

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Velocity Obstacle Avoidance Algorithm for Heterogeneous Robotic Vehicles

Francisco Pires da Silva



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Andry Maykol Gomes Pinto

August 1, 2025

Resumo

Veículos robóticos não tripulados são cada vez mais uma solução eficaz para operações de inspeção e manutenção, reduzindo custos e riscos para os operadores humanos. Garantir a navegação segura desses veículos requer métodos de desvio de obstáculos que lidem com obstáculos estáticos e dinâmicos em espaços tridimensionais, calculando manobras que previnam colisões. Esses algoritmos devem ser generalizáveis para diferentes tipos de veículos, mas configuráveis de acordo com as limitações cinemáticas de cada um.

Este trabalho apresenta o desenvolvimento e validação do JAVOA (3D Adaptive Velocity Obstacle Algorithm), um algoritmo de desvio de obstáculos para veículos heterogêneos em ambientes dinâmicos e incertos. O JAVOA utiliza uma estrutura de amostragem do espaço de velocidades gerada a partir de restrições cinemáticas em 6 graus de liberdade, possibilitando a adaptabilidade a diferentes veículos robóticos. Métodos de detecção e tracking de obstáculos foram integrados numa representação local do ambiente, e o algoritmo desenvolvido foi extensivamente testado em simulações com diversos tipos de veículos robóticos e em cenários marítimos reais num veículo de superfície no ATLANTIS Test Center. Os resultados obtidos validam a abordagem como adequada e robusta para representar ambientes dinâmicos e selecionar preventivamente manobras de desvio de obstáculos.

Abstract

Uncrewed robotic vehicles are an effective solution for inspection and maintenance tasks, reducing operational costs and risks for human operators. Ensuring the safe navigation of these vehicles requires obstacle avoidance methods that address both static and dynamic obstacles in three-dimensional spaces, calculating maneuvers that prevent collisions. These algorithms must be generalizable to different types of vehicles but configurable to the kinematic limitations of each one.

This work presents the development and validation of JAVOA (3D Adaptive Velocity Obstacle Algorithm), an obstacle avoidance algorithm for heterogeneous vehicles in dynamic, uncertain environments. JAVOA uses a velocity space sampling framework generated from 6 Degrees of Freedom kinematic constraints, enabling adaptability across heterogeneous robotic vehicles. Obstacle detection and tracking methods were integrated into a local environment representation, and the developed algorithm was extensively tested in simulations for various types of robotic vehicles and in real maritime scenarios on a surface vehicle at the ATLANTIS Test Center. The results obtained validate the approach as suitable and robust for representing dynamic environments and preemptively selecting obstacle avoidance maneuvers.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a set of 17 goals for achieving a better, safer, more inclusive, sustainable, and efficient future. This work aligns with the following SDGs:

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

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 14 Conserve and sustainably use the oceans, seas and marine resources for sustainable development

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

Target 8.8 Protect labour rights and promote safe and secure working environments for all workers, including migrant workers, in particular women migrants, and those in precarious employment

Target 9.4 By 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities

Target 14.2 By 2020, sustainably manage and protect marine and coastal ecosystems to avoid significant adverse impacts, including by strengthening their resilience, and take action for their restoration in order to achieve healthy and productive oceans

SGD	Target	Contribution	Performance Indicators and Metrics
8	8.8	Development of safe obstacle avoidance algorithms to minimize human error in vehicles operation	Reduce the number of incidents leading to injuries or fatalities in hazardous workplace environments
9	9.4	Enhancing autonomous vehicle systems for inspection and maintenance tasks, enabling more frequent operations and increasing overall industrial efficiency	Increase in operational efficiency and infrastructure longevity; Reduction in downtime and operational costs
14	14.2	Mitigate the impacts of vehicle operation in marine and coastal ecosystems	Reduction in operation incidents that affect marine ecosystems

Official Acknowledgments

This work is partly funded by the European Union's Horizon 2024 research and innovation programme under grant agreement No 101189723 (AEROSUB).

Francisco Pires da Silva

Agradecimentos

Gostaria de manifestar os meus agradecimentos, em primeiro lugar, ao Prof. Andry, pela confiança nas ideias que propus ao ponto de possibilitar testá-las em veículos reais e em ambiente real, o que foi mesmo espetacular.

Um obrigado especial ao Daniel, que eu sei que as "dúvidas de 5 segundos" que fui fazendo se acumularam em muitas horas. Agradeço cada uma delas e tudo o que fui aprendendo.

Um obrigado metafórico à máquina de café do CRAS, pelas conversas que me mantiveram motivado e a cafeína que me manteve acordado.

Um obrigado a todo o MOTUS: Dionísio, Rafa, Luís, Maria, Leite, Celso, MecaPedro, Francisco, Campanhã, Tipinho e Lupinho, pelo apoio psicológico para lidar com documentações, plugins, e especialmente pelo esforço físico de baterias de 40 kg a flutuadores puxados com cordas ou empurrados à mão.

Aos bambis Bruno Costa (obstáculo estático), Alexandre Costa (obstáculo dinâmico) e Tiago Claro, cujo génio, às vezes assustador, ajudou muitas vezes a por as minhas coisas a funcionar. Obrigado Bárbara, por me aturares a falar desta tese praticamente todos os dias.

E por último aos meus pais e à minha irmã, por perguntarem incansavelmente como estava a correr e se podiam ajudar com alguma coisa.

Francisco Pires da Silva

“For that very reason they’re necessarily plastic, malleable,
subject to error and distortion.”

“Fallible,” I said.

“But without fallibility there is no art.
And without art there is no truth.”

Alastair Reynolds, *Zima Blue and Other Stories*

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Objectives	3
1.3	Document Structure	3
2	Literature Review	4
2.1	Environment Perception	5
2.1.1	Object-Centric Representations, Obstacle Detection and Obstacle Tracking	6
2.2	Planning	8
2.2.1	Obstacle Avoidance methods overview	9
2.2.2	Velocity Obstacles	14
2.3	Critical Review	27
3	JAVOA: 3D Adaptive Velocity Obstacle Algorithm	29
3.1	Introduction	29
3.2	Velocity Space representation based on Obstacle Detection	31
3.3	Sampling-Based Obstacle Avoidance	35
3.4	JAVOA Considerations	44
4	Methodology and Results	45
4.1	Introduction	45
4.2	Heterogeneous Vehicle Simulation	46
4.2.1	Evaluation Metrics	48
4.2.2	Defined Parameters and Testing Scenarios	48
4.2.3	2D Holonomic Vehicle Results	51
4.2.4	2D Diff-Drive Vehicle Results	57
4.2.5	3D Multirotor Simulation Results	61
4.2.6	Ideal Information vs Simulated LiDAR	64
4.3	NEST Simulation Environment	67
4.3.1	Defined Parameters and Testing Scenarios	69
4.3.2	Simulated USV Results Analysis	74
4.4	Real World Testing	75
4.4.1	Real-World Results	76
5	Conclusions	81
5.1	Future Work	81
	References	83

List of Figures

2.1	Software for Autonomous Vehicles	5
2.2	Volumetric Representation of the environment	6
2.3	Extracting a circle from a segment	7
2.4	Topologic representation of a port environment	8
2.5	An example of a potential field with local and global minimum.	10
2.6	VFH Polar Histogram	10
2.7	Predictive DWA Collision Avoidance	11
2.8	Object Avoidance with Bendy Ruler	12
2.9	Agent and moving obstacle	14
2.10	Minkowski Sum, relative velocities, and Collision Cone	16
2.11	Feasible and Reachable Avoidance Velocities	17
2.12	Reciprocal Velocity Obstacle	18
2.13	ORCA halfplane	19
2.14	Feasible velocity limited by non-holonomic constraints	19
2.15	HRVO example	21
2.16	Uncertainty-adjusted Velocity Obstacle	21
2.17	Non-holonomic tracking error	22
2.18	AVOA protective zones for static and dynamic obstacles	24
2.19	Two aligned agents	25
2.20	Closely Spaced Agents	25
2.21	Buffer Velocity Set	26
2.22	Avoidance planes	27
3.1	Examples of obstacles in maritime environments	30
3.2	Autonomous Vehicle Navigation with Obstacle Avoidance Pipeline	31
3.3	Obstacle radius as a sum of its estimated radius, radius standard deviation, and position standard deviation.	32
3.4	Obstacle detection for different shapes	33
3.5	Omnidirectional surface vehicle velocity sampling space	36
3.6	Diff-Drive surface vehicle velocity sampling space	37
3.7	Top view of the multicopter sampling space.	38
3.8	Side view of the multicopter sampling space.	38
3.9	Comparison between full and truncated Velocity Obstacle representations.	39
3.10	Pruned Samples from a holonomic planar vehicle when facing a static obstacle	41
3.11	Heading cost	42
3.12	Magnitude cost	42
3.13	Danger cost for static and moving obstacles	43
3.14	Examples of velocity sample cost evaluation	44

4.1	Simple Simulation environment	46
4.2	Pipeline Overview for the Simplified Gazebo Simulation	47
4.3	Holonomic Scenario S0: XY robot trajectory	52
4.4	Holonomic Scenario S0: Velocity components over time	52
4.5	Holonomic Scenario S0: Velocity magnitude over time	52
4.6	Holonomic Scenario D0: XY robot trajectory	53
4.7	Holonomic Scenario D0: Velocity components over time	53
4.8	Holonomic Scenario D0: Velocity magnitude over time	53
4.9	Holonomic Scenario D1: XY robot trajectory	54
4.10	Holonomic Scenario D1: Velocity components over time	54
4.11	Holonomic Scenario D1: Velocity magnitude over time	54
4.12	Holonomic Scenario C4: XY robot trajectory	55
4.13	Holonomic Scenario C4: Velocity components over time	55
4.14	Holonomic Scenario C4: Velocity magnitude over time	55
4.15	Diff-Drive Scenario S0: XY robot trajectory	58
4.16	Diff-Drive Scenario S0: Command Velocity components over time	58
4.17	Diff-Drive Scenario S0: Command Velocity magnitude over time	58
4.18	Diff-Drive Scenario D0: XY robot trajectory	59
4.19	Diff-Drive Scenario D0: Command Velocity components over time	59
4.20	Diff-Drive Scenario D0: Velocity magnitude over time	59
4.21	Diff-Drive Scenario D1: XY robot trajectory	60
4.22	Diff-Drive Scenario D1: Velocity components over time	60
4.23	Diff-Drive Scenario D1: Velocity magnitude over time	60
4.24	Multirotor Scenario C4: 3D robot trajectory	62
4.25	Multirotor Scenario C4: XY trajectory projection	62
4.26	Multirotor Scenario C4: XZ trajectory projection	62
4.27	Multirotor Scenario C4: Velocity magnitude over time	62
4.28	Multirotor Scenario C4: Velocity components in 3D	62
4.29	Multirotor Scenario D3: 3D robot trajectory	63
4.30	Multirotor Scenario D3: XY trajectory projection	63
4.31	Multirotor Scenario D3: XZ trajectory projection	63
4.32	Multirotor Scenario D3: Velocity magnitude over time	63
4.33	Multirotor Scenario D3: Velocity components in 3D	63
4.34	Detection-based Scenario S0: Velocity Components.	64
4.35	Detection-based Scenario S0: XY agent and obstacle trajectories.	64
4.36	Detection-based Scenario D0: Velocity Components.	65
4.37	Detection-based Scenario D0: XY agent and obstacle trajectories.	65
4.38	Detection-based Scenario D1: Velocity Components.	65
4.39	Detection-based Scenario D1: XY agent and obstacle trajectories.	65
4.40	Detection-based Scenario D0: Estimated obstacle velocity magnitude over time.	65
4.41	Simulation environment and vehicle configuration	67
4.42	Pipeline Overview for the Complex Gazebo Simulation	68
4.43	Control System Architecture for Velocity to Thrust Conversion	68
4.44	NEST Scenario S0: XY robot trajectory	71
4.45	NEST Scenario S0: Velocity components	71
4.46	NEST Scenario S0: Velocity magnitude over time	71
4.47	NEST Scenario D0: XY robot trajectory	72
4.48	NEST Scenario D0: Velocity components	72

4.49	NEST Scenario D0: Velocity magnitude over time	72
4.50	NEST Scenario D1: XY robot trajectory	73
4.51	NEST Scenario D1: Velocity components	73
4.52	NEST Scenario D1: Velocity magnitude over time	73
4.53	NEST USV deployed at ATLANTIS Test Center in Viana do Castelo, Portugal	75
4.54	Assisted Teleoperation Pipeline implemented for real-world testing in ATLANTIS Test Centre	76
4.55	Real-World Scenario 1	77
4.56	Real-World Scenario 2	77
4.57	Real-World Scenario 3	78
4.58	Real-World Scenario 4	78
4.59	Real-World Scenario 5	79
4.60	Real-World Scenario 6	79
5.1	Goal behind but close to wall problem.	82
5.2	Goal before obstacle problem.	82

List of Tables

2.1	Symbolic notation used to describe VO approaches	15
3.1	Obstacle Message Definition with Uncertainty Parameters	32
3.2	Maximum angular and linear velocity and acceleration values for an example omnidirectional surface vehicle.	36
3.3	Maximum angular and linear velocity and acceleration values for an example diff drive surface vehicle.	37
3.4	Maximum angular and linear velocity and acceleration values for an example multicopter vehicle.	38
4.1	Configuration parameters for the Obstacle Avoidance Layer for Heterogeneous Vehicle Simulation.	49
4.2	Configuration parameters for the Navigation Layer for Heterogeneous Vehicle Simulation.	49
4.3	2D Scenarios with corresponding visual, label, and description.	50
4.4	Holonomic Scenario S0 Metrics Table	52
4.5	Holonomic Scenario D0 Metrics Table	53
4.6	Holonomic Scenario D1 Metrics Table	54
4.7	Holonomic Scenario C4 Metrics Table	55
4.8	Diff-Drive Scenario S0 Metrics Table	58
4.9	Diff-Drive Scenario D0 Metrics Table	59
4.10	Diff-Drive Scenario D1 Metrics Table	60
4.11	Multirotor Scenario C4 Metrics Table	62
4.12	Multirotor Scenario D3 Metrics Table	63
4.13	Configuration parameters for the Obstacle Avoidance Layer for Simulated NEST.	69
4.14	Configuration parameters for the Navigation Layer for Simulated NEST.	69
4.15	PID controller gains for Simulated NEST.	69
4.16	2D Scenarios with corresponding visual, label, and description.	70
4.17	NEST Scenario S0 Metrics Table	71
4.18	NEST Scenario D0 Metrics Table	72
4.19	NEST Scenario D1 Metrics Table	73

List of Acronyms

2D	Two-dimensional
3D	Three-dimensional
6DOF	Six Degrees Of Freedom
APF	Artificial Potential Fields
ASV	Autonomous Surface Vehicle
AVOA	Adaptive Velocity Obstacle Avoidance
CC	Collision Cone
CL	Center Line
DRL	Deep Reinforcement Learning
FA	Feasible Accelerations
FEUP	Faculdade de Engenharia da Universidade do Porto
GVO	Generalized Velocity Obstacle
ID	Identifier
ICAO	International Civil Aviation Organization
LiDAR	Light Detection and Ranging
MPC	Model Predictive Control
NLVO	Nonlinear Velocity Obstacle
ORCA	Optimal Reciprocal Collision Avoidance
PTEB	Proactive Timed Elastic Band
PVO	Probabilistic Velocity Obstacle
PZ	Protective Zone
RAV	Reachable Avoidance Velocities
RL	Reinforcement Learning
RU	Relative Utility
RVO	Reciprocal Velocity Obstacle
RVP	Relative Velocity Paradigm
RV	Reachable Velocities
SDG	Sustainable Development Goals
TEB	Timed Elastic Band
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
UP	Universidade do Porto
UxV	Unmanned Vehicle
VO	Velocity Obstacle

Chapter 1

Introduction

The use of renewable energies has been growing toward the development of offshore wind farms for several reasons: the limited space on land and its environmental impact, wind speeds increasing with the distance from the coast and reduced turbulence effects, which allow for higher energy production, less visual impact on the landscape, and lower noise impact [1]. The European Union aims to achieve 111GW of offshore renewable energy by 2030, increasing to 317GW by 2050¹. This increasing number of offshore wind farms brings the need for operation and maintenance services to enable early detection of defects and preventive maintenance, therefore minimizing downtime, preventing structural failures, extending the lifespan of turbines, and therefore reducing costs and optimizing energy production [2]. In Europe, there is not only a trend in increasing the number of offshore wind plants but also in their capacity, size, distance to shore, and water depth [3].

Currently, inspection and maintenance operations in offshore wind farms are based on human operators working in dangerous environments. The G+ Offshore Wind 2023 Incident Data Report [4] identifies 1679 incidents, including one fatality. Accidents involving Service Operation Vessels, Jack-up vessels, and Crew Transfer Vessels account for 25% of these accidents. Operating vessels, working at heights, and harsh weather conditions are not only responsible for these safety concerns but also for specialized equipment and resources that contribute to high operational costs. As a result, projects such as TALOS² and AEROSUB³ are designing and demonstrating unmanned robotic solutions for these inspection and maintenance operations. Unmanned Vehicles (UxVs) such as Unmanned Aerial Vehicles (UAVs), Unmanned Surface Vehicles (USVs), and Unmanned Underwater Vehicles (UUVs) are being deployed in offshore sites, leading to safer operations, improved efficiency, and reduced costs and environmental impact.

Different types of vehicles must be used to tackle the challenges of their different operating domains and technologies. It allows different vehicles to specialize in different aspects and tasks, complementing strengths and compensating for limitations. The advantages include operating on

¹<https://energy.ec.europa.eu/> (Accessed on June 25, 2025)

²<https://talosproject.eu/about/> (Accessed on June 25, 2025)

³<https://www.enlit.world/renewable-energy> (Accessed on June 25, 2025)

different domains: above and underwater, improving localization and navigation, extending operational times, and increasing safety and redundancy. The different characteristics and environments of the different types of vehicles must be addressed to ensure safe navigation, and path planning and obstacle avoidance techniques must consider not only their different constraints but also the interactions between them. Ensuring safe navigation and interactions among robotic vehicles for offshore wind farm operations is necessary to make the unmanned approach possible and reduce the costs and risks associated with human operators. Safer inspection and maintenance enable lower maintenance costs, reduced downtimes, and improved efficiency and lifespan of wind turbines. Developments in these areas lead to optimizing the deployment of offshore wind farms, contributing to the transition to renewable energy sources.

1.1 Motivation

Obstacle avoidance methods must be implemented on a navigation pipeline to ensure the safety of the autonomous vehicles and their environment. In real-world navigation scenarios involving interactions between different vehicle types, agents and obstacles must be modeled and mapped in the 3D (three-dimensional) space. While static obstacles are easier to predict and plan for, dynamic obstacles, such as other vessels, pose a more significant challenge. To address dynamic environments, an obstacle avoidance algorithm should have a preemptive behavior, i.e., enabling the system to predict and plan for potential collisions, and perform an early action accordingly. One other issue is that real-time detection systems have limitations that add uncertainties in obstacle detection and prediction, such as sensor noise, environmental conditions (waves and wind), occlusions, and limited sensor range. For dynamic obstacles, there is an added challenge in predicting behavior, which can range from simple linear motion to a complex avoidance strategy. Lastly, the kinematic and dynamic constraints of robotic vehicles must be considered: A planned maneuver must not only avoid obstacles but also remain feasible given the specific characteristics and motion capabilities of the vehicle. All of these challenges have to be addressed while still maintaining real-time feasibility in terms of computational power.

Considering these different challenges, generic approaches for obstacle avoidance that ensure safety in both static and dynamic 3D scenarios, employing collision-preemptive behavior, are needed. The algorithm's design should be vehicle-agnostic, but at the same time, implementations of the algorithm should reflect the specific vehicle's characteristics and constraints.

This thesis is focused on these requirements by building upon the Velocity Obstacle approach [5], as a means to directly integrate the velocity of the obstacles into the configuration space. Lastly, the integration of detection systems with planning methods will be addressed, using LIDAR-based methods to obtain information about the obstacles' position, size, and velocity.

1.2 Objectives

The primary objective of this thesis is to develop and validate a dynamic obstacle avoidance algorithm for heterogeneous autonomous vehicles that navigate in 3D environments and deal with obstacles moving in the same space. The main challenges to be addressed are: integrating real-world LiDAR detection systems into a velocity space environment representation, planning safe avoidance maneuvers in the 3D space, and constraining dynamic and kinematic limitations. To address these challenges, this thesis will focus on the following objectives:

- Formulate a 3D Velocity Obstacle approach able to incorporate heterogeneous kinematic constraints;
- Integrate LiDAR-based perception methodologies into a local environment representation;
- Validate the methodology in simulated scenarios for different vehicles and environments;
- Validate the algorithm in near-real scenarios in the ATLANTIS Test Center using a real robotic surface vehicle.

1.3 Document Structure

This document is structured as follows: Chapter 2 contains a literature review on the problem of obstacle avoidance, and the state-of-the-art environment perception and motion planning methods for dynamic environments, with a focus on Velocity Obstacles approaches. Chapter 3 describes the proposed algorithm, JAVOA and its Velocity Space representation based on Obstacle Detection. Chapter 4 describes the followed methodology and experimental setup, and the results that validate the proposed approach. Chapter 5 presents the main conclusions and contributions and future work to be developed.

Chapter 2

Literature Review

Obstacle avoidance is essential to ensure the safety of unmanned robotic vehicles and the surrounding environment. Path-planning approaches must be able to work in dynamic and complex environments, where heterogeneous robotic vehicles share the same space, such as busy ports or shared workspaces for inspection, maintenance, or transportation operations. Navigation in dynamic environments is a complex task, and even human operators need safety regulations for navigation maneuvers on head-on approaches, overtaking, and crossing paths, such as COLREGs¹ (Convention on the International Regulations for Preventing Collisions at Sea) or ICAO (International Civil Aviation Organization)² [6].

The software to provide autonomous robotic vehicles with these capabilities can be decomposed into three categories: Environment Perception, Planning, and Control [7]. The first regards gathering information from the environment, such as where obstacles are located, and where in the environment the vehicle is; The second step, Planning, is a decision-making process that follows an objective, like reaching a goal location while avoiding obstacles; Lastly, the Control step ensures the plan is followed. For dynamic environments, prediction software can be added to track obstacles and anticipate movements so planning can be done preemptively. An overview of this process that connects sensor data to actuator commands is represented in figure 2.1, courtesy from Prof. Dr.-Ing. Johannes Betz [8]. Considering that this dissertation focuses on obstacle avoidance, the steps in the pipeline that will be addressed are the detection and mapping of the obstacles (Section 2.1) and the motion planning for the obstacle avoidance itself (Section 2.2).

This chapter proceeds with a discussion of different methodologies for scene representation and how they fit for obstacle avoidance. After that, an overview of the main obstacle avoidance algorithms is presented, with a focus on Velocity Obstacles (VO) and the related improvements as well as implementations.

¹<https://www.imo.org> (Accessed on June 25, 2025)

²<https://www.icao.int/> (Accessed on June 25, 2025)

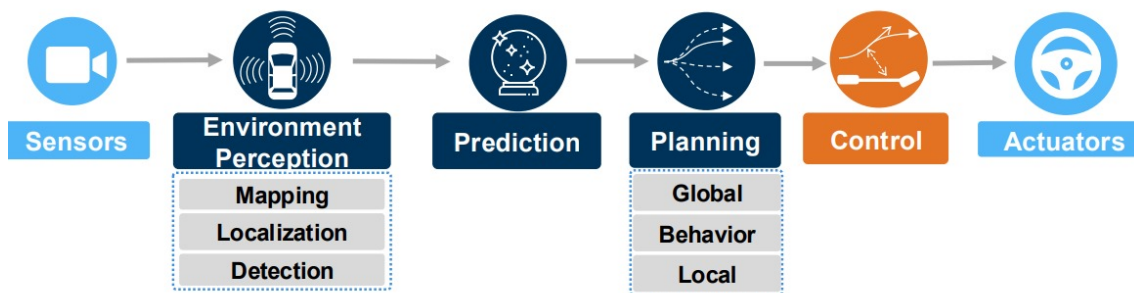


Figure 2.1: Software for Autonomous Vehicles [8]

2.1 Environment Perception

Autonomous vehicles rely on sensors to gather information about the surrounding environment and perceive their surroundings. For autonomous navigation, LiDAR (Light Detection And Ranging) and cameras are the most common, followed by short/long-range radars and ultrasonic sensors [7].

Environment perception refers to the ability of autonomous systems to collect information from the environment, developing a contextual understanding of the surroundings [7], [9], [10]. The perception layer of an autonomous robot encompasses mainly 3 tasks: Detection - where objects are located in the environment; Localization - positioning of the robot in the environment; and Mapping - building local knowledge about the objects in the scenario. Often, Mapping involves incomplete Scene Representations, due to occlusions, limited field of view, and sensor noise, and therefore scene completion techniques [11] and fusion techniques [12] are used to derive a more detailed representation from the sensor data.

For autonomous vehicle navigation, Scene Representation is the most important step of environment perception, as it serves as the input for planning and obstacle avoidance algorithms. Mascaro and Chli [10] distinguish three levels of scene information: metric, semantic, and topological. These levels are used to categorize 3D representations into metric, metric-semantic, and metric-semantic-topological.

Metric representations focus on the geometry of an environment and are usually enough to support simple navigation and obstacle avoidance tasks. One type of metric representation is Feature-based representation, which identifies and maps specific distinctive features in the environment, involving less computational power and memory. Dense point-based representations, on the other hand, use point clouds to capture and store in detail the full 3D geometry of the scene [13]. Surface mesh representations decompose the environment into sets of polygons to represent the surface of objects. Other types of Metric representations include Neural and Radiance Field representations. The last type of metric representation is volumetric representation, which models the 3D geometry and the free space within the observed environment.

Beyond geometry information, incorporating semantic information allows for the identification and semantic labelling of the different elements in the scene, allowing autonomous agents to

better understand their environment. Metric-Semantic representations can be divided into object-centric, semantically-labeled, panoptic, and open-set representations [10]. Object-centric representations isolate a set of objects of interest from the background, identifying and reconstructing them. Semantically labeling focuses on assigning class labels or the probability of classes to data elements. Panoptic representations incorporate labeling for background regions, and Open-set representations allow new categories outside the predefined set to be recognized. Lastly, to address complex tasks with high-level task planning, Metric-Semantic-Topological Representations incorporate information about relationships between different objects, either at a single level of abstraction or at a hierarchical level, understanding the relationships between "*objects, places, rooms, and buildings*" [10].

For simple navigation tasks like obstacle avoidance in static environments, typically, metric representations are enough, as these tasks are based on geometric information. Classical volumetric representations include occupancy grid-based representations, commonly used for these tasks, explicitly distinguishing free and occupied space. Voxel grids are 3D grids that encode the information from the environment in a grid of equal-sized cells (voxels). As this data structure is memory expensive, Hornung *et al* (2013) [14] proposes OctoMaps, which use octrees and probabilistic occupancy estimation to represent a 3D model of the environment, storing occupancy information at different resolutions in a much more efficient manner for map insertions and finding nearby points. Figure 2.2, taken from [15] illustrates an example of a volumetric 3D grid-based representation of obstacles. Grid-based methods have challenges in memory and computation requirements, as well as discretization inaccuracies.

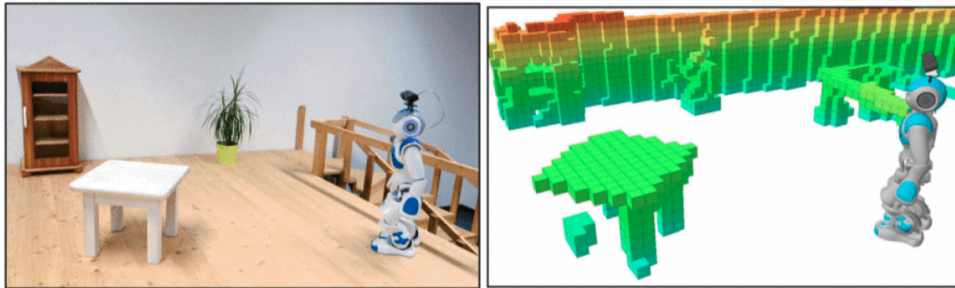


Figure 2.2: Volumetric Representation of the environment [15].

2.1.1 Object-Centric Representations, Obstacle Detection and Obstacle Tracking

Dynamic obstacles pose a new level of difficulty for environment representations, and of course, navigation. It is necessary to isolate and track individual elements from the environment to estimate their velocity, or even reason about interactions with them using semantic information.

P. Y. Leong *et al.* (2024) [16] state how LiDAR is one of the most commonly used sensors for autonomous vehicles navigation and obstacle avoidance. LiDAR sensors provide real-time distance measurement, with a large field of view, enabling fast and precise real-time 3D representations, therefore being a common solution for these problems. The authors also describe

how LiDAR limitations can be addressed with sensor fusion: Object Detection capabilities can be added using trained object classification models, but also by using cameras, a "natural complement to LiDARs" [16]. One other combination of sensors is LiDAR and Radar, since Radar is able to provide precise obstacle velocity information, but has low precision in determining position, creating a robust system that is commonly used in dynamic environment tasks such as autonomous driving.

2D LiDARs, despite the obvious disadvantage of capturing data in a single plane, are often preferred due to lower cost, power consumption, and simplified processing requirements, and still provide good capability on detecting and tracking obstacles.

M. Przybyla (2017) [17] proposes a method for detecting and tracking obstacles from 2D laser scanner data. The environment representation consists of circular and line segment obstacles. The proposed method for object extraction from Laser Scan data consists on the following steps: *Grouping* consecutive points according to distance into subsets; *Splitting* the subsets into smaller segments; *Segmentation*, approximating the subsets with line segments; *Merging* segments that have similar parameters to reduce redundancy; *Circles extraction* from the line segments. To extract the circular objects, an equilateral triangle is constructed coinciding with the line segment. The central point of this triangle will define the circle center, and the radius r_n is the distance to one of the points, as in expression 2.1. where l_n is the segment length. An example can be seen represented in figure 2.3. The author [17] also proposes an approach for obstacle tracking to improve the quality of the obstacle detection and enable velocity estimation. After each scan, detected obstacles are matched with previous detections solving an assignment problem, with the possibility for *obstacle fusion* and *obstacle fission*, merging two obstacles into one and breaking one obstacle into two, respectively. To update the obstacle information, a separate Kalman Filter is employed for each of the obstacles, with a prediction assuming constant linear velocity.

$$r_n = \frac{\sqrt{3}}{3} l_n \quad (2.1)$$

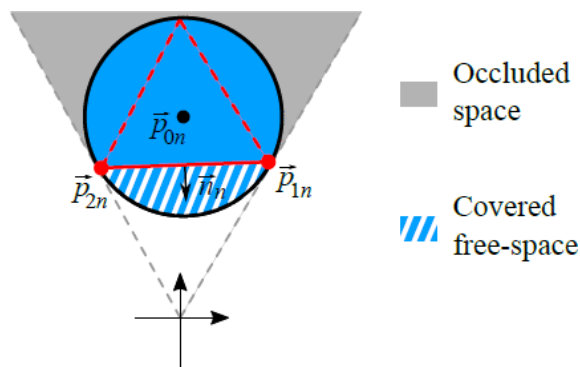


Figure 2.3: Extracting a circle from a segment. Image taken from [17].

Campos (2024) [9] proposes a metric-semantic-topologic scene representation framework for autonomous navigation in maritime environments that maps obstacles according to an object class (e.g., dock, vessel, foundation), size, pose with covariance, and velocity. Figure 2.4, taken from

[9], shows this topological representation of a port environment at ATLANTIS, as well as the USV's trajectory during this representation, and using a vessel detection and tracking method [18]. This framework allows for distinguishing between static and dynamic obstacles, their velocities, and semantic meaning. It allows for improved decision-making capabilities on autonomous systems and much more complex and safer obstacle avoidance and path planning approaches.

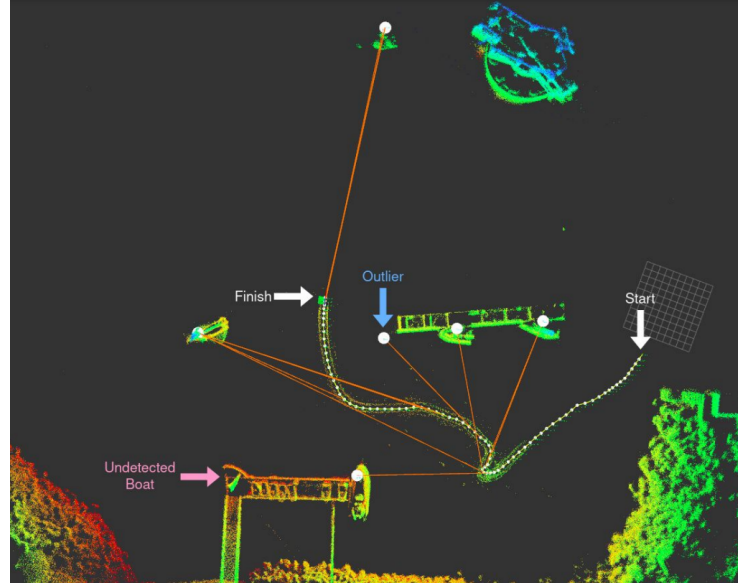


Figure 2.4: Topologic representation of a port environment [9].

2.2 Planning

Although there is no standard framework for planning for autonomous vehicles, a common and promising strategy is to implement a "*three-level hierarchical planning framework*" [7], consisting of having a *mission planner* (or route / global planner), a *behavioral planner* (or decision maker), and a *motion planner* (or local planner).

The *Mission planning* layer concerns addressing high-level objectives, such as what goal locations to consider or which path to take. Behavior planning generates local objectives, making the decisions important for interacting with other agents or following traffic and navigation rules while making progress along the mission planner's output. Setting local goal objectives, virtual obstacles, and finite state machines are common strategies for this layer. Motion/Local planning focuses on the local objectives and the appropriate paths and actions that lead to the best outcome. This layer has diverse applications in robotic systems, generating the paths that lead to the specified goal while avoiding obstacles and ensuring safety. The performance of these methods depends on computational efficiency and completeness - if an existing safe solution is always returned. In dynamic environments, the robot needs to constantly update for changes in the surroundings and account for uncertainties in this process. To plan in a time horizon, robots need to make some assumptions on how the dynamic environment is going to behave, such as constant velocity, bounded

velocity, or bounded acceleration, but more complex decision-making structures can be made in the behavior planning layer. Motion planning can be addressed directly in the configuration space, or the control space, assessing which controls lead to a collision. In addition, restrictions according to the control limitations of the robot must be made, to ensure that only feasible controls are selected.

Regarding multi-agent system operations, Vesentini *et al* [19] distinguish centralized from decentralized methods. Centralized motion planning involves all agents sharing their information at every instant with a common processing unit. These methods do not scale well in crowded scenarios, becoming computationally heavier as the number of agents increases. Additionally, a global communication system may be unfeasible in real-world applications. Lastly, they are dependent on complete knowledge of the environment, making them unsuitable for situations with unexpected or dynamic obstacles. Decentralized methods, which do not use a shared processing unit, rely on information gathered from the environment and other vehicles, such as position and velocity. However, they face greater challenges in dealing with heterogeneous agents and do not guarantee that all agents will reach their goals, as they may become stuck in local minima.

2.2.1 Obstacle Avoidance methods overview

Motion planning and obstacle avoidance is a very broad field of research, with applications from robotics to manufacturing, medical and security systems [7]. Extensive research has been conducted in this field, presenting a diverse set of approaches. This section proceeds with a description of some of the most widely used and efficient motion planning approaches for obstacle avoidance of mobile robots in dynamic environments.

Artificial Potential Fields (APF) [20] is a collision avoidance method that uses virtual forces created by the environment: obstacles create repulsive forces, making robot A avoid them, and the goals create attractive forces, making A go towards them. The resulting potential field $U(t)$ is the sum of all attractive and repulsive potentials. The optimal control is then given according to the negative gradient of the APF, $-\nabla U(t)$, by using a controller proportional to this force, for example. This method is simple and fast but prone to get stuck in a local minimum of the artificial force field. Figure 2.5, taken from [21], represents a Local Minimum created by a U-shaped obstacle where an agent using APF might become stuck. More recent developments focus on adapting APF for dynamic environments and overcoming the local minimum challenges, using genetic or optimization algorithms [22].

Vector Field Histograms (VFH) [23] start with a two-dimensional Cartesian histogram grid, where each cell is filled by range sensors, storing a probability of that cell containing an obstacle. This grid is then reduced to a polar histogram: For each angle step, the value is a function of the cells in that direction, their probability, and their distance to the robot (Equation 2.2):

$$m_{i,j} = (ci, j^*)^2(a - bd_{i,j}). \quad (2.2)$$

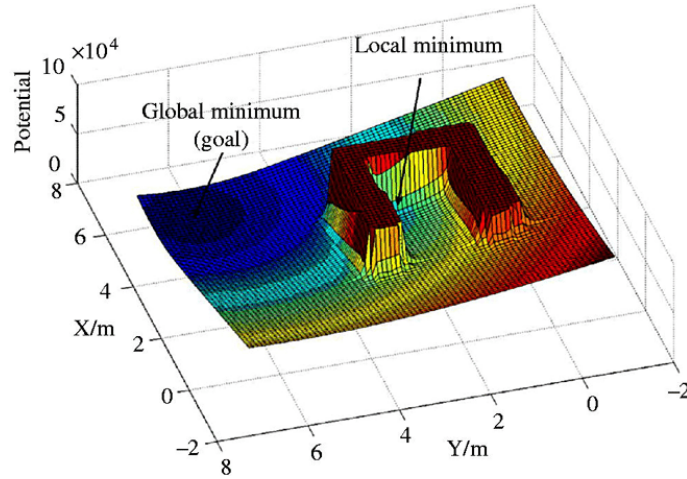


Figure 2.5: An example of a potential field with local and global minimum. Taken from [21]

where $m_{i,j}$ is the magnitude of cell (i, j) , $c_{i,j}^*$ is the certainty value, $d_{i,j}$ is the distance to the robot center, and a, b are constants. Lastly, the algorithm chooses the updated control direction between the candidate valleys, determined by a threshold, represented in figure 2.6, adapted from [24].

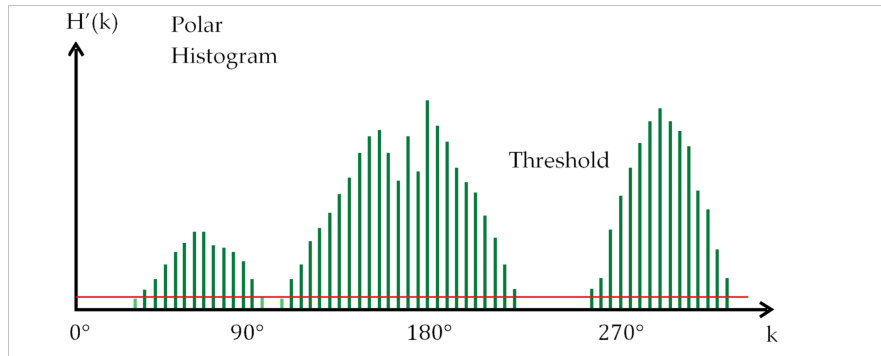


Figure 2.6: VFH Polar Histogram [24].

The Dynamic Window Approach (DWA) [25] approximates trajectories by circular curvatures, creating a search space on translational and rotation velocities (v_i, ω_i) that leads the robot to a safe stopping. Assuming a robot in a 2D world (x, y) and a finite number of acceleration commands, the equations of motion can be simplified to the discrete domain, between t_0 and t_n . Lastly, approximating the velocities within a time interval by a constant value obtains equation 2.3:

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} (F_x^i(t_{i+1})), \quad (2.3)$$

where F_x^i is according to equation 2.4:

$$F_x^i(t) = \begin{cases} \frac{v_i}{\omega_i} (\sin(\theta(t_i)) - \sin(\theta(t_i) + \omega_i \cdot (t - t_i))), & \omega_i \neq 0, \\ v_i \cdot \cos(\theta(t_i)) \cdot t, & \omega_i = 0. \end{cases}, \quad (2.4)$$

and an analogous expression can be obtained for the y coordinate. These approximated equations of motion allow DWA to perform a search on the velocity space (v_i, ω_i) , and infer the best velocity command. However, this search is first reduced to a "window" of velocities around the current velocity, which are reachable under the dynamic constraints, and second to those that are considered safe. The kinematic restrictions are based on the given limited accelerations of the robot, and a pair (v_i, ω_i) is considered safe/admissible if the robot can stop before it reaches the obstacle in the curvature that is closest. Lastly, an objective function according to equation 2.5 is maximized, evaluating: *Target heading* - maximal if the robot moves straight to the target; *Clearance* - the smaller the distance $dist(v_i, \omega_i)$ to the closest obstacle, the better; *Velocity* - the projected translational velocity, so that the robot progresses in the corresponding trajectory. Figure 2.7 is taken from Missura and Bennewitz (2019) [26], where DWA is adapted for dynamic environments and collision between obstacles modeled as polygons can be predicted by taking into account their motion.

$$G(v, w) = \sigma(\alpha \cdot \text{heading}(v, w) + \beta \cdot \text{distance}(v, w) + \gamma \cdot \text{velocity}(v, w)) \quad (2.5)$$

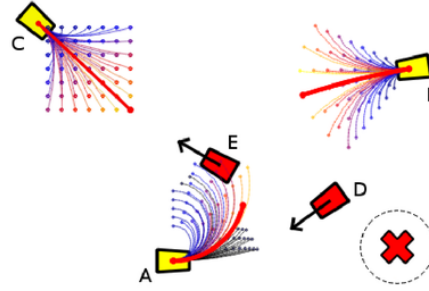


Figure 2.7: Predictive DWA Collision Avoidance. Agent A predicts future collisions with Agent D and discards collisions with E since it is moving away. Agent B is using circular trajectories and Agent C is using a holonomic model [26].

Model Predictive Control (MPC) is a control technique that uses a predictive model of the system to optimize over a finite time horizon [27]. The key components of MPC are the System Model, usually represented in state-space form, an objective function, and constraints on control inputs and states. This method predicts future system behavior over a finite time horizon and solves an optimization problem of minimizing the objective function J while subject to constraints. The state space system dynamics, in the continuous case, can be modeled by a function f (Equation 2.6):

$$\dot{x} = f(t, x(t), u(t)), \quad (2.6)$$

where $x(t)$ is the state of the system at time t and $u(t)$ is the control input. The objective function J defines the optimization criteria that includes a running cost, $L(t, x(t), u(t))$, and can also include an end cost $\Phi(x(t_f))$. The cost function J with the running cost L of a control input u , between

times t_0 and t_f is defined as (Equation 2.7):

$$J(u) = \int_{t_0}^{t_f} L(t, x(t), u(t)) dt \quad (2.7)$$

Lastly, constraints on system states and control inputs can be modeled as inequalities, for example (Equation 2.8):

$$g(t, x, u) \leq 0 \quad (2.8)$$

Although it is a control strategy, with common implementations for process control, it is well-suited for systems with complex dynamics and constraints and is therefore used for path planning in dynamic environments. The modeled constraints can vary from dynamics to constraints of safety or traffic rules. Liu *et al* (2017)[28] propose using MPC as a unified path-planning approach for autonomous driving, instead of high-level rule base decision-making processes for lane keeping, lane changing, ramp merging, or intersection crossing. The cost function includes terms for the distance to the center line of a lane, distance to other vehicles, distance to the destination, minimizing jerk or following the curvature of the road. Constraints model collision avoidance and vehicle dynamics on acceleration and turning rate. The biggest challenges for MPC are the high computational demand for solving the optimization problem in the defined time horizon and the dependency on the system's dynamics model.

ArduPilot³ is a widely used open-source autonomous navigation software for drones, rovers, and marine vehicles. It includes built-in implementations of several basic obstacle avoidance algorithms. The simpler algorithms are *Stop Avoidance*, *Minimum altitude*, *Back away from obstacles*. These 3 methods are purely reactive and are directly controlled by sensor feedback to stop when an obstacle is detected or to maintain a safe distance from the ground or obstacles. Bendy Ruler is another object avoidance algorithm also implemented in ArduPilot that consists of probing around the robot in several directions, evaluating open spaces where the vehicle safely passes through, and choosing the best open space to advance toward the goal destination. It can be configured to perform this search on the horizontal plane (Horizontal Bendy Ruler) or to perform it in four directions: towards the destination, up, down, or backward (Vertical Bendy Ruler). Figure 2.8 represents two iterations of Bendy Ruler, where the lines represent the probed directions, and a new waypoint is chosen based on safe distance from obstacles and proximity to the goal position.

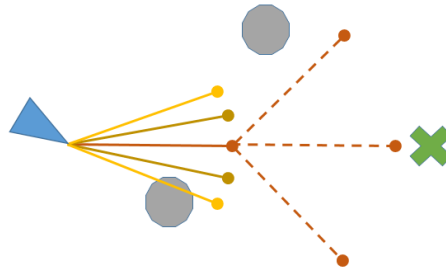


Figure 2.8: Object Avoidance with Bendy Ruler³

³<https://ardupilot.org/>. Accessed on June 25, 2025.

Elastic Band methods on path planning [29] take a path from a global planner and deform it into a shorter and smoother collision-free path, that continues to deform as the online agent detects objects and changes in the environment. The Timed Elastic Band approach (TEB) [30] considers temporal aspects of the dynamic constraints of robots, converting an initial path into a trajectory dependent on time. The TEB defines a tuple of two sequences: Q as a sequence of robot poses x_i and τ as a sequence of time intervals δT_i . $B := (Q, \tau)$ represents the configurations of the robot and the corresponding time differences from one configuration to the other. The referred "deformation" is done by optimizing in terms of a weighted multi-objective function, $f(B)$, which includes velocity and acceleration constraints formulated with penalties. The temporal information allows TEB to add to the objective function an incentive for faster paths, shorter paths like the traditional elastic band does, or even a combination of both. Non-holonomic kinematics can be enforced by requiring two adjacent configurations to be in a common arc segment. As TEB provides this framework to account for velocity and acceleration limitations but lacks the preemptive behavior needed for more complex interactions with other moving vehicles, Nguyen *et al* (2020) [31] propose a proactive adaptation of TEB (PTEB - Proactive timed elastic band) that combines the advantages of a Hybrid Reciprocal Velocity Obstacle [32]. Magyar *et al* (2019) [33] propose using TEB to optimize motion planning for robotic manipulators, due to its ability to generate smooth trajectories and bound velocity acceleration and jerk.

In recent years, Reinforcement Learning (RL) approaches have been increasingly more popular for obstacle avoidance [34] and other path planning tasks, like docking of autonomous vehicles [35] or autonomous landing [36]. In RL, an agent interacts with the environment and receives feedback in the form of rewards or punishments. Positive reinforcement is given when the outcome of an action taken by the agent has a favorable effect, and negative reinforcement is given when the action leads to an undesirable outcome. The learning part in RL will make the agent update his policy and repeat more often the actions that led to better outcomes, and avoid those that led to negative feedback. In Mobile Robotics, agents can learn from their environment and therefore adapt to dynamic scenarios, improving the decision-making process. Research in Reinforcement Learning for path planning has focused on Deep Reinforcement Learning (DRL), multimodal inputs, and assuring safety. Using RL for path planning and obstacle avoidance brings adaptability without explicit behavior programming, which is frequently difficult due to the numerous and diverse situations that the robot may encounter. Even for humans, choosing the right avoidance maneuver can be challenging and RL provides total independence from labeling but requires effective reward functions. Other disadvantages are the exploration risks in the learning step, that can be attenuated by using complex simulators. There is, however a gap between real-world and simulated environments, and the best policy in the simulator may not be transferable to the real world. Lastly, especially for DRL, neural networks bring a lack of interpretability, making it difficult to ensure safety requirements are being met [34].

2.2.2 Velocity Obstacles

Velocity Obstacles (VO), as described by Fiorini [37] is a first-order method of obstacle avoidance in dynamic environments, consisting of selecting avoidance maneuvers in a velocity space, based on the position and velocities of the robot and obstacles. The Velocity Obstacle of A is the set of velocity controls that would lead the agent A to a collision. Table 2.1 describes the symbolic notation based on [19] that will be used in the following section to describe VO-based approaches.

The environment representation consists of an agent A and a set of obstacles B_i , all characterized by a position (x_A, x_{B_i}) , a radius (r_A, r_{B_i}) , and a velocity (v_A, v_{B_i}) . They are represented in the *Velocity Space* according to the Relative Velocity Paradigm (RVP) [5]. Defining the velocities from a common inertial frame allows us to visualize both the position and velocity at each point, adding the velocity vector to the center of the object, as represented in figure 2.9, taken from [37]. Equation 2.9 describes the motion of a point p in the Velocity Space, where $x_p(t)$ and $v_p(t)$ describe, respectively, the point's coordinates and velocity.

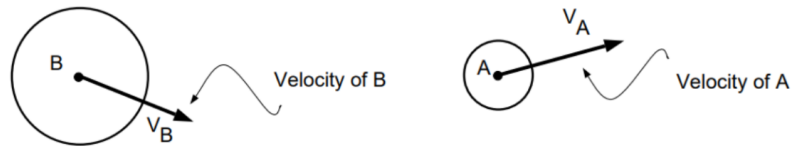


Figure 2.9: Agent and moving obstacle [37]

$$p(t) = (x_p(t), v_p(t)). \quad (2.9)$$

For an agent A , we can consider the relative velocity of A with respect to B as follows (Equation 3.5):

$$v_{AB} = v_A - v_B, \quad (2.10)$$

and the relative trajectory between A and B is defined as Equation 2.11:

$$trj_{AB} = \{(x_{AB}, v_{AB}(t)) \mid x_{AB}(t_0) = x_A(t_0) - x_B(t_0), v_{AB}(t_0) = v_A(t_0) - v_B(t_0)\} \quad (2.11)$$

A collision between A and B is considered if (Equation 2.12):

$$trj_{AB} \cap B \neq \emptyset. \quad (2.12)$$

The relative collision cone CC_{AB} of A induced by B is defined by the set of velocities for which (equation 2.13):

$$CC_{AB} = \{v_{AB} \mid trj_{AB} \cap B \neq \emptyset\}. \quad (2.13)$$

Table 2.1: Symbolic notation used to describe VO approaches

Symbol	Meaning
A	Autonomous agent A
$\hat{A}$	A in the configuration space
B_i	i^{th} obstacle
r_A	Radius of A
x_A	Position of A
θ	Heading Angle of A
v_A	Velocity of A
x_{AB}	Position of A relative to B
v_{AB}	Velocity of A relative to B
trj_{AB}	Trajectory of A relative to B
CC_{AB}	Relative collision cone of A induced by B
VO_{AB}	Velocity Obstacle of A induced by B
u	Actuator efforts
$\mathcal{U}$	Set of admissible actuator efforts
u_s	speed control
u_ϕ	steering angle control
u_{ORCA}	ORCA vector to closest boundary point
$f(x, \dot{x}, u)$	Robot dynamics
FA	Feasible Accelerations
RV	Reachable Velocities
RAV	Reachable Avoidance Velocities
$\oplus$	Minkowski Addition
CL	Center Line
$\mathcal{E}$	Tracking Error
θ_H	Holonomic equivalent velocity
d_m	Minimum distance between A , B
PZ	Protective zone
BV	Buffer Velocity Set

In the *configuration space obstacle*, we reduce the agent A to a point $\hat{A}$ and add its shape to the obstacle B , performing a Minkowski Sum⁴ represented here by $\oplus$. For two spheres, the Minkowsky Sum corresponds to a sphere with the radius of both spheres radii summed. Figure 2.10 represents objects A and B with their velocities, the relative velocity v_{AB} , the Minkowsky Sum of B and A , and the Relative Collision Cone of A induced by B .

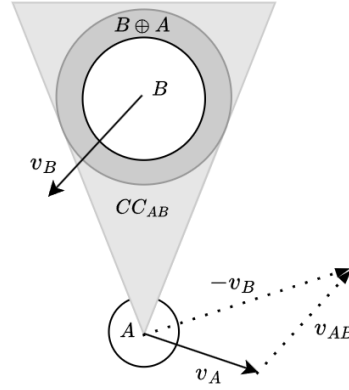


Figure 2.10: Minkowski Sum, relative velocities, and Collision Cone

The set of all colliding velocities for A is the Relative Collision Cone, defined geometrically as the velocities inside the straight lines tangent to $\hat{B}$ that go through $\hat{A}$. Translating the Relative Collision Cone CC_{AB} by v_B , creates the Velocity Obstacle VO_{AB} . For multiple obstacles, the velocity chosen by A should now be outside the union of all the VO_{AB} .

The velocities that are reachable by A over a given time interval Δt are obtained by transforming the dynamic constraints of the robot into bounds on its acceleration [38]. Defining $FA(t)$ as the set of *Feasible Accelerations* at time t , $f(x, \dot{x}, u)$ as the robot dynamics, u as the actuator efforts, and U as the set of admissible controls Equation 2.14 is obtained:

$$FA(t_0) = \{\ddot{x} \mid \ddot{x} = f(x, \dot{x}, u), u \in U\} \quad (2.14)$$

Considering the set of *Reachable Velocities* $RV(t_0 + \Delta t)$ and the set of *Reachable Avoidance Velocities* RAV , that can be defined as (Equation 2.15 and Equation 2.16):

$$RV(t + \Delta t) = \{v \mid v = v_A(t_0) + \Delta t \cdot FA(t_0)\} \quad (2.15)$$

$$RAV(t_0 + \Delta t) = RV(t_0 + \Delta t) \setminus VO(t_0) \quad (2.16)$$

Figure 2.11 represents the *Feasible Velocities* (FV) for agent A . In Figure 2.11b, The areas in red represent the the feasible velocities in the velocity space, with the darker red for velocities that lead to a collision, and lighter red for the Reachable Avoidance velocities.

To compute the avoidance velocity, Fiorini and Shiller [37] propose computing a discretized RAV set as a grid and at discrete time intervals, and generating a tree of avoidance maneuvers.

⁴<https://encyclopediaofmath.org> (Accessed on June 25, 2025)

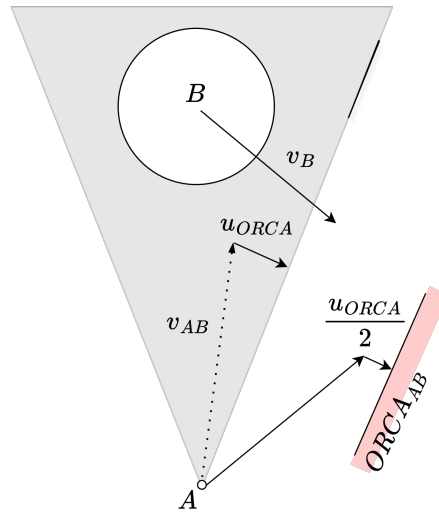


Figure 2.13: ORCA halfplane

vehicles, such as car-like robots since they must follow an arc to achieve a selected velocity, instead of moving in a straight trajectory. Figure 2.14 represents a "feasible" velocity that would result in a collision due to the non-holonomic nature. One way to minimize this effect is to choose small time intervals, but this reduces the set of feasible velocities that are being considered. Wilkie *et al* (2009) [41] propose Generalized Velocity Obstacles (GVO), a generalization for obstacles in the control space to handle non-holonomic constraints. Considering an obstacle B and its position $x_B(t)$, the position of agent A , x_A and a control u , $A(t, u)$ represents the position A will have after the control u for the time interval τ .

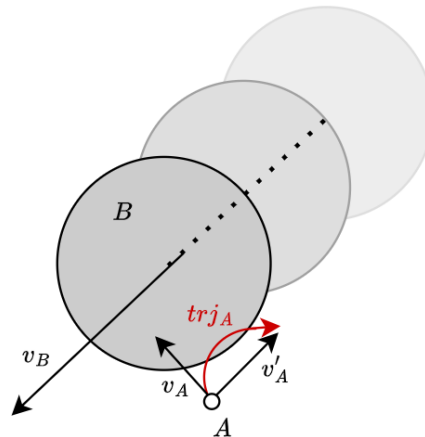


Figure 2.14: A feasible velocity according to VO, which leads to a collision due to the non-holonomic kinematic constraints.

In the control space, the obstacle is defined as the set of all control inputs u that lead to a collision (Equation 2.19):

$$\{u | \exists t > 0 :: \|A(t, u) - B(t)\| < r_A + r_B\}. \quad (2.19)$$

GVO considers the time at which the distances between A and B are minimal, $t_{min}(u)$, and if there is a closed expression for it, then it is possible to look into the distance between the centers of A and B at the time where that distance is minimum for each u , and this way defines the obstacle in the control space. For a holonomic vehicle in 2D and centered around the agent position, the control input corresponds to the velocity v (Equation 2.20):

$$A(t, v) = tv \quad (2.20)$$

$$B(t) = x_B + tv_B \quad (2.21)$$

The minimum distance time can be calculated by equaling the derivative of the distance function to zero. In the case of car-like kinematics, the constraints are given by Equation 2.22, where u_s and u_ϕ are the speed and steering angle controls, respectively. With these differential kinematics equations, a closed expression for the minimum distance can be obtained, and thus a method for evaluating if a non-holonomic control leads to a collision.

$$\begin{aligned} \dot{x}(t) &= u_s \cos \theta(t), \\ \dot{y}(t) &= u_s \sin \theta(t), \\ \dot{\theta}(t) &= u_s \frac{\tan u_\phi}{L}, \end{aligned} \quad (2.22)$$

Snape *et al* (2011) [32] present the Hybrid Reciprocal Velocity Obstacle (HRVO) as a direct extension of *RVO* to address the problem of reciprocal dances. HRVO considers the Center Line (CL), which divides *RVO* into two halves. If v_A is on one side of RVO_{AB} , A should prioritize passing B by that side, so HRVO enlarges the *RVO* of the opposite side. By symmetry, the same thing will happen to agent B trying to pass agent A . On the side opposite to the passing side, the *RVO* is enlarged by replacing the edge on that side with the edge of the original VO_{AB} . This resulting obstacle is called the $HRVO_{AB}$, represented in Figure 2.15, and its apex now lies in the intersection between the passing side's edge of RVO_{AB} and the opposite side's edge of the VO_{AB} . One consequence is that if robot A attempts to pass on the wrong side of B , the algorithm will make it give full priority to B , but if the correct side is chosen, cooperation from B is assumed and the principles in *VO* and *RVO* are fulfilled. For multiple obstacles, the approach is again to calculate the individual HRVO and perform the search for the avoidance maneuver outside the union set of all HRVO. The chosen new velocity is the closest to its preferred velocity, calculated using other path planning algorithms.

Snape *et al* (2011) [32] also propose a way to account for uncertainty in the radius, position, and velocity of obstacles. A Kalman filter [42] is employed to estimate these parameters, but also to estimate the covariance in the measurements, which can be used to adjust the velocity obstacle's

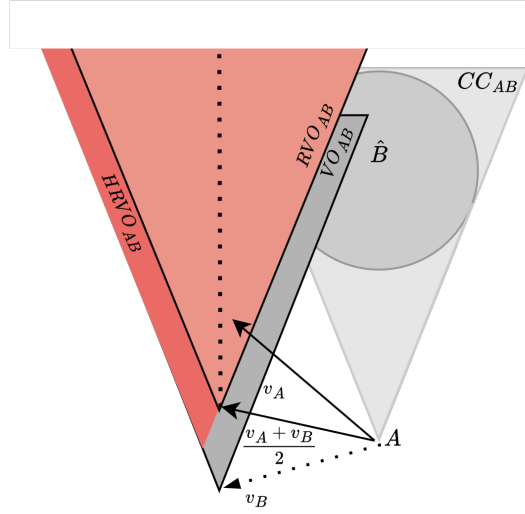


Figure 2.15: HRVO example

shape. Considering Σx_A , Σx_B , and Σx_{AB} as the covariances in measured positions x_A and x_B and relative position x_{AB} , σ_{r_A} , σ_{r_B} , and $\sigma_{r_{A+B}}$ for the variance in the radius of A , B and their sum $A+B$ and lastly Σv_B for the velocity of object B . The Minkowski sum $\hat{B}$ is expanded to a disk of radius $r_A + r_B + \sigma_{r_{A+B}}$, and a linear transformation to account for the uncertainty in position Σx_{AB} . The VO is again expanded around v_B so that geometrically it encompasses a disk of radius Σv_B . This transformation lead to an uncertainty-adjusted VO^* , as can be seen in figure 2.16. In this description and figure 2.16, the values of radius position and velocity refer to the estimated output of the filter and not the real ones.

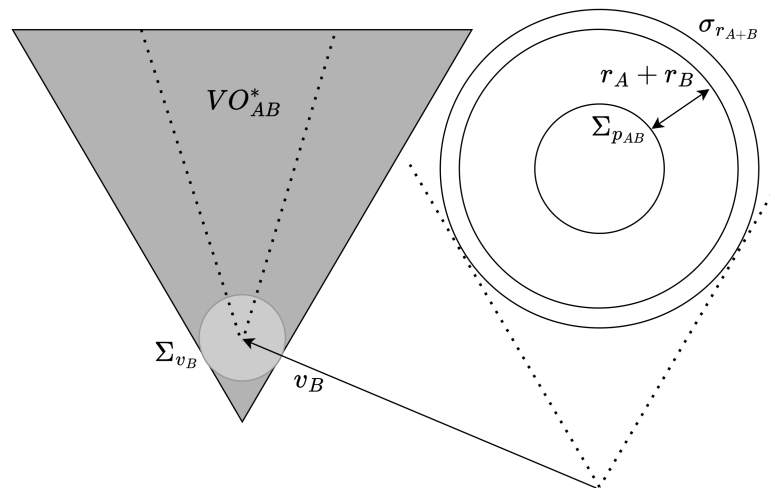


Figure 2.16: Uncertainty-adjusted Velocity Obstacle

An extension toward non-holonomic vehicles for ORCA is done by Alonso-Mora *et al* (2013)

[43]. The idea behind the approach is the assumption that any non-holonomic motion can track a holonomic ideal one within a tracking error $\mathcal{E}$. Ensuring collision avoidance can then be done by increasing the radius of the robot by this error. The kinematics for differential drive robots can decompose trajectories into arcs of circumference and straight-line paths. Figure 2.17 represents an arc with constant velocity v , a straight line with constant velocity v_1 , and a tracking error $\mathcal{E}_H$ to the ideal holonomic velocity, v_H . For a given v_H and control inputs (v, ω) in time step t , the tracking error can be obtained

$$\mathcal{E}^2(v, \omega, v_H, \theta_H) = (v_H t_1 - R \sin(\theta_H))^2 + (R(1 - \cos(\theta_H)))^2 \quad (2.23)$$

This expression is used to define the set of allowed holonomic velocities S_{AHV_i} , computed as those that guarantee a tracking error lower or equal to the given maximum tracking error at all times. In other words, the set of v_H so that there exists one (v, ω) in the Set of Non-holonomic controls S_{NHC} and guaranteeing a tracking error lower or equal to the defined maximum $\mathcal{E}$. In each time step, NH-ORCA builds the original VO, then the ORCA halfplanes for the holonomic equivalent robot with radius $r + \mathcal{E}$. S_{AHV_i} is computed for fixed $\mathcal{E}_i$ and approximated by a convex polygon P_{AHV_i} that lies inside S_{AHV_i} . As P_{AHV_i} is formed by linear constraints, a quadratic optimization problem with linear constraints is formulated, and computational geometry methods can be used to find the optimal avoidance maneuver.

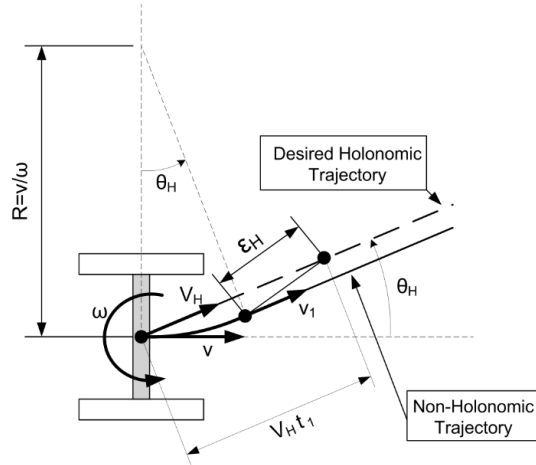


Figure 2.17: Non-holonomic tracking error [43]

Kluge and Prassler (2006) [44] include a probabilistic extension and a Reflective Navigation approach, modeling obstacles as intelligent decision-making agents recursively. The Probabilistic Velocity Obstacle (PVO), is an extension to the original VO algorithm that deals with uncertainty in the shape and velocity of the objects, reflecting the limitations and errors of real-world sensors. Instead of a binary "feasible" or "unfeasible" zones classification, the PVO becomes a function, according to Equation 2.24:

$$PVO_{v_i} : \mathbb{R}^2 \rightarrow [0, 1] \quad (2.24)$$

that maps possible velocities to a probability of colliding with obstacle i . For multiple obstacles, this probability becomes (Equation 2.25):

$$PVO = 1 - \prod_i (1 - PVO_i) \quad (2.25)$$

The policy for the best avoidance maneuver is based on two objectives: reaching the goal and minimizing the probability of collisions. The Relative Utility function RU (Equation 2.26)

$$RU = U \cdot D \cdot (1 - PVO). \quad (2.26)$$

considers also the utility proposed by U for the defined goal, and the feasibility D of that maneuver. Another important idea proposed by Kluge and Prassler (2006) [44] is taking into consideration that each robot using this navigation strategy will also act according to its own relative utility function RU. For this, it is necessary to consider every robot dynamics D , and utility function U , which requires complex motion pattern recognition algorithms. For depth $d = 0$ recursion, the agent maximizes its own $UR = U \cdot D$, ignoring obstacles. For $d=1$, it maximizes $UR = U \cdot D \cdot (1 - PVO)$, where PVO is building according to the current velocity of the obstacle. For recursion depths greater than one, the relative utility function will consider the PVO induced by the velocity that maximizes the obstacle's UR for depth $d - 1$.

Campos *et al* (2019) [45] proposed Adaptive Velocity Obstacle Avoidance (AVOA), as an extension of the VO algorithm for USVs. Similarly to the VO approach, AVOA starts by defining a Protective Zone (PZ_{AB}) as the set of all velocities that would lead A to a collision with a moving obstacle B . To avoid early route changes, the PZ is simplified by removing velocities that would lead to a collision before a predefined time t_H . The removed set PZ_{AB}^H can be represented as in Equation 2.27, where d_m is the minimum distance between A and B .

$$PZ_{AB}^H = \left\{ [v_A, \theta_A] : v_A \in PZ_{AB}, |v_A - v_B| \leq \frac{d_m}{t_H} \right\} \quad (2.27)$$

The kinematic constraints of the USV make it so that the velocity space is defined in velocity and heading pairs, (v, θ) . In this space, maximum linear and angular velocity constraints can be defined to the differential drive kinematics of most USVs. To enhance the Protective zone representation, and avoid tangent crossing and risky maneuvers, AVOA defines a cost decreasing with the distance to the nearest PZ. In figure 2.18, taken from [45] the protective zones for static and dynamic obstacles, considering kinematics and distance to the nearest PZ are represented. Black regions represent unfeasible areas, according to PZ or kinematic constraints. The color space represents v with saturation and θ with color. Lastly, the gradient in gray represents, up to a defined threshold, the distance to the nearest PZ. The evaluation of the quality of a velocity is done in AVOA by using a multicriteria evaluation $h(v)$, described in Equation 2.28, that considers velocity and heading deviation relative to the expected velocity $(\delta v_d, \delta \theta)$, and the proximity-based

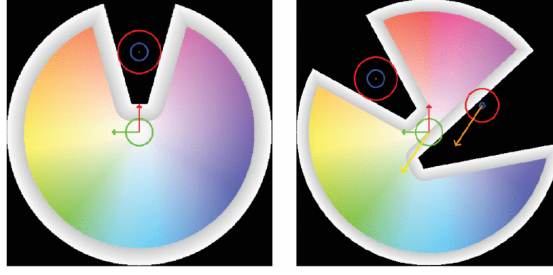


Figure 2.18: AVOA protective zones for static and dynamic obstacles [45]

danger cost that penalizes closeness to obstacles, ∇_i , according to the weights Φ :

$$h(v) = \begin{bmatrix} |\Delta v_d| & |\Delta \theta_d| & \nabla_i \end{bmatrix} \Phi, \quad (2.28)$$

To address the problem of local minima in $h(v)$, AVOA uses Particle Swarm Optimization (PSO) [46] to find the optimal solution according to the multicriteria function. This population-based metaheuristic randomly initializes a swarm of particles within the search space, that evaluate their position using a fitness function, and update their velocity based on their previous velocity (inertia coefficient), their best personal position (cognitive coefficient), the global best position (social coefficient) and a random value to introduce stochastic behavior. The particles iteratively update their velocities and position while exploring the search space and converging towards the global best solution. In AVOA, if a particle falls into an unfeasible zone it respawns, and after a certain number of iterations or dropping below a reasonable solution, PSO returns the updated velocity for the USV to follow.

Battisti and Muradore (2020) [47] present some interesting solutions for expanding to 3D vehicles. Based on ORCA, their work tackles some scenarios where this algorithm does not work as expected in three dimensions, implements some kinematic constraints on acceleration and steering angle, and simulates it for the landing of a UAV on a moving Unmanned Ground Vehicle (UGV) in crowded dynamic environments. The results are then integrated to develop a bilateral teleoperation architecture for the remote control of a UAV, where the operator can receive force feedback proportional to the error between their control input and the velocity calculated by the algorithm.

The ORCA deadlocks addressed by Battistini and Muradore (2020) [47] are represented in figures 2.19 and 2.20. The first issue is related to "Agent alignment," which was also addressed by Snape *et al* [32]. Although this scenario is rare in practice due to measurement noise, two agents can have aligned velocities in opposite directions, making the algorithm unable to choose a side to pass the obstacle. As a result, both remain on the same line and cannot reach their goal coordinates. The second problem concerns motion through closely spaced agents, where the superposition of ORCA half-planes created by obstacles moving towards the agent causes ORCA's linear programming to calculate a maneuvering velocity that moves backward, never passing the obstacles. A system that switches between the goal and "temporary dummy goals" is proposed to

overcome both problems, returning to the main goal when the deadlock conditions are resolved.

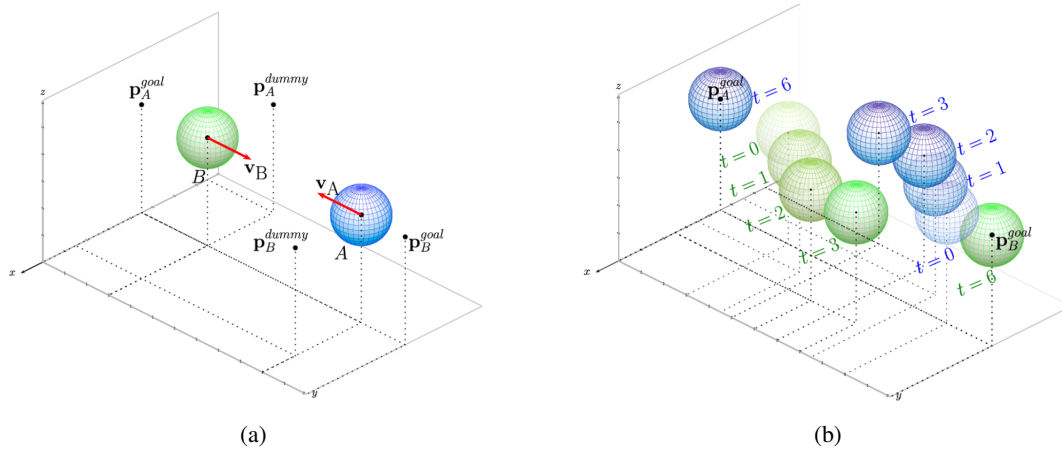


Figure 2.19: Two aligned agents: (a) *Dummy Goals*; (b) *Timelapse* [47].

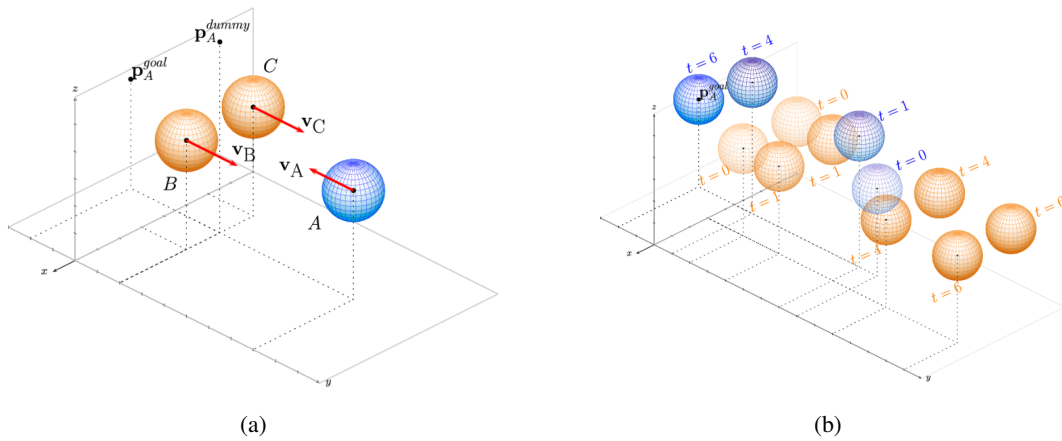


Figure 2.20: Closely Spaced Agents: (a) *Dummy Goals*; (b) *Timelapse* [47].

Jenie *et al* (2016) [48] proposes another 3D Velocity Obstacle approach, introducing some different ideas. The approach is called the Three-dimensional velocity obstacle method. It adds anticipation for the obstacle maneuvers by introducing a *Buffer Velocity Set*, and breaks the problem of choosing an avoidance maneuver into two steps: choosing an avoidance plane and choosing an avoidance turning rate. The *Buffer Velocity Set* is an extra set added to the Velocity Obstacle, to handle obstacle unpredictability. The VO method uses the velocity of the obstacle in the last iteration and assumes that velocity will remain constant. To better address this, a new VO set is anticipated by summing all the possible VO sets created by the possible maneuvers of the obstacle. This set of possible maneuvers is approximated as a circular reachable velocity set, and the new VO is still a cone that is now expanded to be tangent to this circular reachable obstacle velocities set. Figure 2.21, taken from [48], shows a) different VO for considering different possible obstacle velocities in the next iteration (VO' and VO'') and b) The concept of buffer velocity set in 2

dimensions. RV_i is the Reachable Velocity Set for the obstacle i , BV is the Buffer Velocity Set.

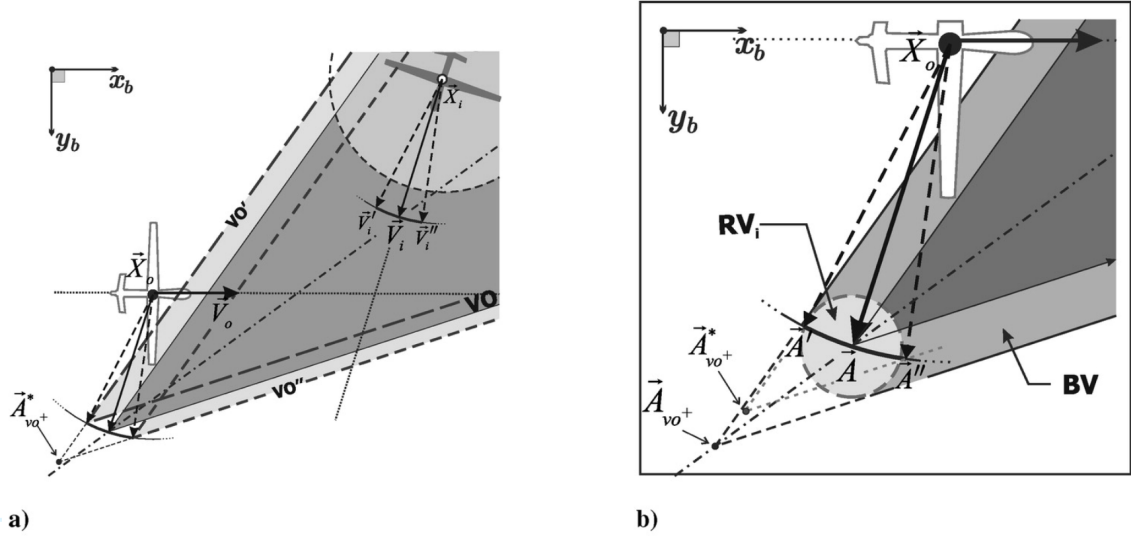


Figure 2.21: a

-) The VO sets for possible obstacle velocities b) Buffer Velocity Set in 2 Dimensions]Buffer Velocity Set a) The VO sets for possible obstacle velocities b) Buffer Velocity Set in 2 Dimensions. Taken from [48]

Jenie *et al* [48] also introduce Avoidance Planes for 3DVO, where the obstacle becomes a cross-section of the plane and the 3-D obstacle. The idea is that instead of trying to address all possible solutions in the velocity space, the algorithm evaluates planes P_ϕ , defined by the angle of rotation ϕ from the vehicle's X axis. Figure 2.22 represents these avoidance planes, projected for different ϕ angles (-90° , 0° , -45° , 45°). Triangular or hyperbolic VO sections are considered more dangerous, and ellipsoidal are preferred. Choosing the best avoidance plane is done by weighing a risk metric based on the section shape, and the smaller angle to the X-axis allowed by that plane. The last part is choosing, inside that plane, an avoidance velocity that complies with the dynamic constraints of the vehicle and is outside all the obstacles' cross sections.

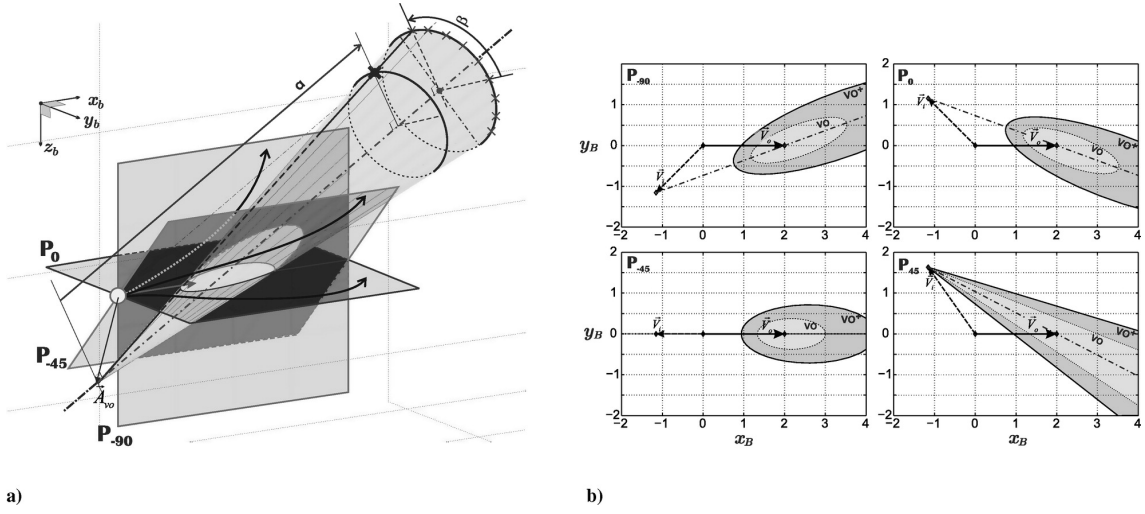


Figure 2.22: Avoidance planes a) The avoidance plane illustration intersects the VO. b) Conic sections of the cone on several avoidance planes [48].

2.3 Critical Review

Obstacle avoidance algorithms can be placed in the motion/local planning layer, adjusting the output of global/mission and behavioral planners to better navigate in dynamic and unknown environments. These algorithms operate based on the outputs from the previous planning layers, specific objective functions, and a detailed representation of the environment that is continuously updated to represent the changes in dynamic scenarios.

Scene representations can be built on 3 different layers of information. Metric information is crucial, and most of the overview methods for obstacle avoidance are based on metric representations. Velocity Obstacles is a method that relies on a representation that is more object-centric, modeling circular obstacles isolated from the background. Uncertainty information can be modeled in the scene representation, as it can be used for more robust decision-making. Multi-agent improvements assume shared responsibility, but semantic information about moving obstacles can be used to infer if they are moving actively or passively through the environment. Lastly, in the same way, that the velocity obstacles can be shaped to account for uncertainty, a similar thing can be done with semantic information to provide different behaviors for different classes of obstacles and improve decision-making.

The described obstacle avoidance methods are straightforward to implement from metric information. The APF transforms the obstacle position and the goal position directly in virtual forces. DWA searches on the velocity space, and extensions to model obstacle velocity make the algorithm suitable for dynamic environments, however, it has difficult scalability since it calculates through potential trajectories. The same problem can be said for MPC, requiring solving several discrete-time interval control problems to evaluate a trajectory, and obtaining results highly dependent on the model given. VFH and Bendy Ruler are the easiest to implement and fastest in real-time computation, but are the least able to address dynamic environments. Lastly, the results

of TEB show its efficiency in modeling kinematic constraints, and the optimization function allows the algorithm to adapt to different requirements, generating smooth paths. However, it faces challenges in modeling interactions with dynamic obstacles or other vehicles, but can be complemented with VO approaches to model potential collisions [31]. The focus on Velocity Obstacles is justified by the algorithm's ability to model crowded scenarios with both static and dynamic obstacles, and widespread use to model multi-agent navigation, with no dependence on communication between them. It has a simple geometric and efficient formulation that directly considers the obstacles' velocities. As VO works in the control space, it allows for dynamic and kinematic restrictions to be made from the beginning, suiting heterogeneous vehicles. The referred improvements on VO can be divided into three goals: multi-agent scenarios, non-holonomic vehicles, and modeling safety/risk. RVO and HRVO improve multi-agent operability. For non-holonomic vehicles, GVO proposes a method to check for collisions in a more realistic non-holonomic setting, and NH-ORCA indirectly guarantees collision avoidance for non-holonomic vehicles by accounting for an approximation error when the model is based on holonomic ideal equivalents. PVO introduces a probability framework for the decision-making process, and AVOA prioritizes safer maneuvers when crossing moving obstacles. Regarding the computation of the obstacle avoidance maneuver, different approaches can be seen: The original VO uses a three-search, where the nodes are cells in a grid of reachable avoidance velocities. ORCA makes the computation more efficient using linear programming. AVOA uses PSO that enables searching larger spaces faster. Instead of searching in the whole control space, 3DVO first defines the avoidance plane by evaluating the projection of the velocity obstacle. Optimal approaches have the problem of leading to tangent crossing of obstacles and are therefore not suitable for maritime environments where environmental factors can cause errors in both the sensors and the control of vehicles. On the other hand, expanding to the tridimensional space brings computation and memory problems, so searching all feasible velocities is not possible, and further restrictions on the search space or heuristics have to be employed for this search to run in real-time for mobile robots.

Although VO proves its efficiency in predicting collisions and acting accordingly, it does not generate a full trajectory to the goal, and therefore, deadlock scenarios in complex situations are possible. This problem is accentuated when expanding to the three-dimensional space, as more directions lead to more ambiguities. The use of virtual obstacles or *dummy goals* as proposed by [47] aligns with the framework described by [7], using a behavioral layer that allows for efficient scalability to complex interactions and navigation rules.

The main gap in the literature regarding VO is the ability to deal with obstacles and vehicles moving in the tridimensional space, although this expansion was theorized by Fiorini and Shiller (1993) [5]. The challenge behind this expansion is the much larger search space for feasible avoidance velocities, which makes real-time feasibility more difficult. This work focuses on exploring how the ideas described in this chapter can be applied to three-dimensional space, optimizations to make the algorithm feasible in real-time, while still complying with the kinematic and dynamic constraints of the vehicle, and integration with detection systems.

Chapter 3

JAVOA: 3D Adaptive Velocity Obstacle Algorithm

This chapter presents the developed algorithm for obstacle avoidance of heterogeneous robotic vehicles in dynamic environments. Section 3.1 introduces the problem and reviews the objectives. Section 3.2 describes the environment representation and the approach implemented for obstacle detection, followed by section 3.3, where the obstacle avoidance approach is presented, from sample generation to pruning and evaluation. Finally, section 3.4 describes the main considerations and approximations of the proposed approach.

3.1 Introduction

Unmanned offshore inspection and maintenance operations using USVs (Unmanned Surface Vehicles), UAVs (Unmanned Aerial Vehicles), and UUVs (Unmanned Underwater Vehicles) are becoming increasingly common. They are playing a significant role in the transition toward renewable energy infrastructure. Using uncrewed vehicles can reduce costs and mitigate risks to human operators. Still, it is crucial to ensure their safe navigation and to avoid collisions that could damage the robotic vehicles and the infrastructure, or even endanger nearby personnel. These vehicles operate in complex environments that include various types of obstacles, as illustrated in Figure 3.1. These obstacles range from static elements, such as buoys (Figure 3.1a) or rocks, to dynamic ones like other vessels (Figure 3.1b). In such dynamic scenarios, it is not enough to react to obstacles: potential collisions must be detected and avoided preemptively, as early as possible.

Additionally, the different domains of operation—underwater, surface, and above water—and the need for heterogeneous vehicles to navigate across them make it essential to sense, interpret, and act within fully three-dimensional environments. Safety and risk minimization must remain top priorities, especially since collisions in maritime environments often lead to high costs and irreparable equipment damage.

¹ www.almarin.es. Accessed on June 25, 2025.



(a) Marine Marker Buoy



(b) Surface vehicles photographed in Viana do Castelo, Portugal

Figure 3.1: Examples of obstacles found in maritime environments. (a) Marine Marker Buoy¹; (b) Surface vehicles photographed in Viana do Castelo, Portugal.

To increase the safety of autonomous vehicles in dynamic environments, an obstacle avoidance layer must be integrated into the navigation pipeline. This layer should have at least two inputs: The first is the desired path or control that is given by a Global Planner, for example, and that allows for a desired goal or objective to be followed. The second is the obstacle information used to guarantee safety while trying to achieve the given goal. This information can come from various sensors, but it should first be processed to understand the environment better. With these two inputs, the obstacle avoidance can guarantee a safe action towards the objective and pass the computed command to a lower-level control layer, which will directly control the vehicle actuators to ensure the command is followed. In the specific case of Velocity-Obstacles-based approaches, the input plan should have the form of a desired velocity to be followed, and the obstacle information should consist of isolated obstacles, creating a command velocity output. An overview of this Autonomous Navigation Pipeline is represented in figure 3.2. This pipeline fits the standard approach of having a Global Planner addressing the static obstacles known a priori, while a Local Planner deals with the unforeseen and dynamic components of the environment.

The main design objective of the proposed obstacle avoidance method is to ensure easy adaptability to the different kinematic constraints of various vehicles. One of the most significant gaps in the literature is the lack of generalization of these algorithms to different types of platforms, such as holonomic planar vehicles, differential drive systems, multicopter UAVs, and Ackermann steering vehicles. The developed algorithm must consider the kinematic constraints of such platforms, ideally accommodating Six Degrees of Freedom (6DOF), while enabling specific implementations to be tailored to the constraints of a particular vehicle.

This generalization effort requires handling three-dimensional dynamic environments, not just allowing the agent to move in 3D space, but also accounting for the movement of 3D obstacles

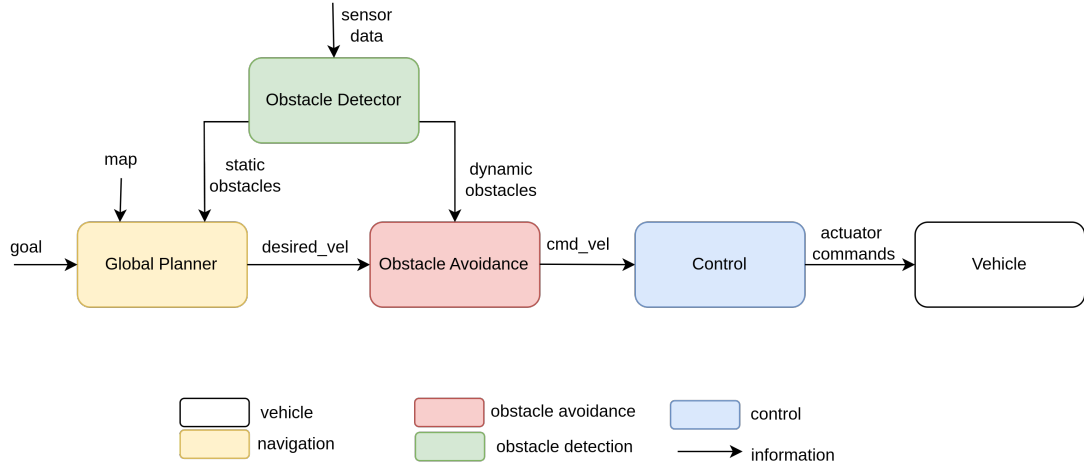


Figure 3.2: Autonomous Vehicle Navigation with Obstacle Avoidance Pipeline

even in 2D implementations. The expansion to 3D space introduces a direct challenge: the representation space scales exponentially with the number of dimensions. Therefore, another goal is to choose an approach that allows efficient traversal of the search space and can scale.

One of the advantages of Velocity Obstacles approaches is that they leverage information from individual obstacles in the environment. Unlike grid-based navigation using occupancy maps, extracting and using individual obstacle features more closely resembles how a human driver perceives and navigates in a dynamic scene. Furthermore, incorporating scene understanding with semantic information [45] and obstacle uncertainty [32] can enable more risk-aware decision-making.

3.2 Velocity Space representation based on Obstacle Detection

The scene representation requires obstacles with static or dynamic nature to be represented in a velocity space, which must characterize the extrinsic features related to geometry and motion of all elements surrounding the vehicle that can be combined from heterogeneous perceptual systems. To develop an uniform representation principle, these elements are represented as spheres characterized by a radius, a position, and a velocity, along with the uncertainty associated with each parameter. In addition to geometric and uncertainty characteristics, each obstacle is assigned an unique Identifier (ID), a semantic class, and a static or dynamic flag. The complete obstacle definition used is presented in table 3.1.

The position and radius standard deviation information are considered directly in the obstacle radius. The obstacle is enlarged by the standard deviation on position (σ_p), and the standard deviation on radius (σ_r), that come from an obstacle detection and tracking layer that uses, for example, Kalman Filtering, as proposed by Snape et al (2011) [32]. For simplicity, from this point on, when referring to the radius of obstacle B (r_B), this refers to an enlarged radius that is the sum of the true radius with radius standard deviation and position standard deviation, as represented in

Table 3.1: Obstacle Message Definition with Uncertainty Parameters

Field	Symbol	Description	Units
Time Stamp		Timestamp	s, ns
ID		Identifier for the obstacle (0...N)	-
Class		Object class definition (0...M)	-
Dynamic		Static (False) or dynamic (True)	Boolean
Frame		Working frame for position and velocity information	-
Pose	$\mathbf{x}_B$	Current pose of the observed object	m
position_x_std	σ_x	Position uncertainty in x-direction	m
position_y_std	σ_y	Position uncertainty in y-direction	m
position_z_std	σ_z	Position uncertainty in z-direction	m
velocity	$\mathbf{v}_A$	Velocity vector (V_x, V_y, V_z)	m/s
velocity_x_std	σ_{vx}	Velocity uncertainty in x-direction	m/s
velocity_y_std	σ_{vy}	Velocity uncertainty in y-direction	m/s
velocity_z_std	σ_{vz}	Velocity uncertainty in z-direction	m/s
Size	r_B	Geometric dimensions of the obstacle	m
radius_std	σ_r	Radius estimate uncertainty	m
protective_zone		Safety margin around the obstacle	m

figure 3.3. Since the obstacles are modeled as spheres, σ_p is chosen as the maximum from σ_x , σ_y and σ_z .

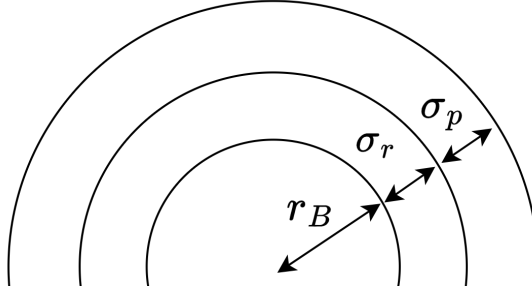


Figure 3.3: Obstacle radius as a sum of its estimated radius, radius standard deviation, and position standard deviation.

The defined obstacle representation allows for the algorithm to perceive three-dimensional obstacles in dynamic environments. This information comes from different obstacle detection and tracking technologies, or even a fusion of different ones.

Considering that LiDAR and Laser Scanners are standard sensors on surface vehicles, an obstacle detection and tracking approach based on LiDAR information was implemented [17]. The algorithm operates in two stages: first, it extracts obstacles from scan points, and then it applies independent Kalman filters to track each obstacle. Figure 3.4 illustrates this obstacle detection for circular and square obstacles, both modeled for the obstacle avoidance algorithm as spheres, showing the progression from raw sensor data to final obstacle representation with a protective

radius to ensure safety. The implemented method works with two-dimensional information, so a constant height is assumed to model the obstacles.

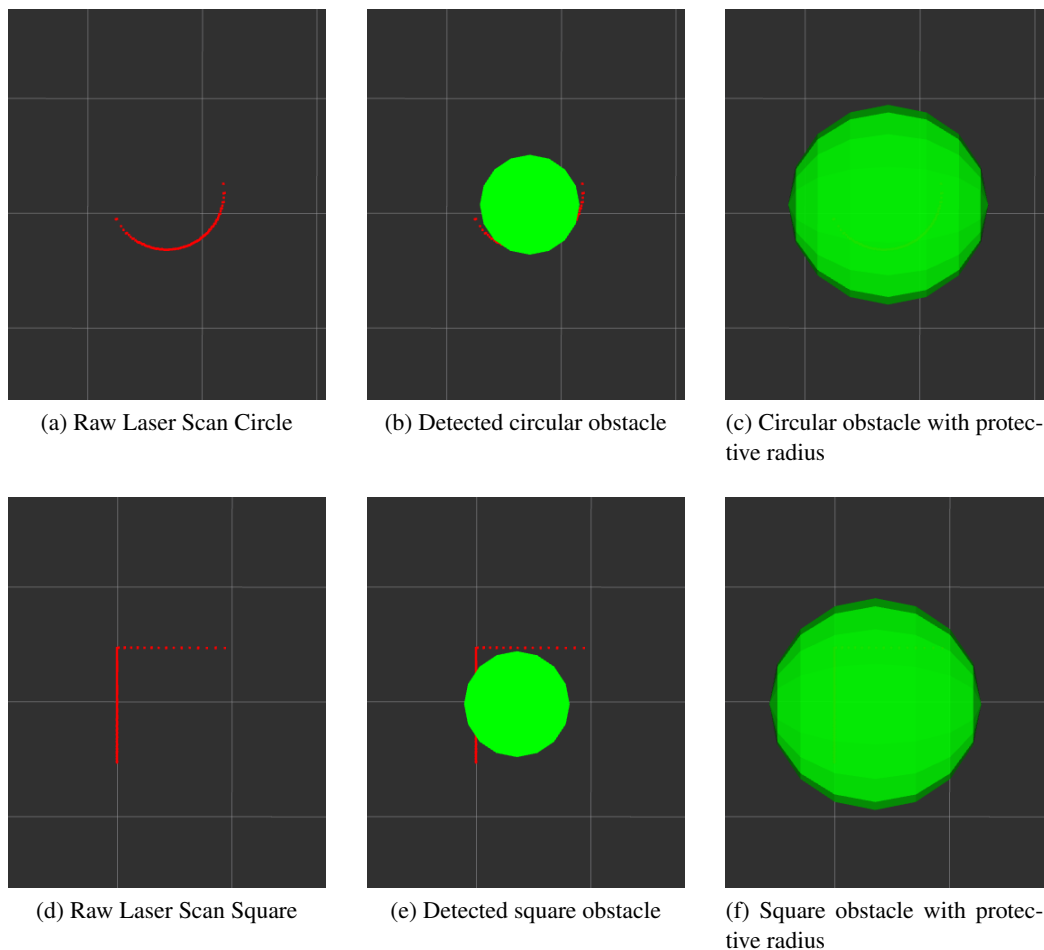


Figure 3.4: Obstacle detection for different shapes: The top row shows circular obstacles: (a) raw LaserScan data, (b) detected circular obstacle, and (c) final representation with protective radius. The bottom row shows square obstacle processing as a circle: (d) raw LaserScan data, (e) detected square obstacle, and (f) final representation with protective radius.

To implement this approach for the environment representation described, some specific changes were made to both detection and tracking methods. The first modification was applied to the obstacle extraction stage. The implementation uses a parameter named Max Circle Radius, which is used to classify detected shapes either as circular obstacles or linear segments. Given that the Velocity Obstacle model, the environment using circular obstacles (spheres in 3D or circles in 2D), this parameter was set to 5 meters. While this change classifies some free space in front of linear segments as obstacles, it will reinforce the preventive nature of the method, which is a positive thing for deployment in real mobile robots.

The second modification is related to obstacle tracking. Performing obstacle tracking while navigating is a complex case, as the system attempts to track the velocity of moving obstacles under the assumption that they follow linear motion, while the reference frame (i.e., the agent) itself

is also moving, not only linearly but also rotationally. The original approach supports compensating obstacle velocities based on the agent's linear and angular motion, but it does not handle agent acceleration, easily leading to oscillatory behavior in both the estimated velocities of obstacles and the resulting avoidance maneuvers. Obstacles are extracted in the agent's frame, then their geometric positions are transformed into world coordinates. Tracking and velocity estimation are performed in the world frame, after which the obstacles are transformed back into the agent's frame, enabling a scene representation in the world frame based on multiple cooperative agents in the future. This modified approach effectively eliminates tracking discrepancies in velocity estimation that were observed when comparing with ground truth velocity information, even under scenarios where the agent is accelerating, decelerating, or rotating.

The drawback of this solution is the dependence on a reliable agent's pose and orientation for estimating obstacle relative positions and velocities. In a simulated environment, the agent's world coordinates (or global poses) can be modeled; however, in a real-world scenario, these coordinates are provided by a localization system that introduces uncertainty. The positioning and orientation information from localization that connect world coordinates to agent coordinates, is represented mathematically by the homogeneous transformation matrix $H_{\text{world}}^{\text{agent}}$, which maps a point from the agent's frame to the world frame, and denoting the coordinates of an obstacle B in the agent frame as p_B^{agent} , and in the world frame as p_B^{world} , the transformation is given by equation 3.1. Obstacle velocity tracking based on world coordinates depends on successive obstacle position measurements and agent positioning, and therefore, inconsistency in successive robot position and orientation information leads to inconsistent homogeneous transformation matrices $H_{\text{world}}^{\text{agent}}$, and leads to errors in velocity estimation.

$$p_B^{\text{world}} = H_{\text{world}}^{\text{agent}} \cdot p_B^{\text{agent}} \quad (3.1)$$

By using separate Kalman Filters to track multiple objects previously detected in the scene, the uncertainties estimated by them are provided by the obstacle detection method to enhance the robustness of the scenario characterization and, at the end, the safety of robot navigation.

3.3 Sampling-Based Obstacle Avoidance

To estimate a collision-free solution for each robot, the linear velocity space is sampled for velocities that the vehicle can achieve within a given interval, according to the vehicle's kinematic limitations. This strategy offers significant advantages over alternative representations since the geometric definition of continuous regions for collisions and kinematic constraints can quickly lead to high-complexity optimization problems when performed in continuous spaces. Voxel or grid-based approaches for velocity space often suffer from an inherent trade-off between resolution and size that can significantly compromise resolution, risking the loss of optimal solutions, especially in 3D environments. Sampling the velocity space allows for merging the benefits of both referred approaches. By operating with a finite set of candidate velocities, the search space is manageable, and these samples are efficiently pruned according to geometric collision constraints with obstacles and the vehicle's kinematic limitations. The result is a discrete and collision-free set of velocity options that can be evaluated effectively.

The basic process for finding a safe velocity output consists of three steps: first, a finite set of control samples, depending on the vehicle kinematics and dynamics, is created and translated to their holonomic equivalent, as the linear velocity with which the robot would end up after the time interval parameter. Secondly, this set of linear velocity samples is pruned according to the obstacles' relative positions and velocities. Lastly, according to the desired velocity and the obstacles, the remaining samples are evaluated using a multicriteria heuristic function. This section proceeds with a more detailed description of these three steps.

Generating Samples

Each sample is defined by: v_x , v_y , v_z , and a cost. These linear velocities are obtained from the vehicle kinematics and represent the velocities that can be achieved within a given planning horizon Δt , considering limits on linear and angular velocities and accelerations, defining the set of Reachable Velocities (RV).

Formally, let $\mathcal{U}$ denote the set of admissible actuator efforts $\mathbf{u}$, Δt the planning time interval, and $\mathbf{v}_A(t_0)$ the velocity of agent A at the time t_0 . Then, with f representing a generic kinematic function that maps actuator efforts into resulting linear velocity, the set of reachable velocities is given by equation 3.2:

$$RV(t_0 + \Delta t) = \{\mathbf{v} \mid \mathbf{v} = f(\mathbf{u}, \mathbf{v}_A(t_0), \Delta t), \mathbf{u} \in \mathcal{U}\} \quad (3.2)$$

The kinematics of heterogeneous vehicles modeled according to 12 values, defines constraints on 6DOF. These values correspond to maximum linear accelerations, maximum angular accelerations, maximum linear velocities, and maximum angular velocities ($a_{x,\max}$, $a_{y,\max}$, $a_{z,\max}$, $\alpha_{\text{roll},\max}$, $\alpha_{\text{pitch},\max}$, $\alpha_{\text{yaw},\max}$, $v_{x,\max}$, $v_{y,\max}$, $v_{z,\max}$, $\omega_{\text{roll},\max}$, $\omega_{\text{pitch},\max}$, $\omega_{\text{yaw},\max}$). The minimum values are assumed to be the symmetrical counterparts of their maximum. These constraints are the basis for defining the set of admissible actuator efforts $\mathcal{U}$ and the function f . Sampling is

performed in the set of admissible actuator efforts, following an uniform distribution in this space, and then mapped with the function f to create RV . In addition to the randomly generated samples, two specific samples are always added: one corresponding to the vehicle's current velocity and another to the desired velocity, provided that both satisfy the kinematic constraints.

This section proceeds with three examples that illustrate sample generation from motion constraints: an omnidirectional surface vehicle, a differential drive surface vehicle, and an omnidirectional aerial vehicle. For each example, a table is presented with the kinematic constraints considered, and a figure captured from the 3D visualization environment *RVIZ* where the corresponding velocity samples are represented. The figures represent 10,000 samples of velocities that can be achieved within a three-second interval, starting with the agent at rest. The representation of velocities is done at a 1:1 distance scale, meaning 1 m/s appears represented as 1 m, corresponding to the resolution of the grid depicted in the figures. Orientation can be interpreted from the axes, where red, green, and blue correspond respectively to the x , y , and z directions. Table 3.2 shows a first example of sample generation for an omnidirectional surface vehicle, characterized by maximum accelerations in x and y of 1 m/s^2 and maximum linear velocities of 2 and 1 m/s in x and y , respectively. These restrictions form an ellipse with corresponding to the kinematic limits. Figure 3.5 allows visualizing a set of samples distributed within this elliptical velocity space.

Table 3.2: Maximum angular and linear velocity and acceleration values for an example omnidirectional surface vehicle.

Maximum Linear Acceleration	
$a_{x,\max}$	1.0 m/s^2
$a_{y,\max}$	1.0 m/s^2
$a_{z,\max}$	0.0 m/s^2
Maximum Angular Acceleration	
$\alpha_{\text{roll},\max}$	0.0 rad/s^2
$\alpha_{\text{pitch},\max}$	0.0 rad/s^2
$\alpha_{\text{yaw},\max}$	0.0 rad/s^2
Maximum Linear Velocity	
$v_{x,\max}$	2.0 m/s
$v_{y,\max}$	1.0 m/s
$v_{z,\max}$	0.0 m/s
Maximum Angular Velocity	
$\omega_{\text{roll},\max}$	0.0 rad/s
$\omega_{\text{pitch},\max}$	0.0 rad/s
$\omega_{\text{yaw},\max}$	0.0 rad/s

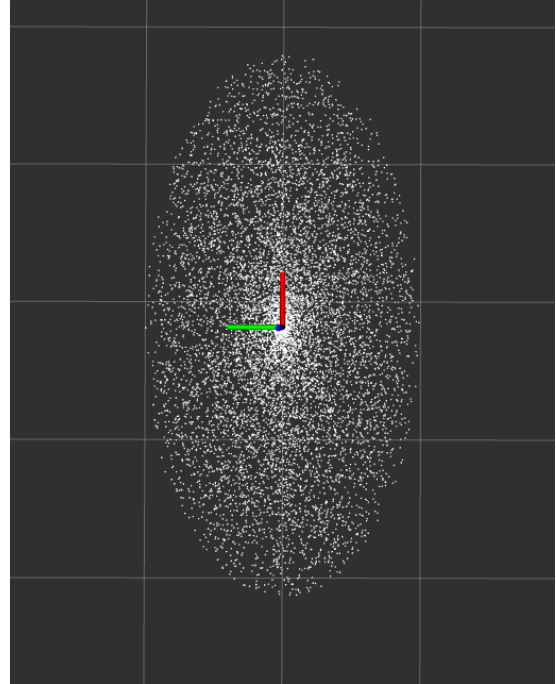


Figure 3.5: Omnidirectional surface vehicle velocity sampling space

Table 3.2 presents a second example of sample generation for a differential drive surface vehicle, characterized by a maximum linear acceleration of 1 m/s^2 and a maximum linear velocity of 1 m/s . For angular motion, the maximum angular acceleration is 1 rad/s^2 and the maximum

angular velocity is 1 rad/s. The mapping between velocities, from (v, ω_z) to linear velocities v_x and v_y , was performed according to the equations in 3.3, and the inverse mapping followed the equations in 3.4.

$$v = \sqrt{v_x^2 + v_y^2}, \quad \omega_{yaw} = \frac{\text{atan2}(v_y, v_x)}{\Delta t}; \quad (3.3)$$

$$v_x = v \cdot \cos(\omega_{yaw} \cdot \Delta t), \quad v_y = v \cdot \sin(\omega_{yaw} \cdot \Delta t). \quad (3.4)$$

These constraints result in a circular shape with two missing sectors, as illustrated by the set of samples in figure 3.6. The outer radius of the shape corresponds to the maximum achievable linear velocity. The empty sectors represent angular velocities that the vehicle is physically unable to achieve due to its dynamic limitations.

Table 3.3: Maximum angular and linear velocity and acceleration values for an example diff drive surface vehicle.

Maximum Linear Acceleration	
$a_{x,\max}$	1.0 m/s ²
$a_{y,\max}$	0.0 m/s ²
$a_{z,\max}$	0.0 m/s ²
Maximum Angular Acceleration	
$\alpha_{\text{roll},\max}$	0.0 rad/s ²
$\alpha_{\text{pitch},\max}$	0.0 rad/s ²
$\alpha_{\text{yaw},\max}$	0.25 rad/s ²
Maximum Linear Velocity	
$v_{x,\max}$	1.0 m/s
$v_{y,\max}$	0.0 m/s
$v_{z,\max}$	0.0 m/s
Maximum Angular Velocity	
$\omega_{\text{roll},\max}$	0.0 rad/s
$\omega_{\text{pitch},\max}$	0.0 rad/s
$\omega_{\text{yaw},\max}$	0.25 rad/s

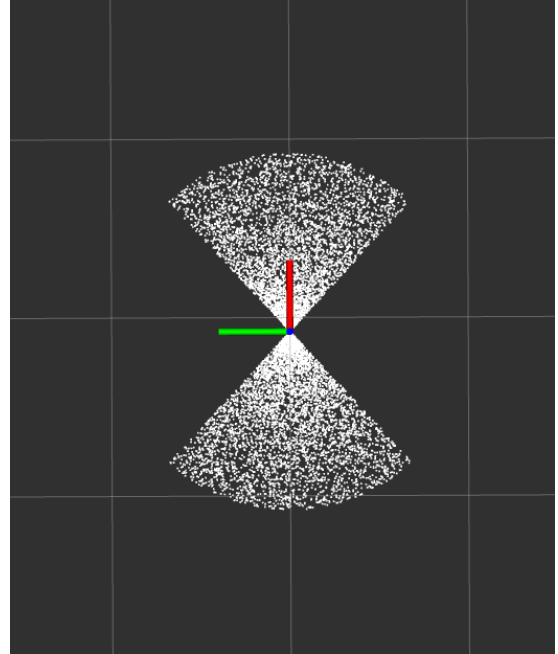


Figure 3.6: Diff-Drive surface vehicle velocity sampling space

As a final example, a multicopter vehicle produces a three-dimensional velocity space. Table 3.4 presents an example set of motion constraints: maximum linear accelerations of 1 m/s² in the x and y directions, and 0.5 m/s² in the z direction, along with maximum linear velocities of 1 m/s in x and y , and 0.5 m/s in z .

These constraints define an ellipsoid in velocity space, represented by a set of samples shown in figure 3.8 and 3.7. The varying semi-axis lengths of the ellipsoid reflect the different maximum velocities achievable in each direction.

Table 3.4: Maximum angular and linear velocity and acceleration values for an example multicopter vehicle.

Maximum Linear Acceleration	
$a_{x,\max}$	1.0 m/s ²
$a_{y,\max}$	1.0 m/s ²
$a_{z,\max}$	0.5 m/s ²
Maximum Angular Acceleration	
$\alpha_{\text{roll},\max}$	0.0 rad/s ²
$\alpha_{\text{pitch},\max}$	0.0 rad/s ²
$\alpha_{\text{yaw},\max}$	0.0 rad/s ²
Maximum Linear Velocity	
$v_{x,\max}$	1.0 m/s
$v_{y,\max}$	1.0 m/s
$v_{z,\max}$	0.5 m/s
Maximum Angular Velocity	
$\omega_{\text{roll},\max}$	0.0 rad/s
$\omega_{\text{pitch},\max}$	0.0 rad/s
$\omega_{\text{yaw},\max}$	0.0 rad/s

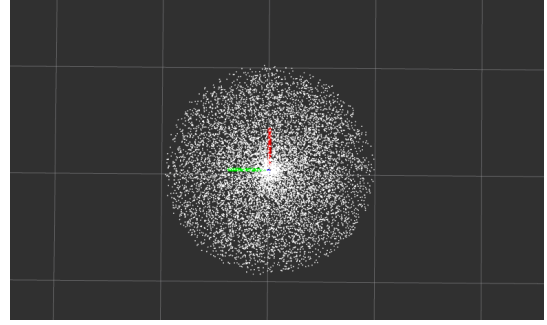


Figure 3.7: Top view of the multicopter sampling space.

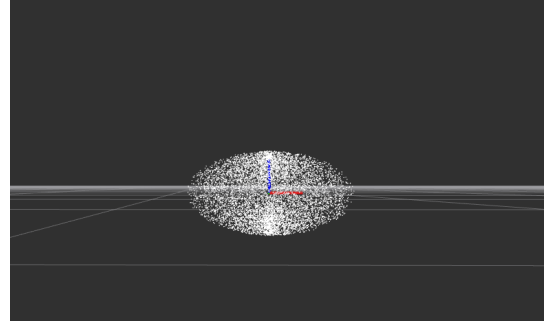


Figure 3.8: Side view of the multicopter sampling space.

Sample Pruning

The sample generation step incorporates the vehicle's kinematic constraints. The second step, presented next, addresses the conditions introduced by the dynamic environment, pruning velocities that would result in a collision, and evaluating a cost for each maneuver.

The method for evaluating whether a velocity leads to a collision is based on the fundamental concept of the Velocity Obstacle approach. First, the spherical representation of each obstacle B is augmented with a Minkowski sum involving the spherical representation of agent A . Each obstacle induces a Relative Collision Cone in A , denoted as CC_{AB} . This cone is defined as the set of velocities for agent A , reduced to a point $\hat{A}$, that would result in a trajectory intersecting B . Geometrically, this corresponds to an infinite-height cone with its vertex at $\hat{A}$, tangent to the augmented obstacle $\hat{B}$. Finally, the Velocity Obstacle induced by B on agent A corresponds to translating the Relative Collision Cone by the obstacle's velocity, v_B . Similarly to other implementations [45], a time horizon simplification is made on velocities that, although leading to a collision, do so after a defined threshold Time Horizon. This is done by calculating the minimum time to collision with a given obstacle B . Specifically, the distance d_B between the agent and the center of obstacle B , the combined radius $r_B + r_A$ accounting for both the obstacle's and the agent's size, and the relative velocity v_{AB} between the two entities. The time to collision is estimated and compared with a time

threshold to decide whether the sample is considered safe.

Figure 3.9a depicts the obstacle representation and its Minkowski sum with the agent's size, the Relative Collision Cone, the Velocity Obstacle, and the agent and obstacle velocity. Figure 3.9b illustrates the Time Horizon Truncation.

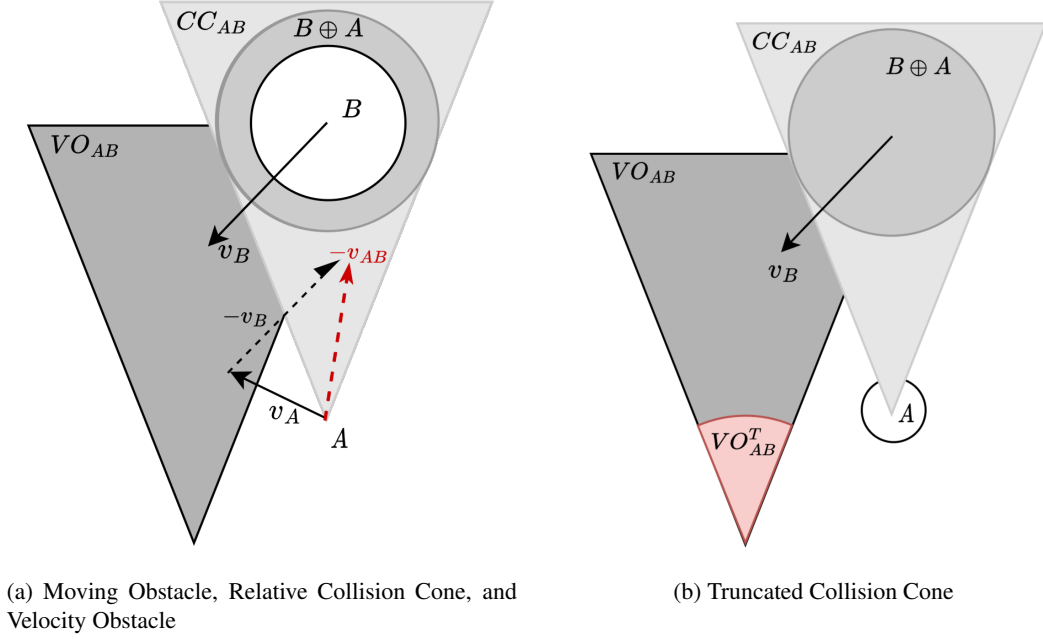


Figure 3.9: Comparison between full and truncated Velocity Obstacle representations.

By definition, the velocities within VO_{AB} are the set of velocities that lead to a collision. As such, the set of samples generated in the previous step is pruned, creating the set of feasible non-colliding velocities. In practice, this pruning is done by verifying a set of geometric conditions.

Firstly, the velocity of the sample is translated by $-\vec{v}_B$, to account for the obstacle's motion, as shown in Equation 3.5:

$$\vec{v}_{AB} = \vec{v}_{\text{sample}} - \vec{v}_B \quad (3.5)$$

This translated velocity is denoted as the relative velocity $\vec{v}_{AB}$. Next, the position of the obstacle $\vec{p}_B$, its geometric radius r_B , the radius of the vehicle r_A , and an additional protective zone $r_{\text{protective}}$ are used to define the collision cone geometry. The effective obstacle radius is computed using Equation 3.6:

$$r_{\text{obs}} = r_A + r_B + r_{\text{protective}}. \quad (3.6)$$

The Euclidean distance from the vehicle to the obstacle is then calculated, as shown in Equation 3.7:

$$d_{\text{obs}} = \|\vec{p}_B\|. \quad (3.7)$$

With these values, the half-angle of the collision cone can be derived using the arctangent of the radius-to-distance ratio, according to Equation 3.8:

$$\theta_{\text{cone}} = \tan^{-1} \left(\frac{r_{\text{obs}}}{d_{\text{obs}}} \right) \quad (3.8)$$

To assess the potential for collision, the projection p of the relative velocity vector onto the obstacle position vector is computed using the dot product, as shown in Equation 3.9. If this projection is negative, the sample moves away from the obstacle, and no collision check is necessary.

$$p = \frac{\vec{v}_{AB} \cdot \vec{p}_B}{\|\vec{p}_B\|}. \quad (3.9)$$

Following Equation 3.9, the expected radial distance of the sample from the cone axis at this projection is calculated by:

$$r_{\text{expected}} = p \cdot \tan(\theta_{\text{cone}}). \quad (3.10)$$

At the same time, the actual radial offset of the relative velocity from the cone axis is given by Equation 3.11:

$$r_{\text{actual}} = \sqrt{\|\vec{v}_{AB}\|^2 - p^2}. \quad (3.11)$$

A potential collision is detected if the actual radial distance falls within the cone, that is, if the condition in Equation 3.12 is satisfied:

$$r_{\text{actual}} < r_{\text{expected}}. \quad (3.12)$$

The time to collision is estimated according to equation 3.13 and compared with a time to collision threshold to decide whether the sample is considered safe.

$$t_{\text{collision},B} = \frac{d_B - (r_B + r_A)}{v_{AB}}. \quad (3.13)$$

Formally, considering the planning time horizon $t_{\text{collision},\text{max}}$, the Velocity Obstacle VO_{AB} is defined as the set of agent velocities satisfying both the collision condition and the time constraint:

$$VO_{AB} = \{ \mathbf{v} \mid r_{\text{actual}}(\mathbf{v}) < r_{\text{expected}}(\mathbf{v}) \wedge t_{\text{collision},B}(\mathbf{v}) < t_{\text{collision},\text{max}} \}, \quad (3.14)$$

where r_{actual} , r_{expected} , and $t_{\text{collision},B}$ are given by Equations 3.11, 3.10, and 3.13, respectively. A representation of a pruned set of samples with a small time-horizon-constraint

Considering the set of detected obstacles $\mathcal{B} = \{B_1, B_2, \dots, B_N\}$, the combined Velocity Obstacle VO_A is defined as the union of the individual velocity obstacles generated by each obstacle:

$$VO_A = \bigcup_{B_i \in \mathcal{B}} VO_{AB_i}. \quad (3.15)$$

Then, the set of Reachable Avoidance Velocities within Δt (RAV), defined as the subset of reachable velocities not leading to collision with any detected obstacle within the planning horizon, is given by equation 3.16:

$$RAV(t_0 + \Delta t) = RV(t_0 + \Delta t) \setminus VO_A. \quad (3.16)$$

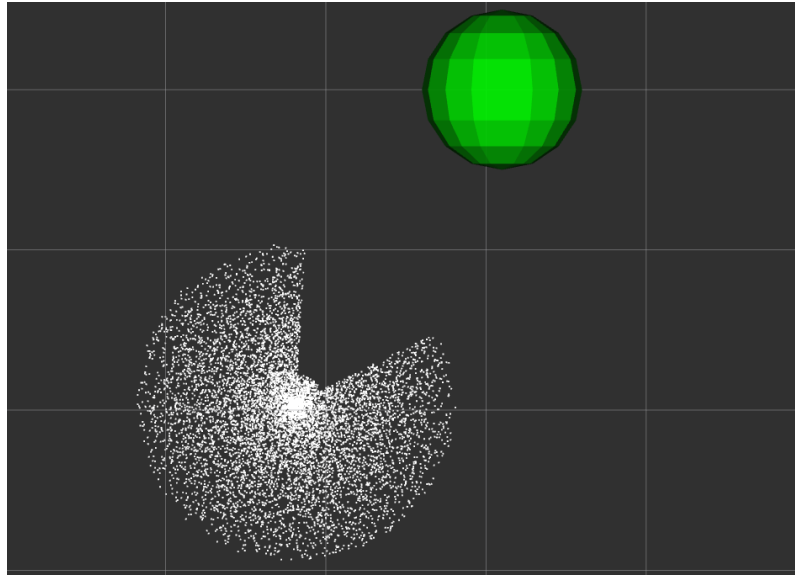


Figure 3.10: Pruned Samples from a holonomic planar vehicle when facing a static obstacle

Sample Cost Heuristic

Each sample is evaluated using a multicriteria heuristic cost function [45]. The cost metric is composed of a weighted sum of three components: the heading cost, which penalizes velocities that deviate from the desired direction; the magnitude cost, which penalizes velocities that do not match the desired velocity magnitude; and the danger cost, which penalizes maneuvers close to Velocity Obstacle, reflecting the risk of leading to a collision. The output velocity is the sample with the lowest total cost according to the defined weights. Figure 3.11 shows the heading cost, Figure 3.12 the magnitude cost, and Figure 3.13 the danger cost for two obstacle scenarios.

The direction error, C_{dir} , quantifies how well the sample velocity vector $\mathbf{v}_s = (v_{sx}, v_{sy}, v_{sz})$ aligns with the desired velocity vector $\mathbf{v}_d = (v_{dx}, v_{dy}, v_{dz})$. It is calculated using the dot product of their normalized vectors.

The dot product between the normalized sample velocity and normalized desired velocity is computed following equation 3.17:

$$\hat{\mathbf{v}}_s \cdot \hat{\mathbf{v}}_d = \frac{v_{sx}v_{dx} + v_{sy}v_{dy} + v_{sz}v_{dz}}{\|\mathbf{v}_s\| \cdot \|\mathbf{v}_d\|}. \quad (3.17)$$

The direction error is then normalized to the range $[0, 1]$ by inverting and scaling the dot product, as shown in equation 3.18:

$$C_{dir} = \frac{1 - (\hat{\mathbf{v}}_s \cdot \hat{\mathbf{v}}_d)}{2} \quad (3.18)$$

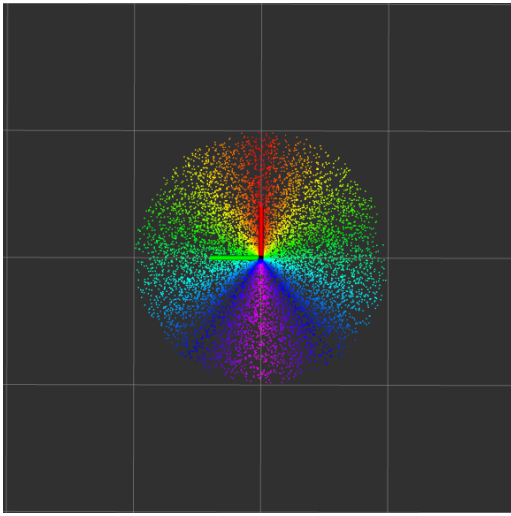


Figure 3.11: Heading cost: goes from low (red) to high (purple) according to the angle between the desired and sample directions.

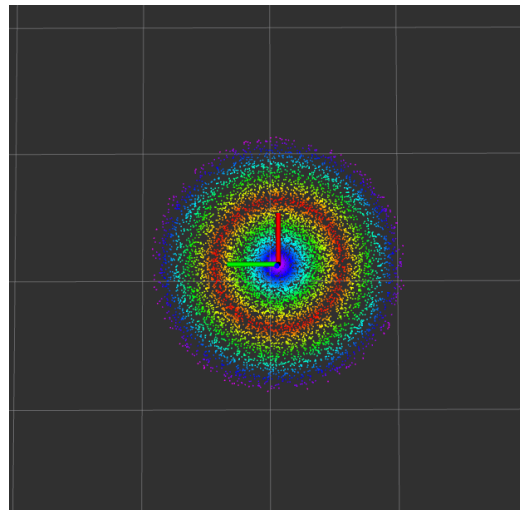


Figure 3.12: Magnitude cost: goes from low (red) to high (purple) according to the normalized magnitude error.

The magnitude error (C_{mag}) quantifies the absolute difference between the sample and desired velocity magnitude. The raw magnitude difference is calculated using the equation in 3.19:

$$\Delta_{mag} = \left| \|\mathbf{v}_s\| - \|\mathbf{v}_d\| \right|. \quad (3.19)$$

To ensure consistent scaling across all operating conditions, this error is normalized by the vehicle's maximum velocity (v_{max}) as shown in the following equation 3.20:

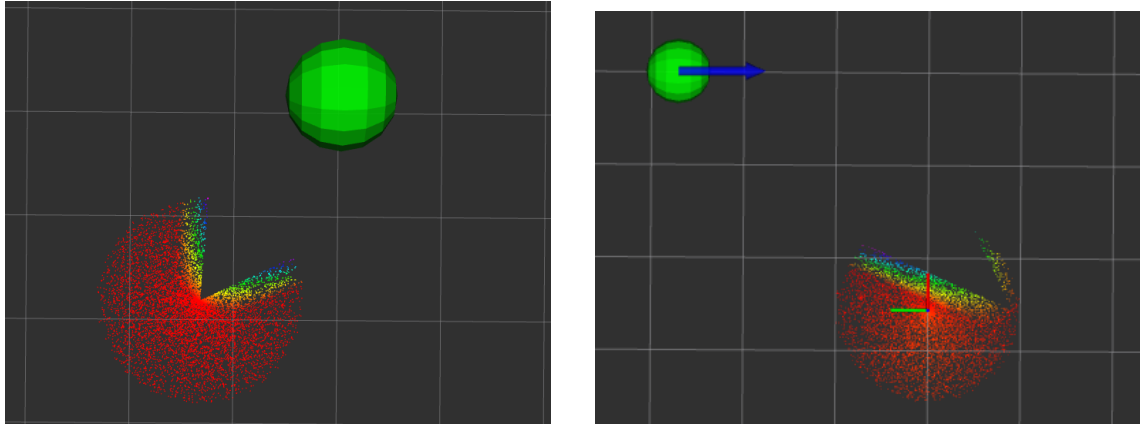
$$C_{mag} = \frac{\Delta_{mag}}{v_{max}} = \frac{\left| \|\mathbf{v}_s\| - \|\mathbf{v}_d\| \right|}{v_{max}} \quad (3.20)$$

The final component of the cost function is a risk-aware penalty applied to velocity samples that are close to but not strictly inside a Velocity Obstacle (VO). This cost helps to avoid tangential movements near obstacles, whether static or dynamic. It supports a preemptive collision avoidance behavior under uncertainty in velocity and position estimates, as proposed by [45].

This penalty is applied when the velocity sample lies near the boundary of the collision cone. In particular, if the projection of the relative velocity onto the obstacle direction is positive, indicating motion toward the obstacle, and the difference between actual and expected radial offsets is smaller than a threshold r_{thresh} , a cost is added to the sample. This cost is computed according to Equation 3.21:

$$C_{\text{danger}} = \left(\frac{r_{\text{thresh}} - (r_{\text{actual}} - r_{\text{expected}})}{r_{\text{thresh}}} \right) \cdot p. \quad (3.21)$$

Equation 3.21 distributes the cost linearly within the range defined by r_{thresh} , with maximum penalty at the cone boundary and no penalty at the outer threshold. Using the projection term p further scales the cost based on how directly the velocity points toward the obstacle, providing stronger penalties for higher and more dangerous approach speeds.



(a) Danger cost around the velocity cone of a static obstacle

(b) Danger cost around a moving obstacle velocity cone

Figure 3.13: Danger cost for a static (a) and a moving obstacle (b). Cost is represented from low cost (red) to high cost (purple).

The resulting cost is a weighted sum of these costs. The weights can be adjusted according to the vehicle and the environment. Figure 3.14 presents examples of the generated samples and their cost for an omnidirectional surface vehicle, dealing with one moving obstacle (Figure 3.13b) and with two static obstacles (Figure 3.14b).

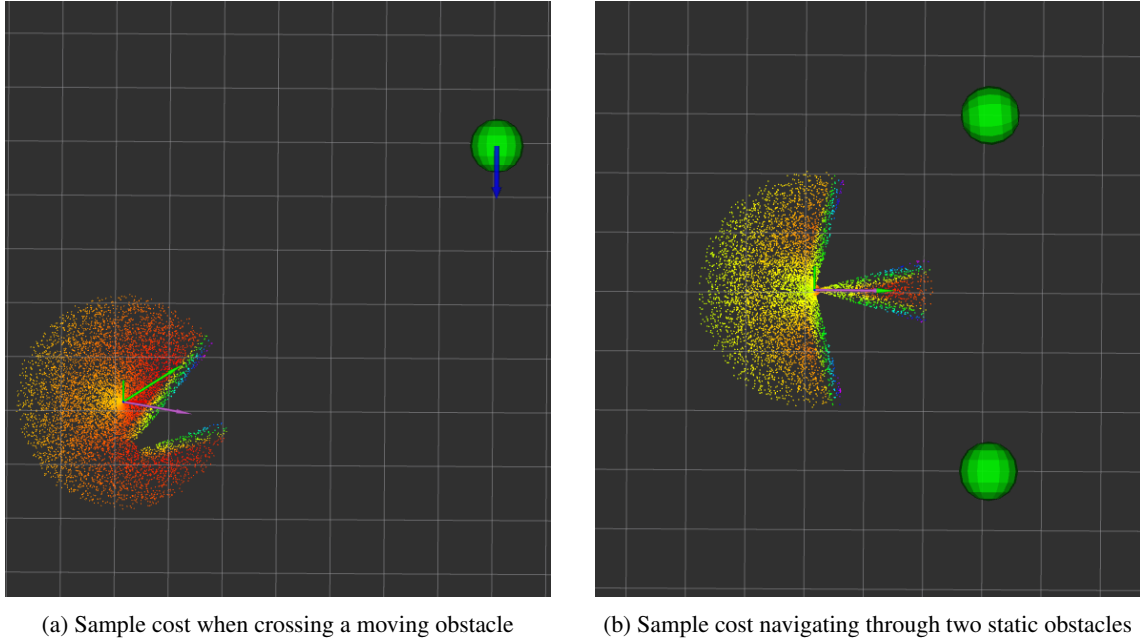


Figure 3.14: Examples of velocity sample cost (a) and cost evaluation (b) for obstacle avoidance. Cost is mapped from low (red) to high (purple). The pink arrow represents the desired velocity vector, and the green one is the command velocity outputted from the obstacle avoidance algorithm. In picture (b), these arrows overlap.

3.4 JAVOA Considerations

JAVOA considers that obstacles are always modeled as spheres, which does not always allow for accurate representation. For example, an obstacle with the shape of a cylinder whose height is much greater than its base radius would require some transformation in the obstacle detection stage to be modeled as a sphere. Any such transformation will not correctly represent the cylindrical obstacle's geometry: one alternative could be to consider a sphere that encloses the entire cylinder, which would incorrectly classify empty space as occupied. Another, geometrically better, alternative would be to decompose the cylindrical obstacle into a stack of spheres, which would significantly increase the complexity of dealing with an otherwise geometrically simple obstacle.

Obstacle velocity information is also reduced to linear velocities. This approximation is generally acceptable for maritime navigation since trajectories are approximately straight, especially at medium to high speeds. However, if a vehicle moves along a circular trajectory, its angular velocity is not considered in the algorithm, leading to incorrect decision-making.

Chapter 4

Methodology and Results

This chapter aims to demonstrate the performance of JAVOA as an effective obstacle avoidance method. A series of extensive experiments showcases its ability to create accurate representations of dynamic environments and follow safe trajectories, handling uncertainty with robustness. This chapter is organized as follows: Section 4.1 introduces the algorithm’s purpose and the experimental objectives. Section 4.2 describes how JAVOA was integrated into heterogeneous robotic vehicles and validates the algorithm on three distinct simulated platforms. Section 4.3 evaluates the algorithm in a more complex and realistic simulation environment, integrating it with a realistic low-level control layer. Finally, Section 4.4 presents the real-world testing setup and demonstrates the algorithm’s performance on a real setup and environment.

4.1 Introduction

The algorithm presented in this work, JAVOA, introduces an obstacle avoidance approach for mobile robots operating in dynamic environments. JAVOA explicitly considers the kinematics of robotics vehicles and allows for easy adaptability between heterogeneous robotic platforms. It also accounts for a risk-based metric that allows for robustness to uncertainty in perception and control.

Two simulation stages were conducted, and their results are presented in this section. To assess the algorithm’s effectiveness in modeling heterogeneous agents, the first set of tests explores applying it to three different kinematic constrained vehicles: A holonomic surface vehicle, a differential drive surface vehicle, and a multirotor aerial vehicle. The second simulation environment validates the algorithm’s robustness in an environment with water dynamics, and a thrust low-level control layer evaluates its robustness.

Real-world integration tests were conducted in the ATLANTIS Test Center in Viana do Castelo, Portugal. These tests validate that the approach can correctly model real-world dynamic maritime environments using LiDAR-based obstacle detection and tracking methodologies. JAVOA was integrated into an assisted teleoperation setup, providing an object-centric environment representation and feasibility and cost evaluation of the velocity sample space.

The testing scenarios presented in this chapter evaluate different behaviours intended for the algorithm to perform: Dealing with static obstacles, giving way to dynamic obstacles, overtaking dynamic obstacles while maintaining a safe distance from them, and reaching the goal.

4.2 Heterogeneous Vehicle Simulation

The first development and testing environment consists of a simple simulation setup in the *Gazebo* simulation environment. The agent is modeled as a cuboid with 1-meter sides, and the obstacles as spheres with a diameter of 1 meter. Figure 4.1 shows the simulation environment with the agent (in blue) and several obstacles (in red). The main goal of this simulated environment was to have a generic vehicle that, by setting different constraints, could be used to model different types of vehicles, supporting the algorithm development and validating its behaviour across different vehicles and environments. Three vehicles were simulated in this environment: an omnidirectional USV, a differential-drive USV, and a multirotor UAV.

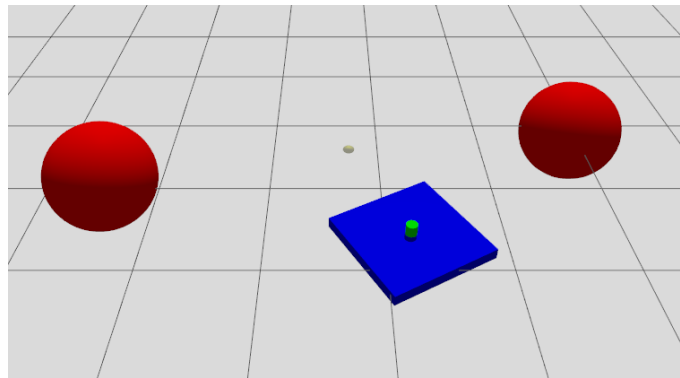


Figure 4.1: Simple Simulation environment. The agent is modeled as a blue cuboid, the obstacles as red spheres, and the Laser Scan Sensor as a green cylinder on top of the agent.

The overall navigation pipeline used in these tests is represented in the diagram in Figure 4.2. The simulated vehicle is directly controlled from the obstacle avoidance layer, using a velocity control plugin, with no control layer. The desired velocity is generated as a vector with constant magnitude pointing toward the goal. Obstacle detection was first done with ground truth obstacle information, but LiDAR Detection and Tracking were also tested.

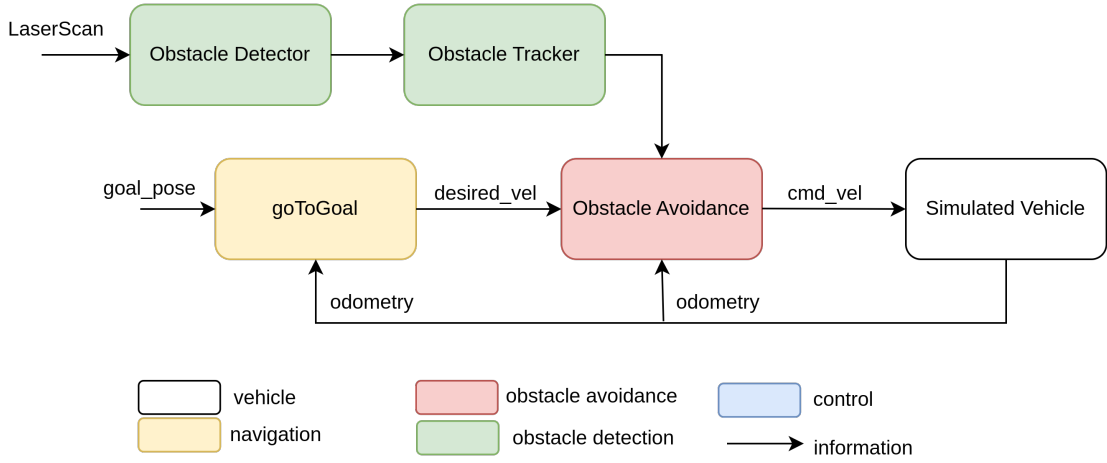


Figure 4.2: Pipeline Overview for the Simplified Gazebo Simulation

The position and velocity information of the agent in the world is ideal and obtained directly from *Gazebo*. Similarly, the command velocity coming from the obstacle avoidance layer was directly converted through a *Gazebo* Velocity Controller plugin. Regarding obstacle information, two different approaches were used. In the first phase, ideal information is considered, directly obtaining obstacle position and velocity data in world coordinates from the simulation, and converting them into agent coordinates based on the agent's position in the simulated world. In a second phase, a Laser Scan sensor is centered on the agent. Laser Scan points were transformed into individual obstacles using the implemented obstacle detection and tracking algorithm. Since both examples work on ideal obstacle information with no noise, the obstacle radius enlargement based on Kalman Filter parameters is not performed.

The global layer used is a simple straight line between the agent and the goal. The velocity command is obtained by first computing the error vector from the agent to the goal (equation 4.1), normalizing its magnitude, and multiplying it by a constant desired velocity magnitude v_{const} .

To achieve more stable navigation and allow the vehicle to stop when the goal is reached, a slow approach and stop can be implemented, gradually slowing down when approaching the goal after the distance $d_{approach}$, and stopping when closer than d_{stop} , as described in equation 4.3,

$$\vec{e} = \vec{p}_{goal} - \vec{p}_{current}; \quad (4.1)$$

$$d = \|\vec{e}\| = \sqrt{e_x^2 + e_y^2 + e_z^2}; \quad (4.2)$$

$$\vec{v}_{desired} = \begin{cases} \vec{0} & \text{if } d < d_{stop} \text{ (goal reached)} \\ \frac{\vec{e}}{d_{stop}} \cdot v_{const} & \text{if } d_{stop} \leq d < d_{approach} \text{ (slow approach) ,} \\ \frac{\vec{e}}{d} \cdot v_{const} & \text{if } d \geq d_{approach} \text{ (constant speed)} \end{cases} \quad (4.3)$$

where $v_{\text{const}} = 1.0$ m/s is a constant desired speed parameter. It is important to mention that the velocity output from this Navigation Layer should be converted to the agent frame to allow the Obstacle Avoidance Layer to work fully in a local coordinate system. In the following results section, the distance thresholds d_{approach} and d_{stop} were set to the agent radius, turning off the slow approach intentionally to isolate the behaviour of the obstacle avoidance layer.

4.2.1 Evaluation Metrics

The results presented focus on different performance metrics that aim to validate the designed algorithm. The first and most important one is the trajectory of the agent and the obstacles, described by a plot of their position over time. The trajectory's length is compared to the straight line between the start and end positions, resulting in a Path Elongation ratio that provides insight into the distance traveled, normalized by what would be a path without obstacles. Since the main focus is to guarantee safety, the Obstacle Clearance metric describes the minimum distance to an obstacle. This metric accounts for the obstacle and agent radius, but not the protective zone¹. Depending on the specific vehicle, the velocity profile for v_x , v_y , v_z , and ω_{yaw} over time is presented, as well as a velocity magnitude over time profile. Lastly, it is indicated whether the goal was reached in that scenario or the vehicle got stuck.

4.2.2 Defined Parameters and Testing Scenarios

The configuration parameters used in the presented tests are detailed in Table 4.1 and Table 4.2, related to the obstacle avoidance layer and the navigation layer, respectively. In addition to the navigation parameters defined for each vehicle, in all scenarios, the obstacles have a fixed protective zone of 0.5 m, and the goal is set to [10, 0, 0] m.

The test scenarios presented are listed in Table 4.16, where each scenario is identified using a short label for easier referencing, along with a preview image and a description. These labels were used during developed to describe the scenarios, and consist of a letter and a number, where the letters help distinguish different types of scenarios (S - Static; D - Dynamic; C - Complex).

¹If for example the obstacle and the agent have radius 1 m, the protective zone is 0.5 m, and the minimum distance between the agent and obstacle position is 3 meters, the minimum Obstacle Clearance would be $3 - 1 - 1 = 1m$

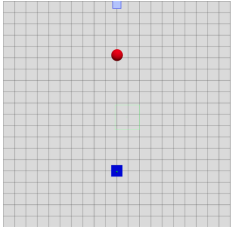
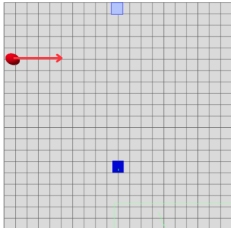
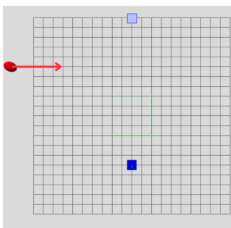
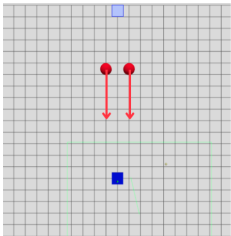
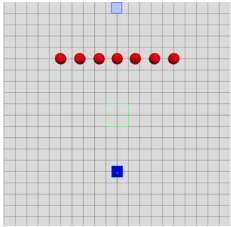
Table 4.1: Configuration parameters for the Obstacle Avoidance Layer for Heterogeneous Vehicle Simulation.

Parameter [unit]	2D Holonomic	2D Diff-Drive	Multicopter (3D)
heading_weight	0.70	0.50	0.70
abs_weight	0.30	0.50	0.30
danger_weight	0.90	0.90	0.90
vehicle_radius [m]	1.4	1.4	1.4
delta_t [s]	3.0	3.0	3.0
num_samples [-]	10 000	10 000	10 000
radius_threshold [m]	0.1	0.1	0.1
time_to_collision_threshold [s]	120.0	120.0	120.0
a_x_max [m/s ²]	1.0	1.0	1.0
a_y_max [m/s ²]	1.0	0.0	1.0
a_z_max [m/s ²]	1.0	0.0	1.0
a_roll_max [rad/s ²]	0.0	0.0	0.0
a_pitch_max [rad/s ²]	0.0	0.0	0.0
a_yaw_max [rad/s ²]	0.0	1.0	2.0
v_x_max [m/s]	1.0	1.0	2.0
v_y_max [m/s]	1.0	0.0	2.0
v_z_max [m/s]	0.0	0.0	2.0
w_roll_max [rad/s]	0.0	0.0	0.0
w_pitch_max [rad/s]	0.0	0.0	0.0
w_yaw_max [rad/s]	0.0	1.0	0.0

Table 4.2: Configuration parameters for the Navigation Layer for Heterogeneous Vehicle Simulation.

Parameter [unit]	2-D Holonomic	2-D Diff-Drive	Multicopter (3-D)
desired_constant_velocity [m/s]	1.0	1.0	1.0

Table 4.3: 2D Scenarios with corresponding visual, label, and description.

Label	Figure	Description
s0		Single static obstacle blocking a direct path. Agent start $[-0, 0, 0]$. Obstacle start $[5, 0, 0]$.
d0		Single dynamic obstacle moving along the y-axis. The relative position favors giving way. Agent start $[-5, 0, 0]$. Obstacle start $[5, 11, 0]$. Obstacle velocity $[0, -1, 0]$.
d1		Single dynamic obstacle moving along the y-axis. The relative position favors overtaking. Agent start $[-5, 0, 0]$. Obstacle start $[5, 13, 0]$. Obstacle velocity $[0, -1, 0]$.
d3		Two dynamic obstacles moving towards the agent along the x-axis. Agent start $[-5, 0, 0]$. Obstacle 1 start $[5.0, 1.0, 0]$. Obstacle 2 start $[5.0, -1.0, 0]$. Obstacle velocity $[-1, 0, 0]$.
c4		Line of seven static obstacles forming a wall that blocks a direct path Agent start $[-5, 0, 0]$. Obstacles at: $[5.0, -5.0, 0]$, $[5.0, -3.3, 0]$, $[5.0, -1.6, 0]$, $[5.0, 0.0, 0]$, $[5.0, 1.6, 0]$, $[5.0, 3.3, 0]$, $[5.0, 5.0, 0]$.

4.2.3 2D Holonomic Vehicle Results

This section validates the algorithm to be applied to a 2D holonomic surface vehicle. In this section, the scenarios to which this simulated vehicle was subjected are initially presented, then, for each scenario, the evaluation metrics are presented in the form of figures, plots, or tables. Finally, an analysis of the results obtained for this vehicle is presented.

- **S0: Single Static Obstacle:** The trajectory is shown in Figure 4.3, the velocity components profile in Figure 4.4, the velocity magnitude profile in Figure 4.5, and the results table in Table 4.4.
- **D0: Single Dynamic Obstacle: Give Way:** The trajectory is shown in Figure 4.6, the velocity components profile in Figure 4.7, the velocity magnitude profile in Figure 4.8, and the results table in Table 4.5.
- **D1: Single Dynamic Obstacle: Overtake:** The trajectory is shown in Figure 4.9, the velocity components profile in Figure 4.10, the velocity magnitude profile in Figure 4.11, and the results table in Table 4.6.
- **C4: Obstacle Wall:** The trajectory is shown in Figure 4.12, the velocity components profile in Figure 4.13, the velocity magnitude profile in Figure 4.14, and the results table in Table 4.7.

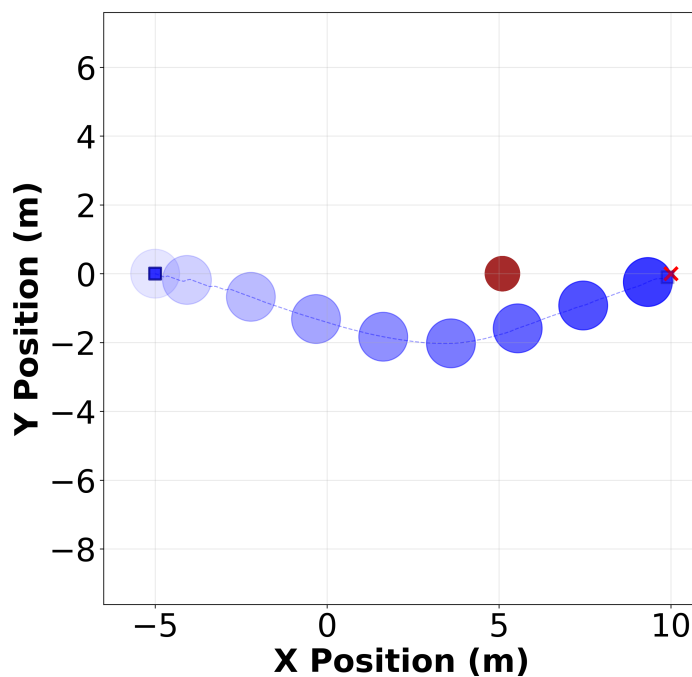
S0: Single Static Obstacle

Table 4.4: Holonomic Scenario S0: Metrics Table

Metric	Value
Start Pos	(-5.00, 0.0, 0.0)
End Pos	(9.40, -0.21, 0.0)
Goal Pos	(10.0, 0.0, 0.0)
Path Length	15.04 m
Path Elongation	1.04×
Min. Clearance	0.44 m
Reached Goal	True

Figure 4.3: Holonomic Scenario S0: XY robot trajectory

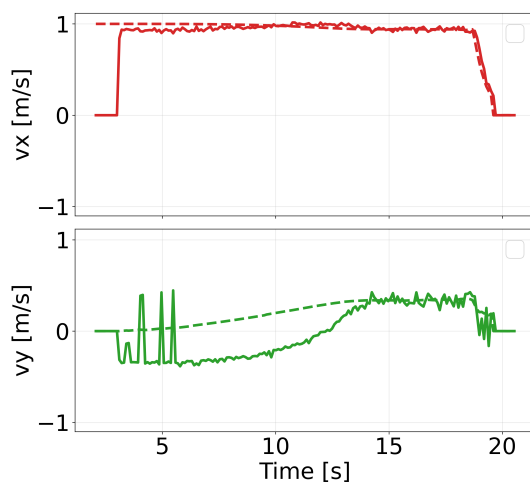
2D Linear Velocity Components Over Time

Figure 4.4: Holonomic Scenario S0: Velocity components over time. The dotted line indicates the desired velocity (desired_vel) and the continuous line indicates the commanded velocity (cmd_vel).

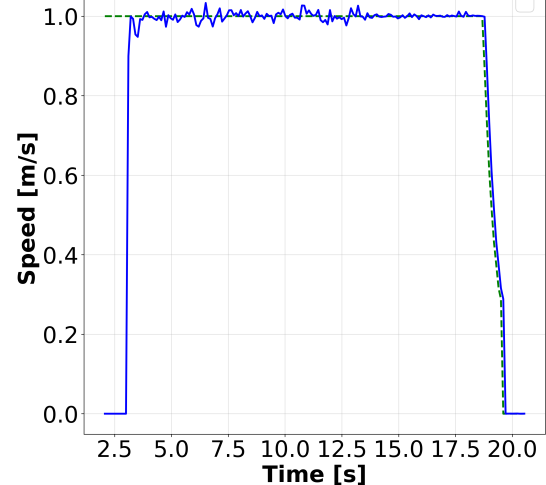
Linear Velocity Magnitude Over Time

Figure 4.5: Holonomic Scenario S0: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line indicates the commanded velocity magnitude.

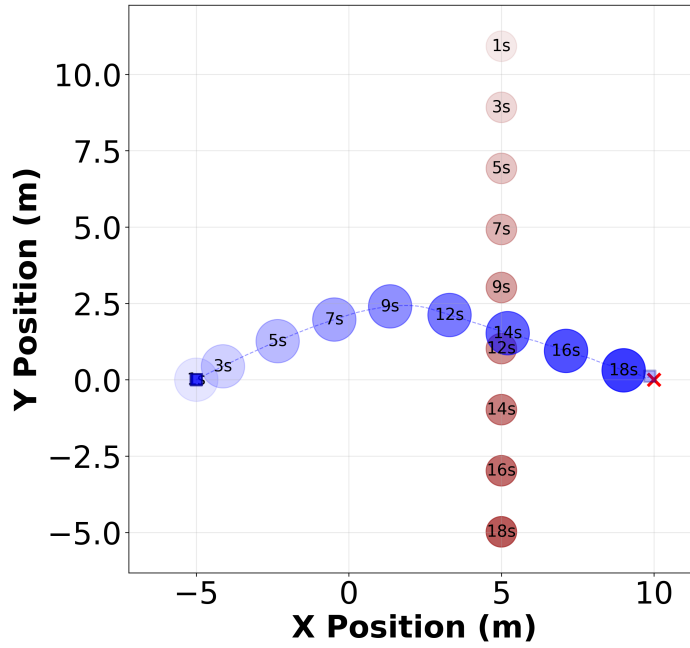
D0: Single Dynamic Obstacle: Give Way

Table 4.5: Holonomic Scenario D0: Metrics Table

Metric	Value
Start Pos	(-5.00, 0.00, 0.00)
End Pos	(9.39, 0.20, 0.00)
Goal Pos	(10.00, 0.00, 0.00)
Path Length	15.21 m
Path Elongation	1.06×
Min. Clearance	0.69 m
Reached Goal	True

Figure 4.6: Holonomic Scenario D0: XY robot trajectory

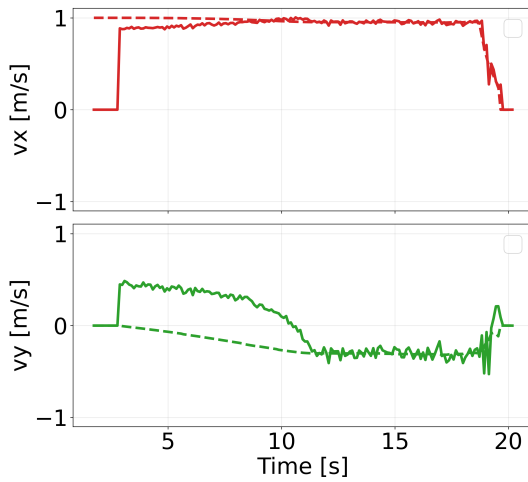
2D Linear Velocity Components Over Time

Figure 4.7: Holonomic Scenario D0: Velocity components over time. The dotted line indicates the desired velocity (desired_vel) and the continuous line indicates the commanded velocity (cmd_vel).

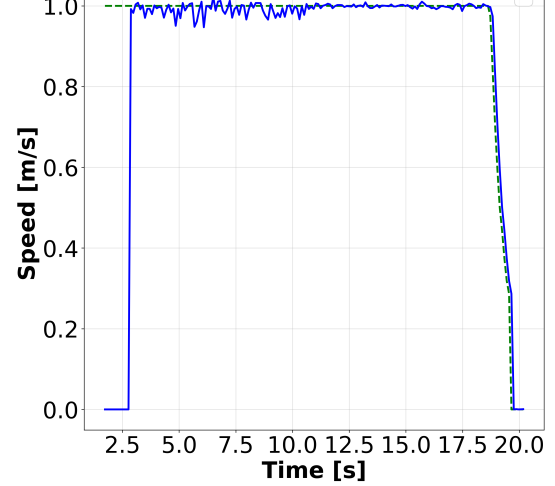
Linear Velocity Magnitude Over Time

Figure 4.8: Holonomic Scenario D0: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line indicates the commanded velocity magnitude.

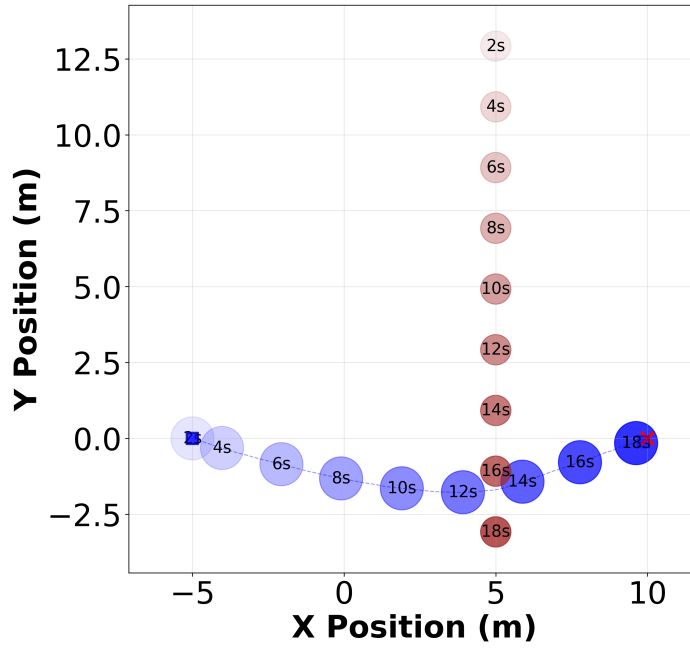
D1: Single Dynamic Obstacle: Overtake

Table 4.6: Holonomic Scenario D1: Metrics Table

Metric	Value
Start Pos	(-5.0, 0.0, 0.0)
End Pos	(9.37, -0.21, 0.0)
Goal Pos	(10.00, 0.0, 0.0)
Path Length	14.84 m
Path Elongation	1.03×
Min. Clearance	0.87 m
Reached Goal	True

Figure 4.9: Holonomic Scenario D1: XY robot trajectory

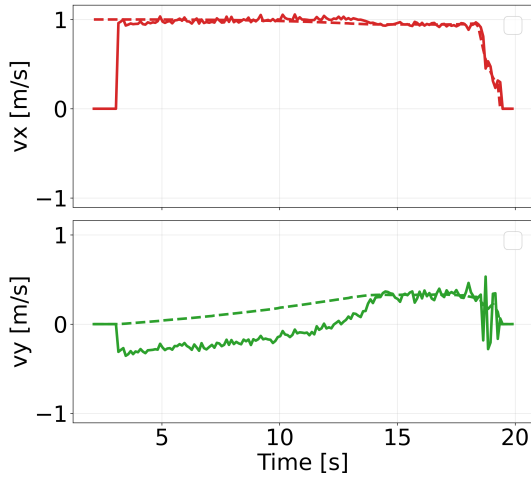
2D Linear Velocity Components Over Time

Figure 4.10: Holonomic Scenario D1: Velocity components over time. The dotted line indicates the desired velocity (desired_vel) and the continuous line indicates the commanded velocity (cmd_vel).

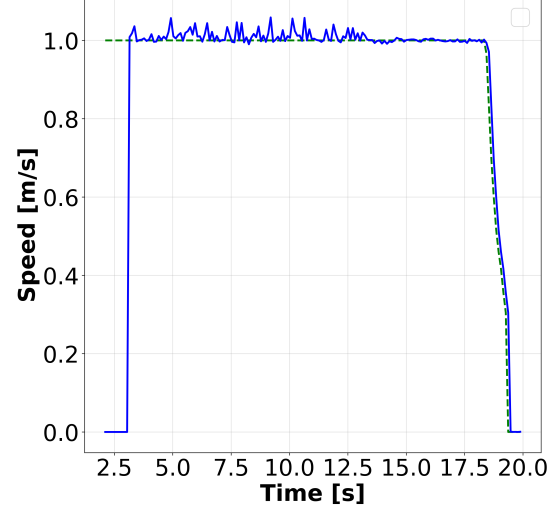
Linear Velocity Magnitude Over Time

Figure 4.11: Holonomic Scenario D1: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line indicates the commanded velocity magnitude.

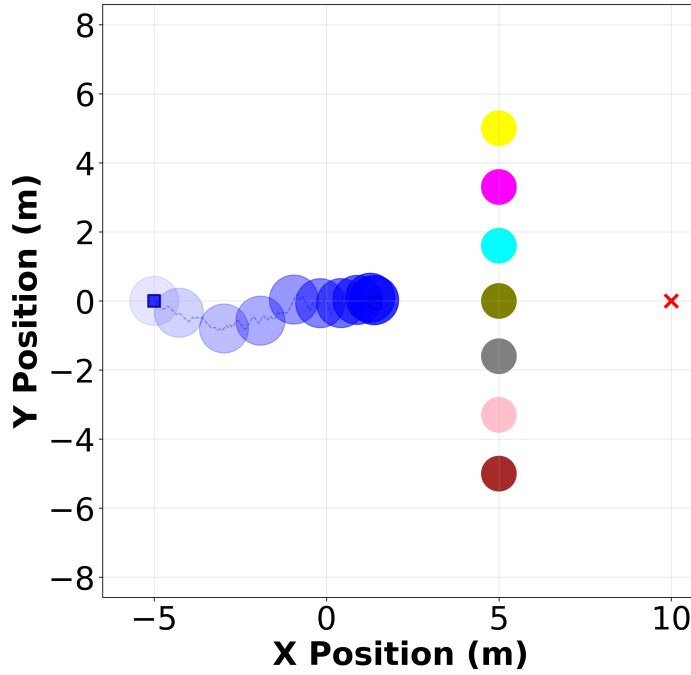
C4: Static Obstacle Wall

Table 4.7: Holonomic Scenario C4: Metrics Table

Metric	Value
Start Pos	(-5.00, 0.0, 0.0)
End Pos	(1.43, -0.07, 0.0)
Goal Pos	(10.0, 0.0, 0.0)
Path Length	-
Path Elongation	-
Min. Clearance	2.36 m
Reached Goal	False

Figure 4.12: Holonomic Scenario C4: XY robot trajectory

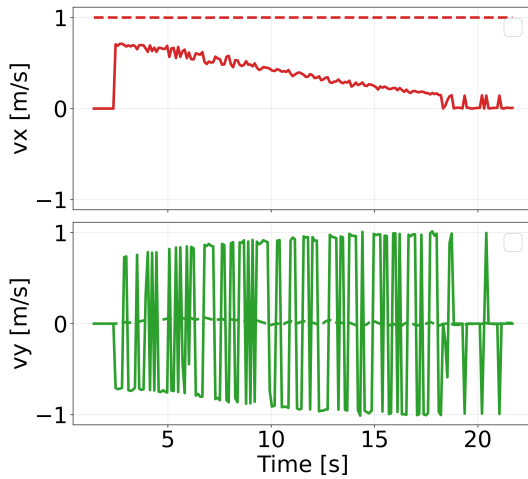
2D Linear Velocity Components Over Time

Figure 4.13: Holonomic Scenario C4: Velocity components over time. The dotted line indicates the desired velocity (desired_vel) and the continuous line indicates the commanded velocity (cmd_vel).

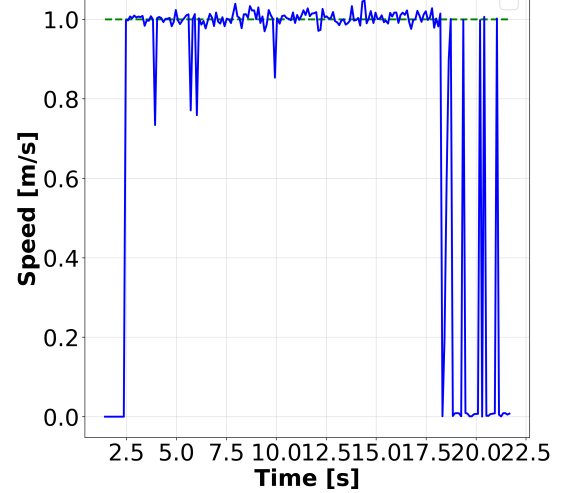
Linear Velocity Magnitude Over Time

Figure 4.14: Holonomic Scenario C4: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line indicates the commanded velocity magnitude.

2D Holonomic Vehicle Result Analysis

The behavior of the obstacle avoidance algorithm was presented through several metrics in four different scenarios. In scenario S0, the straight-line path between the agent's starting position and the goal is blocked by a single static obstacle. Figure 4.3 shows that the agent begins an early maneuver, going around the obstacle to the right (negative y-direction). The velocity components graph in Figure 4.4 shows that the x-component of the velocity remained almost constant during the test, while the y-component is initially negative to bypass the obstacle and then positive to return to the trajectory towards the goal. The velocity magnitude in Figure 4.5 is practically constant and matches the desired velocity set by the global planner. In this scenario, where the obstacle is aligned with the start and goal positions, there is an ambiguity: the cost of going around the obstacle on either side is the same. Although one side is chosen, some initial indecision by the algorithm can be noticed in the first seconds, visible in the v_y component graph.

Scenarios D0 and D1 have essentially the same configuration: a dynamic obstacle moving perpendicularly to the agent's trajectory, intersecting it at $x = 5$. The difference between these scenarios lies in the initial position of the obstacle, causing it to intersect the trajectory at $t = 13$ s in D0 and at $t = 15$ s in D1. These two scenarios aim to demonstrate the algorithm's ability to decide, based on its environment representation, whether to give way or overtake the obstacle, as shown in the trajectories in Figures 4.6 and 4.9, respectively. In both scenarios, the velocity magnitude remained constant and matched the desired speed (Figures 4.8 and 4.11). The velocity components graphs (Figures 4.7 and 4.10) show that, to give way in scenario D0, the y-velocity transitions from positive to negative, while to overtake in scenario D1, it goes from negative to positive.

Scenario C4 features a configuration of seven obstacles forming a 12-meter-wide wall perpendicular to the trajectory. When the algorithm receives a velocity command from the global planner directly toward this wall, it is unable to find a solution to reach the goal (Figure 4.12). The velocity components graph in Figure 4.13 shows that the x-velocity gradually decreases as the agent approaches the wall, and the y-velocity oscillates rapidly between positive and negative values, leaving the vehicle stuck. The behavior of slowing down as it nears the wall is caused by the Time Horizon Constraint, which allows the agent to approach obstacles with velocities that will not result in a collision within the defined time horizon threshold. The oscillating y-velocity behavior is explained by the algorithm's inability to find a solution around the wall. The velocity magnitude graph in Figure 4.14 shows an initially steady speed that becomes increasingly irregular. A deeper analysis of the algorithm's inability to navigate around this type of obstacle, along with a possible approach to address this problem, is presented in the Future Work Section 5.1.

4.2.4 2D Diff-Drive Vehicle Results

This section demonstrates the ability of the algorithm to be applied to a 2D differential drive surface vehicle. First, the scenarios used to test this simulated vehicle are presented. Then, for each scenario, the evaluation metrics are shown in the form of plots and tables. Finally, the results obtained for this vehicle are analyzed.

- **S0: Single Static Obstacle:** The trajectory is shown in Figure 4.15, the velocity components profile in Figure 4.16, the velocity magnitude profile in Figure 4.17, and the results table in Table 4.8.
- **D0: Single Dynamic Obstacle: Give Way:** The trajectory is shown in Figure 4.18, the velocity components profile in Figure 4.19, the velocity magnitude profile in Figure 4.20, and the results table in Table 4.9.
- **D1: Single Dynamic Obstacle: Overtake:** The trajectory is shown in Figure 4.21, the velocity components profile in Figure 4.22, the velocity magnitude profile in Figure 4.23, and the results table in Table 4.10.

S0: Basic Straight-Line Navigation

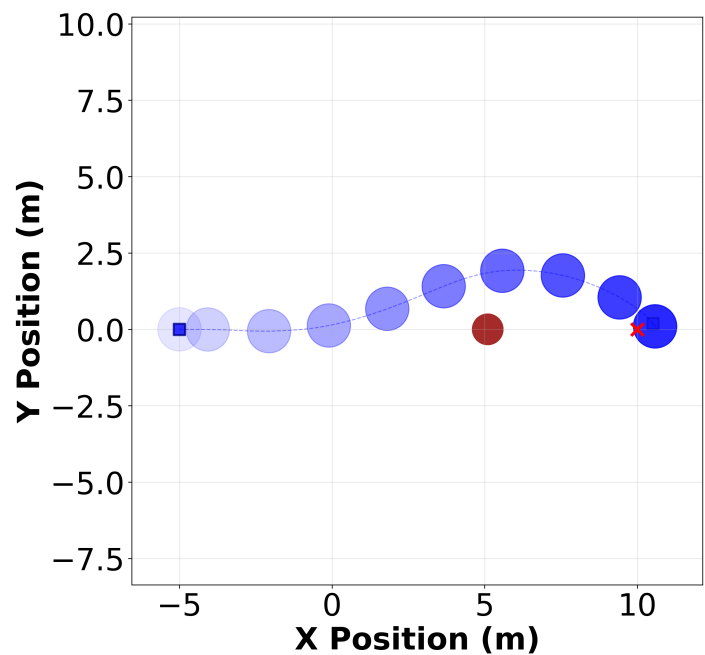


Table 4.8: Diff-Drive Scenario S0: Metrics Table

Metric	Value
Start Pos	(-5.00, 0.0, 0.0)
End Pos	(10.01, 0.67, 0.0)
Goal Pos	(10.00, 0.0, 0.0)
Path Length	15.58 m
Path Elongation	1.04×
Min. Clearance	0.60 m
Reached Goal	True

Figure 4.15: Diff-Drive Scenario S0: XY robot trajectory

Linear vs Angular Velocity (Differential Drive)

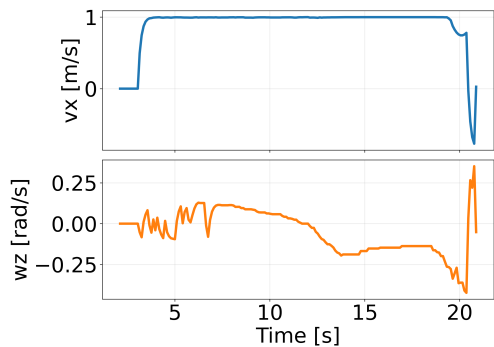


Figure 4.16: Diff-Drive Scenario S0: Velocity components over time (linear velocity v and angular velocity ω).

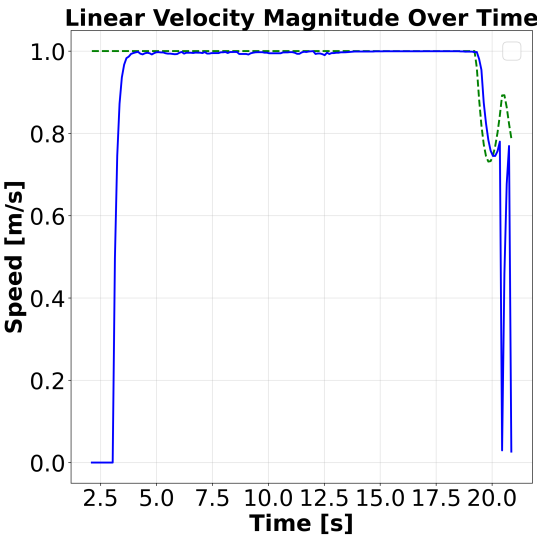


Figure 4.17: Diff-Drive Scenario S0: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line the commanded velocity magnitude.

D0: Diff-Drive Performance Test Scenario

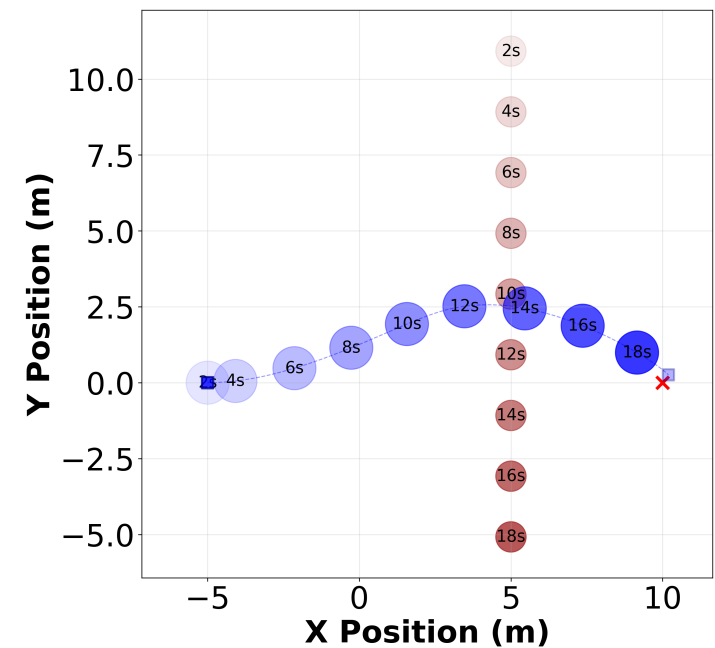


Table 4.9: Diff-Drive Scenario D0: Metrics Table

Metric	Value
Start Pos	(-5.0, 0.0, 0.0)
End Pos	(9.74, 0.61, 0.0)
Goal Pos	(10.0, 0.0, 0.0)
Path Length	15.58 m
Path Elongation	1.06x
Min. Clearance	0.99 m
Reached Goal	True

Figure 4.18: Diff-Drive Scenario D0: XY robot trajectory

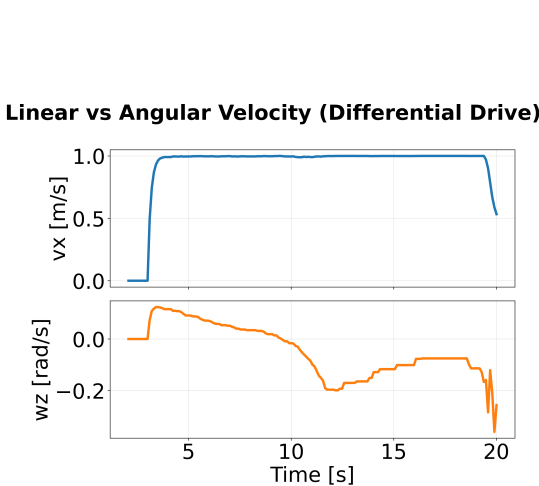


Figure 4.19: Diff-Drive Scenario D0: Velocity components over time (linear velocity v and angular velocity ω).

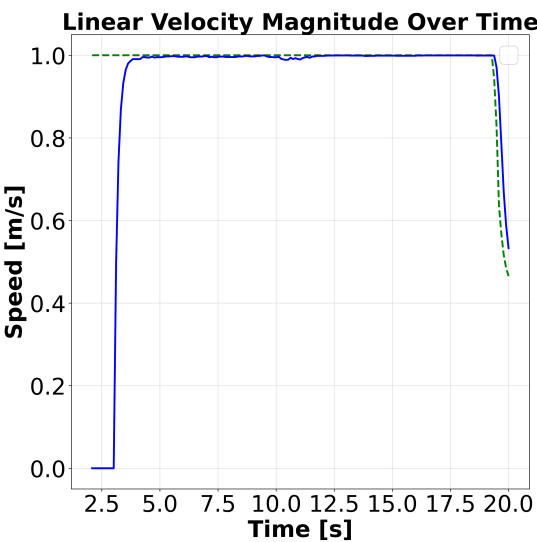


Figure 4.20: Diff-Drive Scenario D0: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line the commanded velocity magnitude.

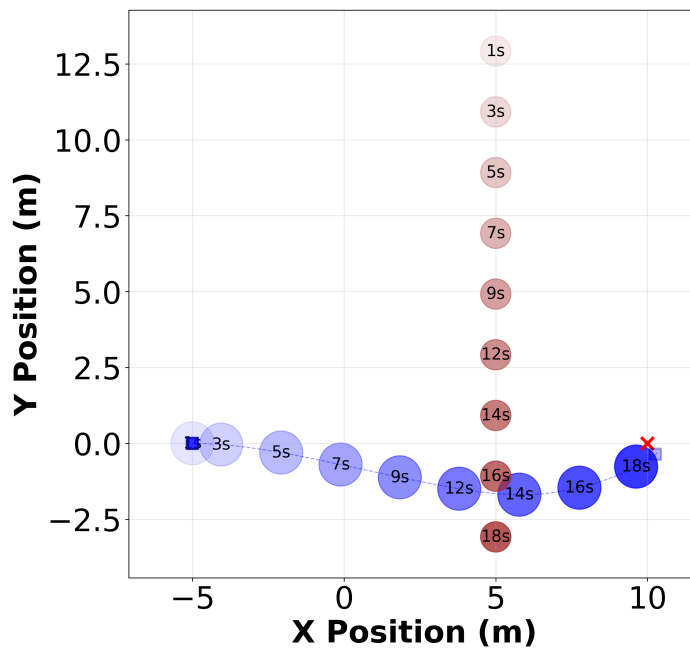
D1: Diff-Drive Performance Test Scenario

Table 4.10: Diff-Drive Scenario D1: Metrics Table

Metric	Value
Start Pos	(-5.0, 0.0, 0.0)
End Pos	(9.80, -0.67, 0.0)
Goal Pos	(10.0, 0.0, 0.0)
Path Length	15.13 m
Path Elongation	1.02×
Min. Clearance	1.15 m
Reached Goal	True

Figure 4.21: Diff-Drive Scenario D1: XY robot trajectory

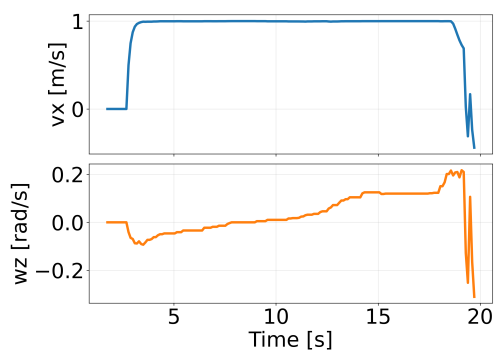
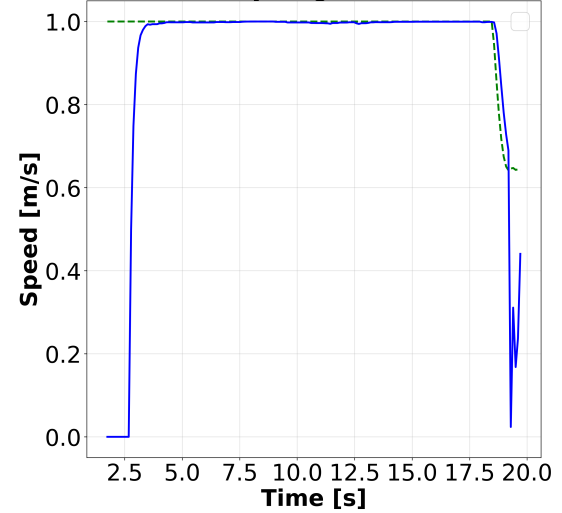
Linear vs Angular Velocity (Differential Drive)Figure 4.22: Diff-Drive Scenario D1: Command Velocity components over time (linear velocity v and angular velocity ω)**Linear Velocity Magnitude Over Time**

Figure 4.23: Diff-Drive Scenario D1: Velocity magnitude over time. The dotted line indicates the desired velocity magnitude, and the continuous line the commanded velocity magnitude.

2D Diff Drive Vehicle Result Analysis

In scenario S0, the straight-line path between the agent's starting position and the goal is interrupted by a single static obstacle. Figure 4.15 shows that the agent begins an early maneuver, going around the obstacle on the left side (positive y-direction). The velocity components graph in Figure 4.16 shows that the linear velocity remained almost constant during the test, and the angular velocity is initially positive to pass the obstacle, then negative to return to the trajectory toward the goal. The velocity magnitude in Figure 4.17 stays constant and matches the desired velocity set by the global planner. In this scenario, where the obstacle is aligned with the start and goal positions, there is ambiguity since the cost of passing on either side is the same. Similarly to the holonomic case, some initial indecision by the algorithm can be seen in the first seconds, visible in the angular velocity graph.

Scenarios D0 and D1 feature a dynamic obstacle moving perpendicularly to the agent's trajectory, intersecting it at $x = 5$ m, with the obstacle crossing the path at $t = 13$ s in D0 and at $t = 15$ s in D1. These scenarios aim to demonstrate the algorithm's ability to decide, based on its environment representation, whether to give way or overtake the obstacle, as shown in the trajectories in Figures 4.18 and 4.21, respectively. In both scenarios, the velocity magnitude stayed constant and matched the desired speed (Figures 4.20 and 4.23). The velocity components graphs (Figures 4.19 and 4.22) show how to give way in scenario D0, the angular velocity goes from positive to negative, while to overtake in scenario D1, it goes from negative to positive.

4.2.5 3D Multirotor Simulation Results

This section demonstrates the ability of the algorithm to be applied to a 3D aerial multirotor vehicle. First, the scenarios used to test this simulated vehicle are presented. Then, for each scenario, the evaluation metrics are shown in the form of figures, plots, or tables. Finally, an analysis of the results obtained for this vehicle is presented.

- **C4: Static Obstacle Line:** The 3D trajectory is shown in Figure 4.24, the XY projection in Figure 4.25, the XZ projection in Figure 4.26, the velocity magnitude profile in Figure 4.27, the velocity components profile in Figure 4.28, and the results table in Table 4.11.
- **D3: Dynamic Obstacle Pair:** The 3D trajectory is shown in Figure 4.29, the XY projection in Figure 4.30, the XZ projection in Figure 4.31, the velocity magnitude profile in Figure 4.32, the velocity components profile in Figure 4.33, and the results table in Table 4.12.

C4: 3D Static Obstacle Line

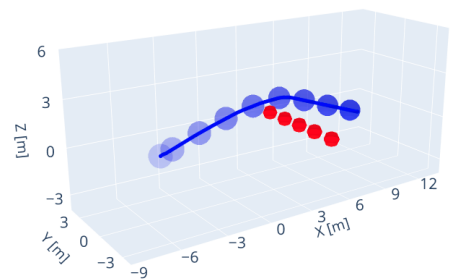


Figure 4.24: Multirotor Scenario C4: 3D robot trajectory

Table 4.11: Multirotor Scenario C4: Metrics Table

Metric	Value
Start Pos	(-5.00, 0.0, 0.0)
End Pos	(9.38, 0.02, 0.21)
Goal Pos	(10.0, 0.0, 0.0)
Path Length	14.95 m
Path Elongation	1.04×
Min Clearance	0.41 m
Reached Goal	True

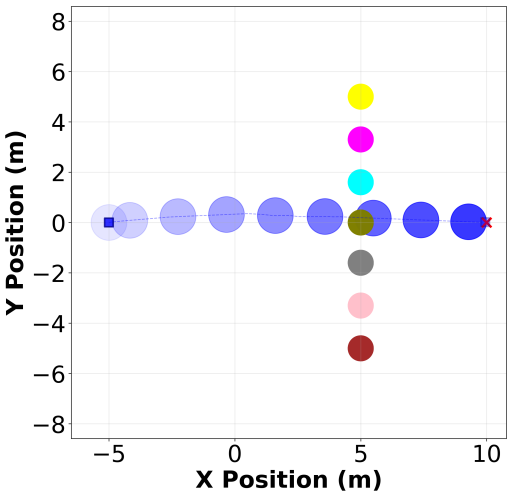


Figure 4.25: Multirotor Scenario C4: XY trajectory projection.

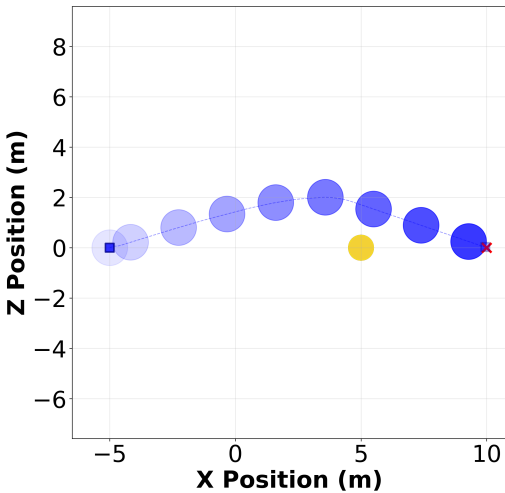


Figure 4.26: Multirotor Scenario C4: XZ trajectory projection.

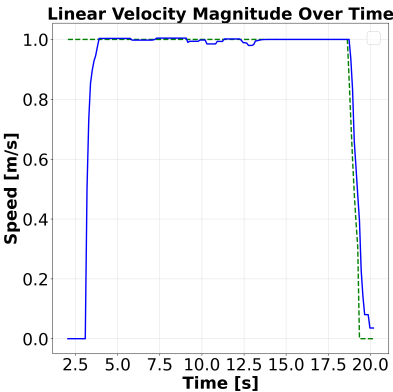


Figure 4.27: Multirotor Scenario C4: Velocity magnitude over time. Dotted line indicates desired velocity magnitude, and continuous line the commanded velocity magnitude.

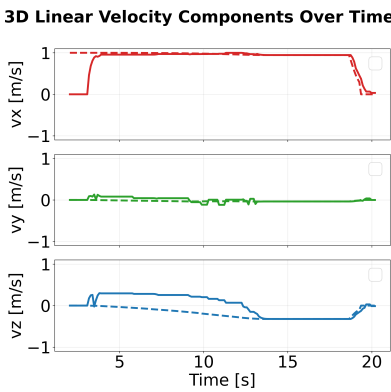


Figure 4.28: Multirotor Scenario C4: Velocity components over time. Dotted lines indicate desired velocities, and continuous lines the command velocities.

D3: Pair of Dynamic Obstacles

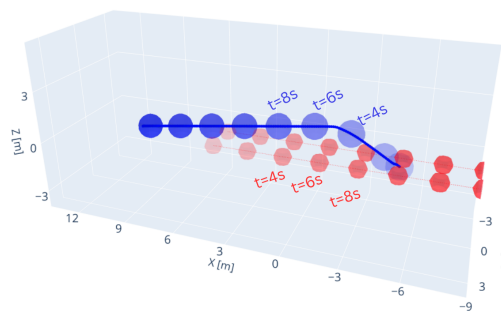


Figure 4.29: Multirotor Scenario D3: 3D robot trajectory

Table 4.12: Multirotor Scenario D3: Metrics Table

Metric	Value
Start Pos	(-5.00, 0.00, 0.00)
End Pos	(9.39, 0.00, 0.09)
Goal Pos	(10.00, 0.00, 0.00)
Path Length	14.96 m
Path Elongation	1.04×
Min Clearance	0.66 m
Reached Goal	True

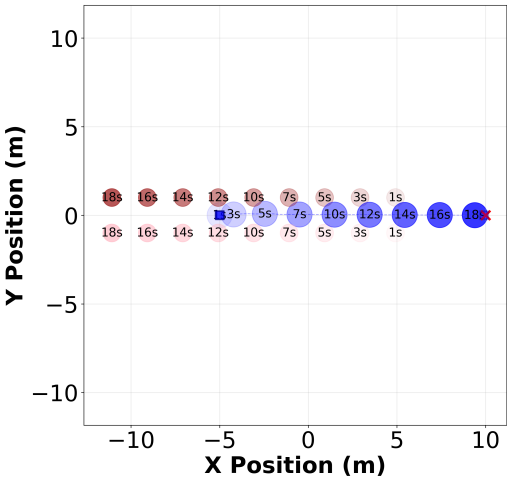


Figure 4.30: Multirotor Scenario D3: XY trajectory projection.

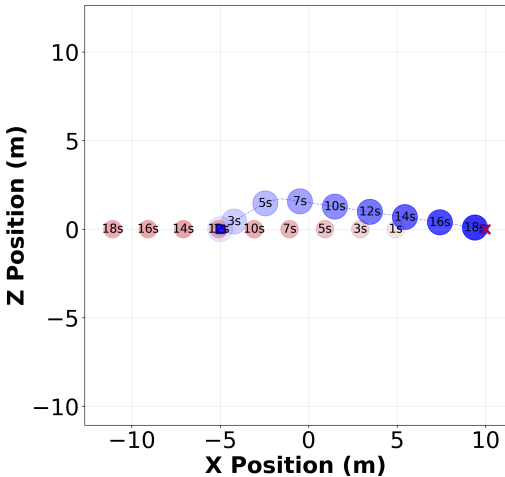


Figure 4.31: Multirotor Scenario D3: XZ trajectory projection.

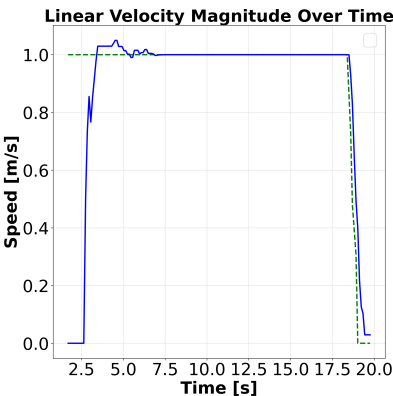


Figure 4.32: Multirotor Scenario D3: Velocity magnitude over time. Dotted line is desired magnitude, continuous line is command magnitude.

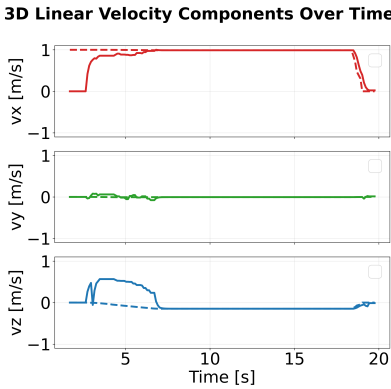


Figure 4.33: Multirotor Scenario D3: Velocity components over time. Dotted lines are desired; continuous lines are commands.

3D Multirotor Vehicle Results Analysis

In scenario C4, the line of obstacles that surface vehicles were unable to bypass becomes an easy problem for the multirotor, thanks to its extra degree of freedom. As shown in the 3D trajectory representation in Figure 4.24 and the XZ plane projection in Figure 4.26, the vehicle flew over the obstacle barrier, maintaining its XY trajectory as seen in the XY projection in Figure 4.25. The graph in Figure 4.27 shows that the velocity magnitude stayed constant, as did its x and y components, shown in Figure 4.28. The only component that changes relative to the desired speed is the z-velocity, which is initially positive to climb over the obstacle wall, and then negative to descend towards the goal.

Scenario D3 presents a pair of dynamic obstacles moving towards the agent. The multirotor again bypasses these obstacles by flying over them, as shown in the 3D trajectory in Figure 4.29 and the XZ projection in Figure 4.31, while maintaining its trajectory in XY as shown in Figure 4.30. The graphs of linear velocity magnitude in Figure 4.32 and the velocity components in Figure 4.33 show more interesting results: the y-component of the velocity is initially positive, but with a higher value than in scenario C4, and the linear velocity magnitude exceeds the desired magnitude. Additionally, the x-component of velocity also changes, initially staying below the desired speed. This behavior is caused by the imminent collision risk from the obstacles moving toward the agent: the z-component and the velocity magnitude increase to quickly move out of the collision path, and the x-component remains lower than desired, avoiding fast movement directly toward the obstacles.

4.2.6 Ideal Information vs Simulated LiDAR

The scenarios presented in the previous sections validated the algorithm using ground truth obstacle information. This section repeats the same tests (S0, D0, and D1) conducted in the differential drive simulation, with the same parameters, but now using the obstacle detection and tracking method described earlier, obtaining similar results presented below, and analysing the differences.

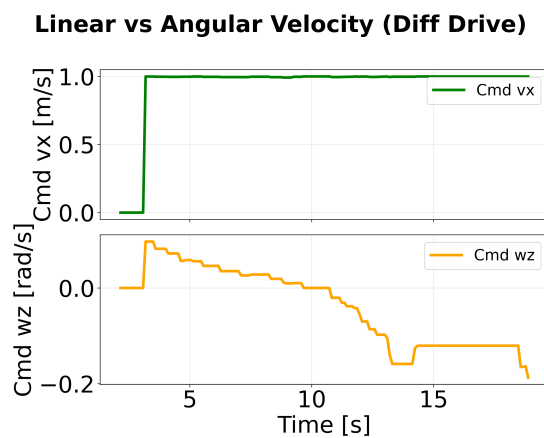


Figure 4.34: Detection-based Scenario S0: Velocity Components.

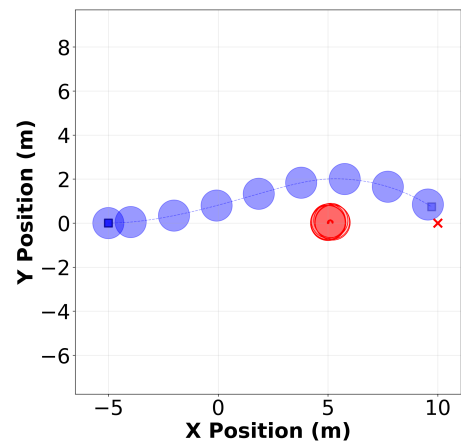


Figure 4.35: Detection-based Scenario S0: XY agent and obstacle trajectories.

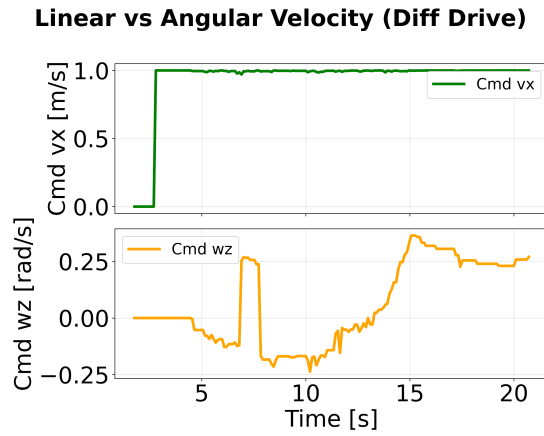


Figure 4.36: Detection-based Scenario D0: Velocity Components.

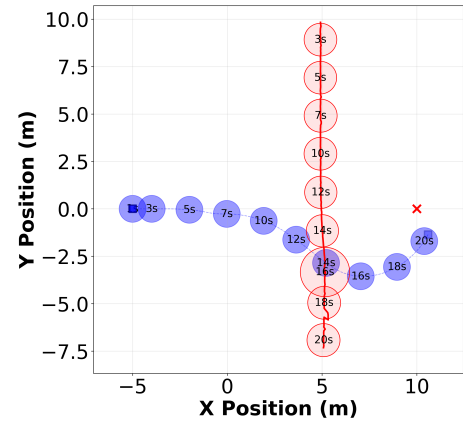


Figure 4.37: Detection-based Scenario D0: XY agent and obstacle trajectories.

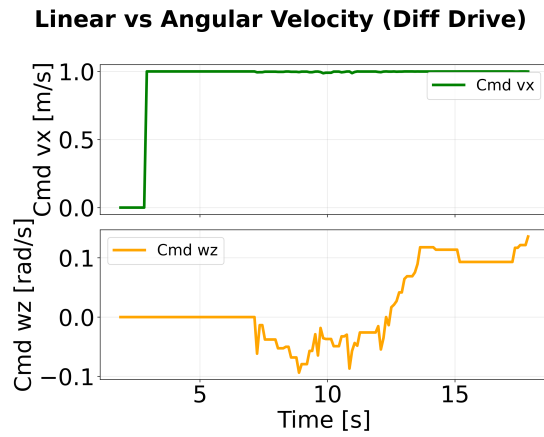


Figure 4.38: Detection-based Scenario D1: Velocity Components.

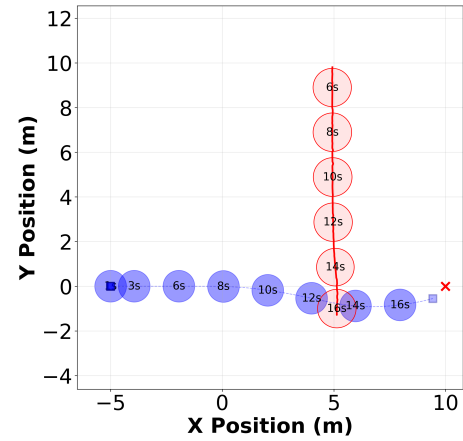


Figure 4.39: Detection-based Scenario D1: XY agent and obstacle trajectories.

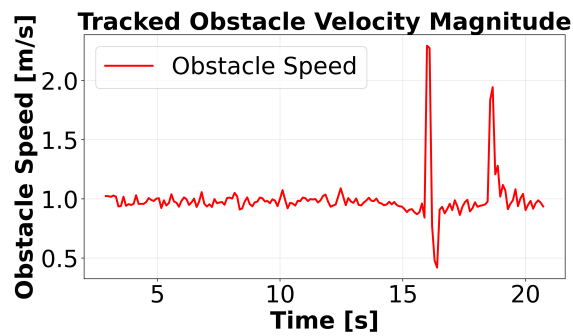


Figure 4.40: Detection-based Scenario D0: Estimated obstacle velocity magnitude over time.

The graphs in Figures 4.34, 4.36, and 4.38 show the velocity profiles, while Figures 4.35, 4.37, and 4.39 show the agent and obstacle trajectories for scenarios S0, D0, and D1, respectively. In the trajectory plots, the estimated obstacle radius is also represented by the size of the circles.

In scenario S0, the algorithm behaves essentially the same as with ground truth information, as confirmed by the similar velocity profile and trajectory. The estimated obstacle position and radius remain constant, and its estimated velocity is zero, so the algorithm's behavior does not change.

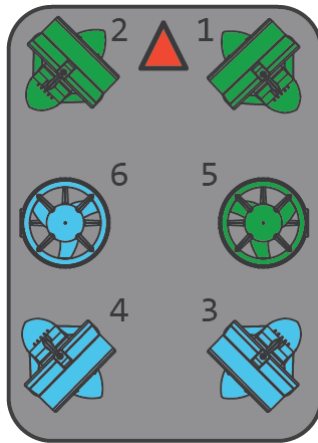
Scenarios D0 and D1 differ in initial positions. The D0 configuration encourages giving way, while D1's setup encourages overtaking. In scenario D1, the trajectory in Figure 4.39 and the velocity profile in Figure 4.38 show behavior similar to the ground truth counterpart. However, in scenario D0, with obstacle detection and tracking, an estimation error in obstacle velocity was enough to change the decision-making from giving way to trying to overtake. This difference in behavior is justified by the estimated obstacle velocity shown in Figure 4.40. The estimated velocity being slightly lower than the actual 1 m/s causes the algorithm to think the obstacle will take longer to intersect the agent's path, so it tries to pass first. Although it does not cause a collision, the angular velocity profile becomes more abrupt, with a noticeable peak representing indecision. In scenario D0, when the agent gets close to the obstacle, estimation errors in both velocity and radius are also observed.

4.3 NEST Simulation Environment

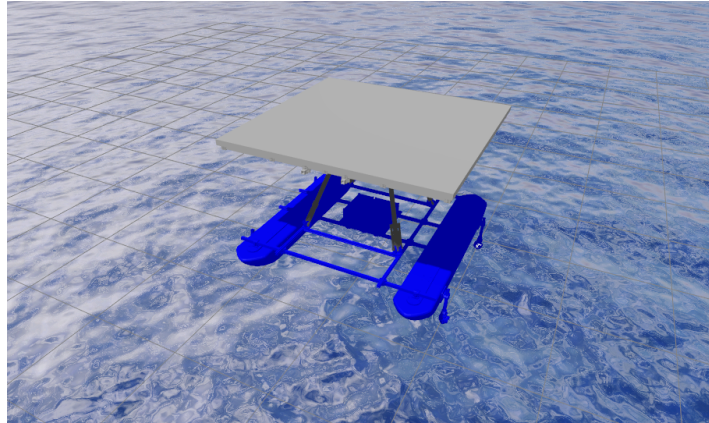
The second simulation phase used a more complex Gazebo simulation environment. In this simulator, a differential-drive surface vehicle was tested, with kinematic and dynamic constraints imposed by the simulation itself. The simulated environment accounts for water drag and buoyancy dynamics, increasing complexity and realism².

The USV used in this environment is a simulated version, represented in figure 4.41b of the real NEST vehicle [49][50] developed by the MOTUS team. This USV operates with a vectored thruster configuration, illustrated in Figure 4.41a, which enables instantaneous movement along the x and y axes, as well as rotation in yaw .

The vehicle was controlled in the simulation through a thrust topic, for which a control layer was integrated. This topic contains the desired force and moment in 3 degrees of freedom: F_x , F_y , and M_z , which are then mapped to the specific thruster configuration mentioned.



(a) Vectored thruster configuration (adapted from ArduSub documentation³)



(b) Complex Gazebo Environment with NEST uSV

Figure 4.41: Simulation environment and vehicle configuration: (a) Thruster vectored configuration; (b) Complex Gazebo simulation environment.

An overview of the pipeline implemented in this simulation is illustrated in Figure 4.42. It consists of the simple global planner providing the desired velocity towards the goal, the obstacle avoidance layer searching for a valid velocity, and a control layer that outputs actuator thrust commands for the simulated vehicle.

During this phase, the vehicle's and the obstacles' position and velocity data were again communicated directly from the simulation environment. The global planner used for these tests was the same one used in the Heterogeneous Vehicle Tests: a constant magnitude linear velocity vector pointing from the agent toward the goal is the input for the obstacle avoidance layer. Since the USV does not receive command velocities directly, velocity commands need to be converted to force

²https://github.com/srmainwaring/asv_wave_sim/, accessed on July 29, 2025

³<https://www.ardubot.com/quick-start/vehicle-frame.html>, accessed on June 19, 2025

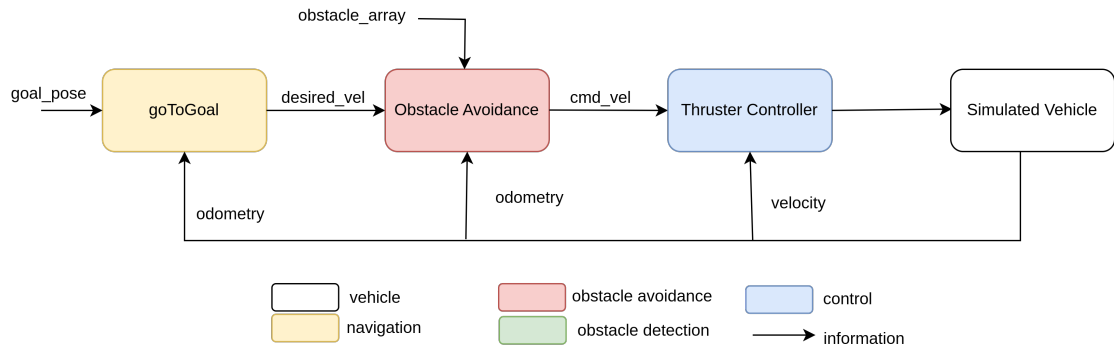


Figure 4.42: Pipeline Overview for the Complex Gazebo Simulation

(thrust), and therefore, a control layer was implemented. Proportional Integrative Derivative (PID) controllers were used to convert velocity commands into thrust for each degree of freedom of the USV: one controller from linear velocity to force in the X direction, one from linear velocity to force in the Y direction, and one from angular velocity in yaw to moment in yaw. The decoupled force and moment outputs from the three PID controllers are then remapped by the simulated vehicle's internal controller to the vectored thruster configuration described. An overview of this control layer is presented in Figure 4.43.

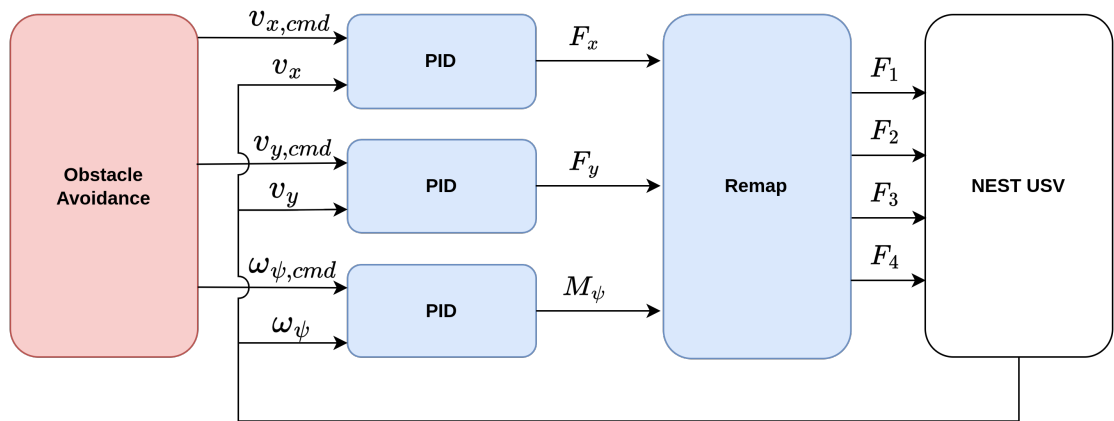


Figure 4.43: Control System Architecture for Velocity to Thrust Conversion

4.3.1 Defined Parameters and Testing Scenarios

The configuration parameters of the algorithm are detailed in Table 4.13 and Table 4.14, corresponding to the obstacle avoidance layer and the navigation layer, respectively. Additionally, the PID controller gains employed for low-level control are shown in Table 4.15. The test scenarios presented are listed in Table 4.16, where each scenario is identified using a short label, a preview image, and a short description.

Alongside these parameters, in all scenarios, the obstacles have a fixed protective zone of 0.5 m, and the goal is set to [10, 0, 0] m.

Table 4.13: Configuration parameters for the Obstacle Avoidance Layer for Simulated NEST.

Parameter [unit]	Simulated NEST
heading_weight	0.50
abs_weight	0.50
danger_weight	0.70
vehicle_radius [m]	2.15
delta_t [s]	1.0
num_samples [–]	3 000
radius_threshold [m]	0.2
time_to_collision_threshold [s]	120.0
a_x_max [m/s ²]	1.0
a_y_max [m/s ²]	1.0
a_z_max [m/s ²]	0.0
a_roll_max [rad/s ²]	0.0
a_pitch_max [rad/s ²]	0.0
a_yaw_max [rad/s ²]	1.0
v_x_max [m/s]	1.0
v_y_max [m/s]	1.0
v_z_max [m/s]	0.0
w_roll_max [rad/s]	0.0
w_pitch_max [rad/s]	0.0
w_yaw_max [rad/s]	1.0

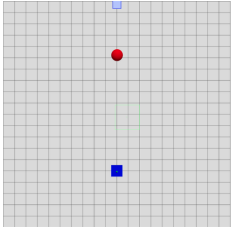
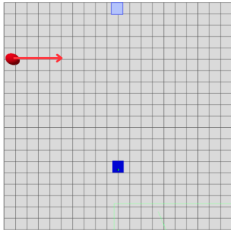
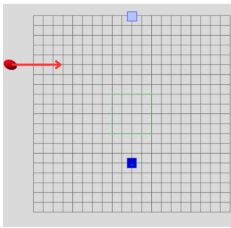
Table 4.14: Configuration parameters for the Navigation Layer for Simulated NEST.

Parameter [unit]	Simulated NEST
desired_constant velocity [m/s]	1.0

Table 4.15: PID controller gains for Simulated NEST.

Parameter	Simulated NEST
Linear PID	
K_p	60.0
K_i	10.0
K_d	0.5
Angular PID	
K_p	50.0
K_i	0.5
K_d	0.5

Table 4.16: 2D Scenarios with corresponding visual, label, and description.

Label	Figure	Description
s0		Single static obstacle blocking a direct path. Referred to in the literature as "agent alignment". Agent start $[-0, 0, 0]$. Obstacle start $[5, 0, 0]$.
d0		Single dynamic obstacle moving along the y-axis. The relative position favors giving way. Agent start $[-5, 0, 0]$. Obstacle start $[5, 11, 0]$. Obstacle velocity $[0, -1, 0]$.
d1		Single dynamic obstacle moving along the y-axis. The relative position favors overtaking. Agent start $[-5, 0, 0]$. Obstacle start $[5, 13, 0]$. Obstacle velocity $[0, -1, 0]$.

S0: NEST Simulation Performance Test Scenario

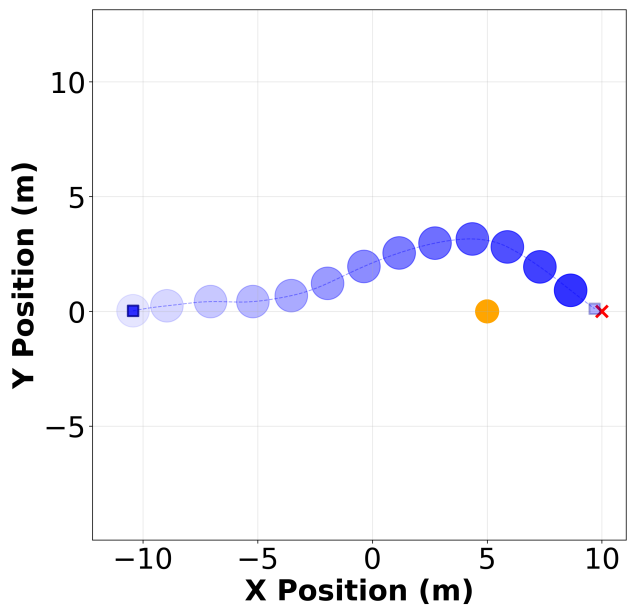


Table 4.17: NEST Scenario S0: Metrics Table

Metric	Value
Start Pos	(-10.43, 0.03, 0.29)
End Pos	(9.49, 0.26, 0.29)
Goal Pos	(10.00, 0.00, 0.00)
Path Length	21.27 m
Path Elongation	1.07×
Min Clearance	1.69 m
Reached Goal	True

Figure 4.44: NEST Scenario S0: XY robot trajectory

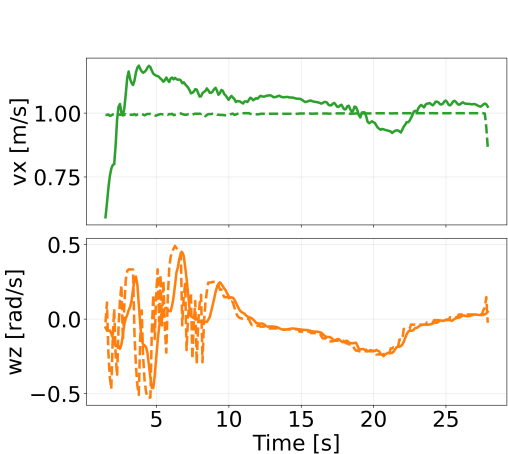


Figure 4.45: NEST Scenario S0: Commanded vs actual velocity components over time. Dotted lines are command velocities, and continuous lines are actual velocities.

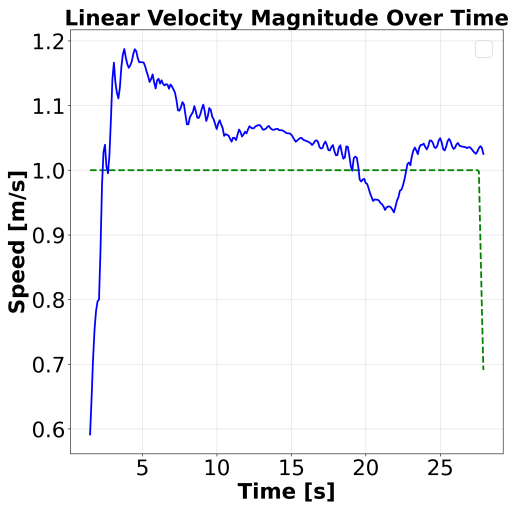


Figure 4.46: NEST Scenario S0: Velocity magnitude over time. Dotted line is desired magnitude, continuous line is actual magnitude.

D0: NEST Simulation Performance Test Scenario

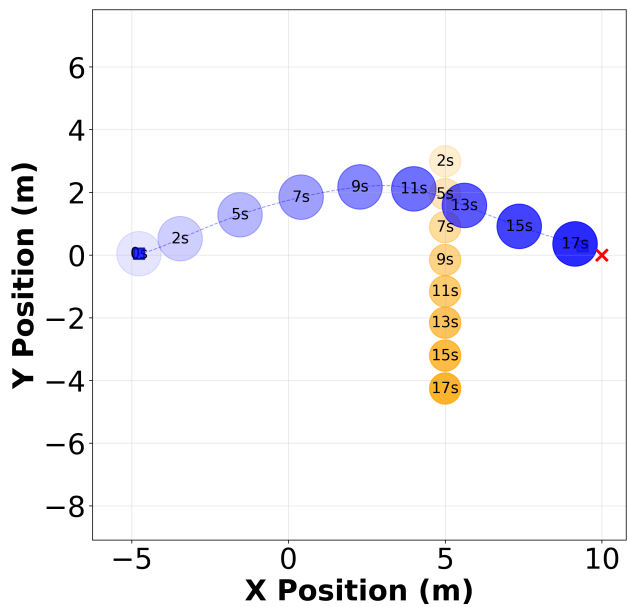


Table 4.18: NEST Scenario D0: Metrics Table

Metric	Value
Start Pos	(-4.77, 0.04, 0.29)
End Pos	(9.38, 0.28, 0.29)
Goal Pos	(10.00, 0.00, 0.00)
Path Length	14.81 m
Path Elongation	1.05×
Min Clearance	2.19 m
Reached Goal	True

Figure 4.47: NEST Scenario D0: XY robot trajectory

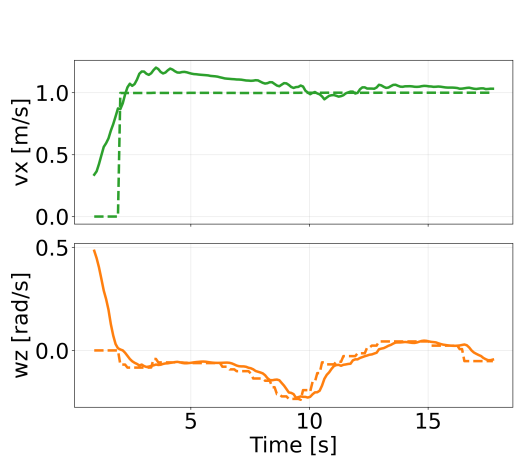


Figure 4.48: NEST Scenario D0: Com-manded vs actual velocity components over time. Dotted lines are command velocities, and continuous lines are actual velocities.

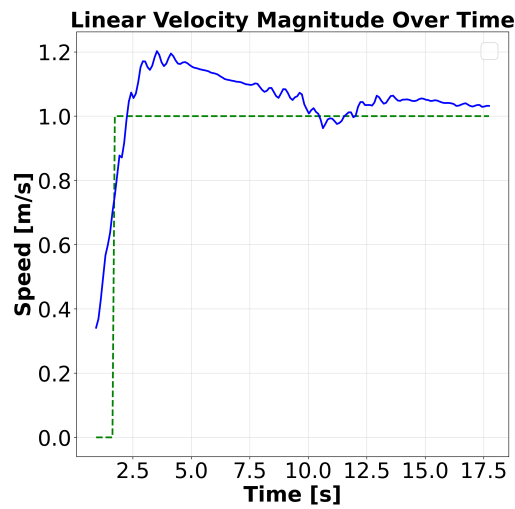


Figure 4.49: NEST Scenario D0: Velocity magnitude over time. Dotted line is desired magnitude, continuous line is actual magni-tude.

D1: NEST Simulation Performance Test Scenario

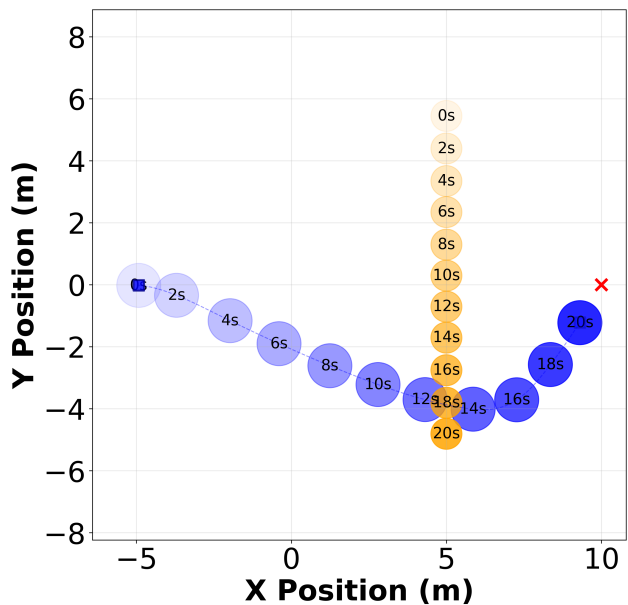


Table 4.19: NEST Scenario D1: Metrics Table

Metric	Value
Start Pos	(-4.92, -0.02, 0.29)
End Pos	(9.31, -1.23, 0.29)
Goal Pos	(10.00, 0.00, 0.00)
Path Length	16.26 m
Path Elongation	1.14x
Min Clearance	1.19 m
Reached Goal	True

Figure 4.50: NEST Scenario D1: XY robot trajectory

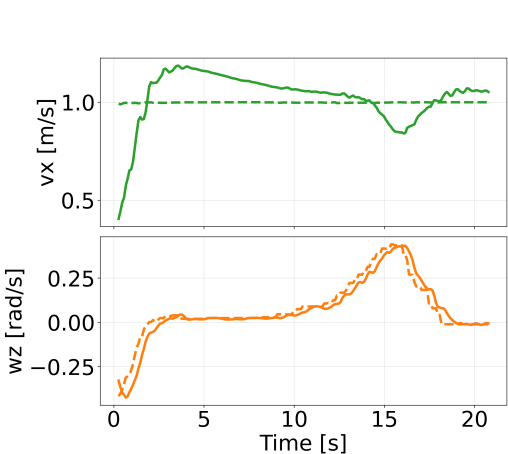


Figure 4.51: NEST Scenario D1: Commanded vs actual velocity components over time. Dotted lines are command velocities, and continuous lines are actual velocities.

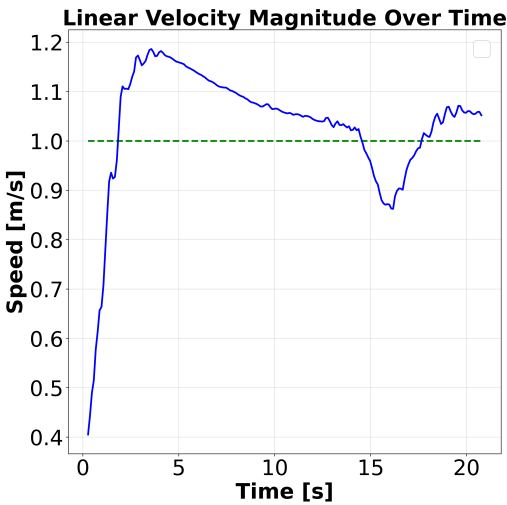


Figure 4.52: NEST Scenario D1: Velocity magnitude over time. The dotted line is the desired magnitude, and the continuous line is the actual magnitude.

4.3.2 Simulated USV Results Analysis

This simulation environment, with a vehicle controlled in a more realistic way, highlights some of the problems that occur with real vehicles. The control is not perfect, and a real vehicle is not capable of reaching exactly and instantaneously a desired velocity. Besides the control itself, the uncertainties and imperfections inherent to a real environment always create some error in following the desired control or path.

The implemented control layer, mainly due to the simplification made by using decoupled PID controllers instead of a more detailed control approach, causes errors in following the command velocity. This is mainly visible in the velocity components plots, where it is clear, especially in the v_x velocity, that there is a discrepancy between the commanded velocity and the velocity actually achieved by the vehicle.

The safety velocity margin around the velocity obstacles added by the Danger Cost is precisely designed to handle this type of problems. A velocity that is not inside a velocity obstacle but is very close to it is penalized, representing the risk that, if that velocity is commanded, the control followed might introduce some deviations and lead to a collision. The presented results show that even with errors in following the commanded velocity, the algorithm is robust enough to avoid collisions and reach the goal.

In scenario S0, a single static obstacle blocks the direct path of the agent to the goal. Figure 4.44 shows that the agent starts an early maneuver, going around the obstacle on the left side (positive y). The velocity components graph in Figure 4.45 shows that the commanded linear velocity was practically constant during the test, but with some error in the actual velocity. The angular velocity is initially positive while bypassing the obstacle, and then negative, returning the heading towards the goal. The velocity magnitude follows the profile of the x velocity, as seen in Figure 4.46. An oscillation in the angular velocity is observed again in the first seconds of this scenario, indicating indecision about which side to pass the obstacle.

Scenarios D0 and D1 consist of obstacles moving perpendicularly to the trajectory, intersecting it at $x = 5m$, with the obstacle intersecting the trajectory at $t = 13s$ in D0 and at $t = 15s$ in D1. These two scenarios show the algorithm's ability to decide, based on the environment representation, whether to give way or overtake the obstacle, as represented in the trajectories of Figures 4.47 and 4.50, respectively. The velocity components graphs (Figures 4.48 and 4.51) show that, to give way in scenario D0, the angular velocity goes from positive to negative, and to overtake in scenario D1, it goes from negative to positive. The velocity magnitude follows the profile of the x velocity (Figures 4.49 and 4.52).

4.4 Real World Testing

The third and final phase implemented the obstacle avoidance algorithm in a real-world environment using the NEST USV previously simulated, represented in Figure 4.53. The position and velocity information used for obstacle avoidance were provided by the localization layer, which relies on GPS and IMU data from the vehicle. The USV is equipped with an onboard LiDAR sensor, providing the sensor data used for obstacle detection.



Figure 4.53: NEST USV deployed at ATLANTIS Test Center in Viana do Castelo, Portugal

This phase included some preparatory integration tests in a laboratory setting, but was primarily carried out at the ATLANTIS Test Center in Viana do Castelo. This center enables testing in near-real scenarios, where navigation conditions, sensor data, and obstacle configurations closely resemble those of an actual deployment environment.

An overview of the pipeline used for these real-world testing is presented in Figure 4.54. The obstacle detection layer implemented in the real environment consists of the following steps: The point cloud obtained by the LiDAR is projected onto the xy -plane, assuming maximum and minimum z values of 30 cm. The resulting 2D information is used by the obstacle detection and tracking method, representing the environment as individual spherical obstacles with position and velocity. The obstacle avoidance layer uses this obstacle information, along with a fixed desired speed, to evaluate feasible velocity samples and estimate the best velocity output. This information is transmitted via a wireless connection established with the vehicle to provide the operator, on shore, with a representation of the vehicle's environment, the position and velocity of the obstacles, as well as the feasibility and cost of the samples in the vehicle's linear velocity space.

This setup, where the obstacle avoidance algorithm does not control the vehicle but provides the human operator with feedback on which controls will lead to a collision, is classified as assisted teleoperation [47]. In this setup, the operator maintains control while receiving continuous guidance from the system, acting accordingly. The feedback consists of visually displaying the

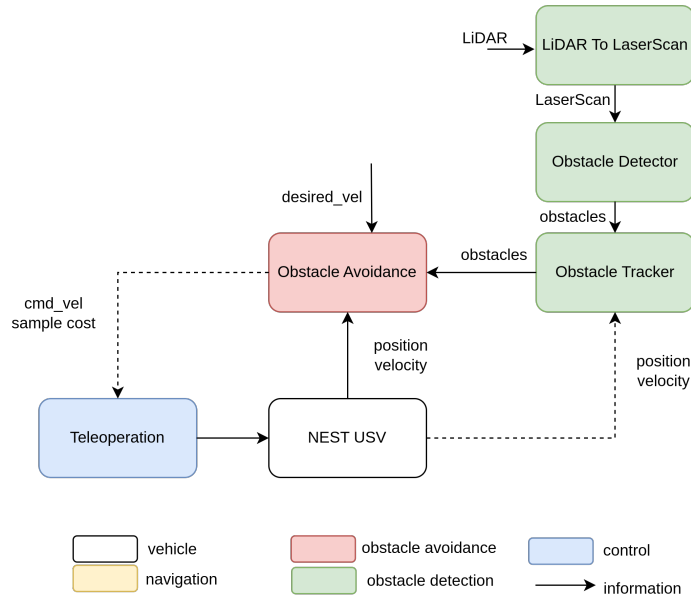


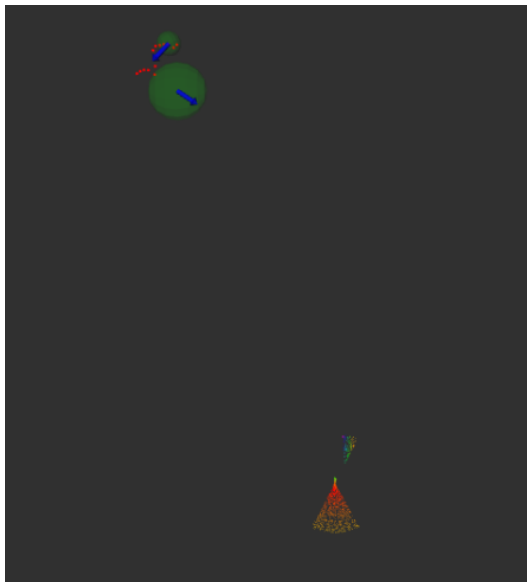
Figure 4.54: Assisted Teleoperation Pipeline implemented for real-world testing in ATLANTIS Test Centre

feasibility and associated cost across the vehicle’s velocity space, enabling the operator to make informed decisions in real time.

The goal of this real-world testing phase was to validate the methodology’s ability to accurately interpret the environment and evaluate a set of kinematically feasible velocities, which was successfully demonstrated across the diverse test scenarios. Given the inherent risks and significant costs associated with testing collision avoidance on real vehicles, particularly in marine environments, and considering the current low Technology Readiness Level (TRL) of the developed obstacle avoidance algorithm, further testing and integration with more advanced control and navigation layers are required to progress towards autonomous operation.

4.4.1 Real-World Results

The vehicle was deployed at sea and manually controlled until a safe distance from the shore was reached. For obstacle simulation, a small rowing boat measuring approximately 2 m by 1 m was used. The two vehicles were placed in different configurations of speed and position, creating small scenarios. The desired speed was fixed at 0, i.e., stationary, which became unfeasible when the dynamic obstacle moved towards the agent. Since in this real environment it is not possible to retrieve ground truth information about the positions and velocities of the agent and obstacles, the analysis of the results is based on a comparison between the scene geometry in real-world video recordings and the displayed environment representation: estimated obstacle positions and velocities, and sample evaluation and pruning. Figures 4.55 to 4.60 present pairs of images comparing the real and perceived environments.

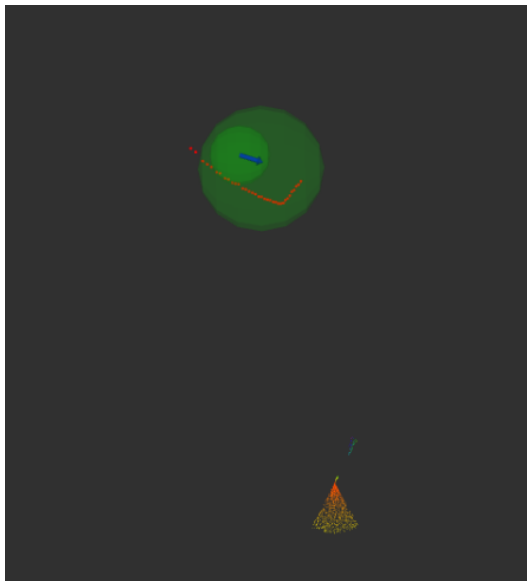


(a) Environment representation 1



(b) Reference image 1

Figure 4.55: Real-World Scenario 1 — Comparison between environment representation (a) and reference image (b).

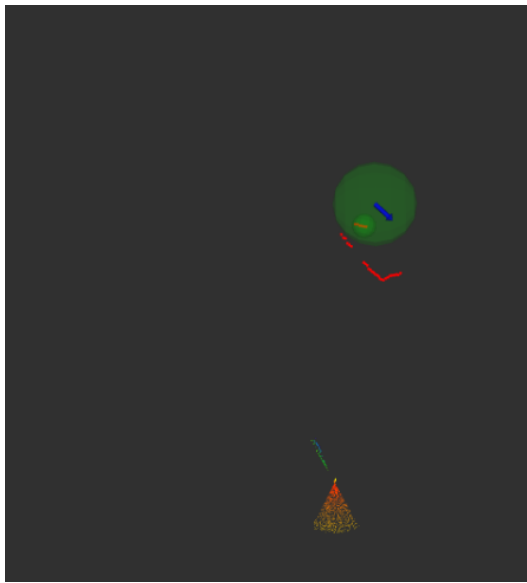


(a) Environment representation 2



(b) Reference image 2

Figure 4.56: Real-World Scenario 2 — Comparison between environment representation (a) and reference image (b).

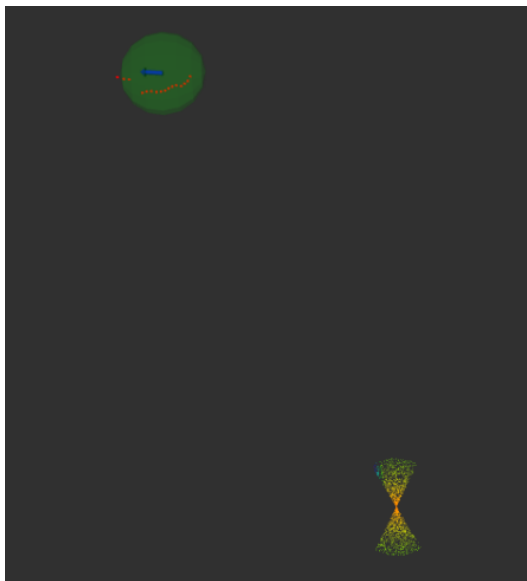


(a) Environment representation 3



(b) Reference image 3

Figure 4.57: Real-World Scenario 3 — Comparison between environment representation (a) and reference image (b).

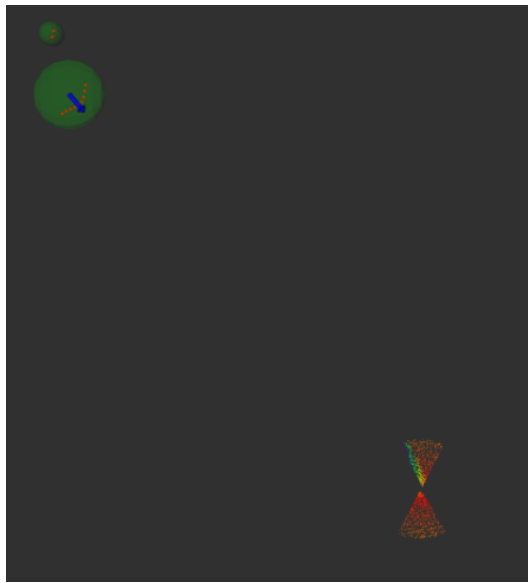


(a) Environment representation 4



(b) Reference image 4

Figure 4.58: Real-World Scenario 4 — Comparison between environment representation (a) and reference image (b).

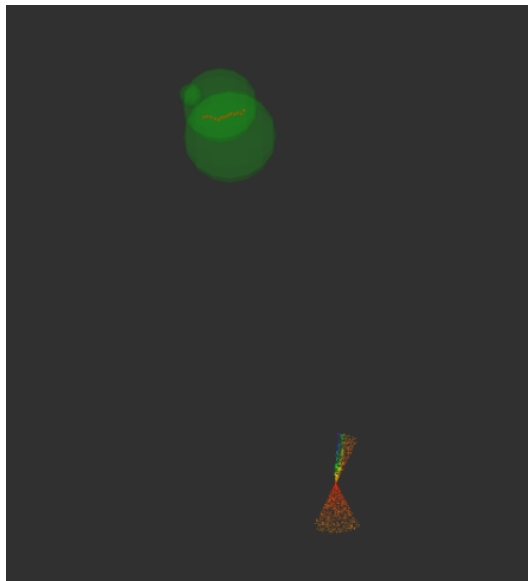


(a) Environment representation 5



(b) Reference image 5

Figure 4.59: Real-World Scenario 5 — Comparison between environment representation (a) and reference image (b).



(a) Environment representation 6



(b) Reference image 6

Figure 4.60: Real-World Scenario 6 — Comparison between environment representation (a) and reference image (b).

The comparison between the obtained representations and the reference images allows for validating the algorithm's ability to interpret the velocity space surrounding the vehicle. In each representation, it is possible to observe the generated velocity samples, where the lowest cost is represented in red and the highest cost in purple. The obstacle is illustrated in red by the laser scan points, and the obstacle's representation is shown in green, along with its estimated velocity depicted by a blue vector.

In Scenarios 1, 2, and 3 (Figures 4.55, 4.56, 4.57), the obstacle is detected crossing in front of the agent. In the representation of Scenario 1, the estimated velocity prunes the agent's forward velocities that would result in a collision, with the obstacle getting closer through Scenarios 2 and 3. Although the obstacle's tracked position and velocity remain consistent with its actual motion and position, in Scenario 3, the environment representation appears approximately one meter misaligned between the lidar points and the surface marked as an obstacle.

Scenario 4, presented in Figure 4.58, shows the obstacle moving perpendicularly in front of the agent and moving away, resulting in all velocities being feasible. In Scenario 5, although the obstacle is distant, it is navigating towards the agent's position; therefore, the velocity samples close to zero appear pruned, making staying in the current position invalid. Finally, in Scenario 6, Figure 4.60 shows a stationary vessel, which is modeled as a static obstacle, and no velocity vector is shown. Despite being static, velocities directed towards this obstacle are pruned, and those passing tangentially to it have a high cost.

In the presented representations, a smaller obstacle is often visible, sometimes even with a different velocity from that of the main vessel. This is likely due to a separate detection of the oar. Although modeling the vessel as one or two obstacles does not affect the obstacle avoidance algorithm itself, the obstacle tracker should be tuned for these environments, potentially incorporating semantic information from other sensors to model the boat's components as a single obstacle correctly.

Chapter 5

Conclusions

Unmanned robotic vehicles in dynamic and uncertain scenarios require robust obstacle avoidance approaches. This work proposed JAVOA, a robust obstacle avoidance algorithm for heterogeneous robotic platforms that builds upon the Velocity Obstacles approach in three-dimensional spaces. The algorithm generates a feasible velocity sample from 6DOF kinematic constraints, prunes the set according to the environment representation, and evaluates the remaining set according to a multicriteria heuristic function that accounts for goal progression and a risk-based metric.

The distinctive features of JAVOA are: the velocity space sampling framework generated from 6DOF kinematic constraints, making it easily adapted to address different vehicles; uncertainty-conscious scene representation, incorporating sensor uncertainty; velocity sample pruning and risk-based cost, assessing future collisions and risk of collision.

JAVOA was implemented and validated through simulation of a holonomic surface vehicle, a differential-drive surface vehicle, and a multirotor aerial vehicle, demonstrating its adaptability to different platforms. Furthermore, the algorithm's integration with other subsystems within a navigation stack was tested. This included LiDAR-based obstacle detection and tracking, as well as a thrust-based low-level control module. Experimental validation at the ATLANTIS Test Center confirmed the ability of the developed approach to perform in real maritime conditions, understanding dynamic environments and creating, pruning, and evaluating the risk of a set of maneuvers.

5.1 Future Work

The developed approach proved to be able to model different robotic vehicles; however, different environments need to be more extensively addressed. Real vehicles must follow strict navigation rules, which were not directly modeled in the obstacle avoidance layer.

Two types of problematic scenarios were identified. The first, exemplified in the obstacle wall scenario (C4), occurs when the agent is unable to reach the goal. The root cause of this issue lies in an indirect assumption of the algorithm: that the desired velocity remains constant while crossing an obstacle. This assumption is often invalid, especially when dealing with large obstacles

and goals located very close to the obstacle. Figure 5.1 shows an example where changes in the desired velocity lead to a deadlock. As the agent navigates to cross the obstacle on one side, the angle between the velocity required to pass on that side and the desired velocity increases, while the angle for the opposite side decreases. Consequently, every few iterations, the agent switches between attempting to pass on the left or right, never successfully crossing the obstacle.

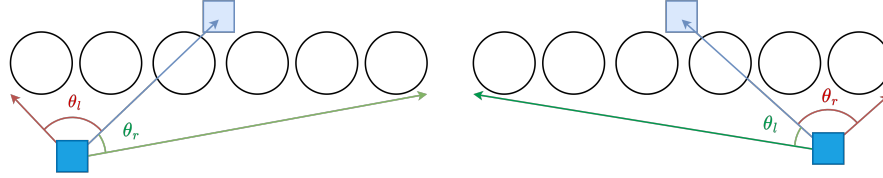


Figure 5.1: Goal behind but close to wall problem, θ_l and θ_r represent the angle of the maneuvers from the left and from the right. The lowest cost maneuver is marked in green

The second identified problem occurs when the goal is located just before an obstacle, as represented in Figure 5.2. With a very high time horizon constraint—meaning the algorithm considers collisions far into the future—the truncated area of the velocity obstacle becomes negligible. As a result, the algorithm oscillates, alternately choosing to pass the velocity obstacle on one side and then the other. The outcome is an “S-shaped” path toward the goal. In contrast, a shorter time horizon allows the cone truncation to enable the agent to navigate at a low velocity directly toward the goal. In such scenarios, a human-like behavior would simply ignore the distant obstacle.

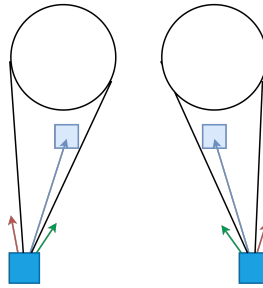


Figure 5.2: Goal before obstacle problem. Green arrow indicates the lowest cost maneuver

All problems described here could be mitigated by implementing a behavior layer in the navigation pipeline to enable more advanced decision-making. This layer could explicitly account for navigation rules, wall-like obstacles, and goals before obstacles. By perceiving these situations, the behavior layer could generate virtual obstacles or dummy goals [47], or dynamically adjust cost weights, guiding the obstacle avoidance layer to act appropriately.

The sampling approach also allows for some interesting improvements. The implemented algorithm used random uniform distributions across the control space, but other alternatives such as normal distributions centered at the current velocity, the desired velocity, or even a two-phase distribution, where a second step refines the sampling resolution around the best candidate from the previous iteration.

References

- [1] R. Perveen, N. Kishor, and S. R. Mohanty, “Off-shore wind farm development: Present status and challenges,” *Renewable and Sustainable Energy Reviews*, vol. 29, pp. 780–792, 2014, ISSN: 13640321. DOI: [10.1016/j.rser.2013.08.108](https://doi.org/10.1016/j.rser.2013.08.108).
- [2] Z. Ren, A. S. Verma, Y. Li, J. J. Teuwen, and Z. Jiang, “Offshore wind turbine operations and maintenance: A state-of-the-art review,” *Renewable and Sustainable Energy Reviews*, vol. 144, p. 110886, 2021, ISSN: 1364-0321. DOI: <https://doi.org/10.1016/j.rser.2021.110886>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364032121001805>.
- [3] E. P. Soares-Ramos, L. de Oliveira-Assis, R. Sarrias-Mena, and L. M. Fernández-Ramírez, “Current status and future trends of offshore wind power in europe,” *Energy*, vol. 202, p. 117787, 2020.
- [4] G. G. O. Wind and E. I. London, “G+ global offshore wind health & safety organisation - 2023 incident data report,” G+ Global Offshore Wind Health & Safety Organisation, Tech. Rep., 2023, Accessed: 2024-12-12. [Online]. Available: <https://www.gplusoffshorewind.com/?a=1538335>.
- [5] P. Fiorini and Z. Shiller, “Motion planning in dynamic environments using the relative velocity paradigm,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, IEEE, 1993, pp. 560–565.
- [6] I. C. A. Organization, *Annex 2 to the convention on international civil aviation: Rules of the air*, Tenth Edition, Accessed: 2024-12-22, 2005. [Online]. Available: https://www.icao.int/Meetings/anconf12/Document%20Archive/an02_cons%5B1%5D.pdf.
- [7] S. D. Pendleton, H. Andersen, X. Du, *et al.*, “Perception, planning, control, and coordination for autonomous vehicles,” *Machines*, vol. 5, Mar. 2017, ISSN: 20751702. DOI: [10.3390/machines5010006](https://doi.org/10.3390/machines5010006).
- [8] J. Betz, *Introduction: Lecture & mobile robot systems*, [Lecture slides], Autonomous Vehicles: Motion Planning & Decision Making, Professorship Autonomous Vehicle Systems (AVS), Technische Universität München, 2024.
- [9] D. F. B. Campos, “Multi-domain contextual awareness using unmanned surface vehicles for offshore wind farms inspection,” Ph.D. dissertation, FEUP, UP, 2024.
- [10] R. Mascaro and M. Chli, “Scene representations for robotic spatial perception,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 8, 2025.
- [11] D. Campos, A. Matos, and A. M. Pinto, “Semantic instance completion for offshore wind farm foundations,” *ESS Open Archive eprints*, vol. 878, p. 87821527, 2024.

- [12] P. N. Leite and A. M. Pinto, “Fusing heterogeneous tri-dimensional information for reconstructing submerged structures in harsh sub-sea environments,” *Information Fusion*, vol. 103, p. 102 126, 2024.
- [13] D. F. Campos, A. Matos, and A. M. Pinto, “Multi-domain inspection of offshore wind farms using an autonomous surface vehicle,” *SN Applied Sciences*, vol. 3, no. 4, p. 455, 2021.
- [14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3d mapping framework based on octrees,” *Autonomous robots*, vol. 34, pp. 189–206, 2013.
- [15] D. Maier, A. Hornung, and M. Bennewitz, “Real-time navigation in 3d environments based on depth camera data,” in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, IEEE, 2012, pp. 692–697.
- [16] P. Y. Leong and N. S. Ahmad, “Lidar-based obstacle avoidance with autonomous vehicles: A comprehensive review,” *IEEE Access*, 2024.
- [17] M. Przybyła, “Detection and tracking of 2d geometric obstacles from lrf data,” in *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, 2017, pp. 135–141. DOI: [10.1109/RoMoCo.2017.8003904](https://doi.org/10.1109/RoMoCo.2017.8003904).
- [18] D. F. Duarte, M. I. Pereira, and A. M. Pinto, “Multiple vessel detection in harsh maritime environments,” *Marine Technology Society Journal*, vol. 56, no. 5, pp. 58–67, 2022.
- [19] F. Vesentini, R. Muradore, and P. Fiorini, “A survey on velocity obstacle paradigm,” *Robotics and Autonomous Systems*, vol. 174, p. 104 645, 2024.
- [20] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.
- [21] L. Palmieri, “A behavioural approach to obstacle avoidance for mobile manipulators based on distributed sensing,” *arXiv preprint arXiv:1502.02465*, 2015.
- [22] J.-F. Duhé, S. Victor, and P. Melchior, “Contributions on artificial potential field method for effective obstacle avoidance,” *Fractional Calculus and Applied Analysis*, vol. 24, pp. 421–446, 2021.
- [23] J. Borenstein, Y. Koren, *et al.*, “The vector field histogram-fast obstacle avoidance for mobile robots,” *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.
- [24] S. Ahmed, B. Qiu, F. Ahmad, C. W. Kong, and H. Xin, “A state-of-the-art analysis of obstacle avoidance methods from the perspective of an agricultural sprayer uav’s operation scenario,” *Agronomy*, vol. 11, 6 2021, ISSN: 20734395. DOI: [10.3390/agronomy11061069](https://doi.org/10.3390/agronomy11061069).
- [25] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.
- [26] M. Missura and M. Bennewitz, “Predictive collision avoidance for the dynamic window approach,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8620–8626.
- [27] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.
- [28] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, “Path planning for autonomous vehicles using model predictive control,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 174–179.

- [29] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, IEEE, 1993, pp. 802–807.
- [30] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, VDE, 2012, pp. 1–6.
- [31] L. A. Nguyen, T. D. Pham, T. D. Ngo, and X. T. Truong, "A proactive trajectory planning algorithm for autonomous mobile robots in dynamic social environments," in *2020 17th International Conference on Ubiquitous Robots (UR)*, IEEE, 2020, pp. 309–314.
- [32] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.
- [33] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3513–3520, 2019.
- [34] K. Almazrouei, I. Kamel, and T. Rabie, "Dynamic obstacle avoidance and path planning through reinforcement learning," *Applied Sciences*, vol. 13, no. 14, p. 8174, 2023.
- [35] M. I. Pereira and A. M. Pinto, "Reinforcement learning based robot navigation using illegal actions for autonomous docking of surface vehicles in unknown environments," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108 506, 2024.
- [36] F. S. Neves, L. M. Branco, M. I. Pereira, R. M. Claro, and A. M. Pinto, "A multimodal learning-based approach for autonomous landing of uav," in *2024 20th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, IEEE, 2024, pp. 1–8.
- [37] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.
- [38] P. Fiorini and Z. Shiller, "Time optimal trajectory planning in dynamic environments," in *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, vol. 2, 1996, pp. 1553–1558.
- [39] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*, Ieee, 2008, pp. 1928–1935.
- [40] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research: The 14th International Symposium ISRR*, Springer, 2011, pp. 3–19.
- [41] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 5573–5578.
- [42] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.
- [43] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems: The 10th international symposium*, Springer, 2013, pp. 203–216.
- [44] B. Kluge and E. Prassler, "Recursive probabilistic velocity obstacles for reflective navigation," in *Field and Service Robotics: Recent Advances in Reserch and Applications*, Springer, 2006, pp. 71–79.

- [45] D. F. Campos, A. Matos, and A. M. Pinto, “An adaptive velocity obstacle avoidance algorithm for autonomous surface vehicles,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 8089–8096.
- [46] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-international conference on neural networks*, iee, vol. 4, 1995, pp. 1942–1948.
- [47] T. Battisti and R. Muradore, “A velocity obstacles approach for autonomous landing and teleoperated robots,” *Autonomous Robots*, 2020, ISSN: 15737527. DOI: [10.1007/s10514-019-09887-8](https://doi.org/10.1007/s10514-019-09887-8).
- [48] Y. I. Jenie, E.-J. van Kampen, C. C. de Visser, J. Ellerbroek, and J. M. Hoekstra, “Three-dimensional velocity obstacle method for uncoordinated avoidance maneuvers of unmanned aerial vehicles,” *Journal of Guidance, Control, and Dynamics*, vol. 39, pp. 2312–2323, Oct. 2016, ISSN: 0731-5090. DOI: [10.2514/1.G001715](https://doi.org/10.2514/1.G001715).
- [49] J. Martins, P. Pereira, R. Campilho, and A. Pinto, “Wave-motion compensation for usv–uav cooperation: A model predictive controller approach,” in *2024 20th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, IEEE, 2024, pp. 1–8.
- [50] P. Pereira, R. Campilho, and A. Pinto, “An inverse kinematics approach for the analysis and active control of a four-upr motion-compensated platform for uav–asv cooperation,” *Machines*, vol. 11, no. 4, p. 478, 2023.