observed across tests is the variability in the estimated velocity of the dynamic obstacles. Because the tracking system relies on a relatively simple approach, based on recent position differences over time, the predicted obstacle velocity can sometimes be noisy or inaccurate, especially when the obstacle's motion is erratic or partially occluded. This can affect the accuracy of TTC estimation and motion classification. However, the system generally maintained safe behavior even under such uncertainties.

Chapter 6

Conclusions and Future Work

This dissertation has presented the development and validation of a real-time dynamic obstacle avoidance system for agricultural robots. The system successfully integrates perception, trajectory generation, and reactive control mechanisms to handle dynamic obstacles.

The algorithm demonstrated robustness in simulation tests, reliably identifying and responding to potential collisions. Specific attention was given to diagonal motion scenarios, where ambiguous obstacle trajectories could otherwise result in unsafe avoidance behaviors. The system's conservative approach, pausing when classification was uncertain, prioritized safety and prevented the generation of unfeasible paths.

Furthermore, the solution builds upon and extends the AgRobPP-CA framework by introducing real-time trajectory replanning and enhancing the system's ability to operate under dynamic conditions. The use of parametric Bézier curves, TTC estimation and directional classification proved to be effective in dynamic agricultural scenarios.

However, terrain-aware navigation and control was not fully addressed within the scope of this work. The current system does not account for terrain features such as slope, inclination, or surface irregularity in dynamic environments. This represents an important area for improvement. To extend the capabilities of the proposed system, the following directions are recommended for future work:

- **Terrain-Adaptive Planning:** Future work should incorporate terrain inclination, in order to improve robot stability and path feasibility, especially in highly sloped or uneven environments.

- **Improved Obstacle Tracking:** The current tracking approach is based on simple position differencing, which can be inaccurate under noisy or partial observations. Future improvements should explore fusing data from multiple sensors, such as cameras, stereo vision, or radar, with the existing LiDAR system. Additionally, more advanced tracking algorithms can be integrated to improve the estimation of obstacle velocity and motion consistency over time.

- **Real-World Field Testing:** Conducting long-term field trials across different seasons and vineyard plots would help evaluate the robustness of the system. These tests should include varying lighting, terrain, and environmental conditions to simulate real agricultural operations.

By addressing these points, the system can evolve into a more complete, terrain-aware, predictive, and adaptive navigation framework, suitable for safe and autonomous operation across a wide range of agricultural scenarios.

References

[1] M. Ryan, "Labour and skills shortages in the agro-food sector," *OECD Food, Agriculture and Fisheries Papers*, no. 189, 2023. DOI: 10.1787/ed758aab-en.

[2] R. Barua, S. Bhowmik, A. Banerjee, and D. Banerjee, "The emerging advancement of robotics in the modern agriculture field," in *Agriculture and Aquaculture Applications of Biosensors and Bioelectronics*, IGI Global Scientific Publishing, 2024, pp. 256–268.

[3] H. L. Nguyen, Q. H. Do, T. T. Le, and A. D. Phan, "Improved obstacle avoidance algorithm for autonomous mobile robot in smart agriculture," in *IOP Conference Series: Earth and Environmental Science*, vol. 804, IOP Publishing, 2021, p. 022 024. DOI: 10.1088/1755-1315/804/2/022024.

[4] L. C. Santos, F. N. Santos, E. J. Solteiro Pires, A. Valente, P. Costa, and S. Magalhães, "Path planning for ground robots in agriculture: A short review," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2020, pp. 61–66. DOI: 10.1109/ICARSC49921.2020.9096177.

[5] N. Sung, S. Kim, and N. Cho, "An efficient path planning algorithm using a potential field for ground forces," *Computation*, vol. 11, no. 1, 2023, ISSN: 2079-3197. DOI: 10.3390/computation11010012.

[6] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, pp. 120–254, 2023, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120254.

[7] S. K. Sahoo and B. Choudhury, "A review of methodologies for path planning and optimization of mobile robots," *Journal of Process Management and New Technologies*, vol. 11, pp. 122–140, Jan. 2023. DOI: 10.5937/jouproman2301122S.

[8] X. Song, "Research on genetic algorithm optimization for agricultural machinery operation path planning," *Applied Mathematics and Nonlinear Sciences*, vol. 9, Aug. 2024. DOI: 10.2478/amns-2024-1933.

[9] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013, ISSN: 0921-8890. DOI: 10.1016/j.robot.2013.09.004.

[10] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021. DOI: 10.1109/ACCESS.2021.3108177.

[11] X. Zhao, K. Wang, S. Wu, *et al.*, "An obstacle avoidance path planner for an autonomous tractor using the minimum snap algorithm," *Computers and Electronics in Agriculture*, vol. 207, p. 107 738, Apr. 2023. DOI: 10.1016/j.compag.2023.107738.

[12] D. Aghi, S. Cerrato, V. Mazzia, and M. Chiaberge, "Deep semantic segmentation at the edge for autonomous navigation in vineyard rows," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3421–3428. DOI: 10.1109/IROS51168.2021.9635969.

[13] I. Beloev, D. Kyuchukova, G. Georgiev, G. Hristov, and P. Zahariev, "Artificial intelligence-driven autonomous robot for precision agriculture," *Acta Technologica Agriculturae*, vol. 24, pp. 48–54, Mar. 2021. DOI: 10.2478/ata-2021-0008.

[14] J. Gai, L. Xiang, and L. Tang, "Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle," *Computers and Electronics in Agriculture*, vol. 188, p. 106 301, 2021, ISSN: 0168-1699. DOI: 10.1016/j.compag.2021.106301.

[15] D. Iberraken, F. Gaurier, J.-C. Roux, C. Chaballier, and R. Lenain, "Autonomous vineyard tracking using a four-wheel-steering mobile robot and a 2d lidar," *AgriEngineering*, vol. 4, no. 4, pp. 826–846, 2022, ISSN: 2624-7402. DOI: 10.3390/agriengineering4040053.

[16] W. Mao, H. Liu, W. Hao, F. Yang, and Z. Liu, "Development of a combined orchard harvesting robot navigation system," *Remote Sensing*, vol. 14, no. 3, 2022, ISSN: 2072-4292. DOI: 10.3390/rs14030675.

[17] A. S. Aguiar, F. Neves dos Santos, H. Sobreira, J. Boaventura-Cunha, and A. J. Sousa, "Localization and mapping on agriculture based on point-feature extraction and semiplanes segmentation from 3d lidar data," *Frontiers in Robotics and AI*, vol. 9, 2022, ISSN: 2296-9144. DOI: 10.3389/frobt.2022.832165.

[18] M. Mammarella, L. Comba, A. Biglia, F. Dabbene, and P. Gay, "Cooperation of unmanned systems for agricultural applications: A case study in a vineyard," *Biosystems Engineering*, vol. 223, pp. 81–102, 2022, New advances in measurement and data processing techniques for Agriculture, Food and Environment., ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2021.12.010.

[19] A. Choudhary, Y. Kobayashi, F. J. Arjonilla, S. Nagasaka, and M. Koike, "Evaluation of mapping and path planning for non-holonomic mobile robot navigation in narrow pathway for agricultural application," in *2021 IEEE/SICE International Symposium on System Integration (SII)*, 2021, pp. 17–22. DOI: 10.1109/IEEECONF49454.2021.9382767.

[20] C. Peng, P. Wei, Z. Fei, Y. Zhu, and S. G. Vougioukas, "Optimization-based motion planning for autonomous agricultural vehicles turning in constrained headlands," *Journal of Field Robotics*, vol. 41, no. 6, pp. 1984–2008, Jun. 2024, ISSN: 1556-4967. DOI: 10.1002/rob.22374.

[21] S. Jiang, S. Wang, Z. Yi, M. Zhang, and X. Lv, "Autonomous navigation system of greenhouse mobile robot based on 3d lidar and 2d lidar slam," *Frontiers in Plant Science*, vol. Volume 13 - 2022, 2022, ISSN: 1664-462X. DOI: 10.3389/fpls.2022.815218.

[22] J. Pak, J. Kim, Y. Park, and H. I. Son, "Field evaluation of path-planning algorithms for autonomous mobile robot in smart farms," *IEEE Access*, vol. 10, pp. 60 253–60 266, 2022. DOI: 10.1109/ACCESS.2022.3181131.

[23] J. Chen, H. Qiang, J. Wu, G. Xu, and Z. Wang, "Navigation path extraction for greenhouse cucumber-picking robots using the prediction-point hough transform," *Computers and Electronics in Agriculture*, vol. 180, p. 105 911, 2021, ISSN: 0168-1699. DOI: 10.1016/j.compag.2020.105911.

[24] S. P. Adhikari, G. Kim, and H. Kim, "Deep neural network-based system for autonomous navigation in paddy field," *IEEE Access*, vol. 8, pp. 71 272–71 278, 2020. DOI: 10.1109/ACCESS.2020.2987642.

[25] X. Liang, B. Chen, C. Wei, and X. Zhang, "Inter-row navigation line detection for cotton with broken rows," *Plant Methods*, vol. 18, no. 1, p. 90, 2022. DOI: 10.1186/s13007-022-00913-y.

[26] F. Sciascio, A. Rosales, G. Scaglia, V. Mut, and J. Nieto, "Trajectory tracking and collision avoidance of mobile robots by using numerical methods," in *Proceedings of the Argentine Conference on Informatics (JAIIO)*, Available: https://www.researchgate.net/publication/255704753, 2008.

[27] K. Svyatov, I. Rubtsov, R. Zhitkov, V. Mikhailov, A. Romanov, and A. Filippov, "Algorithm for controlling an autonomous vehicle for agriculture," in *2024 7th International Conference on Information Technologies in Engineering Education (Inforino)*, 2024, pp. 1–7. DOI: 10.1109/Inforino60363.2024.10551965.

[28] L. C. Santos, F. N. Santos, A. Valente, H. Sobreira, J. Sarmento, and M. Petry, "Collision avoidance considering iterative bézier based approach for steep slope terrains," *IEEE Access*, vol. 10, pp. 25 005–25 015, 2022. DOI: 10.1109/ACCESS.2022.3153496.

[29] A. Danton, J.-C. Roux, B. Dance, C. Cariou, and R. Lenain, "Development of a spraying robot for precision agriculture: An edge following approach," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, 2020, pp. 267–272. DOI: 10.1109/CCTA41146.2020.9206304.

[30] J. Wang, H. Zheng, Y. Yu, Y. He, and Y. Liu, "Robust multiple obstacle tracking method based on depth aware ocsort for agricultural robots," *Computers and Electronics in Agriculture*, vol. 217, p. 108 580, 2024, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2023.108580.

[31] S. Eiffert, N. D. Wallace, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Resource and response aware path planning for long-term autonomy of ground robots in agriculture," *Field Robotics*, vol. 2, pp. 1–33, 2022. DOI: 10.55417/fr.2022001.

[32] J. Yang, J. Ni, Y. Li, J. Wen, and D. Chen, "The intelligent path planning system of agricultural robot via reinforcement learning," *Sensors*, vol. 22, no. 12, 2022, ISSN: 1424-8220. DOI: 10.3390/s22124316.

[33] J. Wang, L. Yang, H. Cen, Y. He, and Y. Liu, "Dynamic obstacle avoidance control based on a novel dynamic window approach for agricultural robots," *Computers in Industry*, vol. 167, p. 104 272, 2025, ISSN: 0166-3615. DOI: 10.1016/j.compind.2025.104272.

[34] Y. Yu, Y. Liu, J. Wang, N. Noguchi, and Y. He, "Obstacle avoidance method based on double dqn for agricultural robots," *Computers and Electronics in Agriculture*, vol. 204, p. 107 546, 2023, ISSN: 0168-1699. DOI: 10.1016/j.compag.2022.107546.

[35] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Department of Computer Science, Tech. Rep., 1995.