La borsa di dottorato è stata cofinanziata con risorse del Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005)
Fondo Sociale Europeo, Azione I.1 “Dottorati Innovativi con caratterizzazione Industriale”

UNIVERSITÀ DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

Information and Communication Technologies

Con il contributo del

Ministero dell’Università e della Ricerca

CICLO

XXXVII

Cooperative Distributed Command Governor Strategies for Surface Marine Vehicles

Settore Scientifico Disciplinare ING-INF/04

Coordinatore: Ch.mo Prof. Giancarlo Fortino

Firma _____ GIANCARLO FORTINO 23.02.2025 14:13:06 GMT+02:00

Supervisore/Tutor: Ch.mo Prof. Francesco Tedesco

Firma__ FRANCESCO TEDESCO 11.02.2025 17:35:09 GMT+01:00 _____

Dottorando: Dott. Ayman El Qemmah

Firma _____ Ayman El Qemmah 11.02.2025 18:30:10 GMT+02:00

Department of Computer Science, Modeling, Electronics and Systems Engineering (DIMES)

Ph.D. Course in

Information and Communication Technologies

Cycle

XXXVII

Cooperative Distributed Command Governor Strategies for Surface Marine Vehicles

by

Ayman El Qemmah

A dissertation submitted in partial fulfillment
of the requirements for the degree of
"Dottore di Ricerca"
(Doctor of Philosophy)
Scientific Disciplinary Sector ING-INF/04

Ph.D. Program Coordinator
Prof. Giancarlo Fortino

Ph.D. Supervisor
Prof. Francesco Tedesco

To my family, my mother Naima, my father Mostafa,
my sisters Houda, Aya and Marwa,
and my nephew Jamil

Acknowledgements

Praise be to God, the Most Gracious and the Most Merciful. Without His blessings and guidance my accomplishments would never have been possible.

I would like to acknowledge many people who helped me and made this journey enjoyable. First, I would like to thank my thesis advisor, Professor Francesco Tedesco, for providing me with tireless support while giving me considerable independence in my research. His great skills and experience, remarkable patience and full supporting attitude, make him an excellent advisor and make me deeply grateful and honored for being able to be his student. I am grateful to Professor Alessandro Casavola for giving me the opportunity to be a part of the research group, for providing me with both constant constructive feedback and financial support, and for giving me the latitude to research what I like most. Their honesty, helpful suggestions, constructive criticisms, and stimulating discussions have greatly influenced my thinking, and many of the presented results are the product of joint work.

I am also grateful to Professor Bruno Sinopoli, for believing in me, for his advice, sincere encouragement and time. His hospitable, amicable, humble, pleasurable nature, combined with his intuitive and stimulating personality, make him a great inspiration to me.

I am also grateful to Franco Torchiaro, with whom I worked the most and shared most of my experiences dating back to the first year of college, for his friendship, never-ending help, assistance and patience. I am thankful also to Gianni Cario for his availability and encouragement, he always had time to answer my questions and to give various suggestions related to my research. I would like to thank Marco Lupia for his efficient and skillful assistance, and for laying down the foundations of the prototype of the marine surface vehicle on which I worked. I would like also to thank Gianfranco Gagliardi, Francesco Cicchello and all the team of Applicon for helping me so many times in their lab.

I would like also to thank Bahram Yaghooti, Minh Vu, Yunshen Huang, Carmel Fisko, Giacomo Vedovati, Haoyu Yin, Jonathan Gornet and all my other WashU alumni friends and staff, too many to be listed here, who have made my stay in the Saint Louis such a pleasant experience.

I sincerely feel I cannot be grateful enough for being able to spend these three years the way I did with all of you in an environment few are able to experience.

Last, but certainly not the least, I would like to acknowledge the support of my parents and my sisters, who have always motivated, supported and believed in me. This thesis is partly theirs too.

Abstract

The deployment of swarming surface marine vehicle offers numerous advantages in aquatic environments, including operational flexibility, task and environmental adaptability, resilience, and scalability. In this context, coordination together with the satisfaction of actuator, state, safety constraints becomes essential to enhance autonomy. Meeting this demand is challenging: while advanced techniques have been proposed, many of them are complex and/or necessitate frequent replanning, which can negatively impact performance and computational intensity. Therefore, readily implementable and low computationally demanding strategies are required.

This dissertation presents: i) a class of novel distributed supervision strategies applied to marine surface multi-vehicles systems to address coordination problems, ii) a novel supervision scheme that bypasses the need for explicit modeling and iii) experimental validation of distributed supervision strategies. By resorting to Command Governor (CG) ideas, predictive supervision algorithms are employed to effectively deal with the above mentioned supervision problems.

While traditional distributed CG strategies have a non-cooperative approach, in this dissertation two distributed schemes that promote cooperation among agents are introduced. Initially, considering the computationally demanding nature of cooperative approaches, the main idea consists of leveraging a non-cooperative non-iterative solution, and enhancing it to induce cooperation between agents, thus preserving the low computational burden. For this reason, the resulting scheme is referred to as "Cooperation-inducing." Subsequently, inspired by a distributed optimization algorithm from literature, a more general distributed approach that allows agents to compute a near-optimal centralized solution, is fully described and analyzed. In this respect, where the resulting scheme is referred to as "Cooperative," agents exchange only auxiliary variables, thus ensuring privacy preservation. The development of both schemes enables the selection of the most suitable approach based on specific application requirements.

Concurrently, to address the challenge of solving the CG design problem when an explicit model representation is not available, an approach referred to as "Data-Driven," is presented. This alternative to the model-based approach avoids the need for time-consuming modeling, identification, and validation phases and, unlike other existing approaches, does not require a new data collection amidst controller modifications.

Finally, with the aim of evaluating computational performances and implementability of CG-based distributed strategies on marine surface vehicles, several experimental results in various environments are presented. These results are obtained using prototypes of highly maneuverable marine surface vehicles, following a comprehensive design, identification, validation, control, and deployment process.

Abstract (Italiano)

Lo schieramento di sciami di veicoli di superficie offre numerosi vantaggi negli ambienti acquatici, tra cui flessibilità operativa, adattabilità alle missioni e all'ambiente, resilienza e scalabilità. In questo contesto, per migliorare l'autonomia, diventa essenziale introdurre vincoli di coordinamento e altri vincoli operativi (es. stato, ingresso). Riuscire in questo intento è impegnativo: anche le tecniche più avanzate spesso sono complesse e/o richiedono frequenti riprogrammazioni, il che può influire negativamente sulle prestazioni e sul carico computazionale. Pertanto, sono necessarie strategie facilmente implementabili e poco impegnative dal punto di vista computazionale.

Questa tesi presenta: i) una classe di nuove strategie di supervisione distribuita applicate a sistemi multi-veicolo di superficie marina per affrontare problemi di coordinamento, ii) uno schema di supervisione innovativo che evita la necessità di modellazione esplicita e iii) la validazione sperimentale di strategie di supervisione distribuita. Facendo ricorso alla strategia del Command Governor (CG), vengono impiegati algoritmi di supervisione predittiva per affrontare efficacemente i problemi di supervisione summenzionati.

Mentre le tradizionali strategie CG distribuite adottano un approccio non cooperativo, in questa tesi vengono introdotti due schemi distribuiti che promuovono la cooperazione tra gli agenti coinvolti. In primo luogo, considerando la natura computazionalmente impegnativa degli approcci cooperativi, l'idea principale consiste nello sfruttare una soluzione non cooperativa non iterativa, potenziandola per indurre cooperazione tra gli agenti al fine di preservare un basso carico computazionale. Per questo motivo, allo schema risultante viene associato il concetto di "Cooperazione-indotta". In secondo luogo, utilizzando un algoritmo di ottimizzazione distribuito preso dalla letteratura, viene descritto e analizzato un approccio distribuito più generale che consente agli agenti di calcolare una soluzione centralizzata quasi ottimale. A questo proposito, lo schema risultante, chiamato "Cooperativo", prevede che gli agenti si scambino solo variabili ausiliarie, garantendo così la privacy.

Contemporaneamente, per affrontare la sfida di risolvere il problema di progettazione del CG quando una rappresentazione esplicita del modello non è

disponibile, viene presentato un approccio noto come "Data-Driven". Questa alternativa all'approccio basato su modelli evita la necessità di fasi di modellazione, identificazione e validazione che richiedono molto tempo e, diversamente da altri approcci esistenti, non richiede una nuova raccolta di dati in caso di modifiche al controllore.

Infine, al fine di valutare le prestazioni computazionali e l'implementabilità delle strategie distribuite basate su CG su veicoli marini di superficie, vengono presentati diversi risultati sperimentali in vari ambienti. Questi risultati sono ottenuti utilizzando prototipi di veicoli marini di superficie ottenuti a seguito di un processo completo di progettazione, identificazione, validazione, controllo ed implementazione.

Contents

List of Figures

List of Tables

Acronyms

AHRS	Attitude and Heading Reference System
ARMA	Autoregressive Moving Average
ASV	Autonomous Surface Vehicle
BLDC	Brushless DC Motor
BMS	Battery Management System
CG	Command Governor
CPU	Central Processing Unit
DPC	Distributed Predictive Control
DOF	Degrees of Freedom
ESC	Electronic Speed Controller
FOC	Field-Oriented Control
FPU	Floating-Point Unit
GNC	Guidance, Navigation, and Control
GPS	Global Positioning System
IMU	Inertial Measurement Unit
IP	Internet Protocol
LTI	Linear Time-Invariant
MCU	Microcontroller Unit
MPC	Model Predictive Control
MPU	Microprocessor Unit
OA	Obstacle Avoidance
OAS	Output Admissible Set
PID	Proportional-Integral-Derivative
PO	Pareto Optimal
PWM	Pulse-Width Modulation
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
RSDD	Relaxation and Successive Distributed Decomposition
SOC	State of Charge
ST	Spanning Tree
UAV	Unmanned Aerial Vehicle

UMV	Unmanned Marine Vehicle
USB	Universal Serial Bus
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
WLAN	Wireless Local Area Network

Chapter 1
Introduction

1.1 Motivations

Drawn by their vital resources, a significant portion of the world's population resides near water bodies, such as oceans, lakes, and rivers. This density is particularly pronounced within 75 miles of coastlines, where it continues to grow [1, 2]. These aquatic environments significantly impact human health [3], both directly and indirectly, and provide essential resources such as food, water, and recreation. However, they face increasing threats such as microbial and chemical contamination, harmful algal blooms, and invasive species, particularly in coastal waters, which can cause serious health problems. These factors, combined with rising global temperatures, environmental abnormalities, and security concerns, present critical challenges that require immediate attention in both oceans and shallow waters, including coastal marine waters. Traditionally, marine vehicles, such as cargo ships, container ships, passenger ships, and tugs, have played a central role in monitoring marine environments. However, these vessels are resource-intensive, significantly contribute to pollution [4], and dependent on human intervention, leading to high operational costs [5]. To address these limitations, innovative monitoring solutions have been sought to replace these environmentally intrusive traditional methodologies. More specifically, Unmanned Marine Vehicles (UMVs) have emerged as a powerful environmentally friendly alternative, offering numerous advantages, including the ability to perform tasks autonomously with minimal human intervention, even when accessing remote or hazardous areas, or in harsh weather conditions. Additionally, UMVs can operate continuously for extended periods when equipped with rechargeable batteries and on-board power recharging modules. They can be designed in a modular way to allow the attachment of various sensing modules, thereby enabling their adaptability to a wide range of missions and their capability to simultaneously fulfill multiple objectives. Among UMVs, Unmanned Surface Vehicles (USVs), often called Autonomous Surface Vehicles (ASVs), closely related

to the families of Unmanned Underwater Vehicles (UUVs) and Autonomous Underwater Vehicles (AUVs), excel in surface navigation. USVs are designed to operate in the ocean and also in a small water column such as rivers [6] and lakes, and to navigate autonomously along programmed transects on the sea surface, holding course with high accuracy in the face of sea currents and wind. Compared to manned surface vehicles, the low weight and compact dimensions give them enhanced maneuverability and deployability in shallow waters (riverine and coastal areas) where larger craft cannot operate effectively. Furthermore, ASVs also have greater potential payload capacity and are able to perform deeper water depth monitoring and sampling compared to other aircraft/Unmanned Aerial Vehicles (UAVs) and spacecraft. ASVs are not designed to dive below the sea surface, and are, therefore, much simpler data platforms to design and construct than AUVs. Furthermore, compared with ROVs and AUVs, these have significantly fewer limitations on communication and localization, and are usually easier to operate and maintain. A collection of high-speed communication devices and high-accuracy localization systems can be conveniently integrated. For these reasons, in recent years, research on ASVs has expanded significantly [7, 8], with different designs (Figure 1.1) and applications spanning various marine domains. Potential applications include tracking of underwater vehicles [9], ocean remote sensing [10], offshore surveillance [11], water sampling [12], ocean monitoring [13], and maritime security operations [14].

While single ASVs offer potential, their limitations in complex tasks, such as those demanding rapid exploration or high fault tolerance, have motivated a growing interest in swarms of ASV systems [15]. This shift in research focus is driven by the need for efficient ocean exploration, encompassing large spatial domains, and ensuring robust data collection characterized by accuracy, precision, and consistency. Illustrative examples include water sampling across diverse locations and the detection of underwater acoustic sources within expansive ocean regions. Rapidly changing water conditions induced by currents can compromise the accuracy of water sample collection and signal attenuation impacting acoustic source detection makes such tasks challenging for a single ASV. In such scenarios, a swarm robotic paradigm emerges as a viable solution. By employing multiple agents, large areas can be explored simultaneously and search and sampling times can be reduced.

In general, the concept of swarming robots has demonstrated the potential for enhanced efficiency in diverse robotic domains, offering several advantages, including operational flexibility to adapt to diverse tasks, such as national security missions (Figure 1.2(b)), and environments, reduced task completion times, resilience to individual failures, and scalability to accommodate varying workspace dimensions and task complexities [16]. For instance, in [17] a team of surface vehicles has been considered acting as a water-based distributed sensor network for coastal applications, while in [18] ASVs, working with an UAV, have been used to encircle the boundary of a

(a) CEE USV.

(b) Kingfisher USV.

(c) Medusa USV.

(d) SL40 USV.

Fig. 1.1 Different types of Unmanned Surface Vehicles.

chemical spill. Additional relevant research on ASV swarming is detailed in [19], [20], and [21].

(a) Three Kayak USVs operating in a network, Massachusetts Institute of Technology (MIT).

(b) Swarm of USVs employed in "cooperative confrontation" in order to "besiege and expel" undesired surface vessels, Yunzhou Tech.

Fig. 1.2 Swarms of USVs.

Existing research often employs conventional ASV designs (Figure 1.2(a)), such as monohull or multi-hull, which, while offering maneuvering stability and higher speeds, often exhibit limitations in maneuverability due to factors

including hull shape, large length-to-width ratio, and actuator placement. As a result, most current ASVs are classified as underactuated systems and, despite their value, are often subject to operational restrictions. More in detail, their shape and turning radius can restrict their maneuverability, particularly in confined spaces. On the other hand, despite their slightly higher cost, recent studies explored overactuated or fully actuated ASV platform designs for applications like diver tracking [22] and swarming of surface drones for environmental monitoring [23]. Omnidirectional ASVs are in fact relatively easier to operate than their under-actuated counterpart and are able to work where precise maneuvering is required, especially in confined areas. For example, highly maneuverable USVs have been employed to track UUVs, compensating for the inherent limitations of their onboard sensors by providing real-time accurate navigation information [24]. Moreover, in contrast to underactuated vehicles, which are subject to nonholonomic constraints that limit the applicability of classical control techniques, fully-actuated and over-actuated systems are more amenable to traditional control design methods, as they can be stabilized using time-invariant continuous state feedback.

To enhance their autonomy, cooperation with safety requirements is essential. More in detail, collision avoidance, such as preventing inter-USV collisions [25, 26] and environmental objects avoidance [27] capabilities need to be included. However, coordinating and ensuring cooperation among multiple surface marine vehicles, while considering state and actuator limits, presents significant challenges. Traditional approaches often involve introducing amplitude/rate limits into the control design phase to handle actuator constraints, while motion planning techniques are employed to ensure dynamically feasible and safe trajectories. However, these methods have limitations and make it difficult to adjust to challenges coming from working in dynamic contexts. Specifically, frequent replanning becomes necessary, significantly impacting performance. Furthermore, imposing formation-keeping constraints on robots to perform increasingly complicated tasks [38, 39] introduces additional complexity that current approaches frequently find difficult to handle.

On the other hand, alternative solutions to path-planning problems that are more attractive in handling complicated tasks have recently emerged from advances in distributed control of dynamic linear/non-linear systems with input and/or state-related constraints. A common feature of these approaches relies upon recasting the problem requirements, such as bounds imposed on process variables for safe and efficient operation and collision/obstacle avoidance conditions, in the form of pointwise-in-time set-membership state constraints involving variables of one or more vehicles of the formation. Notably, *Model Predictive Control* (MPC) has emerged as a viable control strategy for USVs, effectively incorporating actuator and state constraints into the control design process [55, 56].

The Command Governor (CG)

Despite more advanced control methodologies have been extensively researched, PID control remains the dominant approach in USV control system design [8], which often lacks the inherent ability to handle operational limitations such as actuator constraints and process variable bounds. To address the above described challenges of enforcing state and coordination constraints rigorously and efficiently on such pre-compensated systems, while preserving existing controllers that already meet performance requirements, the *Command Governor* (CG) [52, 53] approach emerges as a promising solution.

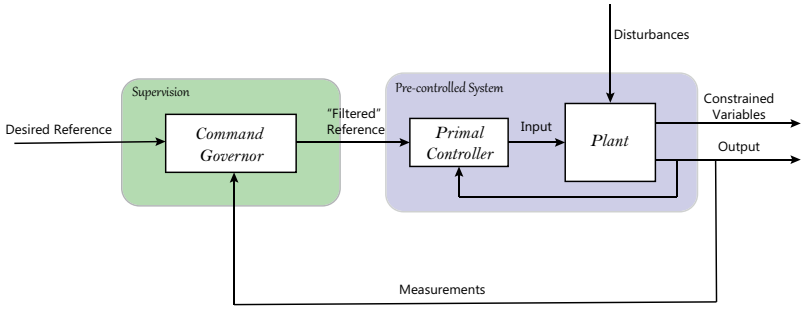


Fig. 1.3 Control scheme with Command Governor.

The CG unit, shown in Figure 1.3, consists of a higher-level nonlinear static device that is used not in place of existing conventional controllers, but in addition to them, as a supervising module for constraint fulfillment and improved performance. More in detail, in a centralized setting, it modifies a nominal reference sequence in tracking control problems whenever its application results in a pointwise-in-time violation of set-membership constraints on pertinent variables of the closed-loop system. On the contrary, the modified reference/command sequence ensures by design that all operative requirements along the system evolutions, driven by it, are satisfied. Furthermore, the modified command sequence is computed as the best approximation of the original one with respect to some selection index. The CG design problem, like MPC techniques, is rooted in the receding horizon paradigm. However, in contrast with other predictive control approaches which attempt to solve stabilization/tracking and constraint fulfillment at the same time, its main benefit is that the numerical burdens of the online optimization phase is usually much lighter than that pertaining to the MPC computations, which does not include in its formulation the stabilization component, i.e. future state

trajectories or control sequences. A more in-depth discussion of this topic can be found in [53, 54].

Originally, model-based CG-based formulations have been expressed in a centralized formulation, with a single supervisory unit solving the optimal control problem for a set of systems. Existing CG approaches have proven to be effective in supervising systems for constraint fulfillment and found different industrial applications [53], attracting considerable attention and leading to the development of numerous approaches to address various challenging aspects in recent years. For instance, schemes have been developed to handle disturbances [57, 58], model uncertainties [59], partial state information [60], and nonlinear systems [62, 63].

More recently, considering the impracticality of using a central unit in complex networked systems, the research activity in this field has shifted towards distributed approaches, in which decision making is distributed among agents corresponding to different subsystems making up the whole. In this case, the optimal control problem is distributed among the agents of a network, each solving a local optimization problem, i.e. each agent in the ensemble should decide its own control action based on only a subset of sensed and/or communicated information from the entire system. Distributed or decentralized forms of MPC and CG have been introduced in literature. For instance, distributed CG strategies have been proposed in [64, 65, 66, 67, 68], as well as more recent formulations that handle time-varying topology and/or constraints [69, 70, 71].

However, existing distributed CG-based solutions often do not induce cooperative behavior and have not yet been experimentally validated on actual marine surface platforms. Furthermore, standard centralized CG formulations depend on explicit mathematical models to compute predictions, thereby motivating the need for the development of novel approaches and experimental validation.

1.2 Cooperation-inducing and Cooperative Distributed CGs

Distributed CG schemes, as mentioned above, have the benefit of overcoming the need of a central computational facility with access to all system information, which often can become impractical for large-scale spatially distributed systems due to unrealistic computational and communication requirements. In these schemes, each agent can solve a localized version of the original problem, assuming fixed values for all other variables in the network and few data exchange occurs only once at a sampling time. More in detail, each agent shares only limited information with its neighbors, enhancing security and robustness against sudden disconnections of individual agents. Moreover, the modular structure enables the incorporation of different constraints, e.g.

collision avoidance [72]. However, due to the fact that the performance index of that each agent locally minimizes contains only its contribution, this leads to a self-fish behaviour, that could, unlike centralized schemes, potentially lead to deadlock situations and/or undesirable Nash equilibria [68, 66]. Moreover, it is necessary to develop efficient methods to handle non-convex obstacle avoidance constraints, by leveraging the inherent modular nature of distributed strategies, to fully realize the potential of this approach in practical applications and to achieve the aforementioned objectives.

Since existing distributed CG approaches, as previously discussed, often employ non-cooperative behavior, in order to enhance overall system performances, it is important to know how to avoid selfish behavior and when cooperation can be induced. Typically, a cooperative behavior is promoted by each local agent through the consideration of a greater portion of the system-wide objective. To address these issues, the primary objective of this research is to develop innovative Command Governor-based schemes that exhibit both cooperation-inducing and cooperative behaviors and validate them on surface marine vehicle formations. Building upon non-cooperative, non-iterative approaches, which are not very computationally intensive, and borrowing tools from separating hyperplanes, obstacle avoidance constraints are integrated. It is assumed that the higher levels of a modularly designed supervision strategy induces a cooperative behavior among the vehicles to ensure connectivity and target reaching. On the other hand, inspired by recent contributions on cooperative distributed MPC [73], cooperative distributed CG schemes can be designed to preserve privacy as well. In this case, at each time step, agents iteratively solve a sequence of optimization problems and exchange only dual variables.

Compared to existing approaches, the cooperation-inducing schemes can offer a degree of cooperation for agents moving in a 2D space, while potentially requiring fewer computational resources. On the other hand, cooperative schemes, that may be more computationally demanding than their counterpart, are expressed in a general formulation and can achieve the same solution as the centralized approach. The development of both schemes enables the selection of the most suitable approach based on specific application requirements. Moreover, although the research is pushed and motivated by ASVs, the resulting schemes can be applied across diverse contexts and areas.

1.3 Data-Driven CGs

Traditional CG approaches use the mathematical model of the system in accordance with a receding horizon strategy to select the optimal reference input. However, one of the main issues lies in the inherent dependence on such explicit prediction model. Such a model-based approach becomes less suitable for applications where thorough modelling, parameter identification

and validation are too costly. Indeed, this thesis demonstrates the challenges associated with these processes, which necessitated few repetitions, motivating the development of a solution for the general class of linear systems.

Motivated by the need to avoid undertaking the time-consuming steps in the modeling, identification and validation phases, in order to have a supervision scheme that bypasses explicit modeling and that does not rely on a parametric system representation, tools from the recent several Data-driven Predictive Control (DPC) algorithms [74] and behavioral systems theory [75] are leveraged. More in detail, a representation of the controller and one input/output trajectory of the plant are used to compute predictions of the overall system. In line with the CG logic, by using the computed predictions, constraint fulfillment can be imposed by modifying the given reference.

Compared to existing approaches, where reference and output trajectories of the closed-loop system are considered without exploiting the possible knowledge of the underlying stabilizing control law, constraints on input and output variables can be written explicitly and the resulting scheme is adaptable even amidst controller modifications, i.e. any change in the control unit does not require to run the entire process of offline data collection and Hankel data matrix verification again.

1.4 Design and Supervision of Surface Marine Vehicles

To validate the theoretical results, both Matlab and a low-physics simulator can be used, especially considering the impracticality of constructing and testing on a large fleet of vehicles. Such environments have enabled both modeling and simulation, as well as low-level physics based simulations [78]. However, real-world validation is also essential to assess the feasibility of the CG-based distributed strategy in terms of computational requirements and applicability to actual marine surface vehicle formations. This requires addressing challenges such as limited computational resources, communication constraints, modeling accuracy, and sensor and actuator modeling.

To ensure a comprehensive understanding, all the design, identification, validation, control and supervision phases, that fully respect the assumption and setting previously introduced, are detailed. The electrical and mechanical design phases take into account the omnidirectional nature of the considered vehicles and the accuracy of a proposed linear time-invariant model of the vehicle is validated also with in field experiments, to further support simulation results. The prototype is controlled and tested, i.e. the overall performance of the controlled system are verified to ensure that the vehicle offers precise maneuvering. Finally, the distributed supervision strategy is implemented, and local, obstacle avoidance and collision avoidance are correctly enforced to verify computational feasibility.

Experimental results, compared to simulations, demonstrate that the computational requirements of distributed CG-based supervision strategies are within the acceptable bounds of the hardware, confirming their suitability and implementability to tackle the above mentioned challenges in real-world applications.

1.5 Thesis Overview

1.5.1 Thesis Outline

The proposed solutions can be selected based on specific objectives and application requirements. However, given that these approaches are based on different concepts and logics, this leads to the need for different theoretical developments. To provide a clear and comprehensive foundation, in Chapter 2 essential concepts and background information, including necessary definitions and notations, are provided. Chapter 3 presents the cooperation-inducing distributed command governor scheme, useful to supervise a group of autonomous vehicles in a 2D environment in order to ensure safe navigation, collision avoidance, while maintaining connectivity between vehicles and formation compactness. Chapter 4 describes the cooperative distributed command governor scheme that, by adopting a distributed optimization framework based on relaxation techniques and duality theory, enables agents to cooperatively contribute individually to the minimization of a global performance index, while preserving the overall privacy. Chapter 5 is dedicated to solving the supervision problem in absence of an explicit model representation, for which a data-driven approach is proposed, by means of leveraging tools from closed-loop data-driven simulation. Chapter 6 presents design and experimental tests on developed real robotic platforms to assess the implementability of distributed command governor approaches, in environments in which supervision would be beneficial. Finally, Chapter 7 draws the main conclusions and highlights some future research directions.

1.5.2 Contributions

This dissertation is largely based, both directly incorporating specific contributions and/or leveraging their results, on the following contributions

Journals

I) [173] "*Preserving Agents Connectivity Amidst Static Obstacles: A Distributed Predictive Supervision Approach within Robot Operating System,*" in IEEE Transactions on Intelligent Vehicles;

II) [108] "*A Matlab-based Toolbox for Supervising Multi-Vehicle Autonomous Systems,*" in IEEE Access;

III) [172] "*Linear System Identification and Control of a Low-Cost High-Performance Omnidirectional Marine Surface Vehicle for Swarming Applications,*" in Journal of Field Robots (under review).

Conferences

IV) [170] "*Distributed Constrained Connectivity Keeping Supervision Scheme in the Presence of Static Obstacles,*" in Proc. of 2022 IEEE 61st Conference on Decision and Control;

V) [171] "*Distributed Connectivity Keeping Supervision Scheme with Collision and Obstacle Avoidance Requirements,*" in Proc. of 2023 IEEE American Control Conference;

VI) [174] "*Data-Driven Command Governors for Discrete-Time LTI Systems with Linear Constraints,*" in Proc. of 2024 IEEE 63rd Conference on Decision and Control;

VII) [169] "*Coordination of Marine Multi-Vehicle Autonomous Systems via the Distributed Command Governor Approach,*" in Proc. of 2023 IEEE Ocean Conference;

VIII) [109] "*CoGoV: A Safe Motion Planning Distributed Supervision Framework for Multi-Vehicle Formations,*" in Proc. of 22nd IFAC World Congress;

IX) [175] "*Privacy-Preserving Cooperative Distributed Command Governor Schemes,*" in Proc. of 2025 IEEE American Control Conference (to appear).

Chapter 2
Background Material

To provide the theoretical foundations on which the results are built and presented, this chapter presents essential definitions, notation, notions and ideas, and it is instrumental for a comprehensive understanding of the proposed schemes in the following chapters. More in detail, after providing the basic definitions, the ideas underlying the standard CG approach are presented, followed by centralized and some distributed solutions, concluding with concepts and definitions useful in the area of behavioral systems theory.

2.1 Definitions

Definition 2.1 (Pontryagin-Minkowski Difference): Given two sets $\mathcal{D}_1$, $\mathcal{D}_2 \in \mathbb{R}^n$, the Pontryagin-Minkowski Difference is defined as

$$\mathcal{D}_1 \sim \mathcal{D}_2 := \{d_1 | d_1 + d_2 \in \mathcal{D}_1, \forall\, d_2 \in \mathcal{D}_2\}.$$

Definition 2.2 (Pontryagin-Minkowski Sum): Given two sets $\mathcal{D}_1$, $\mathcal{D}_2 \in \mathbb{R}^n$, the Pontryagin-Minkowski Sum is defined as

$$\mathcal{D}_1 \oplus \mathcal{D}_2 := \{d_1 + d_2 | d_1 \in \mathcal{D}_1,\, d_2 \in \mathcal{D}_2\}.$$

Definition 2.3 (**Graph**): A graph is an ordered pair $\mathcal{G}(\mathcal{A}, \mathcal{E})$ such that

- $\mathcal{A}$ is the set of nodes;
- $\mathcal{E} \subseteq \mathcal{A} \times \mathcal{A}$ is a subset of pairs of $\mathcal{A}$ known as the set of edges connecting two nodes.

Definition 2.4 (**Undirected Graph**): An undirected graph is a pair $\mathcal{G}(\mathcal{A}, \mathcal{E})$ such that

- $\mathcal{A}$ is the set of nodes;

- $\mathcal{E} \subseteq [\mathcal{A}]^2$ is the set of edges, where each edge is an unordered pair $e = \{i, j\}$ of $\mathcal{A}$.

Definition 2.5 (**Weighted Graph**): A weighted graph $\mathcal{G}(\mathcal{A}, \mathcal{E}, \mathcal{W})$ is an ordered triplet, such that

- $\mathcal{A} \subset \mathbb{Z}_+$ is the set of N nodes;
- $\mathcal{E}$ is the subset of pairs (i, j) of $(\mathcal{A}\times\mathcal{A})$ connected by an edge, i.e. $\mathcal{E} \subseteq \mathcal{A}\times\mathcal{A}$.
- $\mathcal{W}$ is a set of scalar weights assigned to each edge of $\mathcal{E}$ and denoted as $w_{ij} : \mathcal{E} \to \mathbb{R}$.

Definition 2.6 (**Path**): A path is a sequence $(a_0, a_1, \ldots, a_p)$ of nodes such that for k, $0 \leq k < p$,

- If the graph is undirected, $\{a_k, a_{k+1}\}$ is an edge;
- If the graph is directed (a_k, a_{k+1}) is an edge.

Definition 2.7 (**Graph Connectivity**): An undirected graph $\mathcal{G}$ is connected if there exists a path between every two vertices of the graph.

Definition 2.8 (**Neighborhood of the i-th node**): The neighborhood $\mathcal{N}_i$ of the i-th node in $\mathcal{G}(\mathcal{A}, \mathcal{E})$ consists of all its adjacent nodes i.e.

$$\mathcal{N}_i = \{j \in \mathcal{A} : (i, j) \in \mathcal{E}\}. \tag{2.1}$$

Definition 2.9 (**Local states of neighbors of the i-th node**): The sets of all states x_j related to the i-th agent can be characterized as follows

$$[x]_i = \{\text{All subvectors } x_j \text{ of } x \text{ such that } j \in \mathcal{N}_i\}. \tag{2.2}$$

2.2 Basic Command Governor Design

Considering that the CG serves as a foundation for the supervision architectures presented in the following Chapters, the aim of this section is to recall the notions and ideas underlying the standard CG approach for linear systems. Based on a hierarchical philosophy, the approach consists on a separation of the stabilization/tracking and constraint fulfillment requirements, rather than solving them concurrently. More in detail, a primal compensated control system, designed so as to perform satisfactorily under unconstrained conditions, is augmented with the CG supervision module. This non linear device, whenever necessary, "filters" the input to the primal control system to ensure constraint satisfaction. As mentioned in Section 1.1, this approach has received significant attention in the literature for addressing complex systems, including nonlinear and uncertain systems.

2.2.1 System description

Let the closed-loop system illustrated in Figure 2.1, composed of the plant and primal controller, be described by the following discrete-time model

$$\begin{cases} x(t+1) = \Phi x(t) + Gg(t) + G_d d(t), \\ y(t) \quad = H_y x(t), \\ c(t) \quad = H_c x(t) + Lg(t) + L_d d(t), \end{cases} \tag{2.3}$$

where $t \in \mathbb{Z}$, $x \in \mathbb{R}^n$ is the state vector (including the controller states under dynamic regulation), $g \in \mathbb{R}^m$ the manipulable reference vector which, if no constraints (and no CG) were present, would coincide with the desired reference $r \in \mathbb{R}^m$ and $y \in \mathbb{R}^m$ the output vector, which is required to track r. The vector $d \in \mathbb{R}^{n_d}$ is a disturbance signal assumed to belong to the convex and compact set $\mathcal{D} \subset \mathbb{R}^{n_d}$, with $0_{n_d} \in \mathcal{D}$, i.e.

$$d(t) \in \mathcal{D}, \ \forall t \in \mathbb{Z}_+. \tag{2.4}$$

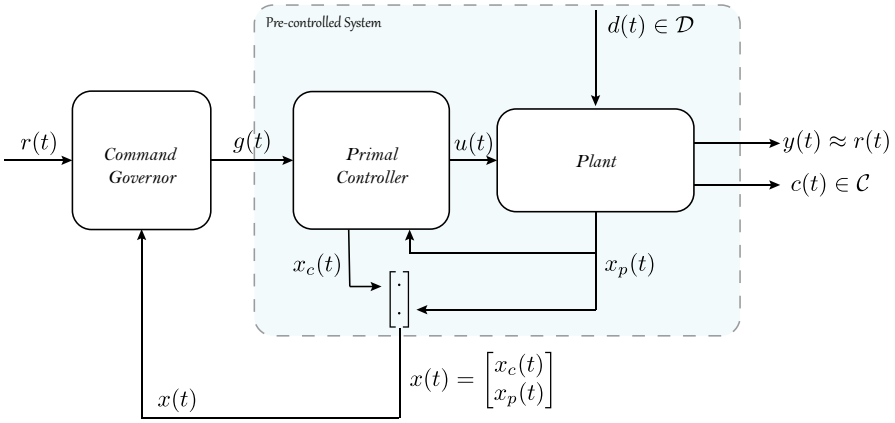


Fig. 2.1 Basic Model-based CG structure.

Furthermore, the vector $c \in \mathbb{R}^{n_c}$ represents the constrained outputs vector which has to fulfill

$$c(t) \in \mathcal{C}, \ \forall t \in \mathbb{Z}_+, \tag{2.5}$$

with $\mathcal{C}$ being a prescribed convex and compact set with non empty interior, regardless of any possible admissible disturbance sequence realization $d(\cdot) \in \mathcal{D}$. Additionally, it is assumed that

A1. The overall system (2.3) is asymptotically stable, i.e. Φ is a Schur Matrix.

A2. System (2.3) is offset free, i.e. $H_y(I_n - \Phi)^{-1}G = I_m$.

2.2.2 Problem Formulation

The CG design problem involves determining at each time step t a suitable command $g(t)$, that best approximates the given reference $r(t)$, such that, if constantly applied from t onward, ensures constraints satisfaction along the system evolutions. Then, the problem to be solved in the model based setting can be formally stated as follows

Problem 2.10 *Given system* (2.3)*, determine at each time instant* t *a suitable command*

$$g(t) = l(x(t), r(t)),$$

that is the "best approximation" of $r(t)$ *according to* (2.5)*,* $\forall d(\cdot) \in \mathcal{D}$*. Moreover, whenever* $r(t) \equiv r$*,* $\forall t$*, with* r *a constant set-point, the sequence of solutions* $g(t)$ *converges to* r*, or to its best feasible approximation* $\hat{r}$ *whenever* r *is not constraints admissible.*

2.2.3 Main Results and Properties

Consider the closed-loop system (2.3), satisfying assumptions ***A1***-***A2***, constraints (2.5), and suppose that the state is measurable at each time instant t. To recast constraints of Problem (2.10) in a form that is more akin to introduce the CG-based supervision strategy, some definitions are introduced. Let

$$x_g := (I - \Phi)^{-1}Gg \tag{2.6}$$

represent the steady-state of the system for a constant *virtual command* g, i.e. $g(k) \equiv g$, while

$$x(k, x, g) := \Phi^k x + \sum_{\tau=0}^{k-1} \Phi^{k-\tau-1}Gg \tag{2.7}$$

are the set-valued disturbance-free future predictions (*virtual evolutions*), computed along the *virtual time* k, of the state variable under a constant *virtual command* g from the initial state x (at virtual time $k = 0$). The same can be done for the c-variable, i.e.

$$\begin{aligned} c_g &:= H_c x_g + Lg, \\ c(k, x, g) &:= H_c \left(\Phi^k x + \sum_{\tau=0}^{k-1} \Phi^{k-\tau-1}Gg \right) + Lg, \end{aligned} \tag{2.8}$$

while virtual prediction the c-variable and for all possible disturbance sequence realizations $\{d(l) \in \mathcal{D}\}_{l=0}^{k}$ can be defined as

$$\tilde{c}(k,x,g,d(\cdot)) = \bigcup_{d(\cdot)\in\mathcal{D}} \left\{ H_c \left(\Phi^k x + \sum_{i=0}^{k-1} \Phi^{k-i-1}(Gg + G_d d(i)) \right) + Lg + L_d d(k) \right\}. \tag{2.9}$$

By linearity, the latter can be rewritten as the sum of two terms, i.e.

$$\tilde{c}(k,x,g,d(\cdot)) = c(k,x,g) + c^d(k,d(\cdot)), \tag{2.10}$$

with the set-valued virtual evolutions due to all possible disturbance sequence realizations

$$c^d(k,d(\cdot)) := \bigcup_{d(\cdot)\in\mathcal{D}} \left\{ \sum_{i=0}^{k-1} H_c \Phi^{k-i-1} G_d d(i) + L_d d(k) \right\}. \tag{2.11}$$

By definition, it holds that

$$c^d(k,d(\cdot)) \subseteq \Delta_k, \tag{2.12}$$

where the set Δ_k can be recursively computed as

$$\Delta_k = \Delta_{k-1} \oplus H_c \Phi^{k-1} G_d \mathcal{D}, \quad \Delta_0 = L_d \mathcal{D}. \tag{2.13}$$

To ensure constraints fulfillment, a possible solution could be considering only the disturbance-free evolutions in (2.8) of system (2.3) and to follow a "worst case" approach. To this end, let

$$\mathcal{C}_k := \mathcal{C} \sim \Delta_k, \quad \mathcal{C}_\infty := \mathcal{C} \sim \Delta_\infty, \tag{2.14}$$

where $k > 0$ and Δ_∞ is the Hausdorff limit of the set sequence Δ_k. Because of asymptotical stability of Φ, it can be proved [57] that Δ_∞ is convex and compact and that, provided all $\mathcal{C}_k$ are non-empty, they are also convex and compact, satisfy the nesting condition $\mathcal{C}_k \subset \mathcal{C}_{k-1}$ and make $\mathcal{C}_\infty$ a nonempty convex and compact set. Then, as a consequence it holds that constraints are always satisfied if

$$\begin{array}{c} c(k,x,g) \in \mathcal{C}_k, \ \forall k \in \mathbb{Z}_+ \\ \Downarrow \\ \tilde{c}(k,x,g) = c(k,x,g) + c^d(k,d(\cdot)) \subset \mathcal{C}, \ \forall k \in \mathbb{Z}_+ . \end{array} \tag{2.15}$$

Formally, the set of all admissible virtual sequences from initial state x can be introduced as

$$\mathcal{V}(x) = \{ g \in \mathbb{R}^m : \ c(k,x,g) \in \mathcal{C}_k, \ \forall k \in \mathbb{Z}_+ \}. \tag{2.16}$$

Analogically, the notion of *Output Admissible Set* (*OAS*) related to set $\mathcal{C}$ (2.5) can be introduced

$$\mathcal{Z} := \left\{ \begin{bmatrix} x \\ g \end{bmatrix} \in \mathbb{R}^{n+m} \,\middle|\, c(k,x,g) \in \mathcal{C}_k,\ \forall k \in \mathbb{Z}_+ \right\}. \tag{2.17}$$

It can be proved [79], that constraints (2.5) are always satisfied if

$$(x(t), g(t)) \in \mathcal{Z},\ \ \forall t \in \mathbb{Z}_+. \tag{2.18}$$

However, these sets are not finitely determinable. To solve this, it can be proved that it is enough to introduce steady-state constraints restricted by a margin $\delta > 0$ and to evaluate the virtual predictions $c(k,x,g)$ within a certain *admissibility index* k_0. Formally, let

$$\mathcal{C}^\delta := \mathcal{C}_\infty \sim \mathcal{B}_\delta, \quad \mathcal{W}^\delta := \{g \in \mathbb{R}^m :\ c_g \in \mathcal{C}^\delta\}, \tag{2.19}$$

where $\mathcal{B}_\delta$ is the ball of radius δ centered at the origin and $\mathcal{W}^\delta$, which we assume non-empty, the closed and convex set of all constant commands g whose corresponding disturbance-free equilibrium points in (2.8) satisfy the constraints with margin δ. Then, set $\mathcal{V}(x)$ in (2.16) can be expressed as

$$\mathcal{V}(x) = \left\{ g \in \mathcal{W}^\delta :\ c(k,x,g) \in \mathcal{C}_k,\ \forall k = 0, \ldots, k_0 \right\}. \tag{2.20}$$

In a scenario where set $\mathcal{C}$ is defined as a polytope described by a set of linear inequalities, a procedure showing how to compute index k_0 is reported in Appendix A. The main idea behind this CG design problem is to choose at each time step a constant virtual command $g(\cdot) \equiv g$, with $g \in \mathcal{W}^\delta$, such that the corresponding virtual evolutions fulfill the constraints over the horizon $k = 0, \ldots, k_0$ and its "distance" from the constant reference of value $r(t)$, valuated by $\|g - r(t)\|^2_\Psi$, $\Psi = \Psi^T > 0$, is minimal. Such a command is applied, and then, after performing a new state measurement, the procedure is repeated at the next time instant, following a *receding horizon* philosophy. Formally, the CG will choose a command $g(t)$ at each time instant t as the solution of the following constrained optimization problem

$$g(t) = \arg \min_{g \in \mathcal{V}(x(t))} \|g - r(t)\|^2_\Psi. \tag{2.21}$$

The following Theorem, proved in [58] and [80], summarizes the properties of the standard CG Algorithm:

Theorem 2.11 - *Let assumptions **A1**-**A2** be fulfilled. Consider system* (2.3)-(2.5) *along with the CG selection rule* (2.21) *and let* $\mathcal{V}(x(0))$ *be non-empty. Then:*

1. *At each decision time t, the minimizer in* (2.21) *uniquely exists and can be obtained by solving a convex constrained optimization problem;*

2. *The system supervised by the CG never violates the constraints, i.e.* $c(t) \in \mathcal{C}$ *for all* $t \in \mathbb{Z}_+$ *regardless of any possible admissible disturbance realization* $d(\cdot) \in \mathcal{D}$*;*
3. *The overall system is asymptotically stable. Moreover, whenever* $r(t) \equiv r$*, with* r *a constant set-point, the sequence of* $g(t)$*'s converges in finite time either to* r *or to its best admissible steady-state approximation* $\hat{r}$

$$\lim_{t\to\infty} \hat{x}(t) = x_{\hat{r}}, \quad \lim_{t\to\infty} \hat{y}(t) = y_{\hat{r}} = \hat{r}, \quad \lim_{t\to\infty} \hat{c}(t) = c_{\hat{r}}, \tag{2.22}$$

and

$$\exists t_f > 0 \ \ s.t. \ \ g(t) = \hat{r} := \arg\min_{g\in\mathcal{W}_\delta} \|g - r\|_{\Psi}^2, \forall t \geq t_f. \tag{2.23}$$

2.3 Command Governor Approaches for Multi-Agent Systems

With the standard Command Governor solution now established, since we are dealing with multi-agent systems, it is worth introducing the formulations of the centralized and the distributed solutions. More in detail, a centralized implementation of the optimization problem (4.1) or a distributed solution involving N computational nodes can be employed to utilize the CG for supervising multi-agent systems. The centralized approach requires a central computational facility with access to all system variables, while the distributed approach involves N computational nodes, each with limited information about the overall system, where each node determines locally an admissible command $g_i(t)$ each time step t. Both approaches have inherent strengths and limitations.

In this case, consider $N \in \mathbb{N}_{>0}$ linear time-invariant closed-loop discrete-time dynamical systems, where each i-th subsystem is modeled as

$$\Sigma_i : \begin{cases} x_i(t+1) = \Phi_i x_i(t) + G_i g_i(t) + \sum_{j\in\mathcal{A}\setminus\{i\}} \Phi_{ij} x_j(t), \\ y_i(t) \ \ = H_i^y x_i(t), \\ c_i(t) \ \ = H_i^c x(t) + L_i g(t), \end{cases} \tag{2.24}$$

where, analogously to (2.3), for all $i \in \{1, \ldots, N\}$, $x_i(t) \in \mathbb{R}^{n_i}$ is the state vector (which includes the controller states under dynamic regulation) at time $t \in \mathbb{Z}_+$. Vector $g_i(t) \in \mathbb{R}^{m_i}$, that is the manipulable command vector, in the absence of constraints (and no CG), would coincide with the desired reference $r_i(t) \in \mathbb{R}^{m_i}$, assuming each system is regulated by a local controller that ensures stability and satisfactory closed-loop performance. The vector $y_i(t) \in \mathbb{R}^{m_i}$ is the output vector related to the tracking performance ($y_i(t) \approx r_i(t)$). Furthermore, $c_i(t) \in \mathbb{R}^{n_i^c}$ is the constrained vector that collects all

constrained local variables, i.e. $c_i(t) \in \mathcal{C}^i,\ \forall t \in \mathbb{Z}_+, i = \{1, \ldots, N\}$, with $\mathcal{C}^i$ being convex and compact polytopic sets. Finally in the general case, coupling between agents can be modeled using matrix Φ_{ij}. For simplicity, disturbances are not considered, as their handling has been previously discussed.

Then, the CG design problem involves determining, at each time step t and for each agent $i = \{1, \ldots, N\}$, a suitable command $g_i(t)$ that best approximates the reference $r_i(t)$ while ensuring no constraint violations, i.e.

$$c_i(t) \in \mathcal{C}^i, \forall t \in \mathbb{Z}_+,\ i \in \{1, \ldots, N\}, \tag{2.25}$$

under the application of such computed sequence of feasible $\{g_i(t)\}_{t=0}^{\infty}$.

2.3.1 Centralized Command Governor

Initially, CG-based formulations have been expressed in a centralized formulation [81]. With a single supervisory unit solving the optimal control problem for a set of systems, from a global point of view, the overall system Σ arising by the composition of the above N subsystems (2.24) can be described as

$$\Sigma : \begin{cases} x(t+1) = \Phi x(t) + Gg(t), \\ y(t) \quad = H^y x(t), \\ c(t) \quad = H^c x(t) + Lg(t), \end{cases} \tag{2.26}$$

where, to describe global (coupling) constraints among states of different subsystems, the vector c_i in (2.24) is allowed to depend on the aggregate state and manipulable commands vectors $x = [x_1^T, \ldots, x_N^T]^T \in \mathbb{R}^n$, with $n = \sum_{i=1}^N n_i$, and $g = [g_1^T, \ldots, g_N^T]^T \in \mathbb{R}^m$, with $m = \sum_{i=1}^N m_i$. Moreover, the other relevant aggregate vectors $r, y \in \mathbb{R}^m$ and $c \in \mathbb{R}^{n^c}$, with $n^c = \sum_{i=1}^N n_i^c$, are defined analogously. $\Phi = [\Phi_{ij}]_{i,j=1,\ldots,N}$, $G = diag(G_1, \ldots, G_N)$, $H^y = diag(H_1^y, \ldots, H_N^y)$, $H^c = [(H_1^c)^T, \ldots, (H_N^c)^T]^T$ and $L = [(L_1)^T, \ldots, (L_N)^T]^T$. Hereafter, following Assumption ***A1***, it is further assumed that *Φ is a strictly Schur matrix*. From a distributed perspective, this assumption implies that the local regulators to each subsystem (2.24) are capable to asymptotically stabilize the overall system (2.26).

Then, the same solution presented in Section 2.2.3 can be adopted. More in detail, due to the polytopic nature of $\mathcal{C}^i$, $\mathcal{V}(x)$ can be generally characterized by a finite number z of inequalities in $\mathbb{R}^m$ ([82]) that can be expressed in matrix form

$$\mathcal{V}(x) := \{g \in \mathbb{R}^m : Rg + \tilde{R}x - v \leq 0\}, \tag{2.27}$$

with matrices R, $\tilde{R}$ and v properly evaluated. By denoting $h(g, x) := Rg + \tilde{R}x - v$, h being an affine function $h : \mathbb{R}^{m+n} \to \mathbb{R}^z$, an equivalent form for (2.21) is

$$\begin{aligned} &\min_{g\in\mathbb{R}^m} \|g - r(t)\|_{\Psi}^2 \\ &s.t. \quad h(g, x(t)) \leq 0. \end{aligned} \tag{2.28}$$

Please notice that throughout this thesis, inequality signs are meant componentwise for vectors. As mentioned above, the solution for the above problem (2.28) represents the *best* feasible approximation of $r(t)$ which, if constantly applied from t onward, never produces constraints violation. It is noteworthy that, if $r(t) \notin \mathcal{V}(x(t))$, the solution of (2.28) is significantly influenced by the shape of the weighting matrix Ψ, which reflects the preferences of the designer. A particular case of interest here, contemplates the use of a block-diagonal matrix Ψ in (2.28), that is

$$\begin{aligned} \Psi &:= diag(w_1\Psi_1, \ldots, w_N\Psi_N), \\ \Psi_i \in \mathbb{R}^{m_i\times m_i},\ \Psi_i &= \Psi_i^T > 0_{m_i}, w_i \in \mathbb{R}_{>0},\ \forall i \in \mathcal{A}. \end{aligned} \tag{2.29}$$

2.3.2 Distributed Command Governor Approaches

While a centralized approach can yield optimal solutions, it suffers from various limitations, including single-point failure, scalability issues, and the impracticality of a centralized communication infrastructure. Consequently, recent research has shifted towards distributed approaches, where the optimal control problem is distributed among network agents, each solving a local optimization problem. This distributed paradigm enhances security and robustness by limiting information sharing to neighboring agents. In contrast to the centralized approach, which requires a central computational facility with access to all system variables to solve the optimization problem (4.1), this section focuses on distributed solutions that involve N computational nodes. Each node operates with limited information about the overall system and locally determines an admissible command $g_i(t)$ at each time step t.

In this context, we consider a network of N agents that communicate via a *connected*, *undirected* graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where $\mathcal{A} = \{1, \ldots, N\}$. A very important observation is that, once $\mathcal{V}(x)$ has been computed as in (2.28), the i-th agent may concur only to the violation of the constraints that directly depend on the command $g_i(t)$. Then each inequality of (2.28) involves just a limited number of agents. Each agent i is assumed to have a limited information scope, i.e. only regarding $j \in \mathcal{N}_i$. To put in evidence the constraints involving the i-th agent only, for each $i \in \mathcal{A}$, it is possible to define the block-matrices $R_i = [R_{i,1}|\ldots|R_{i,N}]$, with $R_{i,j} \in \mathbb{R}^{z_i\times m_i}$, $\tilde{R}_i = [\tilde{R}_{i,1}|\ldots|\tilde{R}_{i,N}]$, with $\tilde{R}_{i,j} \in \mathbb{R}^{z_i\times n_i}$ and vector $v_i \in \mathbb{R}^{z_i}$ collecting respectively the all and only (say z_i) rows r_j of R, rows $\tilde{r}_j$ of $\tilde{R}$ and v_j of v in which the i-th agent is involved, i.e. such that the sub-vectors $R_j^i \neq 0_{m_i}$. As a consequence, the inequalities in $\mathcal{V}(x)$ associated to the i-th agent can be described as

$$R_i g + \tilde{R}_i x - v_i \leq 0. \tag{2.30}$$

At this point, it is important to mention that, in some cases (e.g. for systems with no or weak dynamic interaction and coupling constraints), the matrix R can be quite sparse since only a subset of agents are involved in the constraints associated to the i-th agent. Please notice that, for agent i, in (2.28) $\exists$ at least a line r_p^T in R or a line r_x^T in $\tilde{R}$ such that ($r_p^i \neq 0$ and $r_p^j \neq 0$), ($r_x^i \neq 0$ and $r_x^j \neq 0$) respectively, for all $j \in \mathcal{N}_i$. In this context, from the perspective of agent i, constraints (2.25) become

$$\hat{R}_i g_{i,\mathcal{N}_i}(t) + \hat{\tilde{R}}_i x_{i,\mathcal{N}_i}(t) - \hat{v}_i \leq 0, \tag{2.31}$$

where $g_{i,\mathcal{N}_i}(t) = T^g_{\mathcal{N}_i} g(t)$ and $x_{i,\mathcal{N}_i}(t) = T^x_{\mathcal{N}_i} x(t)$ denote respectively the commands and the states associated to the neighbors of the i-th agent, with $\hat{R}_i = R_i T^{g,T}_{\mathcal{N}_i}$ and $\hat{\tilde{R}}_i = \tilde{R}_i T^{x,T}_{\mathcal{N}_i}$ the associated matrices. More in detail, to extract all rows of g and x related to the agents in $\mathcal{N}_i$, the $T^g_{\mathcal{N}_i}$ and $T^x_{\mathcal{N}_i}$ selection matrices are respectively used.

While several distributed CG approaches, such as the Sequential CG of [68, 65] and the Parallel CG of [83], have been proposed, in the following only the approaches useful in subsequent discussions are detailed.

2.3.2.1 Sequential

In the Sequential approach, only one agent at a time is allowed to compute its local admissible command $g_i(t)$, while all others are instructed to continue to apply the previously computed admissible command [68]. At the end of the computation of the admissible command, the agent transmits it and its aggregate state (system state and controller state) to its neighboring agents, enabling the next agent to perform calculations using the most recent information. At every time instant t, agent i has access to the following vectors

$$\begin{aligned} \xi_i(t) &= [g_1^T(t-1), \ldots, g_i^T(t-1), \ldots, g_N^T(t-1)]^T, \\ \vartheta_i(t) &= [x_1^T(t-1), \ldots, x_i^T(t), \ldots, x_N^T(t-1)]^T. \end{aligned} \tag{2.32}$$

Please notice that the admissible commands in the vector $\xi_i(t)$ are all updated (at the previous time instant). In fact, each agent is required to constantly apply the last calculated control signal over time. The same holds for vector $\vartheta_i(t)$, i.e. all of its values are updated, since at each time instant agent i is required to share its state information, allowing for inconsistency avoidance between exchanged and processed data. The proposed approach can be formulated as follows

Sequential CG Algorithm - *Agent* i

REPEAT AT EACH TIME t

1: BUILD VECTORS $\xi_i(t)$ E $\vartheta_i(t)$ USING RECEIVED DATA FROM $j \in \mathcal{N}_i$;
2: **if** $(t \bmod N) == i$ **then**
3: SOLVE

$$\begin{aligned} g_i(t) &= \arg\min_{g_i} \|g_i - r_i(t)\|^2_{\Psi_i} \\ &\text{S. T.} \\ [g_1^T(t-1), \ldots, g_i^T, \ldots, g_N^T(t-1)]^T &\in \mathcal{V}(x(t)). \end{aligned} \tag{2.33}$$

4: APPLY $g_i(t)$
5: UPDATE $g(t) = [g_1^T(t-1), \ldots, g_i^T(t), \ldots, g_N^T(t-1)]^T$
6: **else**
7: SET $g_i(t) = g_i(t-1)$
8: TRANSMIT $g_i(t)$ AND $x_i(t)$ TO THE NEIGHBORING AGENTS $\mathcal{N}_i$

where $\Psi_i > 0$ are defined in (2.29), $t \bmod N$ is the *modulo* operation that returns the remainder of the division t/N. In the above introduced algorithmic scheme, the operation $t \bmod N$ allows agent i to evaluate the possibility of updating its reference, thus preserving the sequentiality of the solution. Furthermore, the set $\mathcal{V}(x)$ is calculated as in (2.20).

2.3.2.2 Turn-based

The distributed CG strategy here presented is based on a particular coordination of agents that, on the basis of the constraints structure, are grouped into turns [64]. Agents belonging to each turn are instructed to modify their reference according to a precise scheduling procedure established offline. In contrast to the above introduced sequential solution, a more relaxed approach is adopted, where only the neighboring agents of agent i, instead of all other agents, are required to maintain their previously computed admissible command.

Assume that at time t the i-th agent receives from its neighbors the values of their states and of their previously applied commands, i.e. $[x]_i(t)$ and $[g]_i(t-1)$. If at time t all agents in $\mathcal{N}_i$ except the i-th were to hold the commands applied at time $t-1$, then the i-th agent could select a local command g_i satisfying constraints (2.30) by fulfilling the following inequalities

$$\hat{R}_i g_{i,\mathcal{N}_i} \leq \hat{v}_{i,\delta} - \hat{\tilde{R}}_i x_{i,\mathcal{N}_i}, \tag{2.34}$$

where $g_{i,\mathcal{N}_i}$ is set equal to $g_{i,\mathcal{N}_i}(t-1)$ for all entries except g_i. This idea can be extended to a larger number of agents using the following notion of *Turn*.

Definition 2.12 (**Turn**): A turn $\mathcal{T} \subset \mathcal{A}$ is a *subset of non-neighboring nodes*, i.e. $\forall i, j \in \mathcal{T}$ such that $i \neq j$, $j \notin \mathcal{N}_i$ (none of them is a neighbor of the others).

Proposition 2.13 *Let $\mathcal{T}_t \subset \mathcal{A}$ denote a turn selected at time t. Then, under the assumption that $Rg(0) + \tilde{R}x(0) \leq v$, if at each time t all agents not in $\mathcal{T}_t$ keep applying their previously applied commands, i.e. $g_i(t) = g_i(t-1), \forall i \notin \mathcal{T}_t$ and the agents in $i \in \mathcal{T}_t$ update their commands g_i accordingly to* (2.34)*, then the overall constraints* (2.25) *are never violated.*

Proof: It directly follows from the structures of set $\mathcal{V}(x)$ and constraints (2.34). □

At this point, given a sequence of *turns* $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$, and assuming that each agent knows to which turn it belongs to, the problem of locally determining at each time t the best $g_i(t)$ approximating $r_i(t)$ such that global constraints are satisfied can be solved by allowing only the agents in the current turn $\mathcal{T}_t$ to update their commands in accordance with constraints (2.34). This idea can be formalized as follows

Turn-Based CG (TB-CG) Algorithm - *Agent i*
REPEAT AT EACH TIME t
if $i \in \mathcal{T}_t$ **then**
  RECEIVE $x_j(t), g_j(t-1)$ FROM $\mathcal{N}_i$;
  BUILD $[x]_i(t), [g]_i(t-1)$;
  SOLVE

$$g_i(t) = \arg\min_{g_i} \|g_i - r_i(t)\|^2_{\Psi_i} \quad \text{SUBJECT TO (2.34).} \tag{2.35}$$

else
  SET $g_i(t) = g_i(t-1)$
  APPLY $g_i(t)$
  TRANSMIT $g_i(t)$ AND $x_i(t)$ TO THE NEIGHBORING AGENTS

where $\Psi_i = \Psi_i^T > 0, \forall i \in \mathcal{A}$. Please note that the proposed algorithm satisfies the constraints using only local data. In fact, each agent in the turn $\mathcal{T}_t$ only needs to know the constraints in which it is involved and the commands, and the states of its neighbors. This represent a major improvement with respect to previous sequential [68, 65] and parallel [83] schemes where global knowledge of the system dynamics was assumed and the overall system state had to be broadcasted.

2.4 Closed-loop Data-Driven Simulation

This section introduces relevant notation and results from behavioral system theory, useful for the subsequent data-driven command governor formulation. A stacked window of the sequence containing all of its components is denoted as

$$x_{[1,T]} = \begin{bmatrix} x_1, \cdots, x_T \end{bmatrix}^T,$$

and the related Hankel matrix is defined as

$$\mathcal{H}_L(x_{[1,T]}) = \begin{bmatrix} x_1 & x_2 & \cdots & x_{T-L+1} \\ x_2 & x_3 & \cdots & x_{T-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & \cdots & x_T \end{bmatrix}.$$

Moreover, we write x either for the sequence itself or for the stacked vector $x_{[1,T]}$. The vector $\mathbb{1}_L$ denotes a vector of all ones of length L. For a set $\mathcal{X}$, the interior is denoted by $\text{Int}(\mathcal{X})$. The Kronecker product is written as $\otimes$. The following standard definition of persistence of excitation is considered

Definition 2.14 (Persistently exciting of order L): A signal $\{x_k\}_{k=1}^T$ with $x_k \in \mathbb{R}^n$ is *persistently exciting of order* L if *rank*$(\mathcal{H}_L(x)) = nL$.

The behavioral framework [75] provides a suitable foundation for data-driven simulation and control problems. By treating dynamical systems as sets of trajectories, rather than equations, an explicit relationship between trajectories and their generating systems can be established. In this framework, the following definition of dynamical system is considered.

Definition 2.15 (Behavior): In the behavioral setting, a discrete-time dynamical system $\mathscr{B}$ with w manifest variables, that collect inputs and outputs of the system, is a subset of the signal space $(\mathbb{R}^w)^{\mathbb{N}}$.

Notice that $(\mathbb{R}^w)^{\mathbb{N}}$ denotes the set of functions from $\mathbb{N}$ to $\mathbb{R}^w$, i.e. $w \in (\mathbb{R}^w)^{\mathbb{N}}$ is the time series $w = (w(1), w(2), \dots, w(t), \dots)$, where $w(t) \in \mathbb{R}^w$. The finite sequence $w \in (\mathbb{R}^w)^T$ is $w = (w(1), w(2), \dots, w(t), \dots, w(T))$, with $w(t) \in \mathbb{R}^w$. However, with some abuse of notation, $w \in (\mathbb{R}^w)^T$ is viewed also as a wT-dimensional column vector.

In this context, a minimal kernel representation of the controller is considered. More in detail, given two polynomial matrices R_r and R_w that parameterize the backward shift operator σ, i.e. $\sigma x(t) := x(t-1)$, the following representation of $\mathscr{C}$ is considered

$$\mathscr{C} = \{\begin{bmatrix} g \\ w \end{bmatrix} | R_r(\sigma)g + R_w(\sigma)w = 0\}, \tag{2.36}$$

where $g \in (\mathbb{R}^r)^{\mathbb{N}}$. The system given by the feedback interconnection of the plant $\mathscr{B} \subseteq (\mathbb{R}^w)^{\mathbb{N}}$ and the controller $\mathscr{C} \subseteq (\mathbb{R}^r)^{\mathbb{N}}$ is denoted as $\mathscr{B}^{\mathscr{C}}$, with $\mathscr{B}^{\mathscr{C}} = \mathscr{B}^{\text{ext}} \cap \mathscr{C}$ and

$$\mathscr{B}^{\text{ext}} = \{\begin{bmatrix} g \\ w \end{bmatrix} \in (\mathbb{R}^{r+w})^{\mathbb{N}} | w \in \mathscr{B}\}.$$

In general, to obtain a representation of the controller, a relationship between the input-output sequence w and the reference sequence g has to be established, i.e. a relationship between the temporal sequences $w_{[1,L]}$ and $g_{[1,L]}$. More in detail, if a matrix transfer function $C(s)$ linking the input

$U(s)$ to the error signal $E(s)$ is available, ARMA models linking the i-th input to the j-th error can be derived and, in turn, first the temporal sequences $w_{[1,L]}$ and $g_{[1,L]}$ and then subsequently the desired matrices R_r and R_w. In general, the ARMA model linking the i-th input to the j-th error

$$u_i(t) = \alpha_1^i u_i(t-1) + \cdots + \alpha_p^i u_i(t-p) + \beta_1^j e_j(t) + \cdots + \beta_q^j e_j(t-q), \quad (2.37)$$

is relative to the transfer function $C_{ij}(s)$ of the matrix transfer function $C(s)$.

2.4.1 Problem Formulation

The closed-loop data-driven simulation refers to the problem of finding the set of all responses of a closed-loop system to a given reference signal directly from an input/output trajectory of the plant and a representation of the controller [84]. Formally

Problem 2.16 *Given a trajectory $w_d = (w_d(1), \ldots, w_d(T)) \in (\mathbb{R}^w)^T$ of a linear time-invariant system $\mathscr{B} \subset (\mathbb{R}^w)^{\mathbb{N}}$, with a given input/output partition $w = [u \quad y]^T \in \mathscr{B}$, with input $u \in (\mathbb{R}^m)^{\mathbb{N}}$ and output $y \in (\mathbb{R}^p)^{\mathbb{N}}$, and*

- *A linear time-invariant controller $\mathscr{C} \subset (\mathbb{R}^c)^{\mathbb{N}}$, with $c = r + p + m$, with input/output partition $[r \quad w]^T$, with reference signal $r \in (\mathbb{R}^r)^{\mathbb{N}}$;*
- *A reference signal $g = [g(1), \ldots, g(T_r)]^T \in (\mathbb{R}^r)^{T_r}$;*

Find the set of responses w_g of the closed-loop system $\mathscr{B}^{\mathscr{C}}$ to the reference signal g.

2.4.2 Closed-loop data-driven solution

To solve Problem 2.16, a similar approach used in the deterministic subspace identification algorithms is followed. More in detail, in the theoretical analysis of the method, the following assumption is made

A3. The data is assumed exact, i.e. noise free.

Moreover, it is assumed that

A4. System $\mathscr{B}$ is controllable.
A5. The input component u_d of w_d is persistently exciting of order T_r plus the order of $\mathscr{B}$.

Under assumption ***A4***, system $\mathscr{B}$ admits a minimal image representation, i.e.

$$\mathscr{B} = \{w = M(\sigma)l | l \in (\mathbb{R}^m)^{\mathbb{N}}\}. \quad (2.38)$$

However, only the controller is explicitly specified by a kernel representation, while the plant $\mathscr{B}$ is implicitly defined by the trajectory w_d. To avoid the use of an explicit representation, the following Lemma is useful.

Lemma 2.17 *(**Willem's Theorem [85, Theorem 3]**): Suppose $w^d_{[1,T]}$ is a trajectory of a controllable system $\mathscr{B}$, where the input $u^d_{[1,T]}$ of $w^d_{[1,T]}$ is persistently exciting of order $mL+n$. Then, a trajectory $\{w_{k,g}\}_{k=1}^{L}$ is a trajectory of system $\mathscr{B}$ if and only if there exists $\alpha \in \mathbb{R}^{T-L+1}$ such that*

$$w_g = \mathcal{H}_L(w^d_{[1,T]})\alpha. \tag{2.39}$$

The vector α is related to the input and initial conditions of the system that generate the trajectory w_g (where subscript g denotes the closed-loop response to reference signal g). The equivalence of (2.38) and (2.39) holds under assumptions ***A4*-*A5*** [86, 87].

In order to simplify the presentation and abstract from technical details, if the system is assumed single-output then the difference operator $R(\sigma)$ in (2.36) by the structured matrix $\mathcal{F}_{T_r}(R)$, defined below

Definition 2.18 (Banded upper-triangular Toeplitz matrix): The banded upper-triangular Toeplitz matrix $\mathcal{F}_L(r)$ with L block-rows, related to the polynomial $r(z) = r_0 + r_1 z + \cdots + r_{n_r} z^{n_r}$, has form

$$\mathcal{F}_L(r) := \begin{pmatrix} r_0 & r_1 & \cdots & r_{n_r} & 0 & \cdots & 0 \\ 0 & r_0 & r_1 & \cdots & r_{n_r} & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & r_0 & r_1 & \cdots & r_{n_r} \end{pmatrix}. \tag{2.40}$$

Therefore, under assumptions ***A4*-*A5***, the set of solutions of the linear system of equations

$$\begin{aligned} w_g &= \mathcal{H}_{T_r}(w^d_{[1,T]})\alpha, \\ \mathcal{F}_{T_r}(R_w)w_g &= -\mathcal{F}_{T_r}(R_r)g, \end{aligned} \tag{2.41}$$

is equal to the set of trajectories w_g solving the closed-loop data-driven simulation problem. The vector α is related to the input and initial conditions of the system that generate the trajectory w_g. Notice that the mapping from α to the input of the system and initial conditions is not injective, i.e. a solution α of (2.39) does not need to be unique. Furthermore, please notice that (2.41) computes the set of all possible responses to a given reference. However, a single response can be selected by means of adding a condition on α that has to satisfy (2.39) with a given initial trajectory.

Chapter 3
Cooperation-Inducing Distributed Command Governor Schemes

This chapter addresses the connectivity preserving coordination problem for a group of dynamically decoupled agents operating in a 2D environment that are subject to both collision and obstacle avoidance constraints. The approach is based on distributed command governor concepts, which are here expanded to obtain motion coordination of vehicles and to enforce a minimum spanning tree for safe communication between the vehicles. The edges of this tree may be adjusted in real-time to enhance overall performance. Additionally, the on-line distributed determination of particular separation hyper-planes is employed to reduce the inherent non-convexity of obstacle avoidance constraints and a solution is provided to induce a certain degree of cooperation between agents. The resulting scheme, based on solving on-line a linear constrained optimization problem involving a mixture of continuous and integer variables, is proved to fulfill the constraints, as well as feasibility requirements. In order to assist the implementation in actual vehicles, a practical implementation of the control architecture on the Robot Operating System (ROS) is discussed. Simulation results, across various scenarios, demonstrate the effectiveness of the proposed supervision framework.

3.1 Introduction

Connectivity maintenance and safe navigation around obstacles are fundamental challenges for vehicles formation [28]. In fact, when a vehicle follows a trajectory, a smooth motion is desirable while ensuring any potential collision avoidance with the obstacles located in the same environment. Enhancing the safety and autonomy of USVs requires consideration of obstacle avoidance functionality, yet existing research has predominantly focused on inter-vehicle collision avoidance, commonly neglecting other potential environmental obstacles [8]. Traditional approaches [29], [30] usually envisage two operative phases: in the former, a path is determined by a basic refer-

ence planner, while in the second one, a safe robot trajectory complying with the vehicle dynamics is computed around the reference path using trajectory optimization [31]. Early works in this respect explored a variety of methods to enhance safe navigation os autonomous systems, with a focus on static obstacles. These approaches included path planning [32], decision trees [33], graph techniques [34], heuristic search techniques [35], potential field techniques [36] and barrier functions approaches [88]. A more comprehensive survey can be found in [37] and its references, while in-depth discussion of single USV guidance can be found in [8]. However, such methods often suffer from significant replanning cycles when *a priori* unknown obstacles are present in the working space and formation keeping prescriptions are introduced among the vehicles for accomplishing more complicated tasks [38], [39].

More recently, advances in distributed predictive control of dynamic linear/non-linear systems with input and/or state constraints have led to the development of more sophisticated, effective and attractive solutions for path-planning problems, particularly in complex scenarios. These approaches often recast problem requirements, such as safety constraints and collision avoidance, as pointwise-in-time set-membership state constraints involving variables of one or more vehicles in the formation. Examples of applications include proximity networks [48], coverage [51], partitioning [49], and flocking [50]. In particular, predictive control-based schemes [51]–[54] offer a promising approach for guiding vehicles towards their desired trajectories while explicitly considering constraints.

Motivated by these observations and considering the latest advances in the field of constrained distributed predictive control, this chapter aims at providing a novel connectivity-maintaining distributed supervision architecture for a set of surface vehicles with collision and obstacle avoidance requirements. In further detail, at the lower level, the architecture implements a non-iterative non-cooperative distributed *Turn-Based Command Governor* (CG) algorithm [64], where agents not sharing the same constraints are grouped into particular sets and update their control actions in parallel on the basis of local information, while the higher level extends such *Turn-Based CG* solution to adequately deal with connectivity keeping, collision avoidance and obstacle avoidance constraints. More in detail, in order to enhance the system performance associated to reference tracking, following ideas in [89], the communication network is modeled as a weighted time-varying graph and the algorithm enforces the existence of a specific spanning tree amongst all agents at each time in order to guarantee communication connectivity. Whenever more spanning trees are available, the scheme is instructed to select the minimum one with respect to the Euclidean distance among the agent positions. Anyway, the main differences with respect to the approach of [89] regard: *i*) the presentation of a novel algorithm for the minimum spanning tree computation that allows the supervision scheme to instantly and safely switch on the modified constraints structure arising from the new determined minimum spanning tree (please notice that in [89], such a switch is allowed

only after a certain dwell time); *ii*) the introduction of collision avoidance requirements between vehicles due to the fact that, during maneuver, vehicles trajectories may intersect. To this end and due to the non-convex nature of this type of constraint, the same procedure, presented in [72], is used to obtain linear constraints depending on continuous and integer variables; *iii*) the presence of "static obstacles" that makes the problem much more complicated. In this respect, the obstacle avoidance functionality introduced in the proposed approach makes use of a novel algorithm that exploits the *Hyperplane Separation Theorem* [90]. The latter has been traditionally used in robotics to generate collision free trajectories [91, 92]. However, if during the distributed computational process of the hyperplanes, no hierarchy is imposed among the agents, then this could lead under certain conditions, to the activation of the proximity constraints and the stop of the formation during the obstacle avoidance maneuver. By means of exploiting the presence of the above mentioned spanning tree, a solution is proposed to induce self-coordination behavior that translates into arbitrary circumnavigation maneuvers where the agents follow the same directions and are able to safely move toward the prescribed way-points.

Finally, beyond the theoretical contribution, the architectural aspects of a full implementation of a Guidance, Navigation, and Control (GNC) architecture that smoothly combines with the proposed distributed supervision scheme into the Robot Operating System (ROS) environment [76, 77] are described.

The chapter is organized as follows. Section 3.2 introduces the problem at hand. Section 3.3 describes the proposed distributed supervision strategy and its main properties. To enhance real-world portability, in Section 3.4 a ROS-based GNC architecture is described. Matlab-based and Gazebo-based simulations involving a group of agents accomplishing several missions are also presented in Section 3.5 as to showcase the benefits achievable by this method. Finally, Section 3.6 highlights the main conclusions and some future research lines.

3.2 Problem Formulation

Consider a set of N agents, where the i-th subsystem is modeled by the following LTI discrete-time model

$$\Sigma_i : \begin{cases} x_i(t+1) = \Phi x_i(t) + G g_i(t), \\ p_i(t) \;\; = H^p x_i(t) + D^p g_i(t), \\ c_i(t) \;\; = H_i^c x_i(t) + L_i g_i(t), \end{cases} \tag{3.1}$$

where: $t \in \mathbb{Z}_+$, $x_i \in \mathbb{R}^{n_i}$ is the state vector (which includes the controller states under dynamic regulation), $g_i \in \mathbb{R}^{m_i}$ the manipulable reference vector

useful to track the local nominal reference $r_i \in \mathbb{R}^{m_i}$ with $m_i = 2$ and matrices $\Phi \in \mathbb{R}^{n_i \times n_i}$, $G \in \mathbb{R}^{n_i \times m_i}$ and $D^p \in \mathbb{R}^{2 \times m_i}$ and

$$H^p := \begin{bmatrix} 1\ 0\ 0 \cdots 0 \\ 0\ 1\ 0 \cdots 0 \end{bmatrix},\ H^p \in \mathbb{R}^{2 \times n_i}$$

considered the same for all the agents. Notice that H^p is used to extract the first components of the state. In fact, the state can be decomposed as $x_i(t) = [p_i^T(t), \sigma_i^T(t)]^T$, where $p_i^T(t) = [p_i^x(t), p_i^y(t)]^T$ represents the position of each agent i in the earth-fixed frame and $\sigma_i(t)$ includes the additional state components, such as velocities and controller states. The vector $c_i \in \mathbb{R}^{z_i}$ collects all those variables that necessitate constraint enforcement within a local scope. Furthermore, in line with the assumptions introduced in Section 2.2.1, it is assumed that each subsystem represents a closed-loop dynamic regulated by a decentralized local controller that guarantees: *i*) Φ is a strictly Schur matrix and *ii*) Σ_i enjoys the offset-free property (i.e. $H^p(I_n - \Phi)^{-1}G = I_2$). Considering that the following constraints are relevant in many applications, each subsystem is subject to: *Local*, *Obstacle and Collision Avoidance* and *Connectivity Keeping* constraints.

Local Constraints

Effective supervision of vehicles may require enforcing certain limitations, such as speed and saturated input. This necessitates the introduction of *local constraints*, that are typically expressed as follows

$$c_i(t) \in \mathcal{C}_i, \quad \forall t \in \mathbb{Z}_+, \tag{3.2}$$

where $\mathcal{C}_i$ is a compact and convex set, for all $i = 1, \ldots, N$.

Connectivity Keeping

Agents are assumed to exchange information through a communication network that is modeled via a *connected*, *undirected* and *weighted* graph $\mathcal{G}_C(t) = (\mathcal{A}, \mathcal{E}_C(t), \mathcal{W}_C(t))$, with vertex set $\mathcal{A} = \{1, \ldots, N\}$ and edge set $\mathcal{E}_C(t) \subseteq \mathcal{A} \times \mathcal{A}$. If direct communication between agents i and j is possible at a specific time instance t, then $(i, j) \in \mathcal{E}_C(t)$. It is important to note that this direct communication capability is possible when their relative Euclidean distance is smaller than a predefined radius R_{co}, i.e.

$$\|p_i(t) - p_j(t)\| \leq R_{co} \Leftrightarrow (i, j) \in \mathcal{E}_C(t). \tag{3.3}$$

Therefore, at edge $(i,j) \in \mathcal{E}_C(t)$ has been assigned a weight $w_{ij}(t) := \|p_i(t) - p_j(t)\|$, that will be used for enforcing the following proximity constraints

$$w_{ij}(t) \leq R_{co}, \tag{3.4}$$

here introduced to preserve the connectivity of the graph $\mathcal{G}_C(t)$ at each time instant t and improve system performance. Please notice that, following Definition 2.7, graph connectivity is ensured if there exists a path between every two vertices of the graph.

Collision Avoidance

Because vehicles may operate in restricted areas and their trajectories may intersect during their actions, adding collision avoidance constraints becomes crucial. Specifically, each agent i is subject to the following interactive nonlinear and nonconvex constraints

$$\|p_i(t) - p_j(t)\|_\infty \geq D_{ca}, \ \forall j \in \mathcal{A} \setminus \{i\}, \tag{3.5}$$

where D_{ca} represents the minimum distance vehicles have to keep in order to avoid collisions between themselves.

Obstacle Avoidance

Definition 3.1 (**Family of Static Obstacles**) Let a family of fixed obstacles be defined as $\mathcal{H} = \{\mathcal{H}_1, \ldots, \mathcal{H}_D\}$ where each obstacle $\mathcal{H}_j$ is represented by the convex hull defined by its vertices $V_j = \{v_1^j, \ldots, v_{M_j}^j\}$, with $v_k^j \in \mathbb{R}^2, \forall\, k \in \{1, \ldots, M_j\}$.

In order to comply with obstacle avoidance constraints, the following condition has to be satisfied

$$p_i(t) \notin \bigcup_{j=1,\ldots,D} \mathcal{H}_j, \quad \forall t \in \mathbb{Z}_+, \quad \forall i \in \mathcal{A}. \tag{3.6}$$

To handle obstacle avoidance constraints, the following Assumption is required

Assumption 1. *(**Vision Module**) Each agent i is assumed to be equipped with a* Vision Module *(i.e. a LiDaR, stereo camera), capable of identifying at each time instant a collection of points belonging to the visible borders of the obstacles. Furthermore, it is assumed that each vehicle is able to compute the*

distance between the vehicle itself and the convex hull $V_{i,seen}(t) \subseteq \bigcup_{j=1}^{D} V_j$ *of such a collection.*

Problem Statement

The problem to be solved in the next sections can be stated as follows.

Problem 3.2 *Given systems* (3.1)*, locally determine, at each time instant* t *and for each agent* $i \in \mathcal{A}$*, a suitable reference signal* $g_i(t)$ *that is the "best approximation" of* $r_i(t)$ *(according to a performance index specified below), such that its application complies with conditions (3.2), (3.5), (3.6), preserves the connectivity of the communication graph and ensures overall formation compactness.*

3.3 Main Results

In this section, following the lines presented in Section 2.2.3, some definitions are introduced to recast constraints of Problem (3.2) in a form that is more suitable to be tackled by the proposed distributed Turn-Based CG supervision strategy. More in detail, let

$$x_{g_i} := (I - \Phi)^{-1} G g_i, \tag{3.7}$$

represent the steady-state of the i-th system for a constant g_i [97], while

$$x_i(k, x_i, g_i) := \Phi^k x_i + \sum_{\tau=0}^{k-1} \Phi^{k-\tau-1} G g_i + L g_i \tag{3.8}$$

is the virtual prediction of the state of system i along the virtual time k, when a constant command sequence $g_i(k) \equiv g_i$ is applied to Σ_i starting from the initial state x_i.

Then, leveraging the concept of *Output Admissible Set* (*OAS*) presented in (2.17) and associated with the set $\mathcal{C}_i$ in (3.2), the following set is formally defined as

$$\mathcal{Z}_i^l := \left\{ \begin{bmatrix} x_i \\ g_i \end{bmatrix} \in \mathbb{R}^{n+m} \,\middle|\, \begin{array}{l} c_i^{g_i} \in \mathcal{C}_i \sim \mathcal{B}_\delta, \\ c_i(k, x_i, g_i) \in \mathcal{C}_i,\ \forall k \in \mathbb{Z}_+ \end{array} \right\}, \tag{3.9}$$

where $c_i(k, x, g_i) := H_c^i x_i(k, x, g_i)$, $c_i^{g_i} := H_c^i x_{g_i}$ and $\mathcal{B}_\delta := \{x \in \mathbb{R}^n : \|x\| \leq \delta\}$ is a generic ball in an Euclidean space $\mathbb{R}^n$ and $\delta > 0$. It can be proved [79] that local constraints (5.2) are always satisfied if

$$(x_i(t), g_i(t)) \in \mathcal{Z}_i^l, \forall t \in \mathbb{Z}_+. \tag{3.10}$$

As far as connectivity keeping constraints (3.4) are concerned, one approach to guarantee the connectedness of $\mathcal{G}_C(t)$ is to ensure the existence of a spanning tree, $\mathcal{G}_I(t) = (\mathcal{A}, \mathcal{E}_I(t), \mathcal{W}_I(t))$, where $\mathcal{E}_I(t) \subseteq \mathcal{E}_C(t)$ and $\mathcal{W}_I(t) \subseteq \mathcal{W}_C(t)$, from time t onwards. For this purpose, the following set is introduced

$$\mathcal{Z}^{ck} := \left\{ \begin{bmatrix} x \\ y \\ g \\ w \end{bmatrix} \in \mathbb{R}^{2(n+m)} \;\middle|\; \begin{array}{r} \|p^g - p^w\| \leq R_{co} - \delta \\ \|p(k,x,g) - p(k,y,w)\| \\ \leq R_{co}, \\ \forall k \in \mathbb{Z}_+, \end{array} \right\}, \tag{3.11}$$

where $p(k,x,g) := H^p x(k,x,g)$ and $p^g := H^p x_g$. Then, to guarantee the existence of $\mathcal{G}_I(t)$ $\forall t \in \mathbb{Z}_+$, the following relations must hold

$$[x_i(t),\, x_j(t),\, g_i(t),\, g_j(t)]^T \in \mathcal{Z}^{ck}, \quad \forall\, (i,j) \in \mathcal{E}_I(t). \tag{3.12}$$

It is important to note that the graph $\mathcal{G}_I(t)$, and consequently its edge set $\mathcal{E}_I(t)$, can change over time. This time-varying characteristic has significant implications for the supervision strategy, which will be discussed in detail in the next section.

Regarding *Collision Avoidance* constraints (3.5), it is useful to introduce the following set

$$\mathcal{Z}^{ca} := \left\{ \begin{bmatrix} x \\ y \\ g \\ w \end{bmatrix} \in \mathbb{R}^{2(n+m)} \;\middle|\; \begin{array}{r} \|p^g - p^w\|_\infty \geq D_{ca} + \delta, \\ \|p(k,x,g) - p(k,y,w)\|_\infty \\ \geq D_{ca}, \\ \forall k \in \mathbb{Z}_+ \,. \end{array} \right\}. \tag{3.13}$$

The non-convex constraints in the set $\mathcal{Z}^{ca}$, for the whole formation, can be reformulated into linear constraints depending on continuous and integer variables [94, 72]

$$\begin{array}{rrl} \forall i,j | i > j: & p_i^x - p_j^x & \geq D_{ca} - \mu b_{ij}^1 \\ & \text{and } p_i^y - p_j^y & \geq D_{ca} - \mu b_{ij}^2 \\ & \text{and } p_j^x - p_i^x & \geq D_{ca} - \mu b_{ij}^3 \\ & \text{and } p_j^y - p_i^y & \geq D_{ca} - \mu b_{ij}^4 \\ & \text{and } \sum_{k=1}^4 b_{ij}^k & \leq 4 - 1, \end{array} \tag{3.14}$$

where $b_{ij}^k \in \{0,1\}$ represents a binary variable and μ a sufficiently large scalar [95]. Then, in order to ensure collision avoidance, the following relations have to be satisfied $\forall i \in N$ and $\forall t \in \mathbb{Z}_+$

$$[x_i(t),\, x_j(t),\, g_i(t),\, g_j(t)]^T \in \mathcal{Z}^{ca}, \quad \forall j \in \mathcal{A} \setminus \{i\}. \tag{3.15}$$

These types of constraints are typically non-convex and therefore are difficult to treat. To overcome this drawback, an approach similar to the one proposed in [91] has been used. Each vehicle is instructed to stay within a safe convex region that is determined by resorting to the following *Hyperplane Separation Theorem* [90].

Definition 3.3 (**Hyperplane**) Let $n_{\mathcal{P}} \in \mathbb{R}^m$, and $d_{\mathcal{P}} \in \mathbb{R}$ be a nonzero constant vector and a constant scalar respectively. Then, the set $\mathcal{P}(n_{\mathcal{P}}, d_{\mathcal{P}}) := \{y \in \mathbb{R}^m : y^T n_{\mathcal{P}} = d_{\mathcal{P}}\}$ is a subspace of $\mathbb{R}^m$ denoted as *hyperplane* where $n_{\mathcal{P}}$ represents its normal vector and $d_{\mathcal{P}}$ its distance from the space origin.

Lemma 3.4 *(**Hyperplane Separation Theorem**) Let $V_A, V_B \subset \mathbb{R}^n$ be two disjoint nonempty closed convex sets, one of which is compact. Then, there exists an hyperplane $\mathcal{P}(n_{\mathcal{P}}, d_{\mathcal{P}}) \subset \mathbb{R}^n$, such that*

$$\begin{aligned} x^T n_{\mathcal{P}} &> d_{\mathcal{P}}, \quad \forall\, x \in V_A, \\ y^T n_{\mathcal{P}} &< d_{\mathcal{P}}, \quad \forall\, y \in V_B. \end{aligned}$$

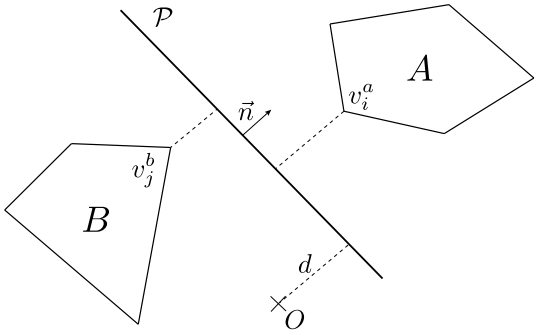


Fig. 3.1 Graphical representation of the Separating Hyperplane.

It is possible to compute the separation hyperplane $\mathcal{P}$, defined above through its normal vector $n_{\mathcal{P}}$ and its orthogonal distance $d_{\mathcal{P}}$ with respect to the origin of the reference system, by solving an optimization problem having the following form

$$\begin{aligned} &\min_{n_{\mathcal{P}}, d_{\mathcal{P}}, \epsilon_{\mathcal{P}}} \ \epsilon_{\mathcal{P}} \\ &\text{s. t.} \quad \begin{cases} x^T n_{\mathcal{P}} + \epsilon_{\mathcal{P}} \geq d, & \forall x \in V_A, \\ y^T n_{\mathcal{P}} - \epsilon_{\mathcal{P}} \leq d + d_{safe}^{ob}, & \forall y \in V_B, \\ \|n_{\mathcal{P}}\| = 1, \end{cases} \end{aligned} \tag{3.16}$$

where $\epsilon_{\mathcal{P}}$ represents a slack variable and $d_{safe}^{ob} > 0$ represents the distance to kept between the computed separating hyperplane and points of the obstacle.

Following this logic, it is possible to compute a separation hyperplane $\mathcal{P}_i$ for agent i where the two objects V_A and V_B are respectively the agent itself and the obstacle. Once $\mathcal{P}_i(n_{\mathcal{P}_i}(\bar{t}), d_{\mathcal{P}_i}(\bar{t}))$ has been found that separates the i-th vehicle from the closest obstacle $\mathcal{H}_{\bar{t}}$ at time $t = \bar{t}$, the convex safe region for agent i is represented by all positions p_i such that

$$p_i^T n_{\mathcal{P}_i}(\bar{t}) \geq d_{\mathcal{P}_i}(\bar{t}), \quad \forall t \geq \bar{t}. \tag{3.17}$$

Following the CG philosophy and in order to ensure that (3.17) holds true, the following additional constraints on the admissible reference $g_i(t)$ need to be satisfied

$$(x_i(t), g_i(t)) \in \mathcal{Z}_i^{oa}(\mathcal{P}_i), \quad \forall t \in \mathbb{Z}_+, \tag{3.18}$$

where

$$\mathcal{Z}_i^{oa}(\mathcal{P}_i) := \left\{ \begin{bmatrix} x_i \\ g_i \end{bmatrix} \in \mathbb{R}^{n_i+m_i} \;\middle|\; \begin{array}{r} p_i(k, x_i, g_i)^T n_{\mathcal{P}_i} \geq d_{\mathcal{P}_i}, \, \forall k \in \mathbb{Z}_+ \\ g_i^T n_{\mathcal{P}_i} \geq d_{\mathcal{P}_i} + \delta. \end{array} \right\}. \tag{3.19}$$

Please note that the pair $(n_{\mathcal{P}_i}, d_{\mathcal{P}_i})$, that represents the hyperplane $\mathcal{P}_i$, characterizes the set $\mathcal{Z}_i^{oa}$.

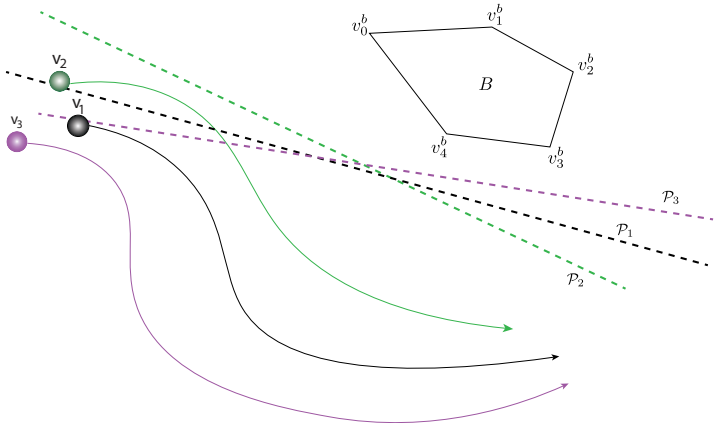


Fig. 3.2 Obstacle avoidance using the hyperplanes approach. Each agent i computes its own hyperplane $\mathcal{P}_i$.

In Figure 3.2 the basic idea behind this approach is presented. Each vehicle i identifies its own hyperplane $\mathcal{P}_i$ to avoid the obstacle B.

A procedure for the computation of the *admissibility index* k_0 ensuring the finite determinateness of the Output Admissible Sets (3.9), (3.11), (3.5) and (3.19) is reported in the Appendix A.

To tackle Problem (3.2), the Supervision Layered Architecture depicted in Figure 3.3 is introduced. Please observe that each layer of the strategy

serves the layers below and works within the same communication network. The two lower-layers, namely the *Control Level* and the *Physical Level*, represent the physical pre-compensated plant Σ_i in (3.1). The *Supervision Level*

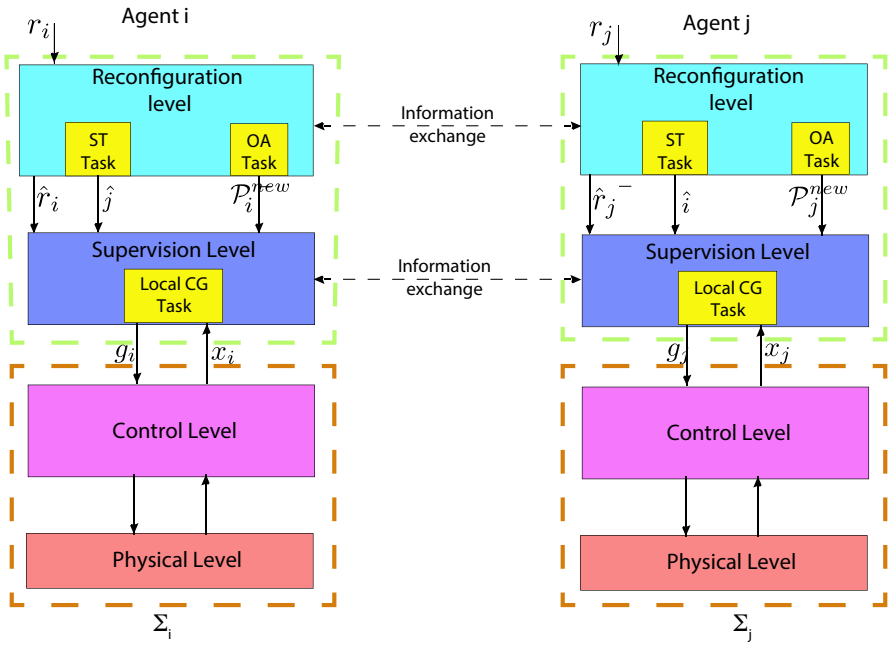


Fig. 3.3 Multi-layer Supervision Architecture

implements the local CG task while the *Reconfiguration Level* is in charge of running both the Spanning Tree (ST) task for connectivity keeping and the Obstacle Avoidance (OA) task, which is in charge of localizing and computing the distance from an obstacle. Please notice that the *Supervision Level* and the *Reconfiguration Level* are connected by horizontal links that represent the requirement for them to communicate with the same layer of all other agents. The next sections describe in more detail each component of the proposed Architecture.

3.3.1 Distributed Turn-Based Command Governor

As mentioned in the introduction, the Turn-Based Distributed CG [64], presented in Section 2.3.2.2, is exploited. This scheme relies on partitioning the set of agents into subsets denoted by $\{\mathcal{T}_1, \ldots, \mathcal{T}_{\mathcal{X}}\}$, also referred to as turns

(Definition 2.12). The determination of a set of turns is closely related to the well-known problem of graph coloring [93].

Let us assume that at each time step t, each agent i knows all its adjacent nodes and thus the time dependent set $\tilde{\mathcal{N}}_i(t) \subseteq \mathcal{N}_i$ of neighbors related to $\mathcal{G}_I(t)$. Thus, the aggregate state and aggregate command of all agents in $\tilde{\mathcal{N}}_i(t)$ are denoted by $x_{\tilde{\mathcal{N}}_i}(t)$ and $g_{\tilde{\mathcal{N}}_i}(t-1)$, respectively. Then, if a certain turn $\mathcal{T}_k(t)$ is in charge of updating at time t, all agents $i \notin \mathcal{T}_k(t)$ are instructed to *freeze* their previous applied commands, i.e. $g_i(t) = g_i(t-1)$. This allows one to formulate the following strategy

Algorithm 1: The Turn-Based CG (TB-CG)

(AGENT i) REPEAT AT EACH TIME t

1: **if** $i \in \mathcal{T}_k(t)$ **then**
2: RECEIVE $\tilde{x}_j(t), \tilde{g}_j(t-1)$ FROM NEIGHBORS $j \in \tilde{\mathcal{N}}_i(t)$;
3: RECEIVE COMPUTED $\mathcal{P}_i^{new}(t) = \left(n_{\mathcal{P}_i}^{new}(t), d_{\mathcal{P}_i}^{new}(t)\right)$ FROM THE *Reconfiguration Level*;
4: **if** $(x_i(t), g_i(t-1)) \in \mathcal{Z}_i^{oa}(\mathcal{P}_i^{new})$ **then**
5: SET $\mathcal{P}_i = \mathcal{P}_i^{new}$
6: SOLVE

$$
\begin{aligned}
& g_i(t) := \arg\min_{g_i} \|g_i - \hat{r}_i(t)\|_{\Psi_i}^2 \\
& \text{s.t.} \begin{cases} [x_i(t),\, x_j(t),\, g_i,\, g_j(t-1)]^T \in \mathcal{Z}^{ck},\ \forall j \in \tilde{\mathcal{N}}_i(t) \\ [x_i(t),\, x_j(t),\, g_i(t),\, g_j(t-1)]^T \in \mathcal{Z}^{ca},\ \forall j \in \mathcal{A} \setminus \{i\} \\ [x_i(t), g_i]^T \in \mathcal{Z}_i^{oa}(\mathcal{P}_i) \cap \mathcal{Z}_i^{l}. \end{cases}
\end{aligned} \tag{3.20}
$$

7: **else**
8: SET $g_i(t) := g_i(t-1)$
9: APPLY $g_i(t)$
10: TRANSMIT $g_i(t)$ AND $x_i(t)$ TO THE NEIGHBORING AGENTS

where $\Psi_i = \Psi_i^T > 0, \forall i \in \mathcal{A}$. Building upon the in-depth analysis of the properties of the algorithm presented in [64], the following sections will explore how the introduction of the higher-level strategic layer modifies the constraints in (3.20) to guarantee that the algorithm effectively addresses, connectivity keeping, obstacle and collision avoidance requirements.

3.3.2 OA Task within the Reconfiguration Level

In principle, multiple distributed coordination approaches exist for determining a set of separating hyperplanes $hy = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ to avoid obstacles. However, the multi-agent approach implemented in Algorithm 1 within the *Supervision Level* logic, which does not prescribe any hierarchy among the deciders, could lead, under certain conditions, to the activation of the proximity constraints and the stop of the formation. Though it may seem restrictive,

activating proximity constraints serves as a crucial safety feature. However, leader-follower formations might benefit from a hierarchical network structure to impose formation compactness, which is frequently a desired feature. In this scenario, the *OA Task* exploits the presence of the spanning tree $\mathcal{G}_I(t)$ and it is carried out by the root agent of $\mathcal{G}_I(t)$, here denoted as $i_r(t)$ acting as the leader agent. Specifically, $i_r(t)$ determines the nearest agent with respect $V_{i,seen}(t)$ at time instant t, denoted hereafter as $\hat{i}(t) := \arg\min_{i\in\mathcal{A}} d_i(t)$ after receiving from each agent i the current distance $d_i(t)$ defined as

$$d_i(t) = \|p_i(t) - p^*_{OA}(t)\|, \tag{3.21}$$

where $p^*_{OA}(t)$ is determined by solving the following QP optimization problem

$$\begin{aligned} p^*_{OA}(t) = \text{argmin} \quad & \|p_{OA} - p_i(t)\|^2 \\ \text{s. t.} \quad & p_{OA} \in V_{i,seen}(t). \end{aligned} \tag{3.22}$$

Then, the nearest agent, $\hat{i}(t)$, is tasked with finding the separation hyperplane $\mathcal{P}_{\hat{i}}$ by solving the following optimization problem

$$\min_{n_{\mathcal{P}_i}, d_{\mathcal{P}_i}, \varepsilon_{\mathcal{P},i}} \varepsilon_{\mathcal{P},i}$$

$$\text{s. t.} \quad \begin{cases} v^T n_{\mathcal{P}_i} \le d_{\mathcal{P}_i}, & \forall v \in V_{i,seen}(t), \\ p_i^T n_{\mathcal{P}_i} \ge d_{\mathcal{P}_i} + d_{safe}, \\ g_i^T n_{\mathcal{P}_i} \ge d_{\mathcal{P}_i} + d_{safe}, \\ r_i^T n_{\mathcal{P}_i} \ge d_{\mathcal{P}_i} + d_{safe} - \varepsilon_{\mathcal{P},i}, \\ \|n_{\mathcal{P}_i}\| = 1, \end{cases} \tag{3.23}$$

where d_{safe} is a scalar chosen to satisfy

$$d_{safe} > R_{co}. \tag{3.24}$$

Subsequently, it finds a point $\hat{r}_i$ belonging to $\mathcal{P}_{\hat{i}}$ that is separated from $V_{i,seen}(t)$ by a second hyperplane, as shown in Figure 3.4, say it $\hat{\mathcal{P}}_{\hat{i}}$, i.e.

$$\min_{n_{\hat{\mathcal{P}}_i}, d_{\hat{\mathcal{P}}_i}, \hat{r}_i} \|r_i - \hat{r}_i\|_2^2$$

$$\text{s. t.} \quad \begin{cases} v^T n_{\hat{\mathcal{P}}_i} \le d_{\hat{\mathcal{P}}_i}, & \forall v \in V_{i,seen}(t), \\ r_i^T n_{\hat{\mathcal{P}}_i} \ge d_{\hat{\mathcal{P}}_i} + d_{safe}, \\ \hat{r}_i^T n_{\hat{\mathcal{P}}_i} \ge d_{\hat{\mathcal{P}}_i} + d_{safe} + \varepsilon, \\ \|n_{\hat{\mathcal{P}}_i}\| = 1. \end{cases} \tag{3.25}$$

Please notice that the latter can be convexified by resorting to a simple binary search. Once (3.25) is solved, it conveys $\mathcal{P}_{\hat{i}}$ and $\hat{r}_i$ back to the root agent $i_r(t)$ that spreads them throughout $\mathcal{G}_I(t)$ along with the sign of the

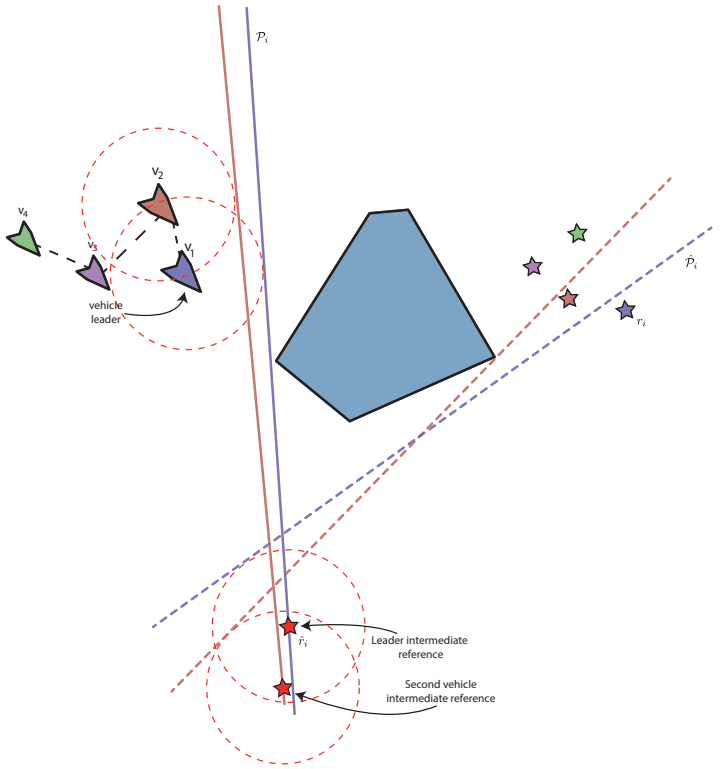


Fig. 3.4 Graphic representation of the computation for $\hat{r}_i$.

slope of $\mathcal{P}_{\hat{i}}$, i.e. $\frac{n_{\mathcal{P}_{\hat{i},1}}}{n_{\mathcal{P}_{\hat{i},2}}}$, with the aim of "imposing" a common direction on the entire formation, facilitating, for example, obstacle circumnavigation. Finally, each agent within the formation solves problem (3.23) subject to the following additional constraints

$$sign(\frac{n_{\mathcal{P}_{\hat{i},1}}}{n_{\mathcal{P}_{\hat{i},2}}})\, n_{\mathcal{P}_{i,1}} \geq 0, \tag{3.26}$$

$$n_{\mathcal{P}_{i,2}} > 0, \tag{3.27}$$

$$|\hat{r}_{\hat{i}}^T n_{\mathcal{P}_i} - d_{\mathcal{P}_i}| \leq R_{co}. \tag{3.28}$$

Then, it attempts to solve also problem (3.25) with the additional constraint

$$\|\hat{r}_{\hat{i}} - \hat{r}_i\| < R_{co}. \tag{3.29}$$

The above described algorithm can be codified as follows

Algorithm 2: Separation Hyperplane Computation

AGENT $i_r(t) \in \mathcal{G}_I(t)$:

1: REPEAT AT EACH TIME t
2: RECEIVE $d_i(t)$ FROM ALL $i \in \mathcal{A}$
3: **if** $d_i(t) == M,\ \forall i \in \mathcal{A}$ **then**
4: SET $\mathcal{Z}_i^{oa} = \mathbb{R}^{n_i+m_i}$ IN (3.20) $\forall i \in \mathcal{A}$ ▷ IF NO OBSTACLE IS SEEN, DISABLE OBSTACLE AVOIDANCE CONSTRAINTS FOR ALL AGENTS
5: **else**
6: FIND $\hat{i}(t) := \arg\min_{i \in \mathcal{A}} d_i(t)$
7: REQUEST THE HYPERPLANE $\mathcal{P}_{\hat{i}}$ TO AGENT $\hat{i}(t)$ THAT SOLVES (3.23)
8: RECEIVE $n_{\mathcal{P}_{\hat{i}}} = [n_{\mathcal{P}_{\hat{i},1}}, n_{\mathcal{P}_{\hat{i},2}}]^T$ FROM AGENT $\hat{i}(t)$
9: SEND $sign\left(\frac{n_{\mathcal{P}_{\hat{i},1}}}{n_{\mathcal{P}_{\hat{i},2}}}\right)$, $\hat{r}_{\hat{i}}$

GENERIC AGENT $i \in \mathcal{A} \setminus \{\hat{i}(t), i_r(t)\}$:

11: RECEIVE $sign\left(\frac{n_{\mathcal{P}_{\hat{i},1}}}{n_{\mathcal{P}_{\hat{i},2}}}\right)$, $\hat{r}_{\hat{i}}$
12: ENABLE (3.26)-(3.28)
13: SOLVE (3.23)
14: FIND $\hat{r}_i$ BY MEANS OF (3.25) WITH (3.29)
15: **if** $\hat{r}_i \equiv \emptyset$ **then**
16: $\hat{r}_i \leftarrow \hat{r}_{\hat{i}}$

Upon determining the separation hyperplanes, each agent proceeds to solve the original optimization problem (3.20) incorporating the upgraded established obstacle avoidance constraints $\mathcal{Z}_i^{OA}(\mathcal{P}_i)$. It's important to note that the hyperplanes $hy = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$, computed autonomously by the agents $i \in \mathcal{A}$, have a similar slope $n_{\mathcal{P}_i}$ because of the constraint (3.26). This shared slope ensures that all agents navigate around the obstacle(s) while keeping it on the same side. Moreover, the further constraint (3.28) guarantees that during the possible circumnavigating operation, connectivity keeping constraints are never saturated so that the whole formation can proceed without undesired stops.

Remark 3.5 The modular nature of the proposed strategy allows for seamless extensions to handle dynamic obstacles with known steady or slowly varying trajectories. By modeling such obstacles as virtual agents and applying the same collision avoidance constraints, the system ensures safe navigation. In cases where obstacles exhibit antagonistic or rapidly changing behaviors, the system can activate a global planner in an emergency recovery mode to modify the nominal reference fed to the CG-based supervision level, implementing an alternative obstacle avoidance technique. This adaptability highlights the flexibility and robustness of the strategy in dynamic environments, ensuring continued performance in complex, real-world scenarios.

3.3.3 Spanning Tree determination within the Reconfiguration Level

While a static spanning tree $\mathcal{G}_I(t) = \mathcal{G}_I(t_0)$, $\forall t \geq t_0 \in \mathbb{Z}_+$ offers simplicity, it can limit the ability of the network to achieve certain configurations. This limitation can hinder performance in terms of reaching the given references $r_i(t)$ for all agents $i \in \mathcal{A}$. This limitation can be addressed by allowing agents to switch between different spanning trees, thus using a dynamic approach. In particular, when the system reaches an equilibrium and the admissible command becomes constant, i.e. when $g_i(t) \equiv \bar{g}_i$, it could become advantageous to switch to a different spanning tree if it can further improve performance. Therefore, among all available spanning trees $\mathcal{G}_I(t)$, selecting the one with the minimum total weight, denoted by $\mathcal{G}_I^\star(t)$, is a desirable strategy, i.e.

$$\sum_{(i,j)\in\mathcal{E}_I^\star(t)} w_{ij}(t) \leq \sum_{(i,j)\in\mathcal{E}_I(t)} w_{ij}(t). \tag{3.30}$$

This approach ensures that each agent only needs to store and update information regarding at most two other agents. Specifically, each node stores information solely about its parent, denoted as $i_f(t)$, and its child while the root node only stores information about its children. Within this task, each agent i identifies the closest vehicle $\hat{j}$ to itself. If this closest vehicle is not its current parent, i.e. $\hat{j} \neq i_f(t)$, agent i asks for a possible parent change to the root agent $i_r(t)$, by sending the information of the edges to be modified. Due to its access to the entire spanning tree information through recursive queries to its children, the root agent can efficiently verify if the proposed new spanning tree, denoted by $\mathcal{G}_I'(t)$, maintains the connectivity property. In the positive, agent i performs the *Parent-Change Operation*. This logic is summarized by the following algorithm

Algorithm 3: Parent Change Operation

(AGENT i)

1: RECEIVE $(p_j(t), g_j(t))$ FROM $j \in \mathcal{N}_i$
2: COMPUTE $\hat{j}$ SUCH THAT $\hat{j} = \arg\min_{j\in\mathcal{N}_i} \|p_i^{g_i} - p_j^{g_j}\|$
3: **if** $\hat{j} \neq i_f(t)$ **then**
4: SEND $(i, i_f(t), \hat{j})$ TO $i_r(t)$ ▷ REQUEST PARENT CHANGE OPERATION TO ROOT NODE
5: **if** APPROVAL IS RECEIVED FROM $i_r(t)$ **then**
6: SET $\mathcal{G}_I(t) = \mathcal{G}_I'(t)$, SET $i_f(t) = \hat{j}$ AND MODIFY $\tilde{\mathcal{N}}_i(t)$

It is worth remarking that after step 4 of the previous algorithm, the root agent computes $\mathcal{E}_I^{'}(t) = \mathcal{E}_I(t) \setminus \{(i, i_f(t))\} \cup \{(i, \hat{j})\}$ and builds a new graph $\mathcal{G}_I'(t)$ such that $(i, \hat{j}) \in \mathcal{E}_I^{'}(t)$ and $(i, i_f(t)) \notin \mathcal{E}_I^{'}(t)$. If $\mathcal{G}_I'(t)$ is connected, the root agent $i_r(t)$ proceeds to send an approval back to the requesting node.

Remark 3.6 Algorithm 3 works thanks to the underlying assumptions that at time t_0 a spanning tree $\mathcal{G}(t_0)$ exists and it is determined by resorting in either centralized or distributed way. Several algorithmic approaches have been proposed for the computation of such spanning tree (see e.g. [96]).

Lemma 3.7 *: A parent change operation performed by agent i from agent $\bar{i}$ to agent $\hat{i}$ ensures that*

$$\begin{aligned} &[x_i(t),\, x_{\bar{i}}(t), g_i(t),\, g_{\bar{i}}(t)]^T \in \mathcal{Z}^{ck}, \\ &\qquad\Downarrow \\ &[x_i(t),\, x_{\hat{i}}(t), g_i(t),\, g_{\hat{i}}(t)]^T \in \mathcal{Z}^{ck}. \end{aligned} \tag{3.31}$$

Proof. This simply follows from the definition of $\mathcal{Z}^{ck}$ in (3.11). In fact,

$$\begin{aligned} &[x_i(t),\, x_{\bar{i}}(t),\, g_i(t),\, g_{\bar{i}}(t)]^T \in \mathcal{Z}^{ck} \\ &\qquad\Downarrow \\ &\begin{cases} \|p_i^{g_i} - p_{\bar{i}}^{g_{\bar{i}}}\| \leq R_{co} - \delta \\ \|p_i(k, x_i, g_i) - p_{\bar{i}}(k, x_{\bar{i}}), g_{\bar{i}})\| \leq R_{co},\ k = 0, \ldots, k_0 \end{cases} \end{aligned}$$

Moreover, the latter equation jointly with the step 2 of Algorithm 3 leads to

$$\begin{aligned} &\begin{cases} \|p_i^{g_i} - p_{\hat{i}}^{g_{\hat{i}}}\| < \|p_i^{g_i} - p_{\bar{i}}^{g_{\bar{i}}}\| \\ \|p_i(k, x_i, g_i) - p_{\hat{i}}(k, x_{\hat{i}}), g_{\hat{i}})\| < \|p_i(k, x_i, g_i) - p_{\bar{i}}(k, x_{\bar{i}}), g_{\bar{i}})\|, \\ k = 0, \ldots, k_0 \end{cases} \\ &\qquad\Downarrow \\ &\begin{cases} \|p_i^{g_i} - p_{\hat{i}}^{g_{\hat{i}}}\| < R_{co} - \delta \\ \|p_i(k, x_i, g_i) - p_{\bar{i}}(k, x_{\hat{i}}), g_{\hat{i}})\| < R_{co},\ k = 0, \ldots, k_0 \end{cases} \\ &\qquad\Downarrow \\ &[x_i(t),\, x_{\hat{i}}(t),\, g_i(t),\, g_{\hat{i}}(t)]^T \in \mathcal{Z}^{ck}. \end{aligned}$$

□

Lemma 3.8 *: Let $\mathcal{G}_I(t)$ be the spanning tree adopted in Algorithm 1 at time instant t. Then, the graph $\mathcal{G}'_I(t)$ resulting from the consecutive application of Algorithm 3 on the entire network is a minimum spanning tree, i.e. $\mathcal{G}'_I(t) = \mathcal{G}^\star_I(t)$.*

Proof. By performing step 2 in Algorithm 3 at time t, each agent selects a new parent such that

$$\|p_i(0, x_i, g_i) - p_{\hat{j}}(k, x_{\hat{j}}, g_{\hat{j}})\| \leq \|p_i(0, x_i, g_i) - p_j(0, x_j, g_j)\|, \quad \forall j \in \mathcal{N}_i.$$

that is equivalent to

$$\|p_i(t) - p_{\hat{j}}(t)\| \leq \|p_i(t) - p_j(t)\|, \forall j \in \mathcal{N}_i, \forall i \in \mathcal{A} \tag{3.32}$$

and, in turn, to

$$w_{i,\hat{j}}(t) \leq w_{i,j}(t), \hat{j} \in \mathcal{N}_i, \forall j \in \mathcal{N}_i, \forall i \in \mathcal{A}. \tag{3.33}$$

Then, performing a summation among left and right terms respectively produces

$$\sum_{(i,\hat{j}) \in \mathcal{E}'_I(t)} w_{i,\hat{j}}(t) \leq \sum_{(i,j) \in \mathcal{E}_I(t)} w_{i,j}(t), \forall \text{ possible } \mathcal{E}_I(t) \subseteq \mathcal{E}_C(t). \tag{3.34}$$

□

Before summarizing the main properties of the described supervision scheme, it is useful to introduce the following definition

Definition 3.9 (**Pareto Optimal (PO) Solution):** Consider the following multi-objective problem

$$\begin{array}{l} \min_g [f_1(g_1), f_2(g_2), \ldots, f_N(g_N)] \\ \text{subject to } g = [g_1^T, \ldots, g_i^T, \ldots, g_N^T]^T \in \mathcal{S}. \end{array}$$

The vector $g^{*_p} \in \mathcal{S}$ is a PO solution if there exist no other $g \in \mathcal{S}$ such that: $f_i(g_i) \leq f_i(g_i^{*_p})$, $i = 1, \ldots, N$ for which there exists a j such that $f_j(g_j) < f_j(g_j^{*_p})$.

Theorem 3.10 *Consider systems* (3.1)*, along with the distributed supervision architecture depicted in Figure 3.3. Assume that Algorithm 1 is performed by agents in $\mathcal{A}$, distributed into turns $\mathcal{T}_i$, such that periodically all agents of the network are selected to update their commands, i.e. $\exists t' : \forall t > 0, \bigcup_{i=0}^{t'} \mathcal{T}_{t+i} = \mathcal{A}$. Then*

1. *The overall system actuated by agents implementing Algorithm 1 under the inputs of the* Reconfiguration Level *never violates the constraints, i.e., $c_i(t) \in \mathcal{C}_i$ and $p_i(t) \notin \bigcup_{j=1,\ldots,D} \mathcal{H}_j$, for all agents $i \in \mathcal{A}$ and all time steps $t \in \mathbb{Z}_+$.*
2. *Under the assumption that $\mathcal{Z}_i^{oa}(\mathcal{P}_i(t))$ is fixed for all $i \in \mathcal{A}$ and constraints* (3.5) *are not considered, whenever $r(t) \equiv [\, r_1^T, \ldots, r_N^T \,]^T, \forall t$, is a constant set-point, the sequence of solutions $g(t) = [g_1^T(t), \ldots, g_N^T(t)]^T$ asymptotically converges to a Pareto-Optimal (PO) stationary (constant) solution of the following multi-objective optimization problem*

$$\begin{array}{l} \min_g [\| \, g_1 - r_1 \, \|_{\Psi_1}^2, \ldots, \| \, g_i - r_i \, \|_{\Psi_i}^2, \ldots, \| \, g_N - r_N \, \|_{\Psi_N}^2] \\ s.t. \begin{cases} \left[x_{g_i}, \, x_{g_j}, \, g_i, \, g_j\right]^T \in \mathcal{Z}^{ck}, \forall j \in \tilde{\mathcal{N}}_i \\ [x_{g_i}, g_i]^T \in \mathcal{Z}_i^{oa}(\mathcal{P}_i) \cap \mathcal{Z}_i^l. \end{cases} \end{array} \tag{3.35}$$

3. *Let an arbitrary convex static obstacle $\mathcal{V}$ be given. Then, for each set of constant targets r_i for whom it exists a separation hyperplane $\mathcal{P}^\star(n^\star, d^\star)$, that is*

$$\begin{cases} v^T n^\star \leq d^\star, & \forall v \in V \\ r_i^T n^\star \geq d^\star + d_{safe}, \end{cases} \tag{3.36}$$

whenever $r(t) \equiv [\, r_1^T, \ldots, r_N^T \,]^T, \forall t$, the sequence of $g(t) = [g_1^T(t), \ldots, g_N^T(t)]^T$ asymptotically converges to $r(t) \equiv [\, r_1^T, \ldots, r_N^T \,]^T$ if for each $i \in \mathcal{A}$ it results $\|r_i - r_j\| < R_{co}$ for at least $j \in \mathcal{A} \setminus \{j\}$. Otherwise, it converges to a PO solution complying with (3.36)*.*

Proof. 1. It is well-known that the sets involved in (3.20) are positively invariant [79]. Then, if *OASs* in (3.20) do not change, then the pair $(x_i(t), g_i(t))$ determined at time t is always admissible. Possible hitches on *OASs* modification can arise from the introduction of new hyperplanes for obstacle avoidance reasons. Anyway, feasibility is kept thanks to step 4 of Algorithm 1. Moreover, a Parent Change Operation performed by agent i could represent a drawback. Even in this case, the feasibility of problem (3.20) is guaranteed by Lemma 3.7 because such a modification affects only the *Connectivity Keeping* constraints where agents $(i, \hat{i})$ are jointly involved.

2. Thanks to Lemmas 3.7 and 3.8, constant values for r_i imply static neighborhood $\mathcal{N}_i$. Then, each agent solves Problem (3.20) with a fixed constraint region. In this context, the proposed scheme is equivalent to the TB-CG scheme of [64] that has been proved to converge to a PO for problem (3.35) whenever $r(t)$ is a constant set-point.

3. To prove this statement, let us consider a worst-case scenario where agent $\hat{i}$ computes its hyperplane $\mathcal{P}_{\hat{i}}$ and it is no more able to find a better one in the next time steps. Please notice that a solution to (3.23) for the other agents always exists because, thanks to assumption (3.24), $\mathcal{P}_{\hat{i}}$ is admissible even for other $i \in \mathcal{A}$. On the other hand, the existence of a solution for problem (3.25) is not always ensured. Therefore, in the worst case, all agents are instructed to apply line 16 of Algorithm 2 and set their temporary target $\hat{r}_i$ to $\hat{r}_{\hat{i}}$. Hence, by using the same arguments used in the previous item of this proof, the sequence $g(t) = [g_1^T(t), \ldots, g_N^T(t)]^T$ asymptotically converges to $r = [\hat{r}_{\hat{i}}^T, \ldots, \hat{r}_{\hat{i}}^T]^T$ and agents manage to go through the hyperplane $\hat{\mathcal{P}}_{\hat{i}}$ computed by $\hat{i}$ in (3.25). In this situation, agent $\hat{i}$ can directly move towards $r_{\hat{i}}$, i.e. $\hat{r}_{\hat{i}} \equiv r_{\hat{i}}$ while the others, in the worst case, approach toward $r_{\hat{i}}$ too. However, because of Assumption (3.36), in a finite time they are able to cross $\mathcal{P}^\star(n^\star, d^\star)$. Then, problem (3.23) becomes solvable with a negative $\varepsilon_{\mathcal{P},i}$ and $\hat{r}_i = r_i$. Then, the proof is completed because we enter in the case considered in the previous item of the proof. □

3.4 Robot Operating System Architecture

In this section, details of the Robot Operating System (ROS) implementation of the overall control architecture are given. The latter development environment has gained attention in the robotics community [76, 77] since it provides powerful development tools for each phase of a robotics project and because of its open source license policy. More in detail, a Guidance, Navigation, and Control (GNC) architecture that smoothly combines with the suggested supervisory technique is here described to enhance real-world portability. This GNC design greatly broadens the scope of the proposed approach by making the results more portable to practical applications. In addition, the proposed GNC is extended by introducing a Cooperation module, which provides specific inter-vehicle reliable communication capabilities.

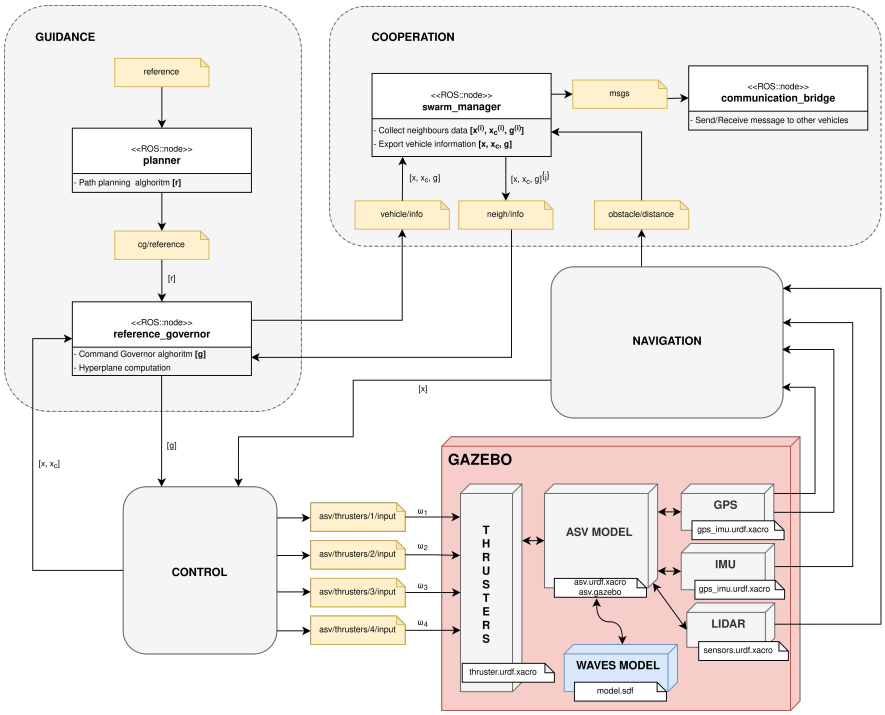


Fig. 3.5 ROS GNC-C Architecture Overview.

Figure 3.5 depicts an overview of the main components of the proposed GNC architecture for a single agent, illustrating the architecture when interfaced with Gazebo (which can be easily replaced with a real vehicle). Please notice that only the gray nodes represent architectural components, which

can be adopted to deal with various surface vehicle types. Below, an overview of the modules of the proposed GNC-C architecture is given.

Control Module

In the *Control* module there are nodes that implement *i*) the primal control law, synthesized as a linear static state-feedback gain, and *ii*) the low-level control law, useful to manage the thrusters. The *Control* module implements the *Control and Physical Level* of the abstract *Multi-Layer Supervision Architecture* previously introduced in Figure 3.3. The whole module receives a reference g from the *Guidance* module and the current state x of the vehicle, and exports the augmented state (the states of the vehicle and the controller). The main node of the *Control* module is the *vehicle_controller* node, which is in charge of computing the state feedback control law used to perform the tracking of the reference. Parameters of the controller are specified in a specific .yaml file. This node computes the desired thrusts, expressed in the body-fixed frame, and publishes them on a specific topic (*vehicle/tau*). Another node, subscribed on this topic, provides the conversion of these values in the desired torques for each motor/propeller, according to an actuator allocation matrix. The required torques are then converted in the signals sent to the actuators (PWM signals on real vehicle, *FloatStamped* messages for Gazebo simulator). The *vehicle_controller* node implements an ActionServer which allows one to enable/disable dynamically its functionalities. This is achieved by means of introducing a custom ROS action which includes a boolean enable flag, a string that indicates its status, and a float array to provide the tracking error as a feedback. During the execution, the primal controller continuously provides the current tracking error as action feedback and publishes the augmented state on a specific topic.

Guidance Module

The main goal of the *Guidance* module is to compute an admissible reference according to the desired target and the imposed dynamic constraints. The constraints that are taken into account are the formation keeping constraints and the local constraints that bind the velocity and the torques of the motors/propellers of the vehicle (*Supervision Level* tasks in Figure 3.3). This is performed through a node that implements the previously introduced distributed *Command Governor* algorithm. Furthermore, this module is designed to compute the hyperplane used to provide obstacle avoidance capability to the vehicle (*Local Reconfiguration Level* task). This module is also equipped of a *Planner* node, which is responsible for the high-level

reference generation, choosing r_i according to the vehicle task. Node *reference_governor* executes Algorithm 1 by using a developed Python-based CG library. This node loads closed-loop data matrices from a .yaml file and, based on augmented state received from the *Control* module, periodically computes admissible commands. Constraints with other vehicles are satisfied taking into account the neighbor-related information, obtained from the *Cooperation* module via the *neigh/info* topic. Whilst *reference_governor* node acts like a local planner, taking into account local perception of the vehicle and guides the latter in reaching the current reference, the *planner* node is a global planner, which determines the global path to be followed by the vehicle according to its mission. The global path can be considered as a list of waypoints that have to be reached by the vehicle. Similarly of what has been done with the *vehicle_controller* node, the *reference_governor* node implements a custom ActionServer to dynamically enable/disable the computation of the admissible commands.

Navigation Module

The *Navigation* module contains all nodes responsible for reading data from sensors. The main nodes of this module read *sensor_msgs/NavSatFix* and *sensor_msgs/Imu* messages from a GPS receiver and from an IMU sensor and use them to compute an estimation of the state of the vehicle [100]. Another ROS node provides the distance from objects $\mathcal{H}_i$ measuring it through a stereo camera sensor mounted on the top of the vehicle. The stereo camera sensor generates *sensor_msgs/PointCloud2* messages, which are subsequently processed. A key challenge was to optimize the processing speed for these 2D structured point cloud messages. To this end, the *pcl_ros* ROS package has been used. The *Vision Module* measures and stores the distance from each point of the obstacle, calculates the minimum distance between these and sends this information to the *Cooperation* module that propagates it to the leader vehicle. In particular, when a simulation tool such as *Gazebo* is used, in order to simulate sensors and actuators, specific plugins are used, which are configured using real parameters taken from commercial sensors/motors/propellers data-sheets.

Cooperation Module

Finally, the *Cooperation* module collects all information received from the other vehicles through the *communication_bridge* node, such as their current augmented state $[x, x_c]$ and their current admissible command g. These values are sent to the *Reference_Governor* node, located in the *Guidance* module, to

allow it to compute an admissible reference that satisfies connectivity keeping, collision and obstacle avoidance constraints. The *swarm_manager* node also receives the minimum distance of each vehicle from obstacle and sends it to the leader vehicle. The leader collects all these distances and selects the nearest vehicle to the obstacle (*Global Reconfiguration Level*), and this vehicle will have to compute a separation hyperplane the slope of which is used then to establish a common direction for all other agents, effectively implementing the logic of the *OA Task* discussed in Section 3.3.2. The *swarm_manager* node also *i*) implements a distributed token-based algorithm to perform the turn synchronization relative to the Turn-Based *Command Governor* approach; *ii*) is tasked with storing the network information, such as the current *spanning tree*, to perform the *Parent Change* operations. Finally, the *communication_bridge* node, which acts as a bridge between different ROS master networks, is introduced to overcome the limitations imposed by ROS. More in detail, a major drawback of ROS in a multi-agent scenario is the presence of a centralized module (ROS master), making it challenging to apply the framework to distributed systems on non-ideal networks [104]. Moreover, the presence of a centralized entity reduces the autonomy of each agent. The implementation of the the *communication_bridge* allows one to run a ROS master on each vehicle, making each agent a standalone system, while information with neighbors are exchanged through a custom UDP protocol, which allows the reduction of jitter and the delay in sending data. Alternatively, a ROS2-based implementation of the proposed GNC architecture could address the limitations imposed by the centralized nature of the architecture of ROS.

3.5 Simulation Results

Considering that the specific domain of interest for the evaluation of the proposed solution is the marine environment, the proposed supervision strategy has been investigated by simulations on a set of N fully actuated marine USVs, modeled as dynamically decoupled systems, as shown in Appendix B, with index set $\mathcal{A} = \{1, \dots, N\}$.

Simulation results have been obtained using both Matlab and the low-level physics-based Gazebo simulator presented in Appendix C. More in detail, this section presents Matlab-based simulations, where rotational dynamics are neglected, for the parent change and cooperation-inducing strategies. Gazebo simulations are also presented for the cooperation-inducing strategy, with a similar setting of the Matlab simulations.

The parameters of model (B.5) have been off-line estimated, by means of an identification procedure exploiting the Least-Mean Squares (LMS) algorithm [98, 99], in the Gazebo simulation environment. More in detail, the parameter m is the mass of the vehicles set to be equal to $23.83\,[Kg]$, the parameter J is the rotational inertia set to be equal to $3\,[Kgm^2]$, while the friction coefficient

and the rotational friction coefficient $\beta = \beta_t = 1$. Starting from (B.5), an augmented model for position tracking had to be built and a continuous-time control law has been designed in Matlab in order to guarantee stability and zero-error tracking for constant set-points. Using the *Zero-Order Hold* discretization technique, with sampling time $T_s^c = 0.1$ [s], the models have been discretized and recast in the form (3.1).

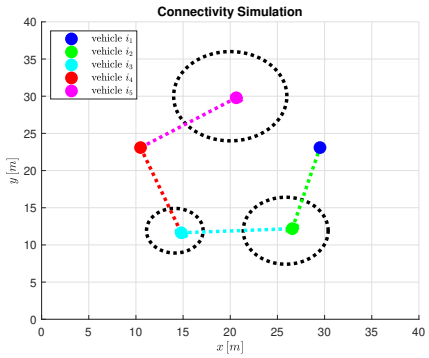
The Matlab-based simulation results reported hereafter were obtained under Matlab 9.7 (Release R2019b) + Simulink 10.0 on a Lenovo Y700 personal computer, while the Gazebo-based simulations were obtained under ROS Noetic + Ubuntu 20.04. To implement the optimization problems used in the proposed approach, The Yalmip interface [105] was used for the Matlab implementation while the CVXPY interface [106] used for ROS/Gazebo implementation, and Gurobi [107] was used as a solver in both cases.

3.5.1 Matlab-based Simulations

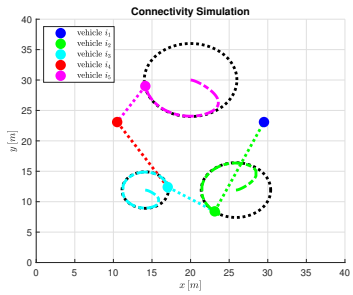
The Matlab-based simulations presented in this section were carried out by means of using the CoGoV toolbox [108, 109]. More in detail, the CoGoV toolbox provides a streamlined and user-friendly approach for implementing Command Governor-based supervisory strategies in the context of multi-vehicle distributed motion planning challenges, and thanks to its modular structure, it can be employed to work with several types of autonomous vehicles across marine, terrestrial, and aerial domains, allowing for the simulation of complex scenarios involving coupled or decoupled dynamical systems with local or global coordination constraints.

3.5.1.1 Dynamic Spanning Tree

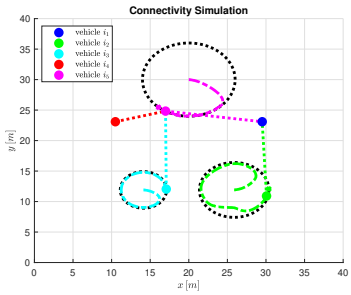
A simulation showing the benefits of implementing Algorithm 3 has been conducted and is shown in Figure 3.6. In this experiment, involving $N = 5$ agents, each agent is instructed to track a predetermined path while maintaining the connectivity of the network, i.e. each agent has to keep its position close to its father in order to preserve connectivity. More in detail, agents i_2, i_3 and i_5 are instructed to track circular paths while agents i_1 and i_4 are instructed to stand still. If a static spanning tree $\mathcal{G}_I$ were used, some agents would lock into their positions as shown in Figure 3.6(b). On the contrary, a dynamic re-configuration of the minimum spanning tree (Figure 3.6(c)) $\mathcal{G}_I(t)$ allows the agents to stay closer to their assigned paths while preserving the connectivity of the whole formation, enhancing the trajectory tracking performance. Finally, a video showing the whole simulation can be found at https://www.youtube.com/watch?v=sRL0oc4tdzc.



(a) Initial configuration of the patrolling simulation.



(b) Simulation carried out using a static spanning tree $\mathcal{G}_I$.



(c) Simulation carried out using a dynamic minimum spanning tree $\mathcal{G}_I(t)$.

Fig. 3.6 Patrolling simulation with the use of a static (a) and dynamic (b) spanning tree.

3.5.1.2 Obstacle Avoidance and Connectivity Keeping

In this case, the simulating scenario depicted in Figure 3.7 has been considered. In that Figure, each vehicle is instructed to track a reference that is located on the right side of the working space, beyond the obstacle. Please notice the spanning tree $\mathcal{G}_I(t_0)$ depicted in the figure. In order to reach the prescribed goal, agents apply Algorithm 1 within two turns.

Moreover, it is assumed that agents are subject to the following local constraints

$$\begin{cases} \|T_{l,i}(t)\| \leq 30\,[N], \\ \|\dot{p}_i^l(t)\| \leq 0.8\,[\frac{m}{s}], \\ \text{with } l = \{x, y\} \text{ and } \forall t \in \mathbb{Z}_+ \end{cases} \tag{3.37}$$

and coordination coupling constraints

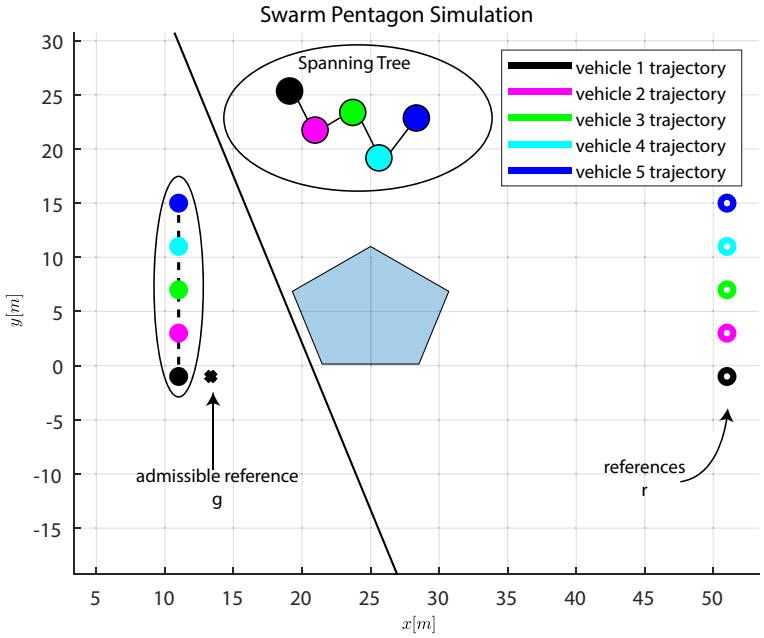


Fig. 3.7 Initial simulation configuration. The vehicles are instructed to reach references that are located on the other side of the obstacle.

$$\begin{cases} \|p_i(t) - p_j(t)\| \leq 7\,[m],\ \forall j \in \tilde{\mathcal{N}}_i(t), \\ \|p_i(t) - p_j(t)\| \geq 1\,[m],\ \forall j \in \mathcal{A} \setminus \{i\}. \end{cases} \tag{3.38}$$

In order to highlight the benefits of the proposed Algorithm 2, Figure 3.8 depicts a simulation where each agent locally determines the separation hyperplane in a decentralized way. Within this context, under the prescribed *Obstacle Avoidance* and *Connectivity Keeping* constraints, a deadlock of the whole formation could occurs as depicted in Figure 3.8. This happens when vehicles get around an obstacle in different directions. In particular, agents i_2 and i_3 have chosen two opposite directions during the obstacle avoidance maneuver. Due to this choice, their distance grows until they are not able to move since they reach the maximum admissible distance allowing full connectivity. In this context, agents get stuck as almost all *Connectivity Keeping* constraints become active. On the contrary, a different behavior can be observed when the proposed Algorithm 2 is considered within the Supervision Architecture. In fact, in Figure 3.9, by exploiting the coordination induced by Algorithm 2, the agents successfully get around the obstacle and reach their targets. The differences between the two above described behaviors stand out even more clearly if the evolution of the admissible reference is observed. Fig-

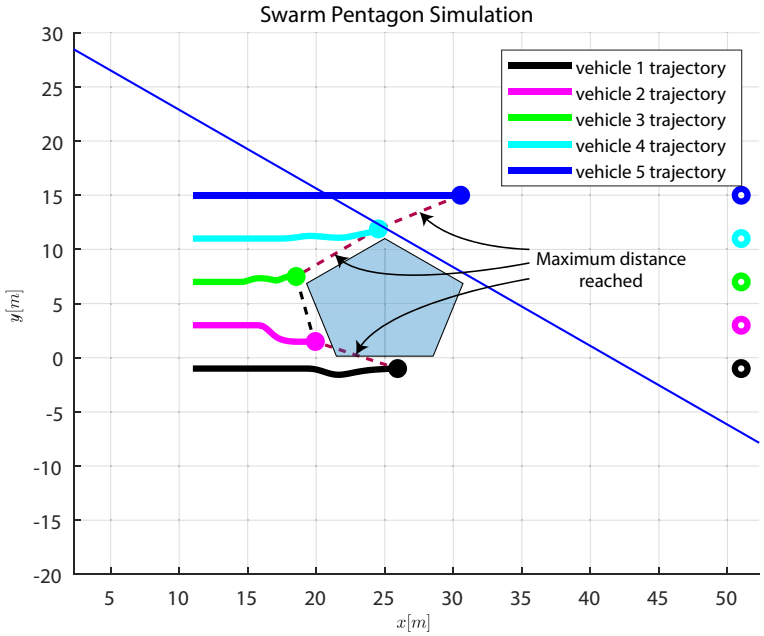


Fig. 3.8 Trajectories evolution with decentralized separation hyperplane computation not implementing condition (3.26).

ure 3.10(b) shows how agent i_1 never reaches the x component of its given reference r_1^x, while Figure 3.10(a) shows how, with the leader/follower approach presented in section 3.3.2, the same agent reaches its given reference. By analyzing the evolution of the measured distance kept between agents i_1 and i_2 in Figure 3.11(a), it is noticeable how the proximity constraints are always fulfilled but never they become active (Figure 3.11(a)), as for the case when Algorithm 2 is not executed (Figure 3.11(b)). This behavior is even more clear in a video simulation available at https://youtu.be/wHqVogWcCYw. The color of the hyperplanes is equal to the color of the agent that computed it. In particular, from the video it is possible to observe that at the time instant $t = 0.0$ [s], agent i_1 performs Algorithm 2 and establishes the direction of avoidance of the obstacle. As a consequence, all other agents in the network find hyperplanes with similar slope thanks to condition (3.26).

3.5.1.3 Collision Avoidance and Obstacle Avoidance and Connectivity Keeping

Another simulating scenario, depicted in Figure 3.12, has been analyzed with $N = \{i_1, i_2, i_3, i_4\}$. The circular regions are introduced to model the vision

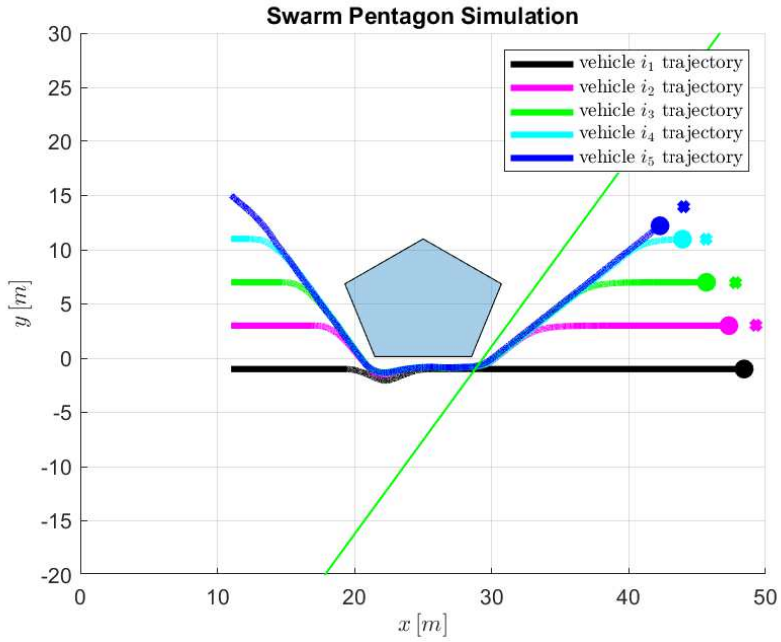
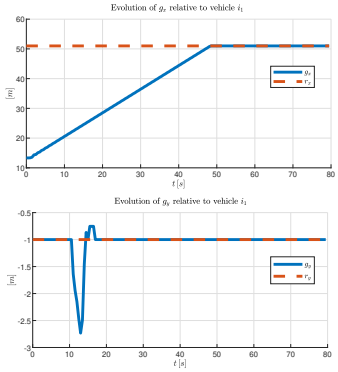
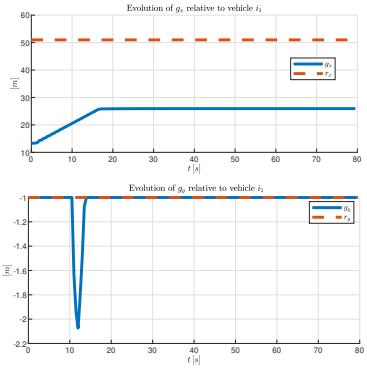


Fig. 3.9 Trajectories evolution when implementing Algorithm 2.



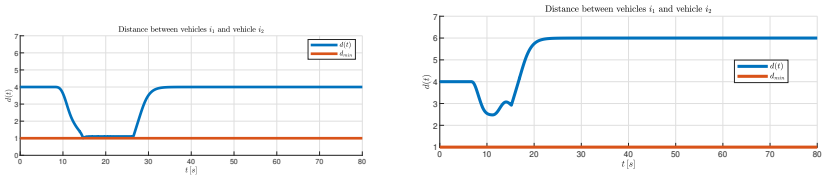
(a) Evolution of $g_1(t)$ with our approach.



(b) Evolution of $g_1(t)$ without our approach.

Fig. 3.10 Comparison of the evolution of $g_1(t)$ with our approach and without.

range of the Vision Module, in accordance with Assumption 1. In this case, two groups of vehicles $\mathcal{F}_1 = \{i_1, i_2\}$ and $\mathcal{F}_2 = \{i_3, i_4\}$, are located initially on opposite sides of a canal shaped environment. Each vehicle is subject to



(a) Evolution of the distance with the proposed approach. The red dotted line represents both the maximum and minimum admissible distance.

(b) Evolution of the distance without the proposed approach. The red dotted line represents both the maximum and minimum admissible distance.

Fig. 3.11 Evolution in time of the distance between vehicle i_1 and vehicle i_2 directly with and without the cooperation-inducing effect.

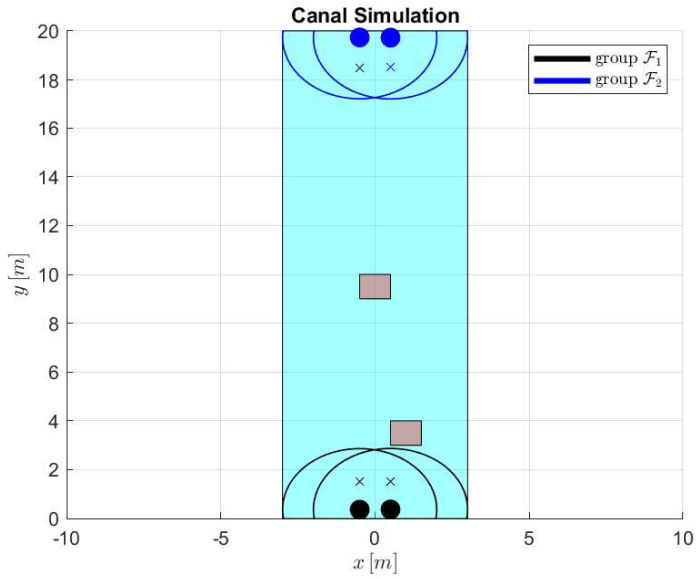


Fig. 3.12 Initial simulation configuration. All vehicles are instructed to reach references that are located on the other side of the canal. Also two rectangular shaped obstacles are contained into the same environment.

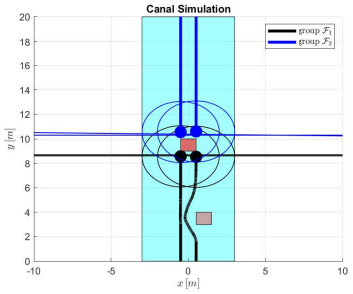
the same local constraints presented in (3.37). Vehicles belonging to a group all share the following coordination coupling constraints

$$\begin{cases} \|p_i(t) - p_j(t)\| \leq 1\,[m],\ \forall j \in \tilde{\mathcal{N}}_i(t), \\ \|p_i(t) - p_j(t)\| \geq 0.2\,[m],\ \forall j \in \tilde{\mathcal{N}}_i(t). \end{cases} \tag{3.39}$$

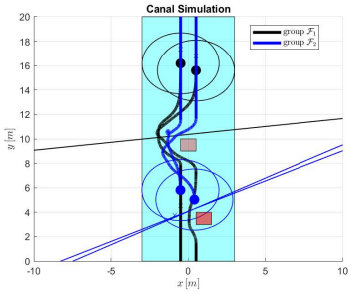
Meanwhile, vehicles belonging to different groups share just *Collision Avoidance* constraints (3.5):

$$\left\{ \|p_i(t) - p_j(t)\| \geq 1\,[m],\ \forall j \in \mathcal{A} \setminus \tilde{\mathcal{N}}_i(t). \right. \tag{3.40}$$

In order to highlight the benefits of the proposed method, Figure 3.13(a) shows a simulation where each agent locally determines the separation hyperplane in a decentralized way. Under the prescribed *Obstacle Avoidance*, *Collision Avoidance* and *Connectivity Keeping* constraints, a deadlock for both groups $\mathcal{F}_1$ and $\mathcal{F}_2$ occurs. This happens when vehicles in the same group get around an obstacle in different directions. Similarly as happened previously, agents in group $\mathcal{F}_1$ have chosen two opposite directions during the obstacle avoidance maneuver. The same happens with agents in $\mathcal{F}_2$. Due to this choice, their distance grows until they are not able to move since they reach the maximum admissible distance allowing full connectivity.



(a) Trajectories evolution with decentralized separation hyperplane computation not implementing condition (3.26).



(b) Trajectories evolution when implementing the proposed method in the canal environment.

Fig. 3.13 Canal simulation without and with the proposed approach.

When the proposed approach is considered, the agents get around the obstacle and reach their targets as shown in Figure 3.13(b). But, this scenario is more complicated since the vehicles of the two groups may collide. Figure 3.14 depicts all the measured distances between vehicle i_1 and all other vehicles, showing the effectiveness of the proposed supervision scheme, as the measured distances never violate the imposed constraints.

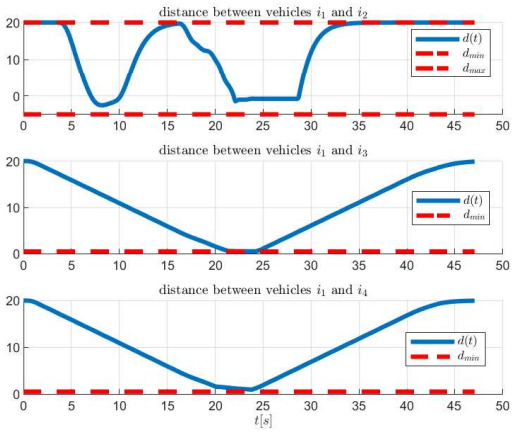
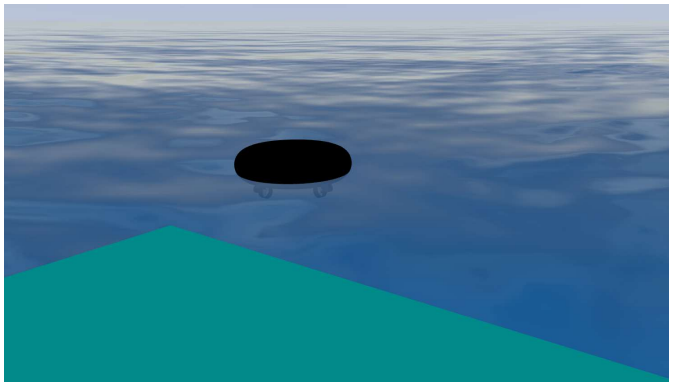


Fig. 3.14 Evolution in time of the distance between vehicle i_1 and all other vehicles with the proposed method.

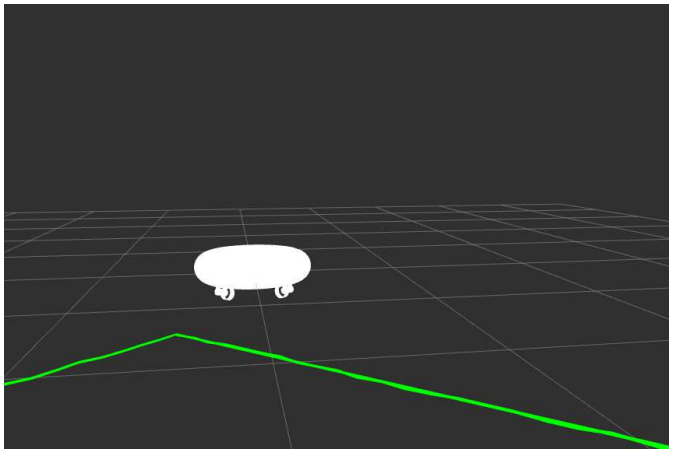
3.5.2 Gazebo-based Simulations

In this section high-fidelity, low-physics-based simulations conducted using the previously derived results using the Gazebo simulator (see Appendix B) are detailed. More in detail, a nonlinear 3-DOF (surge, sway and yaw) surface vehicle is simulated using the proposed ROS framework presented in Section 3.4. In order to satisfy Assumption 1, the vehicle model (Figure C.1) in Gazebo is enriched with a plug-in to integrate a stereo camera sensor. A *ROS/Gazebo* integrated tool *Rviz* (Figure 3.15 (b)) shows the cloud points, belonging to the obstacle, detected by the sensor mounted on top of the vehicle.

To facilitate the interpretation of the results and highlight the similarities between Matlab or Gazebo, a similar setup of section 3.5.1.2 has been considered. More in detail, agents are required to fulfill the same local constraints (3.37) and coordination coupling constraints (3.38). The simulating scenario, depicted in Figure 3.16, has been considered with the aim of validating the GNC architecture and to assess the whole formation performance in terms of maneuverability. More in detail, each vehicle is instructed to reach to a way-point that is located on the opposite side of the working space, beyond the obstacle. Tests have been performed by assigning the GPS coordinates of the arrival points. In order to reach the prescribed goal, agents apply Algorithm 1. Figure 3.17 shows the trajectories recorded for each agents during the experiment in which the vehicles are instructed to fulfill local, collision avoidance, obstacle avoidance and connectivity keeping constraints. In fact, when the proposed Algorithm 2 is considered within the Supervision Archi-



(a) ASV shown in *Gazebo* Simulator during a simulation.



(b) *Rviz* perspective of the *Gazebo* simulation.

Fig. 3.15 ASV shown in the *Gazebo* simulator along with the perspective of the ASV shown in *Rviz*.

tecture, by exploiting the coordination induced by the Algorithm itself, the agents get around the obstacle and reach their targets as shown in Figure 3.17. Positions shown in Figure 3.17 are those obtained with the GPS sensor with a rate of $2\,[Hz]$ and an accuracy of $\pm\ 0.05\,[m]$.

In order to have a comparison and highlight the benefits of the proposed strategy, two simulations have been carried out in two cases, with and without the proposed coordination scheme when obstacle avoidance and connectivity keeping constraints are enforced. As a result, the distance between vehicle i_1 and i_2 is reported in Figure 3.18. It is noticeable how the proximity con-

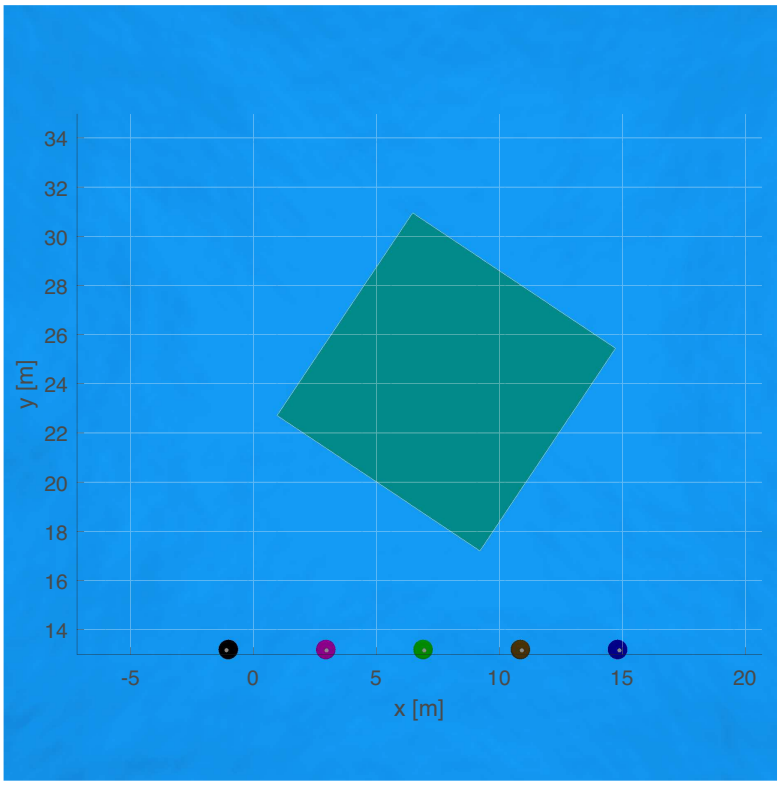


Fig. 3.16 Initial configuration of the Gazebo simulation. Each vehicle is instructed to reach a reference that is located on the other side of the obstacle.

straints between vehicle i_1 and vehicle i_2 are always fulfilled but never they become active when the proposed strategy is enabled.

3.6 Concluding Remarks

In this chapter, the challenge of maintaining connectivity and formation compactness while avoiding obstacles for dynamically decoupled agents with linear discrete-time dynamics, subject to local and collision avoidance constraints, has been addressed. The proposed solution is based on the Turn-Based distributed CG scheme [64] for a network of dynamical systems, that has been here extended by including in its mathematical formulation connectivity keeping, obstacle avoidance and collision avoidance constraints. Notably, the resulting scheme induces cooperation, effectively avoiding unwanted situations where agents become stuck while navigating around obstacles. This

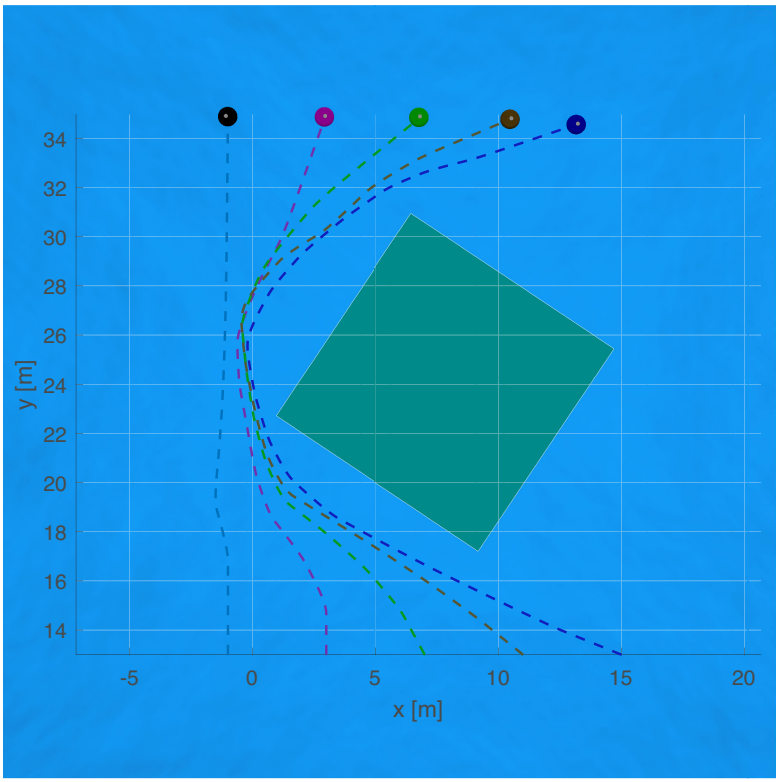


Fig. 3.17 Complete trajectory of the agents with the proposed supervision and coordination scheme using the Gazebo Simulator.

capability stems from an induced self-coordination behavior, leading agents to follow similar directions and to safely navigate around obstacles toward their designated waypoints. This cooperation-induced strategy is based on a non-iterative distributed solution, which is less computationally demanding w.r.t. typically iterative cooperative solutions, making the strategy attractive for practical applications. To further improve performance, a dynamically reconfigurable communication structure is integrated. By addressing the limitations of static spanning tree-based networks, this mechanism enhances the overall supervision architecture. A GNC ROS-based architecture, specifically designed to implement the proposed supervision and coordination scheme, has been developed to facilitate real-world portability. Both MATLAB and Gazebo simulations, involving a group of marine surface vehicles, have been conducted using this architecture. Simulation results, across various scenarios, demonstrate the effectiveness of the proposed supervision framework in preventing agents from becoming stuck while navigating around obstacles.

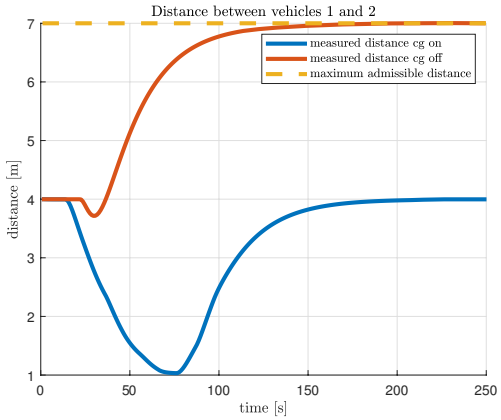


Fig. 3.18 Computed Euclidean distance between agent i_1 and agent i_2 in the Gazebo Simulator. The blue line represents the distance computed with the proposed scheme and with the red line the distance computed without the proposed scheme.

Chapter 4
Cooperative Distributed Command Governor Schemes

A novel distributed Command Governor (CG) scheme, relying on cooperative distributed optimization, is presented for multi-agent networked systems subject to local and coupling constraints. Unlike existing non-cooperative distributed CG schemes, the proposed approach leverages a distributed optimization framework based on relaxation techniques and duality theory to enable agents to cooperatively contribute individually to the minimization of a global performance index. Furthermore, agents exchange only a vector that encodes the resource utilization of other agents, preserving privacy by avoiding the exchange of information about objective functions, constraints, or admissible commands. Even if the resulting scheme is iterative, and thus more computationally demanding with respect to the cooperation-inducing scheme, the cooperative approach is more widely applicable and agents are able to find in a distributed way a near-optimal solution to the original centralized problem. The effectiveness of the proposed approach is validated through an illustrative example involving marine surface vehicles that are subject to connectivity keeping coupling constraints.

4.1 Introduction

Existing distributed CG strategies, such as [68, 65, 101, 102, 64], often employ non-cooperative behavior. These schemes, where agents solve local optimization problems with limited coordination, can lead to deadlock situations and undesirable Nash equilibria [68]. More in detail, each agent can solve a localized version of the original problem, assuming fixed values for all other variables in the network. The performance index that each agent locally minimizes contains only its contribution, leading to selfish behavior. In order to avoid "egoistic behavior," the interest here is moving in the class of cooperative methods. These methods aim to "cooperatively optimize" a global-objective function, which is a combination of local-objective functions, while

relying solely on local data. Recently, for that purpose, there has been a growing interest in distributed control and coordination of multi-agent networks [103]. In general, the optimization set-up of the centralized Command Governor is amenable to distributed computation, since it involves minimizing the sum of local cost functions, each of which depends on a local variable, subject to local constraints for each variable and coupling constraints potentially involving all the decision variables.

The main approach of distributed cooperative methods involves the use of Lagrangian dual-decomposition and dual methods. This leads to distributed optimization algorithms when the problem is "separable," i.e. problems where local-objective functions and constraints can be decomposed based on the components of the decision vector. The networking literature has extensively employed this methodology to design cross-layer resource-allocation mechanisms [110, 111, 112, 113]. Further distributed optimization algorithms for constraint-coupled problems have been explored in [114, 115, 116, 117]. While recent works based on techniques like consensus-based primal-dual perturbation [118], dual decomposition [119, 120], saddle-point subgradient [121], and cutting-plane consensus [122] have been proposed, challenges remain, such as primal feasibility recovery and the need for agents to agree on the complete solution vector. On the contrary, our approach aims to avoid requiring agents to agree on the complete solution vector, as averaging mechanisms can be undesirable in certain applications.

In [66], a distributed iterative CG method that induces cooperation was proposed, based on a distributed penalization method [123], where the local performance index of each agent also incorporates terms related to the collective goal. However, that scheme is not able to obtain in a distributed way the same solution as the centralized CG approach. Moreover, because the local decision variables are made available, the privacy of the agents might result compromised. In order to enhance the resilience, in the proposed approach privacy is prioritized by preventing agents from communicating estimates of local decision variables, costs, or constraints.

Based on the above considerations, in this chapter, we propose a distributed cooperative command governor supervision scheme applying the distributed optimization method proposed in [124] to solve the associated constraint-coupled convex optimization problem. The Relaxation and Successive Distributed Decomposition (RSDD) algorithm, characterized by its simplicity and local optimization structure, leverages relaxation techniques and duality theory to ensure convergence to optimal cost and primal recovery. This approach allows each node to compute its part of the optimal solution without relying on averaging mechanisms, and, notably, privacy preservation through the exchange of only auxiliary variables. By applying this method at each sampling time, agents can asymptotically compute their respective portions of the global optimal solution of the original centralized problem. However, since feasibility for each agent is reached only asymptotically, our approach requires to slightly tighten the coupling constraints to achieve in a

finite number of iterations a solution that is feasible with respect to original constraints, with such a number of iterations being determined by a stopping criterion based on coupling constraint violations. This results in a feasible solution that is a near-optimal solution of the original centralized solution.

While the developed approach has broader applicability, its initial motivation stemmed within the specific context of marine surface vehicles. Consequently, in line with the settings and motivations of this dissertation, simulations are conducted using marine surface vehicles operating in a 2D space with connectivity constraints to assess the effectiveness of the proposed approach.

The chapter is organized as follows. Section 4.2 formally states the problem. Section 4.3 briefly introduces the theory behind the RSDD algorithm, presents the distributed cooperative CG solution and describes its properties, while Section 4.4 presents the simulation results. Finally, Section 4.5 summarizes the main highlights of this approach and concludes the chapter.

4.2 Problem Formulation

Consider $N \in \mathbb{N}$ linear time-invariant closed-loop discrete-time dynamical systems, where each i-th subsystem is modeled as in (2.24). In scenarios involving multiple agents, the enforcement of coupling constraints is often required. These constraints arise in various applications, such as resource sharing among multiple systems or maintaining connectivity between mobile robots. In order to explicitly express the contribution of the i-th agent, the coupling constraint can be expressed in the form in the form $\sum_{i=1}^{N} h_i^c(g_i, x) \leq 0$, where each $h_i^c : \mathbb{R}^{m_i+n} \to \mathbb{R}^s$ is assumed to be convex. Considering (2.29) and $\bar{\Psi}_i = w_i \Psi_i$, the optimization problem 2.28 can be recast as

$$\begin{aligned} \min_{g_1,\dots,g_N} \quad & \sum_{i=1}^{N} f_i(g_i, r_i(t)), \\ \text{s.t.} \quad & g_i \in \mathcal{V}_i^l(x_i(t)), \qquad i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} h_i^c(g_i, x_i(t), [x]_i(t)) \leq 0, \end{aligned} \tag{4.1}$$

where, for all $i \in \{1,\dots,N\}$, $f_i(g_i, r_i(t)) = \|g_i - r_i(t)\|^2_{\bar{\Psi}_i}$, the set $\mathcal{V}(x)$ has been split, $\mathcal{V}_i^l(\cdot) \subseteq \mathbb{R}^{m_i}$ (where the superscript l stands for "local") is a non-empty, compact, convex set, and $h_i^c : \mathbb{R}^{m_i+\check{n}_i} \to \mathbb{R}^s$, with s that depends on κ_0[1][2] and $\check{n}_i = n_i + |\mathcal{N}_i|$, where $|\mathcal{N}_i|$ is the cardinality of $\mathcal{N}_i$.

[1] In this chapter, to avoid confusion, the *admissibility index* k_0, presented in Section 2.2.3, is referred to as κ_0, since k here is later used to indicate the iterations.

[2] s has, for each type of coupling constraint, κ_0 plus 1 (steady state) constraints.

Since a centralized implementation of the optimization problem (4.1) requires a central computational facility with access to all system variables, the problem of interest here focuses on designing a distributed solution that involves N computational nodes, each with limited information about the overall system, where each node determines locally an admissible command $g_i(t)$ at each time step t. In this context, a network of N agents that communicate via a *connected*, *undirected* graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where $\mathcal{A} = \{1, \ldots, N\}$, is considered.

Assumption 2. *It is assumed that each agent $i \in \mathcal{A}$, has knowledge only about its local constraint function $\mathcal{V}_i^l(\cdot)$, its local utility function f_i, and its own contribution h_i^c to the coupling. To preserve privacy, agents are required not to share information about admissible commands, costs, or constraints.*

Problem Statement

The problem to be solved in the distributed setting, under Assumption 2, can be stated as follows.

Problem 4.1 *Locally determine , at each time step t, for all $i \in \mathcal{A}$, a suitable command signal $g_i(t)$ such that the resulting aggregate vector $g(t)$ is the optimal solution for Problem 4.1.*

4.3 Main Results

In this section, the main goal is to formally state a distributed algorithm from the perspective of node i to solve Problem 4.1. More in detail, it is necessary to optimize a global-objective function which is a combination of local-objective functions. Upon inspection of such Problem, it becomes evident that it is a specific instance of a more general optimization problem, i.e. a convex optimization problem with separable costs and coupling constraints. This implies that the local objective functions and constraints decompose over the components of the decision vector. In this particular context, approaches that rely on using Lagrangian dual-decomposition and dual methods for solving the problem are of interest, especially because duality yields distributed methods primarily for separable problems [103].

As mentioned in the introduction of this Chapter, although some works exist on distributed algorithms for constraint-coupled optimization problems, some challenges remain, such as primal feasibility recovery and the need for agents to agree on the complete solution vector. To address these issues, this section reviews the distributed optimization method from [124] and applies it to Problem 4.1. The key contributions of [124] include a novel distributed

algorithm for constraint-coupled optimization problems that offers local computations, direct computation of primal optimal solution components without averaging, and, notably, privacy preservation through the exchange of only auxiliary variables. As presented in [124], the Relaxation and Successive Distributed Decomposition (RSDD) method is an approach that utilizes relaxation and successive dual transformations on a constraint-coupled optimization problem. A key advantage of the relaxation approach is its ability to bypass the need for prior verification of local problem feasibility by permitting (scalar) violations of local coupling constraints and enabling distributed implementation.

As more explicitly detailed in the following, through successive applications of the duality principle, the relaxed optimization problem can be solved by applying dual decomposition and subsequently the subgradient method. In the general formulation, the RSDD is applied to a constraint-coupled optimization problem with separable cost function that shares the same structure optimization problem (4.1) and is subject to the same assumptions. More in detail, the optimization problem (4.1), hereafter referred to as primal problem, satisfies also the well-known Slater's constraint qualification [125]. Note that $f^{t,\star}$ represents the optimal solution value of the primal centralized problem at time t, i.e.

$$\begin{aligned} & f^{t,\star} = \|g^\star - r(t)\|_\Psi^2, \\ & \text{s.t.} \quad g^\star \in \mathcal{V}(x(t)). \end{aligned} \tag{4.2}$$

Since as an initial step a dual problem of (4.1) is derived, strong duality is required to ensure that the primal and dual solutions coincide, and, under the stated assumptions, this condition holds. For sake of simplicity of the presentation, time dependence will be temporarily omitted, assuming a fixed time instant t, unless explicitly stated otherwise. In this case, the objective function $f_i(g_i, r_i(t))$, the optimal solution $f^{t,\star}$ and the coupling constraints $\sum_{i=1}^N h_i^c(g_i, x_i(t), [x]_i(t))$ in (4.1) are respectively referred to simply as $f_i(g_i)$, $f^\star$ and $\sum_{i=1}^N h_i^c(g_i)$.

4.3.1 Relaxation and Duality Steps

Let $\boldsymbol{\mu} \in \mathbb{R}^s$ be a multiplier associated to the inequality constraint $\sum_{i=1}^N h_i^c(g_i) \leq 0$. Then, the dual[3] of problem (4.1) can be formulated as

[3] To improve the visual differentiation between dual and primal variables, dual problem-related elements are denoted in bold.

$$\begin{aligned} \max_{\boldsymbol{\mu}\in\mathbb{R}^s} \ & \sum_{i=1}^{N} q_i(\boldsymbol{\mu}) \\ \text{s.t.} \ & \boldsymbol{\mu} \geq 0, \end{aligned} \tag{4.3}$$

where, for all $i \in \{1, \ldots, N\}$, each term q_i is expressed as

$$q_i(\boldsymbol{\mu}) = \min_{g_i \in \mathcal{V}_i^l} (f(g_i) + \boldsymbol{\mu}^T h_i^c(g_i)), \tag{4.4}$$

and let $q^\star$ be the optimal cost of (4.3), and it is attained at some $\boldsymbol{\mu}^\star \geq 0$, i.e., $q(\boldsymbol{\mu}^\star) = q^\star$, cf. [126, Proposition 5.1.4]. Finally, since $\sum_{i=1}^{N} q_i(\boldsymbol{\mu})$ is the dual function of (4.1), then it is concave on its convex and non-compact domain set $\{\boldsymbol{\mu} \in \mathbb{R}^s \,|\boldsymbol{\mu} \geq 0\}$.

As mentioned above, under convex inequality constraints, the strong duality theorem [126, Proposition 5.3.1] applies, thus optimization problems (4.1) and (4.3) have the same optimal cost $f^\star = q^\star$. Once the dual problem has been stated, by means of using algorithms such as [127], optimization problem (4.3) could be directly solved in a distributed way. However, since primal recovery is not guaranteed with such dual approaches, additional schemes must be employed to regain it, e.g. averaging mechanisms. To avoid the use of such additional schemes, the RSDD distributed algorithm, which overtakes these issues, follows an alternative approach that relies on a further duality step. To this end, the following optimization problem, similar to (4.3), is considered

$$\begin{aligned} \max_{\boldsymbol{\mu}\in\mathbb{R}^s} \ & \sum_{i=1}^{N} q_i(\boldsymbol{\mu}) \\ \text{s.t.} \ & \boldsymbol{\mu} \geq 0, \boldsymbol{\mu}^T \mathbf{1} \leq M, \end{aligned} \tag{4.5}$$

where $M > 0$, $M \in \mathbb{R}$ and $\mathbf{1} = [1, \ldots, 1]^T$. This problem is a restricted version of problem (4.3), since the additional constraint $\boldsymbol{\mu}^T \mathbf{1} \leq M$ has been added. The main purpose of this restriction is to ensure a compact constraint domain set, i.e.

$$\{\boldsymbol{\mu} \in \mathbb{R}^s \,|\boldsymbol{\mu} \geq 0, \boldsymbol{\mu}^T \mathbf{1} \leq M\}. \tag{4.6}$$

A compact constraint domain set is desirable, as the subsequent application of duality to (4.5) necessitates a problem with a compact domain. More in detail, the compact constraint domain in (4.5) guarantees that the subsequent dual problem, derived from (4.5), has a finite solution for all values in its domain, as these solutions involve maximizing over the compact domain set (4.6), thus making the dual of (4.5) unconstrained. A more detailed explanation will be provided later. Notice that, if $\boldsymbol{\mu}^\star$ is an optimal solution of (4.3) and the positive scalar M satisfies $M > \|\boldsymbol{\mu}^\star\|_1$, then $\boldsymbol{\mu}^\star$ is an optimal solution also for (4.5) and problems (4.3) and (4.5) have the same optimal cost $q^\star$ [124, Lemma III.1]. In other words, the presence of the constraint $\boldsymbol{\mu}^T \mathbf{1} \leq M$ in (4.5) does not alter the optimal solutions of the unrestricted problem (4.3).

From a different perspective, the restricted problem (4.5) is the dual of a *relaxed* version of the original primal problem (4.1), i.e.

$$\begin{aligned} \min_{g_1,\dots,g_N,\rho} \quad & \sum_{i=1}^{N} f_i(g_i) + M\rho, \\ \text{s.t.} \quad & \rho \geq 0,\ g_i \in \mathcal{V}_i^l, \qquad i \in \{1,\dots,N\} \\ & \sum_{i=1}^{N} h_i^c(g_i) \leq \rho \mathbf{1}, \end{aligned} \tag{4.7}$$

and strong duality holds between (4.5) and (4.7) [124, Lemma III.2]. Notice that problem (4.7) is a relaxation of problem (4.1) since a positive violation of the coupling constraint is allowed. At the same time, the violation ρ is penalized with a scaling factor M in order to discourage it. The following Lemma states the relationship between the optimal solutions of the original (4.1) and relaxed (4.7) primal Problems.

Lemma 4.2 ***([124, Proposition III.3]):*** *Let M be such that $M > \|\boldsymbol{\mu}^\star\|_1$ with $\boldsymbol{\mu}^\star$ an optimal solution of the dual of problem* (4.1)*. The optimal solutions of the relaxed problem* (4.7) *are in the form $(g_1^\star, \dots, g_N^\star, 0)$, where $(g_1^\star, \dots, g_N^\star)$ is an optimal solution of* (4.1)*, i.e. solutions of* (4.7) *must have $\rho^\star = 0$.*

Now that a relationship has been stated between the unrestricted dual (4.3) and the restricted dual (4.5) problems, the focus will be on an algorithm designed to solve the latter. In order to make problem (4.5) amenable for a distributed solution, a sparsity structure that matches the network can be enforced. To this end, copies of the common optimization variable $\boldsymbol{\mu}$ are introduced and its domain copied. Moreover, coherence constraints among the copies $\boldsymbol{\mu}_i$ having the sparsity of the connected graph $\mathcal{G}$ are also enforced, thus obtaining

$$\begin{aligned} \max_{\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_N} \quad & \sum_{i=1}^{N} q_i(\boldsymbol{\mu}_i) \\ \text{s.t.} \quad & \boldsymbol{\mu}_i \geq 0, \boldsymbol{\mu}_i^T \mathbf{1} \leq M, \qquad i \in \{1,\dots,N\} \\ & \boldsymbol{\mu}_i = \boldsymbol{\mu}_j, \qquad\qquad (i,j) \in \mathcal{E}. \end{aligned} \tag{4.8}$$

Notice that since the following is an equivalent version of (4.5) and merely a rewriting that considers the communication network, it has the same optimal cost $q^\star = f^\star$ under the assumption of $M > \|\boldsymbol{\mu}^\star\|_1$. Now, with the aim of obtaining a distributed algorithm, starting from (4.8) a dual decomposition approach is used. More in detail, the leading idea is to derive the dual of problem (4.8) and apply a subgradient method to solve it. When deriving the dual problem of (4.8), an important step is to dualize only the constraints $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j$ with $(i,j) \in \mathcal{E}$, and not local constraints $\{\boldsymbol{\mu}_i \geq 0 | \boldsymbol{\mu}_i^T \mathbf{1} \leq M\}$ for $i \in \{1,\dots,N\}$. Consider the partial Lagrangian

$$\mathcal{L}(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N, \boldsymbol{\Lambda}) = \sum_{i=1}^{N} (q_i(\boldsymbol{\mu}_i) + \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ij}^T (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)), \tag{4.9}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{s_e}$, with $s_e = s|\mathcal{E}|$ and $|\mathcal{E}|$ the cardinality of $\mathcal{E}$, is the vector stacking each Lagrange multiplier $\boldsymbol{\lambda}_{ij} \in \mathbb{R}^s$, with $(i,j) \in \mathcal{E}$, associated to the constraint $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j = 0$. Since the communication graph $\mathcal{G}$ is *undirected*, the symmetry of the constraints can be exploited, i.e. for each $(i,j) \in \mathcal{E}$ there exists also $(j,i) \in \mathcal{E}$. Consequently, when expanding all the terms in (4.9), for given i and j, both the terms $\boldsymbol{\lambda}_{ij}^T(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$ and $\boldsymbol{\lambda}_{ji}^T(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)$ are always present. Through simple algebraic manipulations, (4.9) can be equivalently expressed as

$$\mathcal{L}(\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N, \boldsymbol{\Lambda}) = \sum_{i=1}^{N} (q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^T \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji})), \tag{4.10}$$

which is separable with respect to $\boldsymbol{\mu}_i$, $i \in \{1, \dots, N\}$. Thus the dual function of (4.8) is

$$\eta(\boldsymbol{\Lambda}) = \sum_{i=1}^{N} \eta_i(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}), \tag{4.11}$$

where, for all $i \in \{1, \dots, N\}$,

$$\eta_i(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i})) = \sup_{\boldsymbol{\mu}_i \geq 0, \boldsymbol{\mu}_i^T \mathbf{1} \leq M} (q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^T \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji})). \tag{4.12}$$

Now, the dual of problem (4.8) can be formulated as

$$\min_{\boldsymbol{\Lambda} \in \mathcal{D}_\eta} \eta(\boldsymbol{\Lambda}) = \min_{\boldsymbol{\Lambda} \in \mathcal{D}_\eta} \sum_{i=1}^{N} \eta_i(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}). \tag{4.13}$$

where the domain of η is expressed as $\mathcal{D}_\eta = \{\boldsymbol{\Lambda} \in \mathbb{R}^{s_e} \,|\eta(\boldsymbol{\Lambda}) < +\infty\}$. Moreover, its cost function $\eta(\boldsymbol{\Lambda})$ is very structured since it is a sum of contributions η_i and each of them depends only on neighboring variables. Please notice that the optimization problem (4.13) is convex and unconstrained [124, Lemma III.5] since the domain $\mathcal{D}_\eta$ of η is $\mathbb{R}^{s_e}$. More in detail, for each $i \in \{1, \dots, N\}$, the function η_i as defined in (4.12) is obtained by maximizing a (concave) continuous function (q_i plus a linear term) over the compact set (4.6) and, thus, has always a finite value for all values of $(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i})$.

Remark 4.3 Please notice that, if during this second dual transformation step the non-restricted dual problem (4.3) (analogously a non-relaxed version of primal (4.1)) is considered, the resulting dual problem would not have an as a domain $\mathbb{R}^{s_e}$ and would thus be constrained. In this case, the counterpart of (4.11) is $\eta^{\mathrm{n}}(\boldsymbol{\Lambda}) = \sum_{i=1}^{N} \eta_i^{\mathrm{n}}(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i})$, with

$$\eta_i^{\mathrm{n}}(\{\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}\}_{j \in \mathcal{N}_i}) = \sup_{\boldsymbol{\mu}_i \geq 0} (q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^T \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ij}^T (\boldsymbol{\lambda}_{ij} - \boldsymbol{\lambda}_{ji})). \tag{4.14}$$

In this case the domain of $\eta^{\mathrm{n}}(\boldsymbol{\Lambda})$ is not $\mathbb{R}^{s_e}$ because the functions η_i^{n} are obtained by maximizing a concave function over a non-compact set, which may not yield finite values for all $\boldsymbol{\lambda}_{ij}, \boldsymbol{\lambda}_{ji}$.

Before presenting a distributed dual decomposition algorithm to solve (4.5) based on a subgradient iteration applied to problem (4.13), it is needed to establish the equivalence of the optimal solutions of (4.1) and (4.13). Strong duality holds between (4.13) and (4.8) (that is an equivalent version of (4.5)[124, Lemma III.6]), and (4.8) has the same solution of (4.3) [124, Lemma III.1], and the established strong duality between (4.3) and (4.1), collectively guarantee this equivalence.

4.3.2 Distributed Subgradient Method

Now that the relationship between primal (4.1) and dual of dual restricted (4.13) problems has been detailed, a distributed formulation can be presented. Exploiting the separability of η in (4.11), the computation procedure of each component of a subgradient of η at a given $\boldsymbol{\Lambda} \in \mathbb{R}^{s_e}$, see e.g., [126, Section 6.1], is recalled. More in detail,

$$\frac{\tilde{\partial}\eta(\boldsymbol{\Lambda})}{\partial \boldsymbol{\lambda}_{ij}} = \boldsymbol{\mu}_i^\star - \boldsymbol{\mu}_j^\star, \tag{4.15}$$

where $\frac{\tilde{\partial}\eta(\boldsymbol{\Lambda})}{\partial \boldsymbol{\lambda}_{ij}}$ denotes the component associated to the variable $\boldsymbol{\lambda}_{ij}$ of a subgradient of η, and

$$\boldsymbol{\mu}_i^\star = \arg \max_{\boldsymbol{\mu}_i \geq 0, \boldsymbol{\mu}_i^T \mathbf{1} \leq M} (q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^T \sum_{h \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ih} - \boldsymbol{\lambda}_{hi})), \tag{4.16}$$

$$\boldsymbol{\mu}_j^\star = \arg \max_{\boldsymbol{\mu}_j \geq 0, \boldsymbol{\mu}_j^T \mathbf{1} \leq M} (q_j(\boldsymbol{\mu}_j) + \boldsymbol{\mu}_j^T \sum_{l \in \mathcal{N}_j} (\boldsymbol{\lambda}_{jl} - \boldsymbol{\lambda}_{lj})). \tag{4.17}$$

Having established the method for computing subgradients of η, the subgradient method can be applied to solve problem (4.13). At each iteration k, each node $i \in \{1, \ldots, N\}$
(A.S.-1) receives $\boldsymbol{\lambda}_{ji}^k$ from neighbors $j \in \mathcal{N}_i$ and computes $\boldsymbol{\mu}_i^{k+1}$ as an optimal solution of

$$\max_{\boldsymbol{\mu}_i \geq 0, \boldsymbol{\mu}_i^T \mathbf{1} \leq M} (q_i(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^T \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^k - \boldsymbol{\lambda}_{ji}^k)), \tag{4.18}$$

(A.S.-2) receives updated $\boldsymbol{\mu}_j^{k+1}$, $j \in \mathcal{N}_i$ and updates $\boldsymbol{\lambda}_{ij}$, $j \in \mathcal{N}_i$ using

$$\boldsymbol{\lambda}_{ij}^{(k+1)} = \boldsymbol{\lambda}_{ij}^{(k)} - \gamma^k \Big(\boldsymbol{\mu}_i^{(k+1)} - \boldsymbol{\mu}_j^{(k+1)} \Big),$$

where γ^k denotes the step-size. Note that the subgradients of η are uniformly bounded, as, from equation (4.17), they are obtained by maximizing a concave

function over a compact domain [124, Lemma III.7]. In contrast, the non-relaxed problem does not ensure the boundedness of $\boldsymbol{\mu}_i^\star$ and $\boldsymbol{\mu}_j^\star$, or their difference, for a given $\boldsymbol{\Lambda}$. Furthermore, considering Remark 4.3, the non-relaxed problem (4.14) η_i^{n} is a constrained minimization. Consequently, to apply the subgradient method, the algorithmic steps (A.S.-1)–(A.S.-2) have to include an additional projection step, i.e. $\Delta^{k+1} = [\tilde{\boldsymbol{\Lambda}}^{k+1}]_{\mathcal{D}_i^{\mathrm{n}}}$, where each component $\tilde{\lambda}_{ij}^{k+1}$ of $\tilde{\boldsymbol{\Lambda}}^{k+1}$ is the result of (A.S.-2), where $[\cdot]_{\mathcal{D}_i^{\mathrm{n}}}$ denotes the Euclidean projection onto $\mathcal{D}_i^{\mathrm{n}}$. Since the entire $\tilde{\boldsymbol{\Lambda}}^{k+1}$ needs to be projected, a distributed implementation of the algorithm is prevented.

In its current form, (4.18) is not directly implementable due to the fact that the functions q_i, for all agents, are still not explicitly available. To address this, a reformulation is necessary. By substituting the definition of q_i (4.4) into (4.18), and using (4.8) to replace $\boldsymbol{\mu}$ with $\boldsymbol{\mu}_i$, the following formulation is derived

$$\max_{\boldsymbol{\mu}_i \geq 0, \boldsymbol{\mu}_i^T \mathbf{1} \leq M} \min_{g_i \in \mathcal{V}_i^l} (f_i(g_i) + \boldsymbol{\mu}_i^T (h_i^c(g_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{ij}^k - \boldsymbol{\lambda}_{ji}^k))). \tag{4.19}$$

It is evident that (4.19) is an explicit reformulation of (4.18). To summarize, (4.19) is an equivalent formulation of (4.18). The distributed problem (4.18) is derived from the dual problem (4.13), which, in turn, as showed above, has the same solution of the original primal problem (4.1). Therefore, solving (4.19) indirectly solves the original problem (4.1). As a final step, [124, Lemma IV.2] establishes the equivalence between (4.19) and the following problem

$$\begin{aligned} &\min_{g_i, \rho_i} \ f_i(g_i) + M\rho_i \\ &\text{s.t. } \ \rho_i \geq 0, \ \ g_i \in \mathcal{V}_i^l \\ &\quad h_i^c(g_i) + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k)} - \boldsymbol{\lambda}_{ji}^{(k)} \right) \leq \rho_i \mathbf{1}. \end{aligned} \tag{4.20}$$

Consequently, (4.13) can be solved using steps (A.S.-1) and (A.S.-2) by replacing in (A.S.-1) (4.18) with (4.20). It is important to note that ρ_i does not represent a local estimate of ρ, but rather represents the individual contribution of agent i to the global violation ρ.

To summarize the above detailed steps, strong duality holds between (4.1) and its dual (4.3), ensuring that they share the same optimal solution. If $M > \|\mu^\star\|_1$ [124, Lemma III.1], then (4.3) and its restricted formulation (4.5) share the same optimal solution. The restricted dual problem (4.8) is an equivalent reformulation of (4.5) that exploits the network structure. Another duality step is applied to (4.8), leading to (4.13). To have a distributed formulation, the subgradient method is applied to (4.13), leading to (4.18). Substituting the definition of q_i into (4.18) leads to problem (4.19). Then, the equivalence between (4.19) and (4.20) is established. The resulting RSDD algorithm,

applied to (4.1), from the perspective of node i at time t (assumed fixed), is based on the following steps

1. calculation of $((g_i^{(k+1)}, \rho_i^{(k+1)}), \mu_i^{(k+1)})$ as a primal-dual optimal solution pair of

$$\begin{aligned} &\min_{g_i, \rho_i} \; f_i(g_i, r_i(t)) + M\rho_i \\ &\text{s.t. } \rho_i \geq 0, \;\; g_i \in \mathcal{V}_i^l(x_i(t)) \\ &\quad h_i^c(g_i, x_i(t), [x]_i(t)) + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k)} - \boldsymbol{\lambda}_{ji}^{(k)} \right) \leq \rho_i \mathbf{1}, \end{aligned} \tag{4.21}$$

2. update of the neighboring variables $\boldsymbol{\lambda}_{ij}$ for all $j \in \mathcal{N}_i$

$$\boldsymbol{\lambda}_{ij}^{(k+1)} = \boldsymbol{\lambda}_{ij}^{(k)} - \gamma^k \left(\boldsymbol{\mu}_i^{(k+1)} - \boldsymbol{\mu}_j^{(k+1)} \right). \tag{4.22}$$

Basically, the resulting algorithm consists in an iterative two-step procedure. Each node $i \in \{1, \ldots, N\}$ maintains a set of variables $((g_i, \rho_i), \boldsymbol{\mu}_i) \in \mathbb{R}^{n_i} \times \mathbb{R} \times \mathbb{R}^s$, representing the primal-dual optimal solution pair of problem (4.21), which is a localized version of the centralized problem (4.1). This localized problem replaces global coupling with local terms involving neighboring variables $\boldsymbol{\lambda}_{ij}$ and $\boldsymbol{\lambda}_{ji}$. Additionally, it relaxes the coupling constraint by introducing a positive violation ρ_i. In addition to minimizing the local f_i, the violation $M\rho_i$, with $M > 0$, is also incorporated into the cost function. At each iteration, neighboring nodes exchange updated $\boldsymbol{\lambda}_{ij}$ values, which are updated using (4.22) based on neighboring $\boldsymbol{\mu}_i$ values. Agents initialize $\boldsymbol{\lambda}_{ij}, j \in \mathcal{N}_i$ arbitrarily and use a step size γ^k.

4.3.3 Cooperative Distributed Command Governor

In the context of the Command Governor, the optimization problem changes at every time step t, with both the local constraint set and the objective function changing due to variations in the initial state $x(t)$ and reference $r(t)$. Even if the specific objective function and initial state may vary at each time step, the problem structure remains consistent with assumptions like convexity and compactness.

Lemma 4.4 ***([124, Theorem II.6]):*** *Let $\boldsymbol{\mu}^\star$ be an optimal solution of the dual problem of problem* (4.21) *and $M > 0$ be such that $M > \|\boldsymbol{\mu}^\star\|_1$ and the step-size $\gamma^k \geq 0$, for all $k \geq 0$ satisfies the diminishing condition*

$$\lim_{k \to \infty} \gamma^k = 0, \;\; \sum_{k=1}^{\infty} \gamma^k = \infty, \;\; \sum_{k=1}^{\infty} (\gamma^k)^2 < \infty. \qquad \square$$

Let $\{g_i^{(k)}(t), \rho_i^{(k)}(t)\}_{k\geq 0}$, $i \in \{1, \ldots, N\}$, *be a sequence generated by the Distributed Optimization distributed algorithm at time* t. *Then, the sequence* $\{\sum_{i=1}^{N} \left(f_i(g_i^{(k)}(t)) + M\rho_i^{(k)}(t)\right)\}_{k\geq 0}$ *converges to the optimal cost* $f^{\star,t}$ *of* (4.1). □

Basically, if the step-size satisfies the diminishing condition, it can be proved that each local solution obtained using the RSDD algorithm is guaranteed to asymptotically converge to the component of an optimal feasible solution of the original problem. Compared to [124], where the optimal solution for each agent is reached asymptotically, in our approach, agents compute an admissible solution in a finite time within an ϵ_ρ-tightened version of coupling constraints, with $\epsilon_\rho > 0$. From the perspective of the generic node i with neighborhood $\mathcal{N}_i$, a pseudo-code implementing the algorithm is reported below

---

Algorithm 1: RSDD-based Distributed Optimization

(AGENT i)
Inputs: $x_i(t)$, $[x]_i(t)$, $\boldsymbol{\lambda}_{ij}(t)$ AND $\boldsymbol{\lambda}_{ji}(t)$ FOR $j \in \mathcal{N}_i$
Outputs: $g_i(t) = g_i^{(k_f)}$, $\boldsymbol{\lambda}_{ij}(t) = \boldsymbol{\lambda}_{ij}^{(k_f)}$

---

initialization

1: SET $k = 0$, $k_f = 0$, $C_i = 0$, $C_i^l = 0$, $C_i^v = 0$, $\boldsymbol{\lambda}_{ij}^{(0)} = \boldsymbol{\lambda}_{ij}(t)$, $\boldsymbol{\lambda}_{ji}^{(0)} = \boldsymbol{\lambda}_{ji}(t)$ FOR $j \in \mathcal{N}_i$

---

main:

1: **if** $C_i == 0$ **then**
2: GATHER $\boldsymbol{\lambda}_{ji}^{(k)}$ FROM $j \in \mathcal{N}_i$
3: COMPUTE $((g_i^{(k+1)}, \rho_i^{(k+1)}), \boldsymbol{\mu}_i^{(k+1)})$ AS A PRIMAL-DUAL OPTIMAL SOLUTION PAIR OF

$$\begin{aligned} &\min_{g_i, \rho_i} \ \|g_i - r_i(t)\|_{\Psi_i}^2 + M\rho_i \\ &\text{S.T. } \rho_i \geq 0, \ \ g_i \in \mathcal{V}_i^l(x_i(t)), \\ &h_i^c(g_i, x_i(t), [x]_i(t)) + \epsilon_\rho \mathbf{1} + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k)} - \boldsymbol{\lambda}_{ji}^{(k)}\right) \leq \rho_i \mathbf{1} \end{aligned} \tag{4.23}$$

4: RECEIVE $\boldsymbol{\mu}_j^{(k)}$ FROM $j \in \mathcal{N}_i$
5: UPDATE $\boldsymbol{\lambda}_{ij}^{(k)}$ FOR ALL $j \in \mathcal{N}_i$

$$\boldsymbol{\lambda}_{ij}^{(k)} = \boldsymbol{\lambda}_{ij}^{(k-1)} - \gamma^k \left(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}\right) \tag{4.24}$$

6: **if** $\rho_i^{(k+1)} \leq \epsilon_\rho$ **then**
7: SET $C_i^l = 1$
8: **else**
9: SET $C_i^l = 0$
10: RECEIVE C_j^l FROM $j \in \mathcal{N}_i$
11: **if** $C_i^l == 1$ AND $[C_j^l]_i == 1$ **then**
12: SET $C_i^v = 1$
13: **else**

14: set $C_i^v = 0$
15: receive C_j^v from $j \in \mathcal{N}_i$
16: transmit $\boldsymbol{\mu}_i^{(k)}$ to $\mathcal{N}_i$
17: **if** $C_i^v == 1$ and $[C_j^v]_i == 1$ **then**
18: set $C_i = 1$
19: set $k_f = k$
20: set $k = k + 1$
21: go to **main**

Please notice that the number of iterations k at each time t is not fixed. This is due to the fact that the stopping criteria, satisfied by all agents, depends on coupling constraint violations. Also, the condition $\rho_i^{(k+1)} \leq \epsilon_\rho$ is eventually met, since, as proved in [124], $\lim_{k\to\infty} \rho_i^{(k)} = 0$.

Remark 4.5 Because the RSDD algorithm does not guarantee feasibility for each agent i when the iterations are stopped at any finite step, it could be useful to select constraints as in (4.23). With this choice, the local coupling constraints penalize violations of the tighter constraints given in ((4.21)) (i.e. $h_i^c(g_i) + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k)} - \boldsymbol{\lambda}_{ji}^{(k)}\right) = 0 \Longleftrightarrow h_i^c(g_i) + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k)} - \boldsymbol{\lambda}_{ji}^{(k)}\right) \leq -\epsilon_\rho \mathbf{1}$) and feasibility can be guaranteed. □

The aforementioned formulation enables us to present the Cooperative Distributed Command Governor scheme when executed at each time instant by all agents

---

Algorithm 2: Cooperative Distributed CG Algorithm (RSDD-CG)
(Agent i)
Initialize $\boldsymbol{\lambda}_{ij}(0)$ for all $j \in \mathcal{N}_i$
at each time t
1: receive $\boldsymbol{\lambda}_{ji}(t-1)$, $x_j(t)$ from each $j \in \mathcal{N}_i$
2: build vectors $[\boldsymbol{\lambda}_j]_i(t-1)$ and $[x]_i(t)$
3: compute $g_i(t)$, $\boldsymbol{\lambda}_{ij}(t)$ by means of **Algorithm 1** with inputs $x_i(t)$, $[x(t-1)]_i$, $[\boldsymbol{\lambda}_j(t-1)]_i$, $\boldsymbol{\lambda}_{ij}(t-1)$
4: apply $g_i(t)$
5: transmit $\boldsymbol{\lambda}_{ij}(t)$ and $x_i(t)$ to $\mathcal{N}_i$

Please notice that, since the $\boldsymbol{\lambda}_{ij}$ can be initialized at arbitrary values, in the proposed scheme agents use the values computed at the previous iteration.

Properties

The main properties of the described Supervision scheme are outlined in the following Theorem.

Theorem 4.6 *Consider the Cooperative Distributed CG Algorithm (Algorithm 2), let the assumptions of Lemma 4.4 hold true. Suppose that* (4.23) *is feasible at time* $t = 0$, $\rho_i \leq \epsilon_\rho$, $\forall i \in \mathcal{A}$. *Then*

1. *It is feasible for all $t > 0$. The global solution $g(0)$ computed by all agents is an admissible solution of the non-relaxed problem* (4.1)*;*
2. *At each time instant $t \in \mathbb{Z}_+$, all agents asymptotically compute their portion of an optimal solution of the original centralized problem $f^{t,\star}$;*
3. *At each time $t \in \mathbb{Z}_+$, the global action of all agents $i \in \mathcal{A}$ solving Algorithm 1, generates an admissible solution that is ϵ_k-close to the optimal solution of the original centralized problem $f^{t,\star}$.*

Proof. 1. By construction, each $g_i^{(k)}(t) \in \mathcal{V}(x_i(t))$ for all $i \in \{1, \ldots, N\}$, so that the sequences $\{g_i^{(k)}(t)\}_{k\geq 0}$ are bounded $\forall i \in \mathcal{A}$. The generated solution $g_i^{(k_f)}(0)$ satisfies the relaxed constraints (4.21)

$$h_i^c(g_i^{(k_f)}, x_i(0), [x]_i(0)) + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k_f)} - \boldsymbol{\lambda}_{ji}^{(k_f)}\right) \leq \rho_i^{(k_f)}\mathbf{1}.$$

However, considering Remark 4.5 and since $\rho_i^{(k_f)} \leq \epsilon_\rho$, then the solution of Algorithm 1 is such that

$$h_i^c(g_i^{(k_f)}, x_i(0), [x]_i(0)) + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k_f)} - \boldsymbol{\lambda}_{ji}^{(k_f)}\right) + \epsilon_\rho\mathbf{1} \leq \rho_i^{(k_f)}\mathbf{1},$$

or equivalently

$$h_i^c(g_i^{(k_f)}, x_i(0), [x]_i(0)) + \sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k_f)} - \boldsymbol{\lambda}_{ji}^{(k_f)}\right) \leq 0.$$

Summing the previous condition over $i \in \{1, \ldots, N\}$,

$$\sum_{i=1}^{N} h_i^c(g_i^{(k_f)}, x_i(0), [x]_i(0)) + \sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k_f)} - \boldsymbol{\lambda}_{ji}^{(k_f)}\right) \leq 0.$$

Denoting by e_{ij} the (i,j)-entry of the adjacency matrix of $\mathcal{G}$, since the communication graph $\mathcal{G}$ is undirected, it holds that

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{(k_f)} - \boldsymbol{\lambda}_{ji}^{(k_f)}\right) = \\ = \sum_{i=1}^{N}\sum_{j=1}^{N} e_{ij}\boldsymbol{\lambda}_{ij}^{(k_f)} + \sum_{i=1}^{N}\sum_{j=1}^{N} e_{ij}\boldsymbol{\lambda}_{ji}^{(k_f)} = 0.\end{aligned}$$

So, if a solution $g_i^{(k_f)}(0)$ exists at $t = 0$, $\forall i \in \mathcal{A}$, it is also a feasible solution of the original non-relaxed problem (4.1). Also, since $\mathcal{V}(\cdot)$ is invariant then if $g_i^{(k_f)}(0)$ exists for all $i \in \mathcal{A}$, then it is a solution for all $t \geq 0$.

2. From Lemma 4.4, we know that, at each time instant t, the sequence $\{\sum_{i=1}^{N}\left(f_i(g_i^{(k)}(t)) + M\rho_i^{(k)}(t)\right)\}_{k\geq 0}$ converges to the optimal cost $f^{\star,t}$ of the centralized problem.

3. Every convergent sequence $\{\sum_{i=1}^{N}\left(f_i(g_i^{(k)}(t)) + M\rho_i^{(k)}(t)\right)\}_{k\geq 0}$ has only one limit point, that is $f^{t,\star}$. Then, at each time t

$$\begin{aligned}
&\forall \epsilon_k \in \mathbb{R}_{>0}\ \exists\, k_f \geq 1 \text{ s.t.,}\\
&\quad k > k_f \implies |(\sum_{i=1}^{N}\left(f_i(g_i^{(k)}(t)) + M\sum_{i=1}^{N}\rho_i^{(k)}(t)\right) - f^{t,\star}| < \epsilon_k\\
&\quad \implies |(\sum_{i=1}^{N}\left(f_i(g_i^{(k)}(t)) + M\bar{\rho}(t)\right) - f^{t,\star}| < \epsilon_k.
\end{aligned} \tag{4.25}$$

□

4.4 Simulation Results

This section presents results, obtained through simulation, to validate the proposed method. While the proposed strategy is general, the motivation stemmed in the context of the supervision problem of omnidirectional marine surface vehicles. To this end, the dynamics of the i-th vehicle are modeled as described in Appendix B, and the same parameters presented in Section 3.5 are used, as this approach has proven effective for simulating the dynamics of the vehicle under consideration. Experiments involving two vehicles operating in a 2D space subject to both local and connectivity keeping convex coupling constraints are conducted using the CoGoV Toolbox [108, 109]. It is assumed that each agent i is subject to the following local

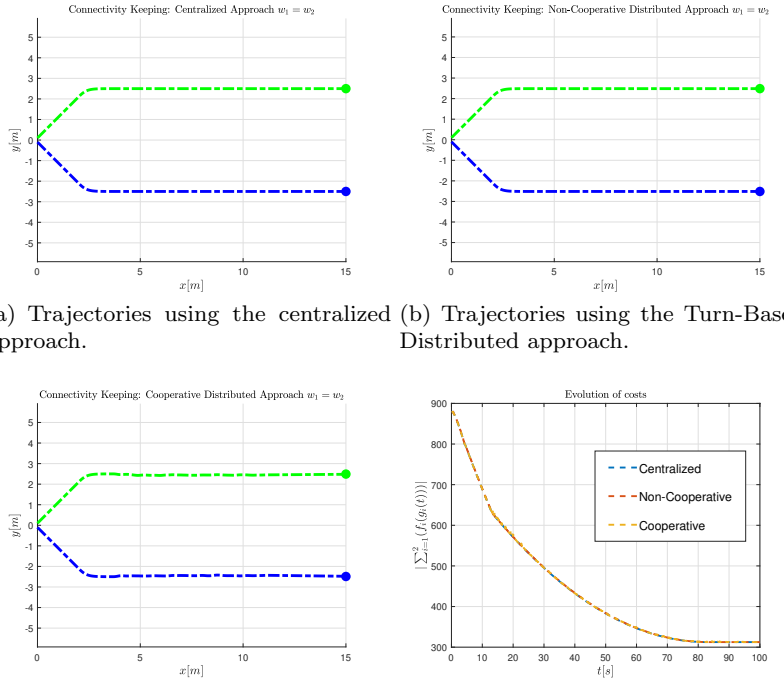
$$\begin{cases}
|T_{l,i}(t)| \leq 0.5\,[N],\\
|p_i^l(t)| \leq 20\,[m],\\
\text{with } l = \{x, y\},\ \forall t \in \mathbb{Z}_+,
\end{cases} \tag{4.26}$$

and following coordination coupling constraints

$$\Big\{\|p_i(t) - p_j(t)\|_\infty \leq d_{max},\ \forall j \in \mathcal{N}_i(t), \tag{4.27}$$

with $d_{max} = 5\,[m]$, $\mathcal{N}_{i_1}(t) = i_2$ and $\mathcal{N}_{i_2}(t) = i_1$, $\forall t \in \mathbb{Z}_+$. The simulating scenario depicted in Figure 4.1 has been considered, where the two agents, that are located initially close to each other, are instructed to reach two references that violate the coupling constraints. More in detail, the initial positions are $p_{i_1}(0) = [0, -0.1]^T$, $p_{i_2}(0) = [0, 0.1]^T$, while the given references $r_{i_1}(t) = [15, -15]^T$ and $r_{i_2}(t) = [15, 15]^T$, $\forall t \in \mathbb{Z}_+$. Due to this choice, the measured distance between the vehicles grows until they reach the maximum admissible distance allowing connectivity.

For comparison, three CG based supervision methods have been considered, that are the standard centralized CG scheme of [52], the non-cooperative Turn-Based distributed CG of [64] and the proposed cooperative distributed CG scheme. The *admissibility index* is $\kappa_0 = 20$ while weighting matrices are $\Psi_1 = \Psi_2 = I_2$. Furthermore, the value of $M > 0$ for the proposed approach has been set as $M = 1400$, $\epsilon_\rho = 1e-14$, a step-size sequence, satisfying the diminishing condition assumption, in the form $\gamma^k = \frac{1}{4}k^{-0.955}$ has been used. Additionally, $\lambda_{i_1 i_2}(0) = \lambda_{i_2 i_1}(0) = 5 \cdot \mathbf{1} \in \mathbb{R}^s$ with $s = 4(\kappa_0 + 1)$.



(a) Trajectories using the centralized approach.

(b) Trajectories using the Turn-Based Distributed approach.

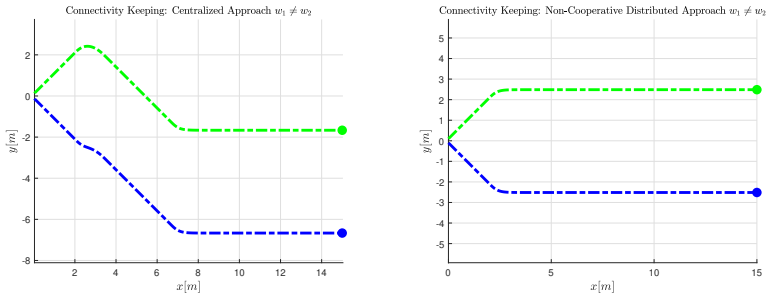
(c) Trajectories using the proposed approach.

(d) Evolution of the costs.

Fig. 4.1 Trajectories of the vehicles using the centralized approach, the Turn-Based distributed approach and the proposed cooperative distributed approach when $w_{i_1} = 1$ and $w_{i_2} = 1$.

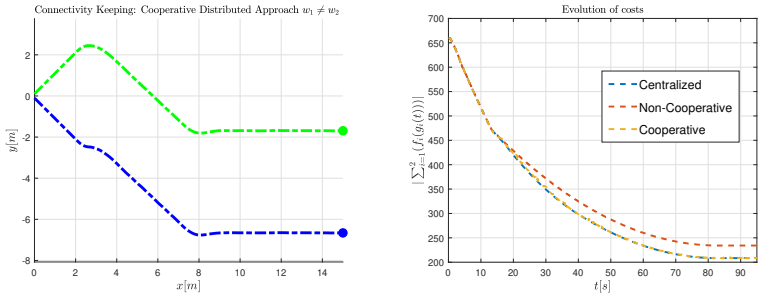
The simulations primarily seek to demonstrate that the cooperative distributed solution closely approximates the centralized solution, whereas the non-cooperative solution does not. To this end, two illustrative case studies are presented. In both cases, we consider different weight assignments for vehicles i_1 and i_2, reflecting varying designer preferences. In the first case, the weights $w_{i_1} = w_{i_2} = 1$ are equal, implying equal priority for both vehicles. Figure 4.1 compares the trajectories of the vehicles under the three

CG schemes. As depicted, all three approaches yield identical behavior and vehicles saturate the connectivity keeping constraints with $p_1^y = -2.5\,[m]$ and $p_2^y = 2.5\,[m]$. However, this is a special case; for any other choice of w_{i_1}



(a) Trajectories using the centralized approach.

(b) Trajectories using the Turn-Based Distributed approach.



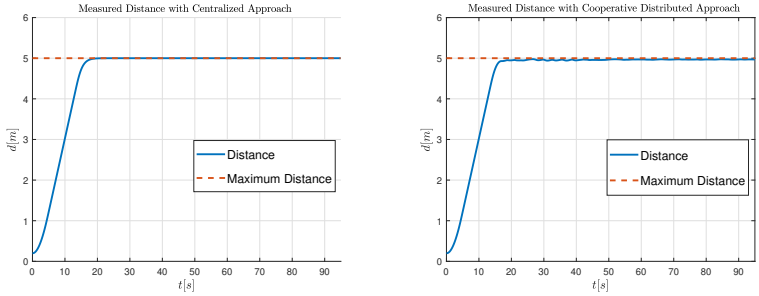
(c) Trajectories using the proposed approach.

(d) Evolution of the costs.

Fig. 4.2 Trajectories of the vehicles using the centralized approach, the Turn-Based distributed approach and the proposed cooperative distributed approach when $w_{i_1} = 1$ and $w_{i_2} = 0.5$.

and w_{i_2}, the trajectories using non-cooperative approach remain unchanged. More in detail, a designer may desires to prioritize one vehicle over another. For instance, in the second case, where vehicle i_1 has been given higher priority than vehicle i_2 ($w_{i_1} > w_{i_2}$), the centralized solution (which minimizes the combined contribution of both) prioritizes the proximity of vehicle i_1 to its target 4.2(a). In contrast, as previously mentioned, the non-cooperative distributed solution does not consider designer preferences and leads to identical trajectories 4.2(b) regardless of weight assignments. On the other hand, the cooperative distributed approach consistently yields results closer to the centralized solution 4.2(c). Furthermore, as illustrated in Figure 4.2(d), the non-cooperative solution results in higher total cost compared to the centralized and cooperative distributed approaches. However, it is important to

note that the proposed solution requires agents to compute multiple optimization problems and exchange information at each time step t, leading to a significantly higher computational load compared to the non-cooperative non-iterative distributed approach.



(a) Evolution of the distance between the two vehicles with the centralized approach.

(b) Evolution of the distance between the two vehicles using the proposed approach.

Fig. 4.3 Evolution in time of the distance between vehicle i_1 and vehicle i_2 with the centralized approach and with the proposed cooperative distributed approach when $w_{i_1} = 1$ and $w_{i_2} = 0.5$.

In Figure 4.3 the measured distance between the two vehicles at each iteration t, both using the Centralized 4.3(a) and the proposed 4.3(b) approaches, is shown. It is interesting to notice that in the latter case the measured distance never violates the connectivity keeping constraints.

Remark 4.7 Thanks to its modular nature, the ROS-based GNC architecture, presented in Section 3.4, can be adapted to take into account Assumption 2 and the proposed results can be easily integrated.

4.5 Concluding Remarks

In this chapter, a novel cooperative distributed CG scheme, for supervising dynamically coupled interconnected linear systems subject to local and coupling constraints, has been presented. The proposed scheme, based on the RSDD algorithm, addressed constrained coordination problems while ensuring privacy by avoiding the exchange of information related to objective functions, constraints, or admissible commands. Agents implementing the overall algorithm are able to find in a distributed way a near-optimal solution to the original centralized problem. The proposed approach has been simulated using the model of two omnidirectional surface vehicles subject to

connectivity constraints, and the effectiveness of the proposed approach has been validated. Future research endeavors could explore extending the results to Command Governor strategies that handle non-convex constraints, which often rely on mixed-integer optimization.

Chapter 5
Data-Driven Command Governor Schemes

Inspired by the recent development of several Data-driven Predictive Control (DPC) algorithms and leveraging conceptual tools borrowed from behavioral system theory, subspace predictive control, unfalsified control and predictive control, a method is described to address the challenge of supervising a linear time-invariant discrete-time system that bypasses explicit modeling and without relying on a parametric system representation. By means of using an input/output trajectory of the plant and a representation of the controller, the proposed data-driven CG handles explicitly both input and output constraints. According to a receding horizon control philosophy, the data-driven CG computes at each time a virtual admissible command by means of solving a convex constrained quadratic optimization problem and then feeds it to the primal system. The overall data-driven approach is proved to fulfill the constraints and it is shown that the data-driven Output Admissible Set is positively invariant and characterizable by a finite-complexity polytope. Although an input-output plant trajectory that is persistently exciting is required, this alternative to the model-based approach avoids the need for modeling, identification, and validation phases, and, compared to other data-driven approaches, the proposed solution does not require, during the online phase, a new data collection amidst controller modifications.

5.1 Introduction

One of the main issues in designing CG lies in its inherent dependence on a prediction model. Such a model-based approach becomes less suitable for applications where thorough modeling, parameter identification and validation are too costly (e.g., in robotics). Furthermore, the time-consuming nature of these steps has become evident also during the the controller design phase of the surface marine vehicles used in this thesis.

As systems become more intricate and data becomes more readily accessible, data-driven methods [128, 129] are emerging as a powerful alternative due to their efficiency and closer alignment with real-world scenarios. Indeed, in practice, one rarely has a given input/state/output representation but often may observe a trajectory of the plant. Furthermore, as exemplified by PID control [130], learning control policies directly from data can be more efficient than model learning. Compared to existing adaptive or learning-based methods [131, 132], the main advantage of data-driven schemes is that they only require an initially measured, persistently exciting data trajectory and an upper bound on the system order, but no (set-based) model description [135].

Similar to [133], the problem is approached from a behavioral systems theory perspective [75, 86, 87]. Willem's fundamental lemma [134] plays an important role in the learning and control of dynamical systems on the basis of measured data. Leveraging these results, recent Data-driven Predictive Controllers (DPC) [74] that rely on structured data to predict the future response of the controlled system (without requiring any explicit identification of a model prior to control design) have been proposed. Within a noise-free scenario, the equivalence between a DPC scheme [136] and Model Predictive Control (MPC) has been proven, thus connecting the established model-based predictive control rationale with the data-driven one. Motivated by these recent results and recognizing the shared principles of MPC and CG approaches, this work presents a data-driven Command Governor supervision scheme for plants regulated by static output-feedback controllers [137].

The recent contribution [138] focuses on the development of a Reference Governor (RG) scheme where a Hankel matrix is built considering reference and output trajectories of the closed-loop system without exploiting the possible knowledge of the stabilizing control law underlying the scheme. Here, on the contrary, a different approach is followed and, in the spirit of closed-loop data-driven simulations [84], a representation of the controller and one input/output trajectory of the plant are explicitly used in the scheme. In this way, the resulting scheme is adaptable even amidst controller modifications and constraints on input and output variables can be written explicitly.

The chapter is organized as follows. Section 5.2 briefly recalls the model-based formulation and formally states the problem in the data-driven setting. Section 5.3 presents the main results that lead to the formulation of data-driven Output Admissible Sets. Section 5.4 presents the supervision strategy and describes its properties, showing that the data-driven Output Admissible Set is positively invariant and characterizable by a finite-complexity polytope. In Section 5.5 an illustrative example is presented to validate the proposed approach, while in Section 5.6 the main conclusion are summarized.

5.2 Problem Formulation

Consider a closed-loop system described by the following LTI discrete-time model

$$\Sigma: \begin{cases} x(t+1) = \Phi x(t) + G g(t), \\ y(t) \;\; = H^y x(t) + L^y g(t), \\ u(t) \;\; = H^u x(t) + L^u g(t), \end{cases} \tag{5.1}$$

where: $t \in \mathbb{Z}_+$, $x \in \mathbb{R}^n$ is the state vector (which includes the controller states), $y \in \mathbb{R}^p$ is the output vector, $u \in \mathbb{R}^m$ is the input vector (to the plant), $g \in \mathbb{R}^p$ the manipulable command vector useful to track the local nominal reference $r \in \mathbb{R}^p$ and matrices $\Phi \in \mathbb{R}^{n \times n}$, which models the plant, with decoupled dynamics, and the controller as well, $G \in \mathbb{R}^{n \times m}$, $H^y \in \mathbb{R}^{p \times n}$, $L^y \in \mathbb{R}^{p \times p}$, $H^u \in \mathbb{R}^{m \times n}$ and $L^u \in \mathbb{R}^{m \times p}$. It is assumed that (5.1) represents an observable closed-loop dynamic system regulated by a local controller that guarantees asymptotic stability (Φ is Schur, all eigenvalues are strictly in the interior of the unit disk). In the absence of state information or measurements, inherent to data-driven settings, an output-feedback controller is considered.

The output $y(t)$ and input $u(t)$ vectors of system (5.1) have to satisfy pointwise-in-time set-membership constraints of the form

$$\begin{aligned} y(t) &\in \mathcal{Y}, \ \forall t \in \mathbb{Z}_+, \\ u(t) &\in \mathcal{U}, \ \forall t \in \mathbb{Z}_+, \end{aligned} \tag{5.2}$$

where $\mathcal{U}$ and $\mathcal{Y}$ are compact and convex sets that satisfy $0 \in \text{Int}(\mathcal{U})$ and $0 \in \text{Int}(\mathcal{Y})$.

The problem to be solved in the model-based setting, considering (5.2), can be similarly stated as presented in Section 2.2.2.

Following the standard model-based CG strategy, analogically to (2.7), virtual predictions for the input to the plant $u(t)$ and output $y(t)$ can be respectively formulated as $u(k, x(t_0), g) := H^u x(k, x(t_0), g) + L^u g$ and $y(k, x(t_0), g) := H^y x(k, x(t_0), g) + L^y g$. Then, the notion of *Output Admissible Set* (*OAS*) (2.17) related to sets $\mathcal{Y}$ and $\mathcal{U}$ in (5.2) can be introduced

$$\begin{aligned} \mathcal{Z}^y &:= \left\{ \begin{bmatrix} x \\ g \end{bmatrix} \in \mathbb{R}^{n+p} \,\middle|\, y(k, x, g) \in \mathcal{Y}, \, \forall k \in \mathbb{Z}_+ \right\}, \\ \mathcal{Z}^u &:= \left\{ \begin{bmatrix} x \\ g \end{bmatrix} \in \mathbb{R}^{n+p} \,\middle|\, u(k, x, g) \in \mathcal{U}, \, \forall k \in \mathbb{Z}_+ \right\}. \end{aligned} \tag{5.3}$$

As shown in 2.2.3, it can be proved [79] that local constraints (5.2) are always satisfied if

$$(x(t), g(t)) \in \mathcal{Z}^y \cap \mathcal{Z}^u, \forall t \in \mathbb{Z}_+. \tag{5.4}$$

In this respect, an interesting scenario is represented by the case where the sets $\mathcal{Y}$ and $\mathcal{U}$ in (5.2) are defined as two polytopes described by a set of linear inequalities in the form

$$y(t) \in \mathcal{Y} \Longleftrightarrow \mathcal{Y} := \{y \in \mathbb{R}^p : Sy \leq s\}, \tag{5.5}$$

$$S = \left[S_1^T, \cdots, S_{l_y}^T\right]^T \in \mathbb{R}^{l_y \times p}, s = \left[s_1, \cdots, s_{l_y}\right]^T \in \mathbb{R}^{l_y}, \tag{5.6}$$

and

$$u(t) \in \mathcal{U} \Longleftrightarrow \mathcal{U} := \{u \in \mathbb{R}^m : Du \leq d\}, \tag{5.7}$$

$$D = \left[D_1^T, \cdots, D_{l_u}^T\right]^T \in \mathbb{R}^{l_u \times m}, d = \left[d_1, \cdots, d_{l_u}\right]^T \in \mathbb{R}^{l_u}, \tag{5.8}$$

with $l_y \geq p$, $l_u \geq m$, S_i^T, D_i^T, s_i and d_i representing the i-th row of S, D, s and d respectively. Furthermore, rank $(S) = l_y$ and rank $(D) = l_u$.

Notice that the sets $\mathcal{Z}^u$ and $\mathcal{Z}^y$ given in (5.3) are not finitely determinable. However, as discussed in Section 2.2.3, it is shown in [139] that finitely-determined inner approximations, $\tilde{\mathcal{Z}}^u$ and $\tilde{\mathcal{Z}}^y$, with similar invariance properties can be obtained. More in detail, it can be proved that it is enough to introduce steady-state constraints restricted by a margin $\delta > 0$ and to evaluate the virtual predictions $y(k, x, g)$ and $u(k, x, g)$ within a certain *admissibility index* k_0. Then, the sets $\tilde{\mathcal{Z}}^u$ and $\tilde{\mathcal{Z}}^y$ take the following form

$$\tilde{\mathcal{Z}}^y := \left\{ \begin{bmatrix} x \\ g \end{bmatrix} \in \mathbb{R}^{n+p} \left| \begin{array}{l} S(H^y(I-\Phi)^{-1}G + L^y)g \leq s - \delta \mathbb{1}_{ly}, \\ SH^y(\Phi^k x(t) + \sum_{\tau=0}^{k-1} \Phi^{k-\tau-1} G g) \\ + SL^y g \leq s,\ k = 0, \ldots, k_0 \end{array} \right. \right\}, \tag{5.9}$$

$$\tilde{\mathcal{Z}}^u := \left\{ \begin{bmatrix} x \\ g \end{bmatrix} \in \mathbb{R}^{n+p} \left| \begin{array}{l} D(H^u(I-\Phi)^{-1}G + L^u)g \leq d - \delta \mathbb{1}_{lu}, \\ DH^u(\Phi^k x(t) + \sum_{\tau=0}^{k-1} \Phi^{k-\tau-1} G g) \\ + DL^u g \leq d,\ k = 0, \ldots, k_0 \end{array} \right. \right\}. \tag{5.10}$$

As shown in Section 2.2.3, here with a slightly different formulation, the CG directly uses $g(t) \in \mathbb{R}^p$ as an optimization variable, and solves at each time instant t the optimization problem (2.21), i.e.

$$\begin{aligned} & g(t) = \arg\min_{g} \|g - r(t)\|_{\Psi}^2 \\ & \text{s.t.} \quad g(t) \in \mathcal{V}(x(t)), \end{aligned} \tag{5.11}$$

where

$$\mathcal{V}(x(t)) := \{g \quad \in \mathbb{R}^p | \, [x \quad g]^T \in \tilde{\mathcal{Z}}^y \cap \tilde{\mathcal{Z}}^u\}. \tag{5.12}$$

Problem Statement

In this chapter, the focus is made on the supervision problem presented in Section 2.2.2 in the case when the model for system (5.1) is *unknown*, but input/output data samples from the plant and a minimal controller representation are available. The problem to be solved in the data-driven setting can be stated as follows

Problem 5.1 *Suppose that the matrices Φ, G, H^y, H^u, L^y and L^u are unavailable for the calculation of $\tilde{\mathcal{Z}}^u$, $\tilde{\mathcal{Z}}^y$ and that length-T input and output noise-free trajectories of plant of system* (5.1)*, denoted by $u^d_{[1,T]}$ and $y^d_{[1,T]}$, and a representation of the controller are available. More specifically, we assume that the input $u^d_{[1,T]}$ is persistently exciting of order $L+n$. Without identifying a parametric model of the system, characterize a data-driven $\tilde{\mathcal{Z}}^u$, $\tilde{\mathcal{Z}}^y$ to formulate a data-driven CG, i.e. determine at each time step t a suitable command signal $g(t)$ that is the "best approximation" of $r(t)$ according to the constraints (5.2).*

The superscript d is used to indicate that these are sequences of data samples collected during an offline procedure from the unknown system. More generally, instead of a single long u_d trajectory, multiple (possibly short) system trajectories could be considered [141]. To solve the problem introduced above, the predictions of the output and the input of the system, $y(t)$ and $u(t)$ as well as the steady-state output and input, y_g and u_g, are needed.

It is assumed that the input/output/state representation of (5.1) is a *minimal representation*, i.e. n is the smallest state dimension (known also as the system order). To define proper conditions on the length of the initial measured trajectory, the concept of system's *lag* has to be defined. To this end, consider the vector

$$\bar{w}(t) = \begin{bmatrix} u(t) \\ y(t) \end{bmatrix} = H^{\bar{w}} x(t) + L^{\bar{w}} g(t), \tag{5.13}$$

that collects both the input and the output of the plant, where $H^{\bar{w}} = \begin{bmatrix} H^u & 0 \\ 0 & H^y \end{bmatrix}$ and similarly for $L^{\bar{w}}$. Then, it is useful to introduce the following definitions of the Extended Observability $O^{\bar{w}}_t$ and Convolution matrices $C^{\bar{w}}_t$

$$O^{\bar{w}}_t = \begin{bmatrix} H^{\bar{w}} \\ H^{\bar{w}}\Phi \\ \vdots \\ H^{\bar{w}}\Phi^{t-1} \end{bmatrix}, C^{\bar{w}}_t = \begin{bmatrix} L^{\bar{w}} & 0 & \cdots & 0 \\ H^{\bar{w}}G & L^{\bar{w}} & \vdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ H^{\bar{w}}\Phi^{t-2}G & \cdots & H^{\bar{w}}G & L^{\bar{w}} \end{bmatrix}. \tag{5.14}$$

It is worth mentioning that, starting from any initial condition $x(t_0)$, and command sequence $g \in \mathbb{R}^{pL}$, over the horizon L (positive integer), the input $u \in \mathbb{R}^{mL}$ and the output $y \in \mathbb{R}^{pL}$ of (5.1) are given by

$$\bar{w} = O^{\bar{w}}_t x(t_0) + C^{\bar{w}}_t g, \tag{5.15}$$

over the same horizon. Finally, the system's *lag* can be defined as follows

Definition 5.2 *The lag (or observability index) of system* (5.1)*, denoted by ℓ, is defined as the smallest integer such that O_ℓ has rank n (i.e. full column rank), where n is the system order.*

Please notice that ℓ exists by definition because of the assumptions made on system (5.1). The key idea is to link the trajectory of the system to the signal reference [142], i.e.

$$\begin{bmatrix} u \\ y \end{bmatrix} = \Gamma g, \tag{5.16}$$

and, leveraging ideas from unfalsified control [140], to use directly the command $g(t)$ as an optimization variable. In unfalsified control, a candidate controller can be tested using a trajectory of the plant without having a model of the plant or applying the controller on the plant. More in detail, as depicted in Figure 5.1, the high-level supervisor module monitors plant P data (u, y) for evidence that would falsify candidate controllers K, and if the currently active controller K becomes falsified by data, then an yet unfalsified controller is switched into the loop to replace it.

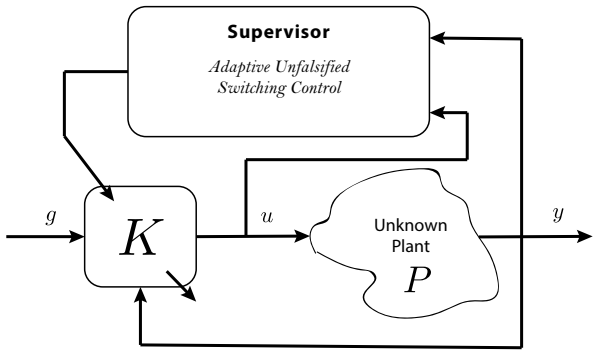


Fig. 5.1 Adaptive Unfalsified Switching Controller scheme.

5.3 Main Results

5.3.1 Data-Driven Offline Phase

Since a system model is not available, instead of using classical system theory, the behavioural system theory [75] will be considered. The key idea is to use the closed-loop data-driven simulation results, presented in Section 2.4, to link the response of a closed-loop system to a given reference signal directly from an input/output trajectory of the plant and a representation of the controller.

By referring to Figure 5.2, it is assumed that the controller $\mathscr{C}$ is specified by a kernel representation and that the plant $\mathscr{B}$ is only implicitly specified by the trajectory w_d (so a representation of $\mathscr{B}$ is not known). Furthermore, linear,

time-invariant, and finite dimensional plants and controllers are considered in this chapter. In this setting, the plant can be seen as a discrete-time dynamical system $\mathscr{B}$ with w manifest variables (inputs and outputs), where $w(t) \in \mathbb{R}^{(m+p)}$ with $t \in \mathbb{Z}_+$, as a subset of the signal space $(\mathbb{R}^w)^{\mathbb{Z}_+}$. We assume that the manifest variables w have a given input/output partition

$$w(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \\ y_1(t) \\ \vdots \\ y_p(t) \end{bmatrix}, w_{[1,T]} = \begin{bmatrix} w(1) \\ w(2) \\ \vdots \\ w(T) \end{bmatrix} = \begin{bmatrix} u(1) \\ y(1) \\ u(2) \\ y(2) \\ \vdots \\ u(T) \\ y(T) \end{bmatrix}. \tag{5.17}$$

A useful definition that guarantees that a trajectory $w_i \in (\mathbb{R}^{m+p})^L$ is a trajectory of $\mathscr{B}$ is the following

Definition 5.3 (Restriction of Behaviour): In the behavioral setting, given a discrete-time dynamical system $\mathscr{B}$ and a positive integer L, the "restricted behavior" $\mathscr{B}_{|L}$ is defined as

$$\mathscr{B}_{|L} = \{w_i \in (\mathbb{R}^{m+p})^L | \, \exists \, w_f \in (\mathbb{R}^{m+p})^{\mathbb{Z}_+} : (w_i, w_f) \in \mathscr{B}\},$$

that means that there is an extension w_f of a finite trajectory $w_i \in \mathscr{B}_{|L}$ of the system, to an infinite trajectory $w = (w_i, w_f) \in \mathscr{B}$. Following the notation presented in Section 2.4, the system given by the feedback interconnection of the plant $\mathscr{B}$ and the controller $\mathscr{C}$ is denoted as $\mathscr{B}^{\mathscr{C}}$. More in detail, a minimal kernel representation of the controller (2.36) is considered.

Remark 5.4 The assumption of a plant with decoupled dynamics serves for the only purpose of simplifying the design of an output-feedback controller, without the need to introduce additional concepts such as decouplers. Indeed, without loss of generality, system $\mathscr{B}$ may have coupled dynamics.

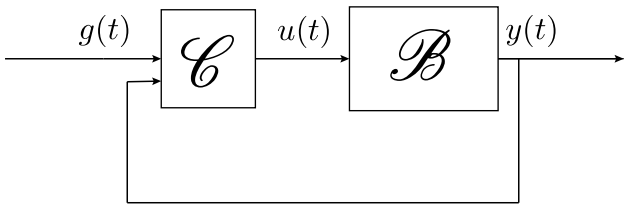


Fig. 5.2 Feedback interconnection.

To guarantee that all possible trajectories of system $\mathscr{B}$ are captured, certain conditions need to be met regarding the values of T and L. More in

detail, T denotes the length of the measured trajectory used to construct the Hankel matrix and L represents the entire time window employed in the Command Governor approach. Furthermore, L can be subdivided as $L = L_i + L_p$, where L_i (subscript referring to "initial") represents the length of the initial measured trajectory, and L_p (subscript referring to "predicted") represents the length of the predicted future trajectory. The relationship between T and L [85] is

$$T \geq (m+1)(L+n) - 1. \tag{5.18}$$

For the sake of simplicity, and to abstract from technical details, the results in the following sections are derived assuming a static output-feedback controller [137]. However, the computation of $R_r(\sigma)$ and $R_w(\sigma)$ is detailed for both Proportional and Proportional-Integral controllers. As discussed in Section 2.4, ARMA models can be extracted from the matrix transfer function of the controller, facilitating the analysis of temporal sequences and thus the explicit writing of $R_r(\sigma)$ and $R_w(\sigma)$. In the context of static proportional output-feedback controller, it is assumed that the control law $\mathscr{C}$ in Figure 5.2 is given by

$$u(t) = \mathcal{K}_r y(t) + \mathcal{K}_v g(t). \tag{5.19}$$

Choosing $\mathcal{K}_r = -\mathcal{K}$ and $\mathcal{K}_v = \mathcal{K}$

$$u(t) = \mathcal{K}(g(t) - y(t)), \tag{5.20}$$

with $\mathcal{K} \in \mathbb{R}^{m \times p}$. For the case of (5.20), the ARMA model (2.37) that relates the i-th input to the j-th tracking error

$$u_i(t) = k_{i,j} e_j(t). \tag{5.21}$$

For single-output systems, the difference operators $R_r(\sigma)$ and $R_w(\sigma)$ in (2.36) can be replaced by the structured matrices $\mathcal{F}_L(R_r) = -kI_L$, and $\mathcal{F}_L(R_w) = I_L \otimes [1 \;\; k]$ respectively, being $k \in \mathbb{R}$ the gain of the controller. In the multi-output case, it is trivial to verify that $\mathcal{F}_L(R_r)$ and $\mathcal{F}_L(R_w)$ can be expressed as

$$\mathcal{F}_L(R_r) = I_L \otimes \hat{T}_r = I_L \otimes -\mathcal{K}, \tag{5.22}$$

$$\mathcal{F}_L(R_w) = I_L \otimes \hat{T}_w = I_L \otimes [I_m \quad \mathcal{K}], \tag{5.23}$$

where $\mathcal{F}_L(R_w) \in \mathbb{R}^{mL \times (m+p)L}$, $\mathcal{F}_L(R_r) \in \mathbb{R}^{mL \times pL}$, $\hat{T}_w \in \mathbb{R}^{m \times (m+p)}$ and $\hat{T}_r \in \mathbb{R}^{m \times p}$. More in detail, in the case of (5.20), the sequences are expressed as

$$\begin{cases} u(1) = \mathcal{K}(g(1) - y(1)) \\ u(2) = \mathcal{K}(g(2) - y(2)) \\ \quad \vdots \\ u(L) = \mathcal{K}(g(L) - y(L)) \end{cases} = \begin{cases} u(1) + \mathcal{K}y(1) = \mathcal{K}g(1) \\ u(2) + \mathcal{K}y(2) = \mathcal{K}g(2) \\ \quad \vdots \\ u(L) + \mathcal{K}y(L) = \mathcal{K}g(L) \end{cases} \tag{5.24}$$

Then,

$$\begin{bmatrix} I_m & \mathcal{K} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & I_m & \mathcal{K} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & I_m & \mathcal{K} \end{bmatrix} \begin{bmatrix} u(1) \\ y(1) \\ u(2) \\ y(2) \\ \vdots \\ u(L) \\ y(L) \end{bmatrix} = - \begin{bmatrix} -\mathcal{K} & 0 & \dots & 0 \\ 0 & -\mathcal{K} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -\mathcal{K} \end{bmatrix} \begin{bmatrix} g(1) \\ g(2) \\ \vdots \\ g(L) \end{bmatrix}. \tag{5.25}$$

As shown in (2.41), the relationship between the input-output trajectory of the plant and the reference can be expressed as

$$\mathcal{F}_L(R_w)w_g = -\mathcal{F}_L(R_r)g. \tag{5.26}$$

Please notice that in [84], a set of closed-loop trajectories is computed. Here, on the contrary under suitable initial conditions, a single trajectory of the closed-loop system is selected. From (5.26) we know that

$$g = -(\mathcal{F}_L(R_r))^\dagger \mathcal{F}_L(R_w)w_g \implies g = \Gamma w_g. \tag{5.27}$$

In the general multi-output case $\mathcal{F}$ has a more complex structure compared to the one presented in (2.40). For instance, let us consider the case where the control law in $\mathscr{C}$ is a PI controller, i.e.

$$u(t) = \mathcal{K}(g(t) - y(t)) + \mathcal{K}_I \sum_{\tau=1}^{t} (g(\tau) - y(\tau)). \tag{5.28}$$

Supposing an approximation $s = \frac{2}{T}\frac{z-1}{z+1}$ in the i-j-th transfer function $C_{ij}(s) = k^c_{i,j} + k^{c,I}_{i,j}\frac{1}{s}$, the ARMA model (2.37) that relates the i-th input to the j-th tracking error has the following form

$$u_i(t) = u_i(t-1) - k_{i,j}e_j(t-1) + k^I_{i,j}e_j(t). \tag{5.29}$$

More in detail, in the case of (5.28), the sequences are expressed as

$$
\begin{cases}
u(1) = u_I(0) + \mathcal{K}(g(1) - y(1)) + \mathcal{K}_I(g(1) - y(1)) \\
u(2) = u(1) - \mathcal{K}(g(1) - y(1)) + \mathcal{K}(g(2) - y(2)) + \mathcal{K}_I(g(2) - y(2)) \\
\quad \vdots \\
u(L) = u(L-1) - \mathcal{K}(g(L-1) - y(L-1)) + \mathcal{K}(g(L) - y(L)) + \mathcal{K}_I(g(L) - y(L))
\end{cases}
$$

$$
= \begin{cases}
u(1) - u_I(0) + \mathcal{K}y(1) + \mathcal{K}_I y(1) = \mathcal{K}g(1) + \mathcal{K}_I g(1) \\
u(2) - u(1) + \mathcal{K}y(2) + \mathcal{K}_I y(2) - \mathcal{K}y(1) = \mathcal{K}g(2) + \mathcal{K}_I g(2) - \mathcal{K}g(1) \\
\quad \vdots \\
u(L) - u(L-1) + \mathcal{K}y(L) + \mathcal{K}_I y(L) - \mathcal{K}y(L-1) = \mathcal{K}g(L) + \mathcal{K}_I g(L) - \mathcal{K}g(L-1)
\end{cases} \tag{5.30}
$$

Then,

$$
\begin{aligned}
&\underbrace{\begin{bmatrix}
I_m & \mathcal{K}_{PI} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
-I_m & -\mathcal{K} & I_m & \mathcal{K}_{PI} & \dots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \dots & -I_m & -\mathcal{K} & I_m & \mathcal{K}_{PI}
\end{bmatrix}}_{\mathcal{F}_L(R_w)}
\begin{bmatrix}
\bar{u}(1) \\ y(1) \\ u(2) \\ y(2) \\ \vdots \\ u(L) \\ y(L)
\end{bmatrix} = \\
&\qquad = -\underbrace{\begin{bmatrix}
-\mathcal{K}_{PI} & 0 & \dots & 0 & 0 \\
\mathcal{K} & -\mathcal{K}_{PI} & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & \mathcal{K} & -\mathcal{K}_{PI}
\end{bmatrix}}_{\mathcal{F}_L(R_r)}
\begin{bmatrix}
g(1) \\ g(2) \\ \vdots \\ g(L)
\end{bmatrix},
\end{aligned} \tag{5.31}
$$

where $\bar{u}(1) = u(1) - u_I(0)$ and $\mathcal{K}_{PI} = \mathcal{K} + \mathcal{K}_I$. With some manipulation, matrix $\mathcal{F}_L(R_r)$ can be written in the same form as the proportional case

$$
\begin{aligned}
&\underbrace{\begin{bmatrix}
I_m & \mathcal{K}_{PI} & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
A_1 & 0 & I_m & \mathcal{K}_{PI} & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
A_2 & 0 & A_1 & 0 & I_m & \mathcal{K}_{PI} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\
A_3 & 0 & A_2 & 0 & A_1 & 0 & I_m & \mathcal{K}_{PI} & \dots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & & & & \ddots & & & & & & \vdots & \vdots \\
A_{L-2} & 0 & A_{L-3} & 0 & A_{L-4} & 0 & A_{L-5} & 0 & \dots & I_m & \mathcal{K}_p & 0 & 0 \\
A_{L-1} & 0 & A_{L-2} & 0 & A_{L-3} & 0 & A_{L-4} & 0 & \dots & A_1 & 0 & I_m & \mathcal{K}_{PI}
\end{bmatrix}}_{\mathcal{F}_L(R_w)}
\begin{bmatrix}
\bar{u}(1) \\ y(1) \\ u(2) \\ y(2) \\ \vdots \\ u(L-1) \\ y(L-1) \\ u(L) \\ y(L)
\end{bmatrix} = \\
&\qquad = -\underbrace{I_L \otimes -\mathcal{K}_{PI}}_{\mathcal{F}_L(R_r)}
\begin{bmatrix}
g(1) \\ g(2) \\ \vdots \\ g(L-1) \\ g(L)
\end{bmatrix},
\end{aligned} \tag{5.32}
$$

where

$$\begin{cases} A_1 = \mathcal{K}_i \mathcal{K}_{PI}^{\dagger}, \\ A_k = -A_1(A_{k-1}) + A_{k-1}. \end{cases} \tag{5.33}$$

Then, Γ can be computed as in (5.27).

5.3.2 Predictions

The following idea has been used in the literature for the purpose of data-driven simulation and prediction [133]. The trajectory of w is usually partitioned into two parts, one with horizon length of L_i that serves to implicitly fix the initial condition, and another of horizon length L_p that serves as the predicted trajectory

$$\begin{aligned} w_{[-L_i, L_p]}(t) &= \begin{bmatrix} w_i(t) \\ w_p(t) \end{bmatrix} \\ &= \left[\underbrace{w_{-L_i}(t), \ldots, w_{-1}(t)}_{\textit{Fixed Initial Trajectory}}, \underbrace{w_t(t), \ldots, w_{L_p}(t)}_{\textit{Predicted Trajectory}} \right]^T, \end{aligned} \tag{5.34}$$

where, with slight abuse of notation, $w_i(t) \in \mathbb{R}^{(m+p)L_i}$, $w_p(t) \in \mathbb{R}^{(m+p)L_p}$ and $L_i + L_p = L$. As a first result from (5.27), we have

$$\begin{bmatrix} \mathbf{g}_i(t) \\ \mathbf{g}_p(t) \end{bmatrix} = \Gamma \begin{bmatrix} w_i(t) \\ w_p(t) \end{bmatrix}, \tag{5.35}$$

where $\mathbf{g}_i(t) = \{g_k(t)\}_{k=-L_i}^{-1}$ and $\mathbf{g}_p(t) = \{g_k(t)\}_{k=t}^{L_p}$. For sake of simplicity of the presentation, the dependency on time t will be omitted. Considering (2.39) and (5.26)

$$\mathcal{F}_L(R_w)\mathcal{H}_L(w_{[1,T]}^d)\alpha = -\mathcal{F}_L(R_r)g. \tag{5.36}$$

Considering Lemma 5.9, if $(w_i, \mathbf{g}_i) \in \mathscr{B}_{L_i}^{\mathscr{C}}$ and $\mathbf{g}_p$ results in an admissible trajectory, one can uniquely solve for the latent initial condition and, thus, uniquely compute w_p solving (5.15). While the solution for α may not be unique [138, 84], the specific value of α itself is not relevant for the simulation problem. In other words, any, not necessarily a unique, solution for α is sufficient for the solution of the simulation problem. Often, the minimum norm solution is chosen, that is

$$\bar{\alpha} = (\mathcal{F}_L(R_w)\mathcal{H}_L(w_{[1,T]}^d))^{\dagger}(-\mathcal{F}_L(R_r))g, \tag{5.37}$$

where † denotes the Moore–Penrose pseudoinverse. Then

$$\begin{aligned}\begin{bmatrix} u_i \\ u_p \end{bmatrix} &= \mathcal{H}^u_L(u^d_{[1,T]})\bar{\alpha} \\ &= \mathcal{H}^u_L(u^d_{[1,T]})(\mathcal{F}_L(R_w)\mathcal{H}_L(w^d_{[1,T]}))^\dagger(-\mathcal{F}_L(R_r))\begin{bmatrix} \mathbf{g}_i \\ \mathbf{g}_p \end{bmatrix} \\ &= A_1\mathbf{g}_i + A_2\mathbf{g}_p = A_1\Gamma_i w_i + A_2,\end{aligned} \tag{5.38}$$

and

$$\begin{aligned}\begin{bmatrix} y_i \\ y_p \end{bmatrix} &= \mathcal{H}^y_L(y^d_{[1,T]})\bar{\alpha} \\ &= \mathcal{H}^y_L(y^d_{[1,T]})(\mathcal{F}_L(R_w)\mathcal{H}_L(w^d_{[1,T]}))^\dagger(-\mathcal{F}_L(R_r))\begin{bmatrix} \mathbf{g}_i \\ \mathbf{g}_p \end{bmatrix} \\ &= B_1\mathbf{g}_i + B_2\mathbf{g}_p = B_1\Gamma_i w_i + B_2,\end{aligned} \tag{5.39}$$

where $A_1 \in \mathbb{R}^{mL\times pL_i}$, $A_2 \in \mathbb{R}^{mL\times pL_p}$, $B_1 \in \mathbb{R}^{pL\times pL_i}$ and $B_2 \in \mathbb{R}^{pL\times pL_p}$. In principle, an accurate data-driven representation of $\mathcal{Z}^y_D$ and $\mathcal{Z}^u_D$ would require $L_p = \infty$, which is not practicable. For this reason, the data-driven version of $\tilde{\mathcal{Z}}^y$ and $\tilde{\mathcal{Z}}^u$ are introduced. Since the command $g(t)$ has to be constantly kept from t onward, the future predicted sequence of the admissible command can be written as $\mathbf{g}_p(t) = \mathbb{1}_{L_p} \otimes g(t)$. Considering sets $\mathcal{Y}$ and $\mathcal{U}$ as two polytopes (5.5)(5.7), thanks to (5.38) and (5.39), the *Output Admissible Sets* for the data-driven formulation can be expressed as

$$\tilde{\mathcal{Z}}^y_D := \left\{ \begin{bmatrix} w_i \\ g \end{bmatrix} \in \mathbb{R}^{(m+p)L_i+p} \,\middle|\, \begin{array}{l} SYg \le s - \delta\mathbb{1}_{ly}, \\ SB_1\Gamma_i w_i \\ +SB_2\mathbb{1}_{L_p} \otimes g \le s \end{array} \right\}, \tag{5.40}$$

$$\tilde{\mathcal{Z}}^u_D := \left\{ \begin{bmatrix} w_i \\ g \end{bmatrix} \in \mathbb{R}^{(m+p)L_i+p} \,\middle|\, \begin{array}{l} DA_1\Gamma_i w_i \\ +DA_2\mathbb{1}_{L_p} \otimes g \le d \end{array} \right\}, \tag{5.41}$$

$Y \in \mathbb{R}^{p\times p}$ being the DC gain matrix. Equation (5.10) holds true in the general case. Notably, when considering a static output-feedback control law (5.20), the input converges to zero in steady-state, and the enforcing of steady-state constraints on the input becomes unnecessary.

Remark 5.5 In the scenario where a different controller is used instead of (5.20), the OASs $\tilde{\mathcal{Z}}^u_D$ and $\tilde{\mathcal{Z}}^y_D$ may need to have a slightly different formulation. For example, the use of a PI controller would necessitate the inclusion of steady-state constraints in $\tilde{\mathcal{Z}}^u_D$ as done in (5.10).

Then, local constraints (5.2) are always satisfied if

$$\begin{aligned} g(t) &\in \mathcal{V}_D(w_i(t)), \\ \mathcal{V}_D(w_i(t)) &:= \{g \in \mathbb{R}^p | \, [w_i \quad g]^T \in \tilde{\mathcal{Z}}^y_D \cap \tilde{\mathcal{Z}}^u_D\}. \end{aligned} \tag{5.42}$$

The computation of $\mathcal{V}_D(w_i(t))$ previously introduced assumes that $L_i \ge \ell$ and $L_p > k_0$ [138].

5.4 Supervision Scheme

In this section, in order to solve the above stated problem 5.1, a novel supervision architecture (Figure 5.3) is introduced. The main objective of the proposed architecture is to choose, on the basis of the current past L_i measurements w_i, at each time instant, a command $g(t)$ such that the pair $(w_i(t), g(t))$ belongs to $\tilde{\mathcal{Z}}_D^y \cap \tilde{\mathcal{Z}}_D^u$ or equivalently that $g(t)$ belongs to $\mathcal{V}_D(w_i(t))$. More in

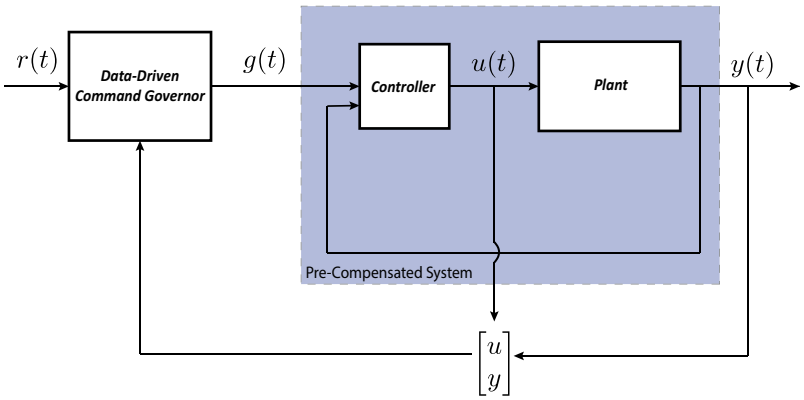


Fig. 5.3 Data-Driven Command Governor Architecture.

detail, the admissible command $g(t)$ is computed by solving the following optimization problem

$$
\begin{aligned}
g(t) &:= \arg\min_{\alpha, g} \|g - r(t)\|_{\Psi}^2 \\
\text{s.t.} &\begin{cases}
\begin{bmatrix} \mathbf{g}_i \\ \mathbf{g}_p \end{bmatrix} = \Gamma \mathcal{H}_L(w_d)\alpha, \\
\mathbf{g}_i = \Gamma_i w_i(t), \\
\mathbf{g}_p = \mathbb{1}_{L_p} \otimes g, \\
g \in \mathcal{V}_D(w_i(t)),
\end{cases}
\end{aligned}
\tag{5.43}
$$

where $\Psi = \Psi^T > 0$. This idea can be formalized as follows

Algorithm 1: Data-Driven Command Governor (DD-CG)

REPEAT AT EACH TIME t

1: COLLECT $w_i(t)$ TAKING THE PAST L_i MEASUREMENTS $u_{[-L_i,-1]}$, $y_{[-L_i,-1]}$ AND SOLVE (5.43)
2: APPLY $g(t)$
3: SET $t = t + 1$ AND GO BACK TO STEP 1.

Notice that the Data-Driven Command Governor algorithm enforces the constraints starting at $t = 0$, so an initial data trajectory $w_i(0)$ is assumed to be known. A schematic illustration of the logic behind the Data-Driven CG algorithm is depicted in Figure 5.4.

Remark 5.6 Please notice that one of the main features of the proposed approach consists in its adaptability even amidst controller modifications, making the approach more practical. Any change of the control unit does not requires reiteration of the entire process of data collection and Hankel matrix verification (using Definition 2.14), which may be inconvenient or not be feasible in all cases (i.e. it may not be possible to take the system offline). In fact, this feature would be particularly interesting in cases where the controller changes online and the closed-loop system is unavailable for offline measurement. Using the proposed approach, the Hankel matrix needs to be computed only once and it would suffice to recompute matrix Γ and sets $\tilde{\mathcal{Z}}_D^u$ and $\tilde{\mathcal{Z}}_D^y$.

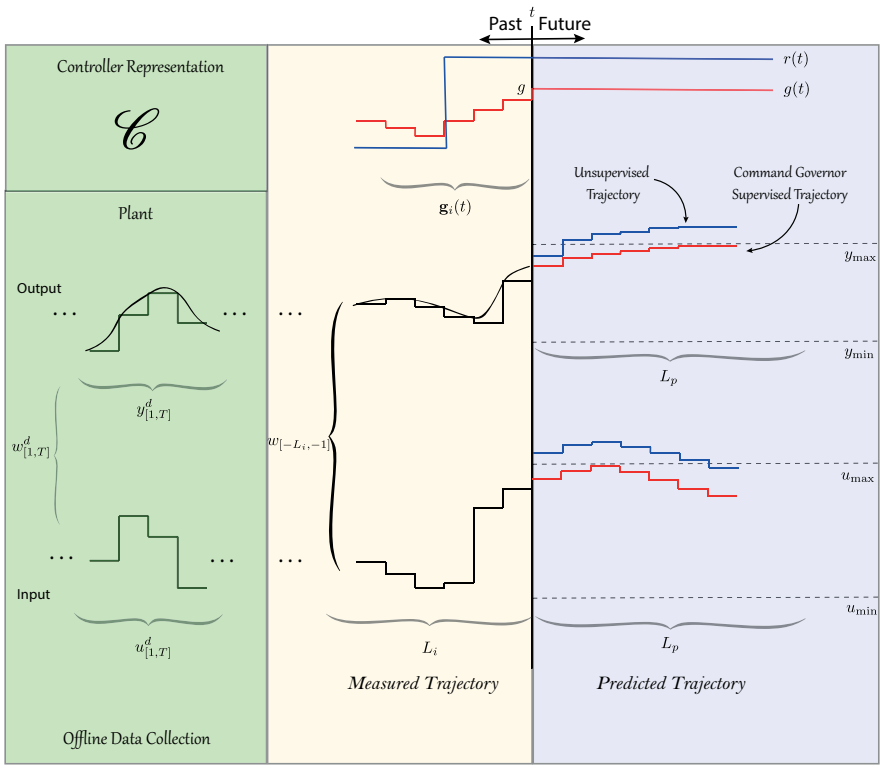


Fig. 5.4 Data-Driven Command Governor Logic.

Remark 5.7 In real-world applications, the output of an unknown LTI system is often corrupted by measurement noise. This noise introduces inaccuracies into the stacked Hankel matrices, and consequently, these matrices no longer perfectly capture the trajectory space of the system, leading to less accurate output prediction. Additionally, noisy measurements of initial conditions further degrade prediction accuracy. However, it is important to note that following ideas presented in [143, 144], the feasibility of the approach may be preserved despite the presence of bounded noise by, for instance, tightening the output constraints, as well as introducing an additional slack variable which renders the equality constraints feasible in (5.43) and whose norm is also penalized in the cost.

Properties

The main properties of the described Supervision Architecture can be summarized in the following Theorem

Theorem 5.8 *Consider system* (5.1) *along with the proposed supervision architecture, where Algorithm 1 is performed. If w_i is updated at every time instant using the noise-free input-output data collected from the system* (5.1)*, $L_i \geq \ell$, $L_p > k_0$, with $L = L_i + L_p$, $w^d_{[1,T]}$ is persistently exciting and a minimal representation of the controller is available. Suppose that* (5.43) *is feasible at time $t = 0$, then:*

1. *It is feasible for all $t > 0$;*
2. *For a constant reference $r(t) \equiv r$, the admissible command $g(t)$ computed using* (5.43) *is bounded $g(t)$ and converges in finite time. It also converges to $r(t)$, if $r(t)$ is steady-state constraint-admissible.*

Lemma 5.9 ***([133, Lemma 1]):*** *Let an output-command sequence $(\bar{w}, g) \in \mathscr{B}^{\mathscr{C}}_L$ be given, with $L \geq \ell$ and let* (5.1) *be a minimal input/output/state representation. Then, the initial condition $x(t_0) \in \mathbb{R}^n$ can be uniquely determined such that* (5.15) *holds true.*

Lemma 5.10 ***([138, Theorem 1]):*** *Following Lemma 5.9, if $g(t) \in \mathcal{V}_D(w_i(t))$ and $(w_i \mathbf{g}_i) \in \mathscr{B}^{\mathscr{C}}_{L_i}$, then there exists a unique $x(t)$ such that $g(t) \in \mathcal{V}(x(t))$.*

Proof. To prove Theorem (5.8), it is needed to analyze the relationship between $\mathcal{V}(\cdot)$ and $\mathcal{V}_D(\cdot)$. More in detail, if the two OASs $\mathcal{V}(\cdot)$ and $\mathcal{V}_D(\cdot)$ are equivalent and linked through a linear map, then the same properties of the model-based OAS, *e.g.* being positively invariant, can be transferred.
The trajectory w_i, because it is collected from the system, represents an admissible trajectory, i.e.

$$(w_i, \mathbf{g}_i) \in \mathscr{B}^{\mathscr{C}}_{L_i}. \tag{5.44}$$

Following Lemma 5.9, if (5.44) holds true, then the initial condition $x(t)$ can be uniquely determined from (5.15). Thanks to Lemma 5.10, it holds

$$g(t) \in \mathcal{V}_D(w_i(t)) \implies \exists!\ x(t) : g(t) \in \mathcal{V}(x(t)). \tag{5.45}$$

Then, the data-driven CG is equivalent to the standard, model-based CG and it produces, at every timestep, the same output $g(t)$. Stated differently, the two OASs, $\mathcal{V}(\cdot)$ and $\mathcal{V}_D(\cdot)$, are equivalent and linked through a linear map, hence the same properties of the model-based OAS, *e.g.* being positively invariant and constraint-admissible [139], can be transferred. □

5.5 Simulation Results

In this section the main goal is to simulate and therefore validate the previously proposed method. The simulations results presented in this section were carried out on a Lenovo Legion Pro 5 16IRX8 Laptop using Matlab 23.2 (Release R2023b), the Yalmip interface [105] and the Gurobi solver [107]. To investigate the proposed supervision strategy by means of simulations, the following pre-compensated discrete time system, with number of outputs $p = 2$, inputs $m = 2$ and reference $p = 2$, has been chosen, with sampling time set at $1\,[s]$, where the values of the matrices of system (5.1) are

$$\begin{aligned} \Phi &= \begin{bmatrix} 0.99 & 0 & -0.56 & 0 \\ 0 & 0.989 & 0 & 1.2708 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, G = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ H^y &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},\ H^u = \begin{bmatrix} 0 & 0 & 0.56 & 0 \\ 0 & 0 & 0 & -1.2708 \end{bmatrix}, \end{aligned} \tag{5.46}$$

with the gain of static output-feedback controller being

$$\mathcal{K} = \begin{bmatrix} -0.56 & 0 \\ 0 & 1.270 \end{bmatrix}.$$

Model (5.46) has decoupled dynamics and resembles model (B.5), used to characterize the closed-loop dynamics of the omnidirectional surface vehicles. The value of T, i.e. the length of the offline collected trajectory, has been set equal to $100\,[s]$, the entire time window employed in the CG scheme $L = 15\,[s]$, with the length of the initial trajectory $L_i = 3\,[s]$ and the length of the predicted future trajectory $L_p = 12\,[s]$. Using the input/output partition and the measured trajectory shown in Figure 5.5, $w^d_{[1,T]}$ has been constructed and it has been verified that the resulting Hankel matrix $\mathcal{H}_L(w^d_{[1,T]})$ has full rank. More in detail, a reference r^d (Figure 5.5) has been given to the

system and both the input to the plant $u^d_{[0,T]}$ and the output $y^d_{[0,T]}$ have been measured. The initial part of the given trajectory is zero ensuring that the initial conditions of the computed response are also zero. Finally, the following constraints are imposed

$$\begin{cases} \|y_1(t)\| \leq 1.5, \ \|u_1(t)\| \leq 0.6, \\ \|u_2(t)\| \geq -2.2, \ \forall t \in \mathbb{Z}_+. \end{cases} \tag{5.47}$$

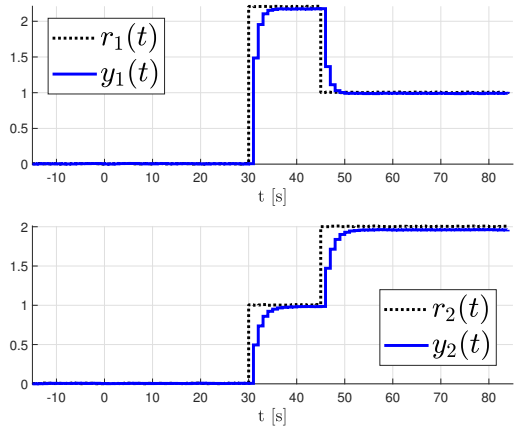
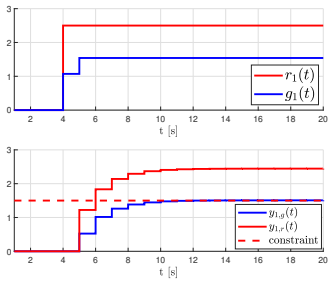


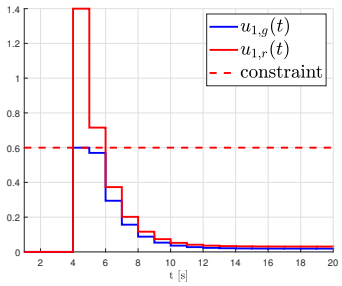
Fig. 5.5 Trajectory used to build Hankel matrix.

We simulate the response of the closed-loop system to a step change in $r_1(t)$ and $r_2(t)$. Figure 5.6(a) and Figure 5.6(c) show the computed admissible reference (blue line) computed by means of solving Problem (5.43). The computed admissible commands $g_1(t)$ and $g_2(t)$ (with a receding horizon approach) have been fed to the system and the trajectories of the input and the output have been measured (Figure 5.6). As can be seen, the response satisfies the constraint for all time, as desired. It is noticeable how, from Figure 5.6(b) and Figure 5.6(d), the constrained measured input (blue line) saturates if the computed admissible commands $g_1(t)$ and $g_2(t)$ is fed instead of the references $r_1(t)$ and $r_2(t)$. It is worth noting that if the reference $r_1(t)$ and $r_2(t)$ have been given directly, the system would have violated the constraints, as shown in Figure Figure 5.6(b) and Figure 5.6(d) (red line).

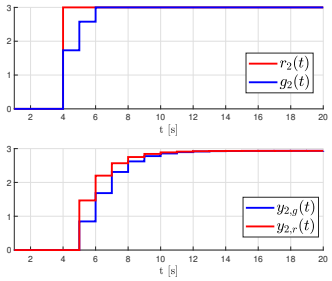
For the sake of completeness and to provide a preliminary comparison, the model-based CG solution, presented in Section 2.2.3, has been used to supervise (5.46) under the same constraints (5.47) and using an admissibility index equal to the size of the prediction window, i.e. $L_p = k_0 = 12$. As a comparative example, Table 5.1 shows the values of the admissible command g and the time required to solve the optimization problem using both the



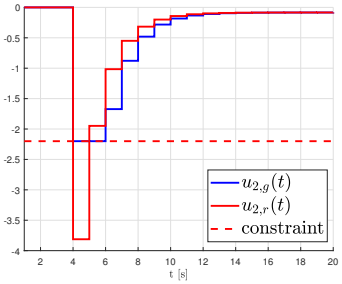
(a) At the top $r_1(t)$ and $g_1(t)$. At the bottom $y_{1,r}(t)$ and $y_{1,g}(t)$.



(b) $u_{1,r}(t)$ and $u_{1,g}(t)$.



(c) At the top $r_2(t)$ and $g_2(t)$. At the bottom $y_{2,r}(t)$ and $y_{2,g}(t)$.



(d) $u_{2,r}(t)$ and $u_{2,g}(t)$.

Fig. 5.6 Simulation results using the proposed DD-CG scheme.

model-based and the data-driven approaches at time instant $t = 4$, using in both cases the Gurobi [107] solver. Interestingly enough, the two approaches yield similar results and require comparable computation times.

Table 5.1 Details of model-based and data-driven computation of admissible command g_2 at $t = 4$.

	Value	Computational Time
Model-based	1.76	0.0010 $[s]$
Data-driven	1.73	0.0014 $[s]$

5.6 Concluding Remarks

This chapter addressed the challenge of supervising a linear time-invariant discrete-time system without relying on the system model. The challenges associated with time-consuming identification and validation processes, that became particularly evident during the study of marine surface vehicles, motivated this research. An alternative to model-based CG supervision schemes was specifically designed for linear systems, which are suitable models for these vehicles, as verified through simulations and real-world experiments. This results was achieved by leveraging concepts from behavioral system theory, subspace predictive control and unfalsified control literature. The proposed approach utilizes offline noise-free data from the plant and a representation of the controller. Future research endeavors will explore extending the results to decentralized and/or distributed settings, as well considering disturbances in both offline and measured data.

Chapter 6
Omnidirectional Surface Marine Vehicles

Motivated by the need to assess the implementability of the distributed command governor strategy on actual omnidirectional surface vehicles prototypes, a comprehensive implementation overview is presented, including comprehensive mechanical and electrical design, model identification and validation, control design and validation procedures. More in detail, under the stated assumptions, such as negligible currents and glassy water surface, the accuracy of the proposed linear time-invariant model of the vehicle, demonstrating satisfactory approximation, is fully described. Then, a controller is designed and the overall performance of controlled system is verified with in field experiments. The distributed supervision strategy is then implemented to assess the capability of the strategy to meet computational requirements of typical applications, and this is verified by means of enforcing local, obstacle avoidance and collision avoidance constraints. Although the initial tests are conducted with two vehicles due to resource constraints, the good scalability properties of the distributed CG-based strategy, demonstrated by simulation results, suggests that, given the ability of the strategy to meet real-time computational requirements, the approach can accommodate additional vehicles and constraints without significantly increasing computational complexity, especially in sparsely characterized networks.

6.1 Introduction

Marine operations traditionally rely on human intervention, a costly and disruptive method. A powerful alternative, due to its capability of operating autonomously for extended periods, and with various sensors for different missions, consists in the use of an ASV. However, the use of a single autonomous vehicle face limitations, such as lacking the flexibility and fault tolerance needed for complex tasks, particularly in applications requiring rapid exploration of large areas. In order to improve mission performance,

to enhance robustness and reliability against system failures, to reduce operational costs, and to optimize strategies for larger coverage of surveillance, communication, and measurement applications, current research goes beyond single ASV systems. Indeed, recent research has highlighted the growing interest in swarms of ASVs for rapid and robust exploration of large ocean workspaces, since their use offers advantages in terms of data measurement accuracy, precision, and consistency. However, conventional ASVs often face limitations due to their size and turning radius, particularly when operating in confined environments, while fully or over-actuated vehicles offer greater maneuverability and can overcome these limitations. Although in the literature examples exist of similar actuated surface vehicles [145, 22, 146], here the design has placed a special emphasis on the envisaged use of these vehicles in swarming applications, where it is necessary to minimize the risk of damage from collisions with other vehicles and obstacles. For instance, the thrusters are positioned underneath the center of the hull, which increases the resilience of the vehicle. Additionally, its modular design allows for the seamless integration of various sensors, making it adaptable to a wide range of missions.

As far as modeling is concerned, developing a unified numerical model for ASV control is challenging due to the requirement for specialized equipment and facilities, as well as the presence of inherent and external nonlinear influences [147]. While a more accurate and comprehensive model, reflecting the real-world physics, is desirable, practical considerations often necessitate simplifications to facilitate controller design. In fact, for this particular class of vehicles, a linear model of the dynamics of the vehicle can be obtained through several simplification steps, considering also that nonlinear dynamics are unnecessary for the planned operative regimes of the vehicle. Moreover, linear control design approaches can be directly applied in this context, offering a notable advantage by simplifying the control system while preserving robustness, stability guarantees, and ease of implementation. However, real-world validation is still needed.

To increase autonomy and safety, local constraints on states such as thrust, speed, position, obstacle and collision avoidance constraints are needed. Due to the generally expensive computational costs associated to optimization-based methods, especially when the previously stated constraints are enforced that could limit real-time implementation, in field tests involving marine surface vehicles are necessary to assess the practical implementability of the proposed distributed CG-based strategies.

To this end, this chapter addresses these challenges by presenting experimental validation on novel small, low-cost, and modular ASV prototypes suitable for swarming applications and precise maneuvering in confined spaces. In contrast with usual practice of deriving physical-based nonlinear vehicle models, a simplified linear time-invariant model, particularly beneficial for reducing the computational burden associated with developing predictive control strategies, is employed and validated through fully described experi-

ments, enabling the use of linear control techniques, further simplifying the control system. Following the validation of the closed-loop system and verification of offset-free behavior, real-world field tests are conducted to evaluate the practical implementation of the distributed CG-based strategy. These tests, that mimic real-world applications, involve various scenarios, including different types of constraints and environments, to demonstrate the ability of the strategy to meet real-time computational requirements.

This chapter is organized as follows. Section 6.2 details the design of the omnidirectional surface vehicle while Section 6.3 presents the basic GNC software architecture. Section 6.4 presents the experimental results, including model identification and validation, control system performance, and the effectiveness of the distributed CG-based supervision strategy in enforcing local, obstacle and collision avoidance constraints. Finally, Section 6.5 concludes the chapter.

6.2 Design

The ASV, referred to also as Chelon ASV, has to be designed to support a broad spectrum of applications that do not require human intervention, such as divers or operators. The structural design prioritizes maneuverability, modularity to facilitate component replacement in the event of failure, while ensuring safe operation. Energy use, cost, reliability, and ease of operation also guided the design. The key properties of the proposed ASV are reported in Table 6.1. The following subsections delve into the specific characteristics of its major components.

Table 6.1 Main specifications of the Chelon ASV.

Parameter	**Value**
Dimension	ϕ80 x h20 $[cm]$
Total weight in air	20.81$[Kg]$
Total buoyancy	49.30$[Kg]$
Thruster Spacing	51$[cm]$
Power Source	19.2$[V]$
Endurance	4-5 h

6.2.1 Mechanical Structure and Propulsion

Designed for maneuvering in confined environments, the Chelon ASV (Figure 6.1) is a holonomic surface vehicle capable of low-speed inspection and moni-

Fig. 6.1 Chelon Omnidirectional Surface Vehicle prototype.

toring. Given the requirements, because of its intrinsic shape stability and low draft and to ensure efficient navigation at low/medium velocities, the hull of the ASV is designed based on a cylindrical shape. The cylindrical hull design allows the ASV to turn almost at its center (i.e. the turning radius almost zero). Additionally, this design has advantage in providing smooth turning while maintaining the ASV position if force is applied by two of the four thrusters. To keep the cost modest and allow it to be launched and recovered from small boats, vehicle size was chosen to be just big enough to carry the constituent parts, namely the batteries and electronics to run the vehicle. A top and cross-sectional view of the ASV is drawn in Figure 6.2. The diameter of the hull is $80\,[cm]$ and height is $20\,[cm]$. The hull (Figure 6.4(a) and 6.4(b)), that accommodates the electronics and the battery pack, is made of resin coated high-density polystyrene, while the top (Figure 6.4(d)) and bottom (Figure 6.4(d)) plates are made of polyoxymethylene (POM-C). Opting for these materials not only ensures durability and cost-effectiveness but also streamlines the construction process itself. The hull alone can support up to $49,30\,[kg]$ load (Figure 6.5), intended for the battery pack, electronics and various sensors. The total weight of the vehicle, including the battery pack, is $20.81\,[kg]$, so that it can be easily carried by two people.

Propulsion is provided by four thrusters, each of which is located at one of the four corners and angled with respect to the forward direction. This layout ensures thrust and drag symmetry while allowing vectoring in the $X_b\,Y_b$ plane and rotation about the Z_b axis. This configuration is generally sufficient for most applications while preserving overall simplicity. The vehicle's hull is designed to enclose the thrusters (Figure 6.3), preventing them from being exposed and safeguarding them from potential collisions. More specifically, if an unexpected impact occurs, the upper structure will absorb the impact, shielding the thrusters from damage.

The vehicle is propelled by four outrunner brushless (BLDC) motors. One of the primary reasons for choosing brushless motors is their suitability for submersion. Compared to traditional brushed motors, brushless motors have

a stator that is easier to seal, making them more suitable for underwater applications. Specifically, we opted for outrunner thrusters each powered at $24\,[V]$ with peak current draw of $43\,[A]$, capable of delivering a thrust of $9.2\,[kgf]$. All four motors are controlled by a $51\,[Ah]$ BLHeli32 Electronic Speed Controller (ESC). This ESC offers high power in a compact size and features a 32-bit ARM Cortex M0 microcontroller capable of handling currents up to $51\,[A]$.

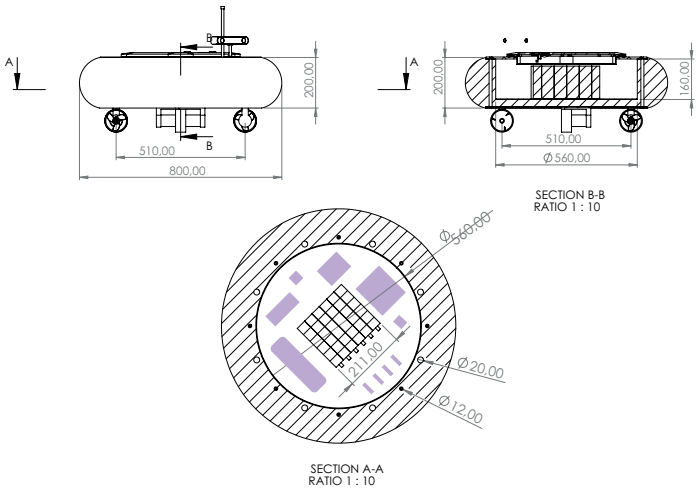
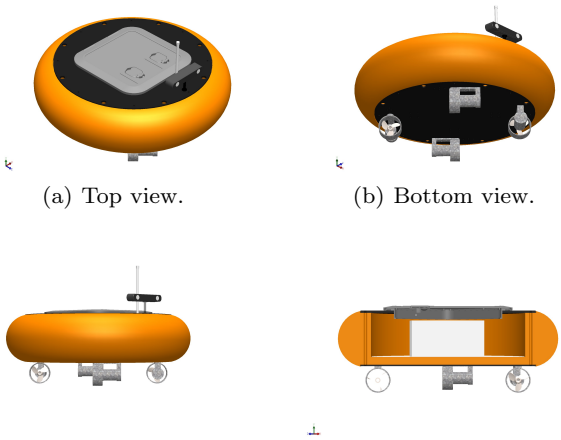


Fig. 6.2 Top and Cross-sectional views of the ASV hull showing placement of battery pack, electronics casing and motor frames attached to the underside.

6.2.2 Electronics

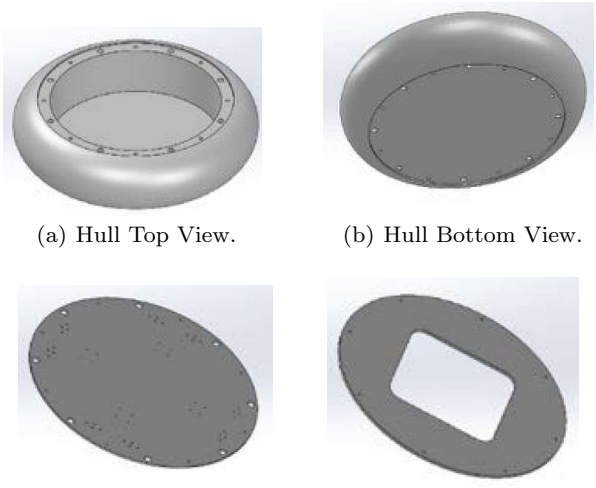
As previously mentioned, the ASV is not targeted towards a single, predefined application. Instead, the primary design objective is to achieve modularity. A modular design philosophy facilitates the seamless integration of diverse sensor suites, enabling the ASV to be adapted for a wide range of missions. The primary requirements for its electrical design are low power, small size, and versatility to allow integration of current and future sensors. The key components are shown in the system block diagram in Figure 6.6. At the core of the system is a single board computer running Linux and ROS (Robot Operating System). The single board computer sends/receives information to/from various components and it capable of communication with other devices (i.e. ASVs, laptops) via a Wireless Local Area Network (WLAN). Onboard systems manage all autonomous functions, while a laptop provides a communication interface for visualization and manual control. In manual mode, a



(a) Top view. (b) Bottom view.

(c) Lateral view. (d) Lateral section.

Fig. 6.3 3D renderings of the ASV.



(a) Hull Top View. (b) Hull Bottom View.

(c) Bottom Plate. (d) Top Plate.

Fig. 6.4 3D renderings of the structural components of the ASV.

joystick can transmit data directly to the onboard computer or indirectly via a laptop. Thrust commands, generated by the onboard computer, are sent through serial connection to a microcontroller, which produces PWM signals for the four ESCs. Powered by the battery, each ESC delivers a three-phase

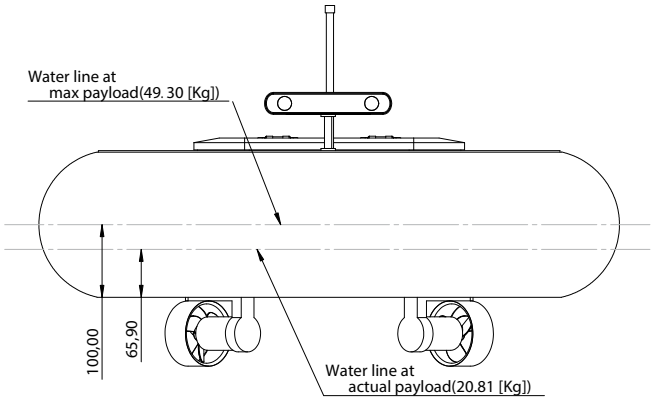


Fig. 6.5 Water line at both the current and maximum payload respectively.

output to drive the corresponding brushless motor. To enhance system resilience in the event of a single-board computer malfunction, redundancy is incorporated through the direct connection of a radio command module to the microcontroller. For a more thorough understanding, a detailed breakdown of the employed components is provided below.

Processor Board

The primary requirements for the ASV electrical design are low power, small size, and versatility to allow integration of current and future sensors. To this end, the USB technical standard has been chosen as the unique connector system to allow connection of peripherals. This choice has been made since the USB standard became commonplace on a wide range of devices and that additional USB hubs may be included, enhancing overall modularity. Due to its high-performance, low-power consumption and small size, the UDOO Bolt V3 has been chosen as the main computer board. An ATmega32U4 microchip is also integrated to a custom designed main circuit board incorporating a wide variety of analog and digital interfaces, power control and conditioning circuits, and interface connectors, allowing easy access to an Arduino Leonardo environment.

Microcontroller

The low-power requirement necessitated a microcontroller unit (MCU) capable of a submilliampere sleep mode and a large variety of power switches to handle potential various subsystems. A large number of serial communication channels as well as analog and digital channels were required for

embedded sensor integration. Due to its cost and size the STM32 Nucleo-64 development board with STM32F446RE MCU was chosen. In fact, this microcontroller utilizes a power conditioning circuitry on a small ($7 \times 8.25\,cm$) printed circuit board. The STM32F446xC/E devices are based on the high-performance Arm Cortex-M4 32-bit RISC core operating at a frequency of up to 180 MHz. The Cortex-M4 core features a floating point unit (FPU) single precision supporting all Arm single-precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) that enhances application security. This device incorporates high-speed embedded memories (Flash memory up to 512 Kbytes, up to 128 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, two AHB buses and a 32-bit multi-AHB bus matrix. It also offers three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers.

Power Budget

As outlined above, the usefulness of the ASV relies heavily on power conservation. To ensure extended operational capability, the vehicle's power system employs a high-capacity Lithium battery pack paired with a high-current balancing board. *LiFePO4* cells were selected for their ability to deliver high discharge currents required for motor operation. These cells exhibit a low self-discharge rate and lack of memory effects. Furthermore, *LiFePO4* technology is inherently safe, eliminating the risk of spontaneous combustion, fire, or explosion, especially in the event of water contact. A significant portion of the vehicle's weight comes from the battery pack, weighing in at $10.9\,[kg]$ (including the Battery Management System). For optimal weight distribution and vehicle stability, the battery pack is positioned centrally as shown in Figure 6.2. The installed battery pack has six $3.2\,[V]$ cells connected in series, resulting in a total voltage of $19.2\,[V]$ and a capacity of $60\,[Ah]$, capable of delivering $1,152\,[Wh]$. To safeguard cell integrity and prolong their lifespan, a $150\,[A]$ Battery Management System (BMS) is integrated into the system. The BMS provides also the State of Charge (SOC) of the battery pack. Energy use is highly dependent upon mission goals and environmental conditions, that predominately determine the energy budget.

Attitude Sensing Package

Due to its small size ($24 \times 22 \times 3\,mm$) and low power consumption ($45\,[mA]$ @ $3.3\,[V]$), the high-performance Inertial Measurement Unit (IMU) and Attitude Heading Reference System (AHRS) Vectornav VN-100, incorporat-

ing Solid-State MEMS inertial sensors, has been chosen to measure vehicle pitch, roll and magnetic heading. The VN-100 is rated to operate at temperatures ranging from −40°C to 85°C. Combining 3-axis accelerometers, gyros, and magnetometers, a barometric pressure sensor and a 32-bit processor, the VN-100 provides high-rate, calibrated IMU data and a real-time 3D attitude solution. Its VectorNav Processing Engine for disturbance rejection, adaptive filtering, dynamic filter tuning and Hard/Soft Iron Compensation has been used.

Navigation Receiver

In order to be able to cover all possible applications, it is important to have accurate GPS data. Due to its high-precision, the Ublox C099-F9P application board has been chosen. The ZED-F9P module provides multi-band GNSS positioning and comes with built-in RTK technology providing centimeter level accuracy. Augmentation services may be formatted as SPARTN 2.0 or RTCM 3.3, and can be provided either by a local device, via Satellite or via IP (Internet). The C099-F9P application board integrates the ZED-F9P module and includes an ODIN-W2 short range module for connectivity options.

Data Telemetry

The ASV uses the Mikrotik Groove 52HPn weatherproof wireless RouterBOARD for bidirectional data communication. It is equipped with an AR9342 600Mhz CPU and Ethernet One 10/100 Mbit/s Fast Ethernet port with Auto-MDI/X, L2MTU up to 2030 and Wireless 5 or 2GHz (software selectable) radio. In this way, with standard WiFi connectivity, the ASV can be conveniently accessed and operated by a laptop computer or smartphone.

Vision module

To achieve obstacle detection, the IP66-rated ZED 2i dual lenses camera, capable of providing depth maps, has been mounted on top of the ASV. More specifically, depth maps captured by the ZED 2i store a distance value (Z) for each pixel (X, Y) in the image. Furthermore, for robustness and redundancy, the ZED 2i stereo camera is equipped with motion sensors (Gyroscope, Accelerometer, Magnetometer) and environmental sensors (Barometer, Temperature) that can also be used.

Scientific Sensors

The basic scientific instrumentation consists of a multiparameter sonde, measuring conductivity, temperature, pH, dissolved oxigen and electrical conductivity. Another micro controller (or the same one used for the actuation of the thrusters) is used to interface the multiparameter sonde but any kind of sensor can be used as long as it is provided with an USB interface.

A summary related to the dimensions, weight and estimated power consumption of the main components of the surface vehicle is reported in Table 6.2.

Table 6.2 Components with their weight and power consumption.

Component	Description	Dimension	Weight (kg)	Estimated peak Power Consumption
Main Board	UDOO Bolt V3	120×120 mm	0.2505 kg	31.04 W (1.634 @ 19V)
Microcontroller	NUCLEO F446RE	70×82.5 mm	0.077 kg	1.5 W (300 mA @ 5 V)
IMU	Vectornav VN-100	$24 \times 22 \times 3$ mm	0.0035 kg	0.1485 W (45 mA @ 3.3 V)
Localization Module	Ublox C099-F9P	110×55 mm	0.0165 kg	5 W (500 mA @ 5 V)
Communication Module	Mikrotik Groove 52HPn	$177 \times 44 \times 44$ mm	0.193 kg	4.56W (0,19A @ 24V)
Obstacle Detection Module	ZED 2i	$175.25 \times 30.25 \times 43.1$ mm	0.166 kg	1.9 W (380 mA @ 5 V)

Remark 6.1 While the Ublox C099-F9P navigation receiver typically relies on IP corrections to achieve an accuracy of approximately $0.1\,[m]$, higher precision is required for operations in small, confined spaces. A GPS base can be used to compute and send corrections, and accuracy can be improved to $0.015\,[m]$. However, the native Mbed OS 3 firmware of the ODIN-W2 module limits corrections to a single device. To enable corrections for multiple devices, the u-connectXpress firmware variant must be used.

Remark 6.2 The presence of large dead-zones in the response of the thrusters, i.e. areas in which the command does not induce any rotation of the propeller, can lead to undesirable oscillatory behavior, particularly in confined spaces. This occurs due to the inability of the thrusters to generate sufficient force for small control inputs, causing the vehicle to drift away from the desired position. When the error becomes sufficiently large, the control signal exit the dead-zone and thrusters become active, but as the vehicle approaches the reference point, the control signal may enter again dead-zone threshold, leading to further deviations. To mitigate this issue, BLHeli32_S firmware on the ESCs was used, and efforts were made to reduce the dead-zone during the characterization of the thrusters (Figure 6.7).

6.3 Software Architecture

The standard modular Guidance, Navigation, and Control (GNC) architecture depicted in Figure 6.8 has been initially adopted and implemented on

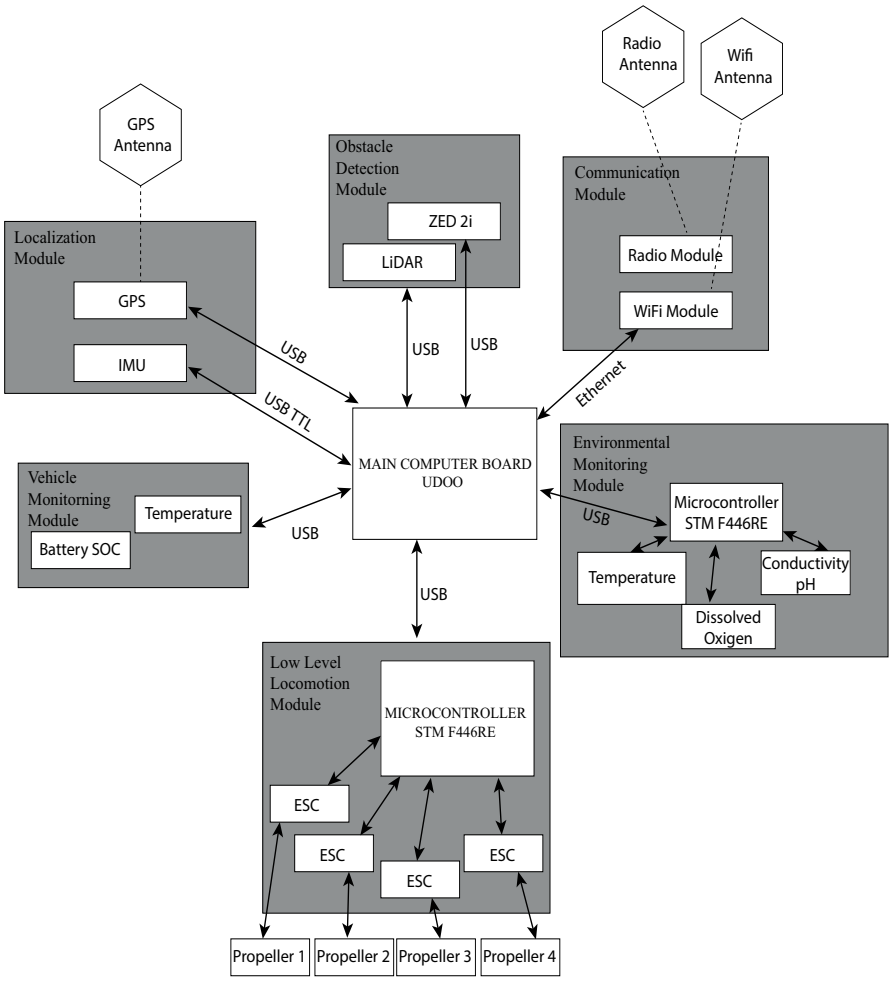


Fig. 6.6 System architecture overview, where the different boards and sensors are shown. The same micro controller can be used for both the Low Level Locomotion Module and the Environmental Monitoring Module.

the Robot Operating System (ROS). More in detail, the UDOO Bolt V3 main board operates on Ubuntu 20.04 and ROS Noetic with Python 3.8. The main components of this basic GNC architecture, which is then expanded to incorporate the supervision strategy, are

- **Navigation Module**: The *Navigation* module contains all *ROS* nodes responsible for reading data from sensors. In particular, a GPS sensor and an IMU sensor are used to compute the state of the vehicle (linear and angular positions and velocities), whilst the distance from objects can be measured using the stereo camera sensor. Specifically, a *Vision Module*

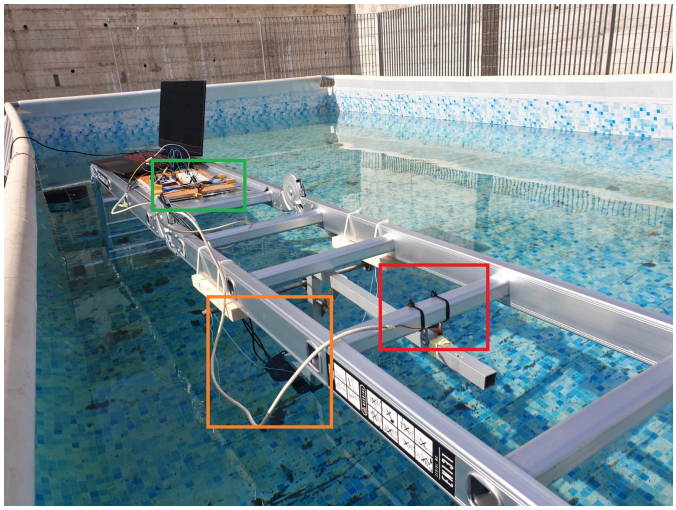


Fig. 6.7 Thruster characterization setup with the ESC (green box), the load cell (red box) and the propeller (orange box).

node measures and stores the distance from each point of the obstacle and computes the minimum distance between these;

- **Guidance Module**: The main goal of the *Guidance* module is to compute a reference according to a specific policy. The estimated state, provided by the navigation system, is used to provide the desired reference vector at each point in time. More in detail, this module has a *Global Planner* node, which is responsible for the high-level reference generation and chooses a reference according to the task assigned to the vehicle. A *Local Planner*, based on information received from the *Vision Module*, changes such reference to avoid collisions;
- **Control Module**: The *Control* receives references from the *Guidance* module. In this module there are two nodes: *i*) a high-level control module that implement the primal control law; *ii*) a low-level control module, useful to manage the thrusters, that takes into account the disposition of the thrusters presented in Section B.1.1.

Remark 6.3 As presented in Section 6.2.1, BLDC propellers have been used. However, since they need to be sealed and pressure tolerant while maintaining efficiency and affordability, these propellers are typically available without sensors, which poses challenges for control, a crucial factor for efficient maneuvering. To overcome this issue, a sensorless Field-Oriented Control (FOC) algorithm has been integrated to control these three-phase marine thrusters.

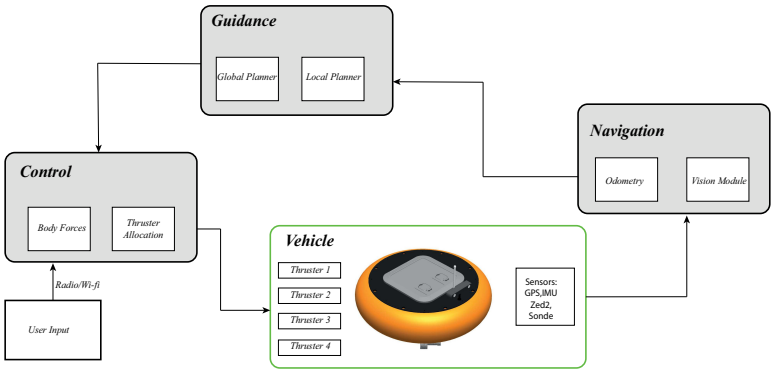


Fig. 6.8 Basic GNC system architecture.

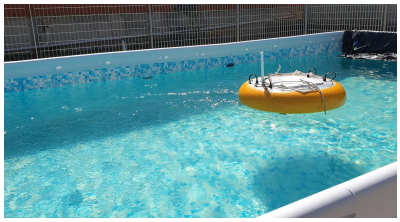
6.4 Experiments and Deployments

This section presents real-world experiments conducted on the proposed ASV to validate the dynamic model, control system, and distributed CG-based supervision strategy.

Given the fundamental assumption that a LTI model can accurately capture the dynamics of vehicle, the initial focus is on validating such model. More in detail, upon the assumptions introduced in Section B.2.1, model (B.5), which is excellent for applications where the goal of the vessel is to track and maintain a desired position and orientation, is here considered. Model parameters estimation is followed by model validation. Then, after controller design, closed-loop tests are conducted to ensure offset-free behavior, a prerequisite for effective CG-based supervision. Finally, the focus shifts to assessing the performance of the distributed CG-based supervision strategy. This strategy is designed to ensure that the vehicle adheres to specific constraints while operating in real-world scenarios.

Three distinct environments are employed: *i*) a $7 \times 4\,[m]$ pool and a testing facility (Figure 6.9(a) and Figure 6.9(d)), *ii*) a lake (Figure 6.9(c)), and *iii*) a pier (Figure 6.9(b)). Table 6.3 summarizes the experiments carried out. The longitudinal and latitudinal measurements provided by the GPS device are transformed with respect to a specific point, that serves as the local origin of the inertial frame. The *E*ast-*N*orth-*U*p (ENU) coordinate system is used to represent the position. More in detail, vehicles gather RTK-GPS data with a $20\,[Hz]$ frequency with an accuracy of $0.014\,[m]$, while orientation (yaw) data is measured at $100\,[Hz]$ with an accuracy of 2.0°.

Prior to experimental validation in a real environment, the same identification, control and validation phases have been carried out in the ROS/Gazebo simulation environment. More in detail, all Gazebo plugins used to simulate sensors and actuators have been configured using real parameters taken from the commercial data-sheets of the components introduced above.



(a) Pool testing environment.

(b) Pier.

(c) Lake.

(d) Testing Facility.

Fig. 6.9 Testing environments.

Table 6.3 In field tests description.

Test	Pool	Testing Facility	Lake	Pier
Open Loop	✓	✓		
Controlled System	✓			✓
Local Constraints			✓(speed)	✓(speed, thrust)
Obstacle Avoidance	✓			✓
Collision Avoidance	✓			

6.4.1 Estimating the Vehicle Model Parameters

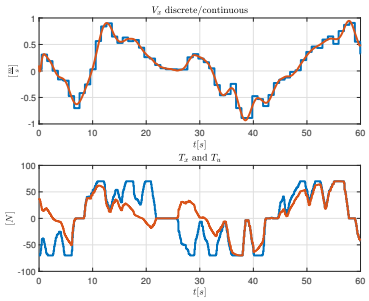
An USV model is typically identified offline using data collected from onboard sensors. Common identification methods include frequency domain techniques [148] and time domain techniques, such as least squares regression [149, 150], onboard sensor-based identification [151], continuous time models [152], hybrid-extended Kalman filtering [153], and artificial neural networks [154]. In this specific case, to identify the model parameters, a time-domain least squares regression approach is employed. This section presents the input datasets, the corresponding system responses in terms of velocity, and the estimated numerical values of the actual vehicle. Initial testing was conducted in the testing facility (Figure 6.9(d)) due to its conditions respecting the as-

sumptions, such as absence of current and glassy water surface. Furthermore, due its large size, this controlled environment ensured safety, especially considering the absence of a control system. The methodology which is used for identification is based on open loop step responses of the system. This method is appropriate for laboratory conditions where external disturbances are negligible. Furthermore, this approach can be used to avoid the risk of control parameters having influence on model coefficients. Four different datasets have been used. Dataset S_1 has been used to collect $V_x = \dot{p}^x(t)$, dataset S_2 for $V_y = \dot{p}^y(t)$, dataset S_3 for $\dot{\theta}$ and dataset S_4 has been used for validation purposes.

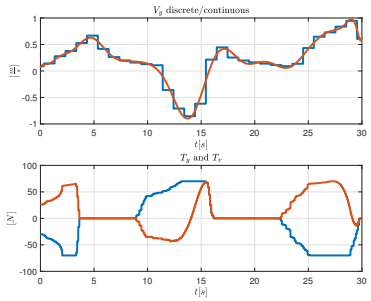
As discussed in Section B.1.1, to convert the thrust values expressed in the body-fixed frame T_{uvr} into thrusts commands for the propellers, the pseudo-inverse of the allocation matrix has to be computed considering the construction values

$$\Sigma^{+} = \begin{bmatrix} 0.35355339 & -0.35355339 & -0.69444444 \\ 0.35355339 & 0.35355339 & 0.69444444 \\ -0.35355339 & 0.35355339 & -0.69444444 \\ -0.35355339 & -0.35355339 & 0.69444444 \end{bmatrix}. \tag{6.1}$$

Figure 6.10(a) and Figure 6.10(b) show respectively velocities (measured



(a) Velocities using dataset S_1.



(b) Velocities using dataset S_2.

Fig. 6.10 Measured velocities relative to the x (left) y (right) axis located on top. Thrusts generated in the body-fixed reference frame T_u (left) T_v (right) in blue and the inertial reference frame T_x (left) T_y (right) in brown on bottom.

values in blue, their interpolation in brown) using dataset S_1 and S_2, while Figure 6.12 depicts the measurements of the orientation (Figure 6.12(a)) and angular velocity (Figure 6.12(b)) of the vehicle. The parameters of model (B.5) have been off-line estimated in Matlab by means of an identification procedure exploiting the Least Mean Squares (LMS) algorithm [98, 99]. More in detail, the estimated numerical parameters $-\beta/m$, $-1/m$, $-\beta_t/J$ and $1/J$ of (B.5) are

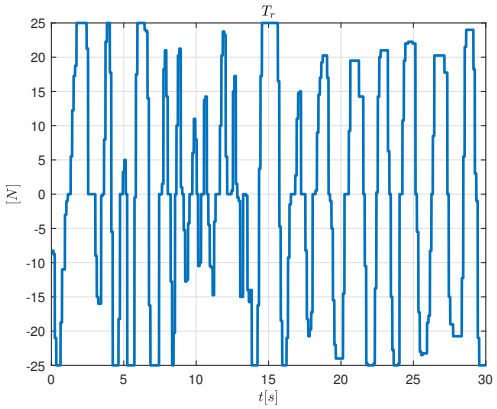
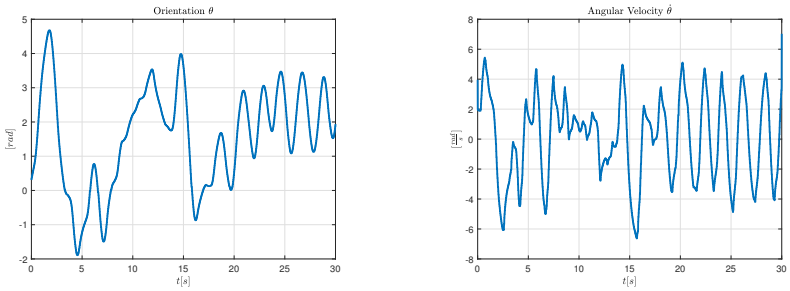


Fig. 6.11 Plot showing a 30 $[s]$ sample of the generated signal T_r.



(a) Measured vehicle orientation with dataset *S3*.

(b) Measured vehicle angular velocity with dataset *S3*.

Fig. 6.12 Measure orientation and angular velocity with dataset *S3*.

$$a = [-0.2032,\ -0.0044,\ -58.0888,\ 43.9945]. \tag{6.2}$$

To validate the estimated model, another validation data set has been used. All results presented in this section are based on a k_0 step ahead velocity prediction, i.e. velocity predictions at time $\bar{t}$ are generated by taking the velocity measurement at time $\bar{t} - k_0 T_c$ and recursively evaluating the system model (B.6), using estimated parameters (6.2), for $k_0 T_c$ subsequent time steps. In other words, the state of the estimated model is updated with the state of the vehicle every $T_c k_0$, with T_c equal to 0.01 $[s]$ seconds and k_0 equal to 50. This horizon length was selected as it was deemed to be appropriate for model-based control. Figure 6.13 shows the real trajectory (brown) and the simulated trajectory (blue) of the vehicle with the previously estimated parameters. Given the distinct characteristics of freshwater and saltwater environments, or potential variations between prototypes, re-identifying and re-validating the model may be necessary, further emphasizing the need for a

data-driven approach. Additionally, even within the same complex environment, such as a pier, slight variations in sea conditions can lead to parameter changes, and frequent model updates may be needed.

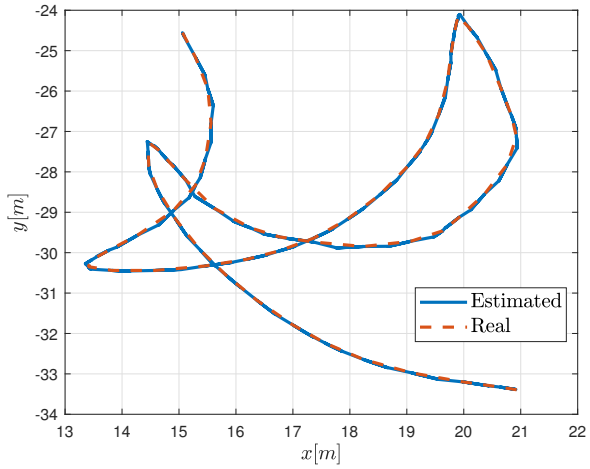


Fig. 6.13 Validation phase using dataset S_4: The actual trajectory is shown in brown, while the simulated trajectory using the estimated point mass model is shown in blue.

6.4.2 Controlled System

Starting from system (B.6), an augmented model with the tracking error has been built

$$\begin{cases} \xi(t+1) = A^e \xi(t) + B^e u^p(t), \\ \quad y(t) = C_y^e \xi(t) + D^e u^p(t), \end{cases} \tag{6.3}$$

where

$$A^e = \begin{bmatrix} A & \mathbf{0}_{6\times 3} \\ -C_y & \mathbf{0}_{3\times 3} \end{bmatrix}, B^e = \begin{bmatrix} B \\ \mathbf{0}_{3\times 3} \end{bmatrix},$$
$$C_y^e = \begin{bmatrix} C_y & \mathbf{0}_{3\times 3} \end{bmatrix}, D^e = \mathbf{0}_{3\times 3},$$

and $\xi(t) = [\delta x(t), e(t)]^T$, where $\delta x(t) = x(t) - x(t-1)$ and $e(t)$ is the tracking error for a given reference $r(t) = [r^x(t), r^y(t), r^\theta(t)]^T$. The control law is expressed as

$$u^p(t) = -F_b \xi(t) + F_f \left[\left(\sum_{\tau=1}^{t} T_s^c e(\tau) \right) + e(0) \right], \tag{6.4}$$

where $F_b \in \mathbb{R}^{3\times6}$ is the feedback matrix, $F_f \in \mathbb{R}^{3\times3}$ is the feedforward matrix and the last terms are the Euler forward approximation of the error $e(t) = r(t) - p(t)$ accumulation process [155] with sampling time T_s^c.

The computed feedback F_b and feedforward F_f gains (6.4), considering the above estimated parameters (6.2), are

```
% Feedback_Gain
Fb =  0.3*[90.96244, 0, 0, 120.79747, 0, 0;
0, 90.922055, 0, 0, 120.78731, 0;
0, 0, 4.2187366, 0, 0, 5.4310626];
% Feedforward_Gain
Ff = 0.3*[11.9796, 0.0, 0.0;
0.0, 11.9722,0.0;
0.0, 0.0, 0.5555];
```

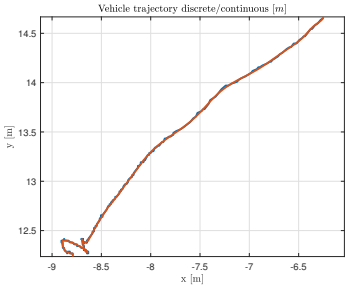
Considering the initial objectives of validating the accuracy of the simplified dynamic model and verifying the overall performance of the controlled system, field experiments are conducted.

Similarly to the approach in Section 6.4.1, the control design and testing phases were initially carried out in a Gazebo simulation environment.

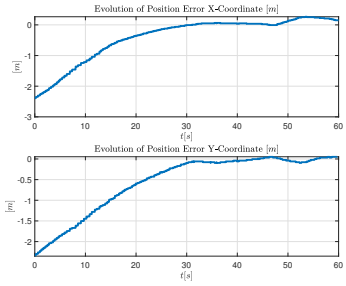
To meet the above stated objectives, initial testing is conducted in a confined environment, demonstrating the suitability for applications in restricted spaces or bathymetric surveys of the designed prototype. Subsequently, a patrolling experiment is conducted in a pier environment to validate the suitability of the system also in typical applications.

Pool

For the initial experiment, carried out in the pool (Figure 6.9(a)) and aimed at validating the controlled system, with an initial pose of $p(0) = [-6.4\,m, 14.6\,m, \pi\,rad]^T$, the vehicle was instructed to reach a reference point located at $r = [-8.65\,m, 12.3\,m, 2.9\,rad]^T$. While other experiments similar to the one presented have been conducted, only the results of this representative trial are shown for brevity and due to the consistent behavior observed across all trials. Figure 6.14(a) shows the trajectory of the vehicle, while Figure 6.14(b) shows the measured error between the reference provided to the vehicle and its position in time along the two coordinates x and y, demonstrating successful reference tracking and control scheme validation, as the error converges to zero. For a comprehensive evaluation, Figure 6.16 shows the evolution of measured orientation, while Figure 6.15(a) and Figure 6.15(b) show the evolution of the measured velocities along the x-axis and y-axis and the computed thrusts T_x and T_y. In a separate experiment, the vehicle was instructed to reach three consecutively provided reference points in the pool environment. A video recording can be found at https://youtu.be/XokVmIBx5Nc.

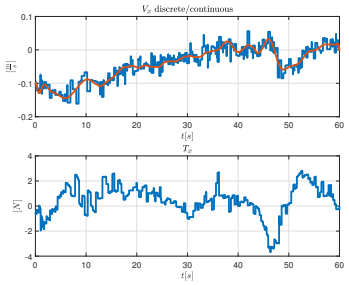


(a) Vehicle trajectory (measured trajectory in blue, interpolated trajectory in brown).

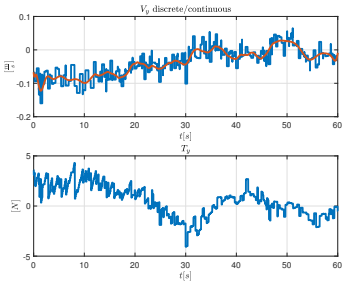


(b) Error measured between the reference provided to the vehicle and its position in time along the two coordinates x and y.

Fig. 6.14 Controlled vehicle validation: Plot showing the trajectory of the vehicle and the tracking error in the pool.



(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.



(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

Fig. 6.15 Controlled vehicle validation: plot showing the measured velocities and computed thrusts of the vehicle in the pool.

Pier Patrolling

This basic patrolling experiment involved implementing a simple guidance node that generated reference points for the vehicle to perform a brief inspection around the entrance of a pier. Figure 6.17 illustrates the experimental setup, including the coordinates of the point that serves as the origin of the inertial frame and the reference points generated by the guidance node.

For simplicity of analysis, two scenarios are shown. An excerpt from the trajectory data, speed and thrust, and error when the vehicle is reaching reference $r_4 = [-56.52\,m, -110.57\,m, 2.81\,rad]^T$ after it has reached the reference $r_3 = [-56.46\,m, -98.9\,m, 2.81\,rad]^T$, is considered. As shown in

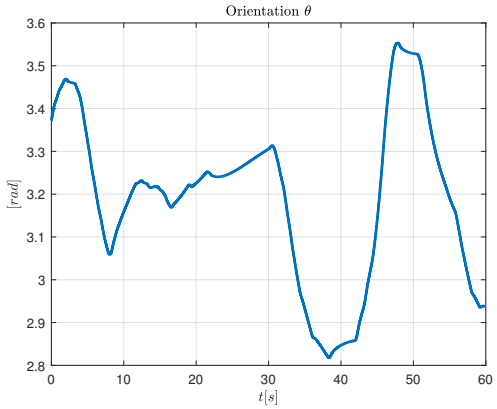


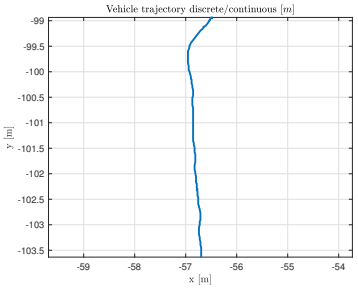
Fig. 6.16 Controlled vehicle validation: evolution of the measured orientation in the pool.

Figure 6.19(a), a negative V_x speed is observed between $t = 64\,[s]$ and $t = 70\,[s]$ despite a positive T_x thrust command. This behavior indicates that the control system is compensating for a current, that is pushing the vehicle in the negative x direction. In this case, with minimal disturbance, the controlled vehicle performs well under the assumptions made, demonstrating successful reference tracking (Figure 6.18(a)) and position error going to zero (Figure 6.18(b)). In another case, recorded portions of the trajectory and the measured error, when the vehicle is reaching a reference $r_7 = [-48.65\,m, -94.5\,m, 3.14\,rad]^T$ after it has reached the reference $r_6 = [-46.46\,m, -107.9\,m, 3.14\,rad]^T$, are shown in Figure 6.21(a) and Figure 6.21(b). In this case, the performance of the controlled vehicle deteriorates significantly when a nearby ship and wave motion introduce external disturbances. This highlights the need for a more robust control strategy in case the assumption of calm water is not made, which can be readily implemented due to the modular design of the GNC architecture.

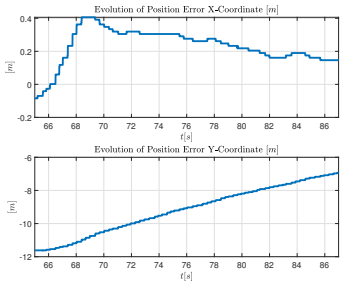
6.4.3 Distributed Supervision

Following the successful validation of the closed-loop system through experiments, the supervision strategy is here tested. The basic ROS-based GNC architecture presented in Section 6.4 was extended to incorporate the GNC-C Architecture described in Section 6.3 and all subsequent tests are conducted using this distributed supervision scheme. To evaluate the capability of the supervision strategy to enforce local constraints on states and control inputs, initial tests are performed on a single vehicle, allowing for focused analysis.

Fig. 6.17 Pier Patrolling experimental setup. The origin of the inertial frame is located at 39°05'56.3" *N*, 16°09'22.5" *E*.



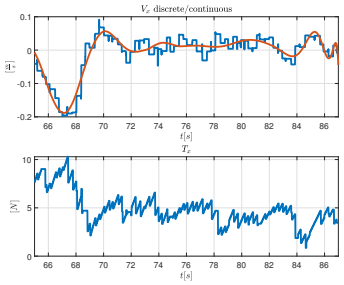
(a) Vehicle trajectory while tracking reference r_4 with glassy water surface.



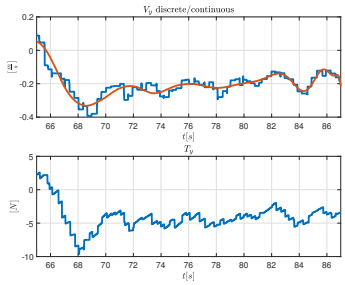
(b) Error measured between the reference provided to the vehicle and its position in time along the two coordinates x and y when reaching reference r_4.

Fig. 6.18 Controlled vehicle validation: plot showing the trajectory of the vehicle and the tracking error in the pier.

Subsequent tests assessed the performance in more complex scenarios, including obstacle avoidance and collision avoidance. It is important to note that the time axis in the following plots represents the duration of the observation, not the actual time of the mission.



(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.



(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

Fig. 6.19 Controlled vehicle validation: plot showing the measured velocities and computed thrusts of the vehicle in the pier.

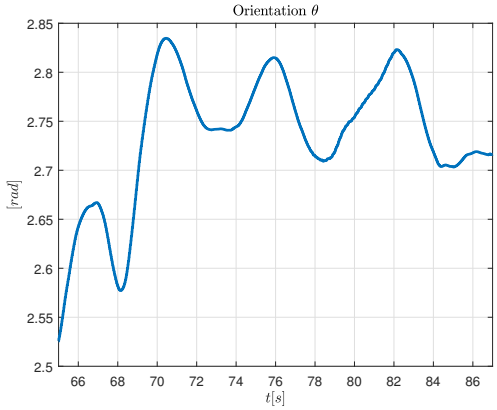


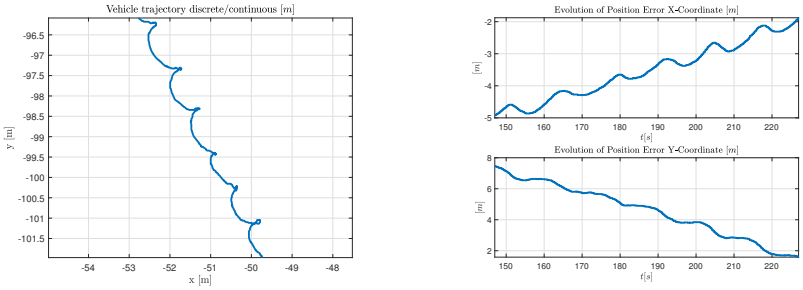
Fig. 6.20 Controlled vehicle validation: evolution of the measured orientation in the pier.

6.4.3.1 Local Constraints

This subsection presents the validation of the supervision scheme for a single vehicle. Two scenarios are considered: *i*) local speed constraints in a lake visual census application and *ii*) speed and thrust constraints in a pier environment.

Speed Constraints in Lake

This section presents experiments conducted in a lake environment to assess the ability of the supervision scheme to enforce local constraints on an omni-



(a) Vehicle trajectory while tracking reference r_7 with rippling water surface.

(b) Error measured between the reference provided to the vehicle and its position in time along the two coordinates x and y when reaching reference r_7.

Fig. 6.21 Controlled vehicle validation: plot showing the vehicle trajectory and the tracking error in the pier with rippling water surface.

directional marine surface vehicle. A visual census experiment is conducted in an environmentally protected lake, where the electric nature of the vehicle allows for non-invasive data collection. A video camera was easily attached to the vehicle (Figure 6.23(a)) to capture underwater images (Figure 6.23). The experimental setup is shown in Figure 6.22.

To ensure the quality of the visual census, it is necessary to limit the speed of the vehicle. The supervision scheme plays a crucial role in enforcing these speed constraints. More in detail, the following speed constraints

$$\begin{cases} \|\dot{p}^x(t)\| \leq 0.25\,[\frac{m}{s}], \\ \|\dot{p}^y(t)\| \leq 0.25\,[\frac{m}{s}],\ \forall t \in \mathbb{Z}_+, \end{cases} \tag{6.5}$$

have been imposed on the system. For simplicity, the analysis focuses on an excerpt of the experiment, as the a consistent behavior was observed throughout its entire duration. Figure 6.24 shows the trajectory of the vehicle during the visual census. To capture the desired underwater images, the vehicle is instructed to follow a predefined zig-zag pattern. This pattern was achieved by providing constant r^x and varying r^y reference values. As shown in Figure 6.25, the supervision strategy modifies the given references to fulfill the imposed speed constraints. Consequently, the vehicle never violates the imposed constraints, as shown in Figure 6.26. For sake of completeness, Figure 6.25(b) depicts the tracking error between the position of the vehicle and the given references. The red line indicates the time instant when step changes of the reference are made.

Fig. 6.22 Lake Visual Census experimental setup. The origin of the inertial frame is located at 38°93'44.96" *N*, 16°20'80.81" *E*.

(a) Waterproof camera mounted on the bottom of the vehicle.

(b) Visual census snapshot.

Fig. 6.23 Waterproof camera and a snapshot of its recording during the visual census.

Speed Constraints in Pier

Another experiment, similar to the previous one, is conducted in a pier environment. While visual census was not performed in this case, the experiment aims at demonstrating the effectiveness of the patrolling when implementing the supervision strategy. The origin of the inertial frame is located approximately as shown in Figure 6.17. The vehicle is subject to the same speed constraints as before (Equation (6.5)). Figure 6.27 illustrates the trajectory

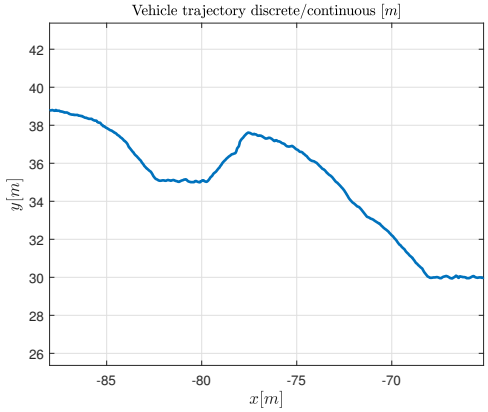
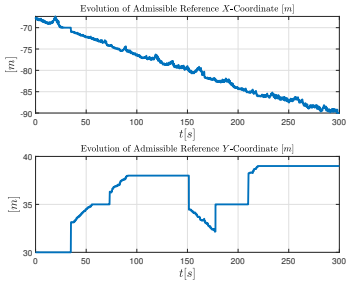
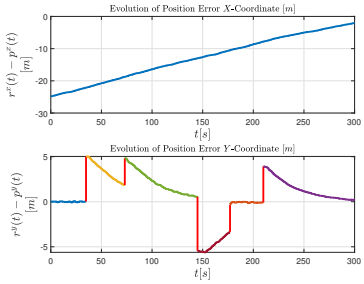


Fig. 6.24 Vehicle trajectory while tracking references during the visual census experiment.



(a) Evolution of the admissible reference.



(b) Error measured between the reference provided to the vehicle and its position in time along the two coordinates x and y.

Fig. 6.25 Local speed constraints validation: Plot showing the evolution of the admissible reference and the tracking error in the lake.

of the vehicle during supervised patrolling. As the vehicle moves towards the reference point $[-56.80\,m, -110.80\,m, 2.81 rad]$, the employment of supervision strategy, as shown in Figure 6.29, leads to successful satisfaction of the constraints. Figure 6.28(b) depicts the tracking error between the position of the vehicle and the given reference. These results further validate the effectiveness of the supervision strategy in ensuring efficient vehicle operation.

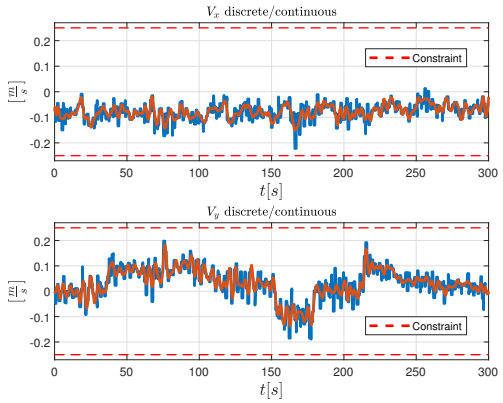


Fig. 6.26 Local speed constraints validation: Plot showing the measured velocities of the vehicle in the lake.

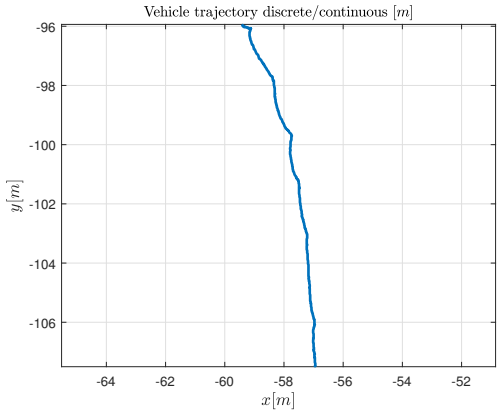
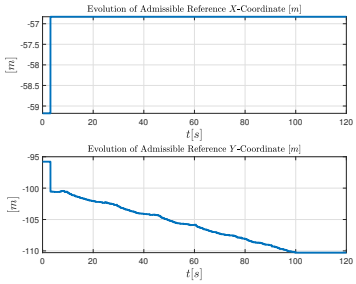


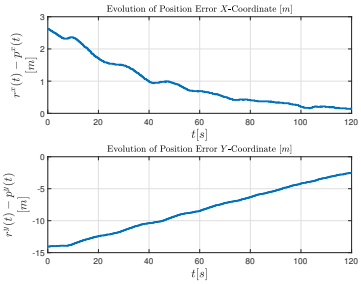
Fig. 6.27 Excerpt of the trajectory of the vehicle while tracking its assigned reference.

Thrust Constraints in Pier

While constraints on plant states, such as speed limits, are important, it would be beneficial to impose constraints on controller states, such as limiting thrust generation to reduce energy consumption and noise. Ignoring actuator constraints can also lead to degraded performance, increased wear and tear, and potential instability. While imposing speed constraints is necessary, it may not be sufficient to ensure efficient operation in all scenarios. As demonstrated by the lake and pier experiments, identical speed constraints can lead to different thrust commands, highlighting the need for additional actuator constraints. The thrust commands have been constrained $\|T_l(t)\| \leq 10\,[N]$,

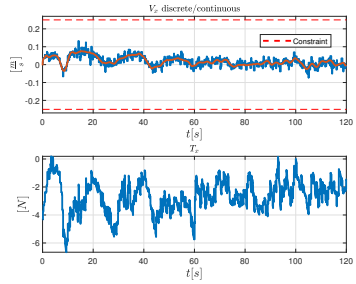


(a) Evolution of the admissible reference.

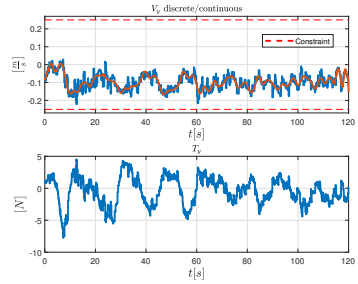


(b) Error measured between the reference provided to the vehicle and its position in time along the two coordinates x and y.

Fig. 6.28 Local speed constraints validation: Plot showing the evolution of the admissible reference and the tracking error in the pier.



(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.



(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

Fig. 6.29 Local speed constraints validation: Plot showing the vehicle measured velocities and computed thrusts in the pier.

with $l = x, y$ and for all $t \in \mathbb{Z}_+$. To induce significant control inputs, the reference of the vehicle is abruptly changed. This scenario requires the vehicle to not only move towards the new reference point but also decelerate from its current motion, resulting in higher thrust requirements. More in detail, Figure 6.30 shows the trajectory of the vehicle as it moves towards the initial reference point $[-56.80\,m, -110.80\,m, 2.81\,rad]$. Before reaching this point, a new reference is assigned at $[-56.80\,m, -95.80\,m, 2.81\,rad]$, effectively reversing the direction in which the vehicle is moving. Throughout this maneuver, the supervised vehicle adheres to the imposed constraints, as shown in Figure 6.31. The measured distance between the position of the vehicle and its assigned reference is shown in Figure 6.32(b). This is an interesting case, be-

cause, unintentionally, although the vehicle is supervised, the reference given is already admissible (Figure 6.32(a)).

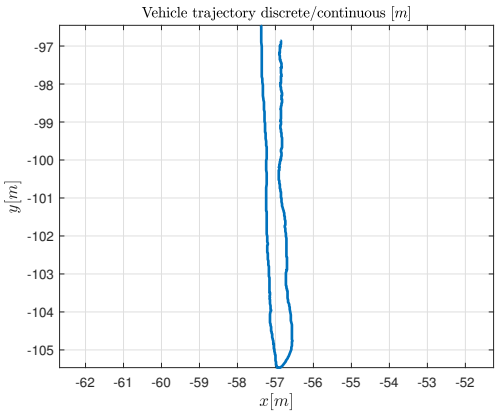
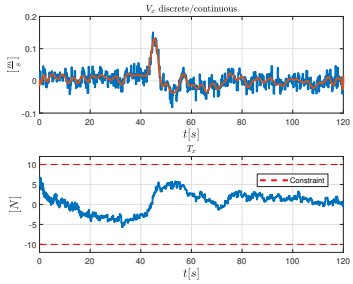
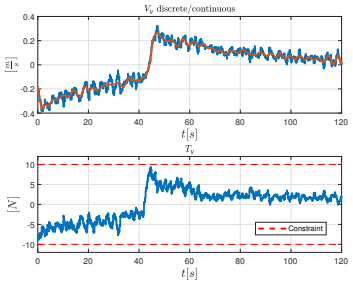


Fig. 6.30 Vehicle trajectory while tracking references.



(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.

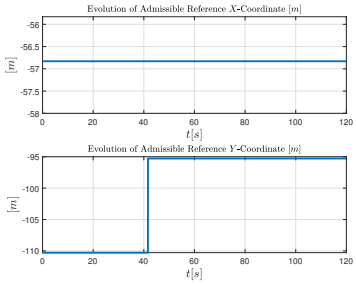


(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

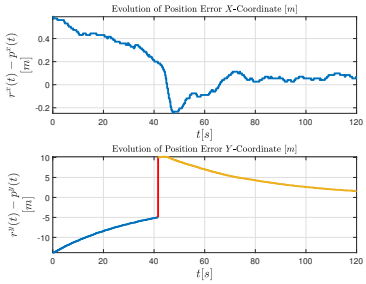
Fig. 6.31 Local thrust constraints validation: Plot showing the vehicle measured velocities and computed thrusts in the pier.

6.4.3.2 Obstacle Avoidance

The proposed vehicles are designed to operate in challenging environments, such as confined harbors and areas with abandoned structures. To validate the ability of the supervision scheme to handle obstacle avoidance, experiments



(a) Evolution of the admissible reference.



(b) Error measured between the reference provided to the vehicle and its position in time along the two coordinates x and y.

Fig. 6.32 Local thrust constraints validation: Plot showing the evolution of the admissible reference and the tracking error in the pier.

are conducted in a pool and a pier environment. Since the vision sensor was not fully developed at this stage, static obstacle positions were directly provided to the supervision scheme, rather than being detected using a stereo camera.

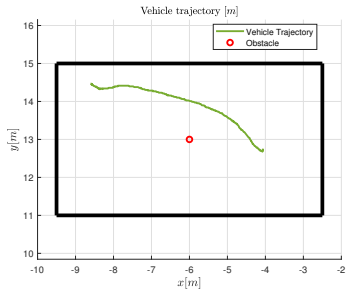
Pool

To assess the effectiveness of the supervision scheme in a challenging environment, initial experiments are conducted in a pool (Figure 6.9(a)). A small obstacle is placed in the center of the pool, and two tests are carried out, both using the supervision scheme and without using the supervision scheme. The system is subject to the following constraints

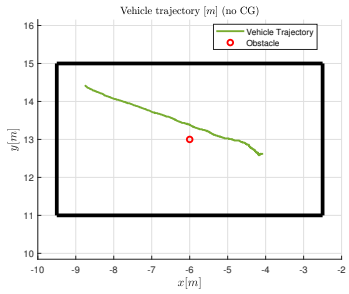
$$\begin{cases} \|\dot{p}^x(t)\| \leq 0.25\,[\frac{m}{s}], \\ \|\dot{p}^y(t)\| \leq 0.25\,[\frac{m}{s}], \\ \|p(t) - p^{ob}(t)\| \geq 0.85\,[m], \forall t \in \mathbb{Z}_+. \end{cases} \tag{6.6}$$

More in detail, $p^{ob}(t) = [0.0\,m, 0.0\,m]^T$ represents where the obstacle has been placed in. Initially, the vehicle is positioned in $[-4.1\,m, 12.8\,m, 3.2\,rad]^T$ and instructed to reach $[-8.6\,m, 14.5\,m, 0.0\,rad]^T$ with the supervision scheme enabled. Upon reaching the target, the supervision scheme is disabled, and the vehicle is instructed to return to its initial position. Figure 6.33 compares the trajectories of the vehicle with and without the supervision scheme (Figures 6.33(a) and 6.33(b)). As shown in Figures 6.34(a) and 6.35, the supervision scheme appropriately modifies the given reference to fulfill speed and obstacle avoidance constraints (6.6). In contrast, without supervision, the vehicle

approaches the obstacle dangerously close, violating both speed and obstacle avoidance constraints (Figures 6.36 and 6.34(b)), highlight the importance of the supervision strategy.

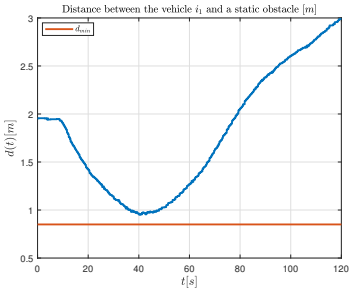


(a) Measured trajectory during the obstacle avoidance experiment with the distributed supervision scheme.

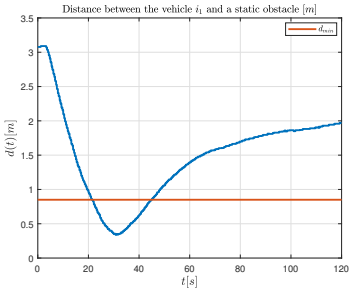


(b) Measured trajectory during the obstacle avoidance experiment without the distributed supervision scheme.

Fig. 6.33 Obstacle Avoidance experiment: Vehicle trajectories in pool with and without the supervision scheme.

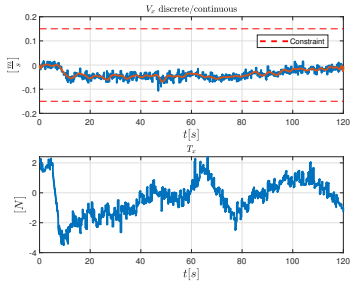


(a) Measured distance during the obstacle avoidance experiment with the distributed supervision scheme.

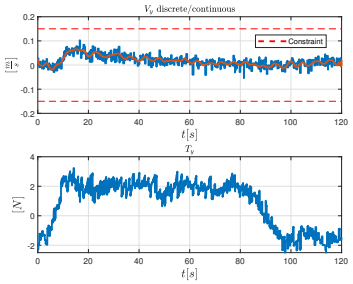


(b) Measured distance during the obstacle avoidance experiment without the distributed supervision scheme.

Fig. 6.34 Obstacle avoidance constraints validation: Plot showing the vehicle measured with and without the distributed supervision scheme in the pool.

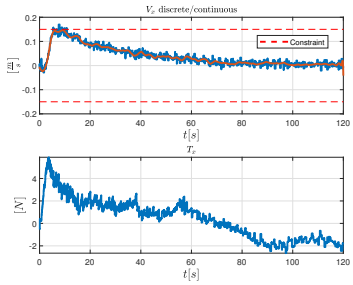


(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.

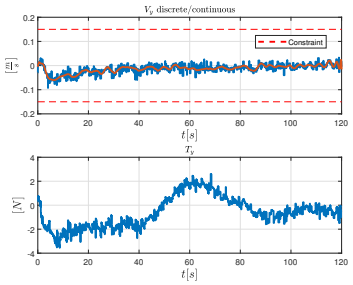


(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

Fig. 6.35 Obstacle avoidance constraints validation: Plot showing the vehicle measured velocities and computed thrusts in the pool.



(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.



(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

Fig. 6.36 Obstacle avoidance constraints validation: Plot showing the vehicle measured velocities and computed thrusts without the supervision scheme in the pool.

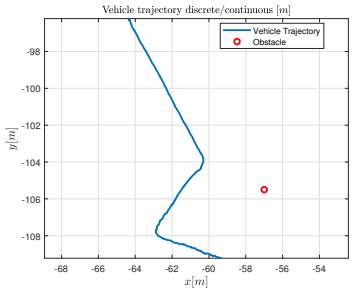
Pier

As mentioned previously, the second obstacle avoidance experiment is carried out at the pier (Figure 6.9(b)). The following constraints

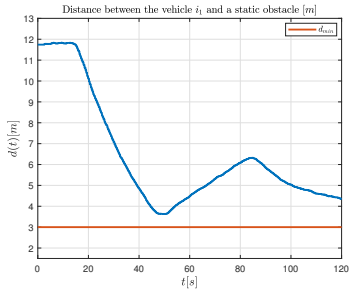
$$\begin{cases} \|T_x(t)\| \leq 10\,[N], \\ \|T_y(t)\| \leq 10\,[N], \\ \|p(t) - p^{ob}(t)\| \geq 3\,[m], \forall t \in \mathbb{Z}_+ \end{cases} \tag{6.7}$$

have been imposed on the system. In this case, $p^{ob}(t) = [-57.0\,m, -105.5\,m]^T$. The vehicle is initially positioned in $[-64.3\,m, -96.3\,m, 4.6\,rad]^T$ and is instructed to reach a reference located in $[-56.8\,m, -110.2\,m, 3.2\,rad]^T$. Figure

6.37(a) depicts an excerpt of the measured trajectory of the vehicle.As it can be seen in Figure 6.38 and Figure 6.37(b), when the supervision strategy is enabled, the given reference is appropriately modified so that both thrust and obstacle avoidance constraints (6.7) are satisfied.

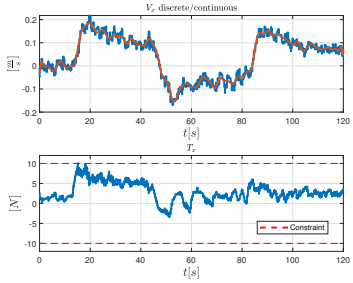


(a) Vehicle trajectory while avoiding an obstacle in the pier.

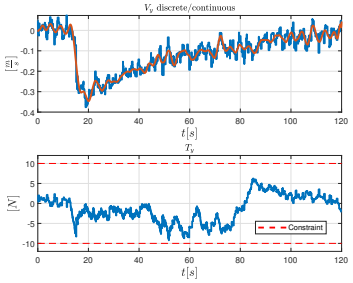


(b) Evolution of the measured distance between the vehicle and the obstacle.

Fig. 6.37 Obstacle avoidance constraints validation: Plots showing the measured trajectory and distance with the obstacle in the pier.



(a) Evolution of the measured velocity along the x-axis and the computed thrust T_x.



(b) Evolution of the measured velocity along the y-axis and the computed thrust T_y.

Fig. 6.38 Obstacle avoidance constraints validation: Plot showing the measured velocities and computed thrusts of the vehicle in the pier.

6.4.3.3 Collision Avoidance

This subsection presents the validation of the distributed Turn-Based supervision scheme for two agents with collision avoidance constraints. The pool

environment was chosen for testing due to its challenging nature, since it is characterized by a confined space, and successful operation in this environment could suggests its effectiveness in less constrained scenarios.

While the tests were limited to two vehicles due to practical constraints, these results provide valuable insights into the computational performance of the proposed strategy. Additionally, the scalability of the distributed approach suggests that adding more agents would not significantly increase the computational burden in relatively sparse networks.

Pool

In these experiments, two omnidirectional surface vehicles were instructed to swap their positions. Each vehicle was instructed to reach to a reference point that initially corresponded to the position of the other vehicle. The values of the initial positions and references for the two experiments Ex_1 and Ex_2 are reported in Table 6.4.

Table 6.4 Collision avoidance experiments initial positions and references.

	Experiment Ex_1	Experiment Ex_2
$p_1(t_0)$	$[-6.0\,m, 14.0\,m, 0.0\,rad]^T$	$[-10.0\,m, 14.9\,m, 0.0\,rad]^T$
$r_1(t)$	$[-10.0\,m, 14.9\,m, 0.0\,rad]^T$	$[-6.0\,m, 14.0\,m, 0.0\,rad]^T$
$p_2(t_0)$	$[-10.0\,m, 14.9\,m, 0.0\,rad]^T$	$[-6.0\,m, 14.0\,m, 0.0\,rad]^T$
$r_2(t)$	$[-6.0\,m, 14.0\,m, 0.0\,rad]^T$	$[-10.0\,m, 14.9\,m, 0.0\,rad]^T$

Each vehicle $i \in \mathcal{A} = \{1, 2\}$ is instructed to fulfill the following local constraints

$$\begin{cases} \|\dot{p}_1^l(t)\| \leq 0.1\,[\frac{m}{s}], \\ \|\dot{p}_2^l(t)\| \leq 0.2\,[\frac{m}{s}], \\ \text{with } l = \{x, y\} \text{ and } \forall t \in \mathbb{Z}_+ \end{cases} \tag{6.8}$$

and collision avoidance coupling constraints

$$\Big\{ \|p_i(t) - p_j(t)\| \geq 0.8\,[m],\ \forall j \in \mathcal{A} \setminus \{i\}. \tag{6.9}$$

During the experiment, the CG-based strategy effectively modified the given references to generate admissible commands that when applied fulfilled the imposed constraints. Figure 6.39 shows the vehicles during experiment Ex_1, while a video recording of such experiment can be found at `https://youtu.be/cRBIl1nqh3o`. During the recording, the camera is positioned at roughly $x = -11\,[m]$, $y = 14\,[m]$.

Figure 6.40 shows the trajectory of vehicle i_1 (green) and i_2 (blue) during both experiments Ex_1 (Figure 6.40(a)) and Ex_2 (Figure 6.40(b)). Please notice that vehicle i_1 has ESCs equipped with BLHeli firmware, while ESCs of

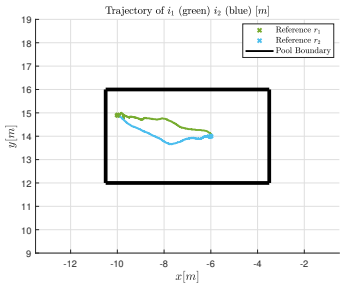
(a) Vehicles before avoidance maneuver.

(b) During avoidance maneuver.

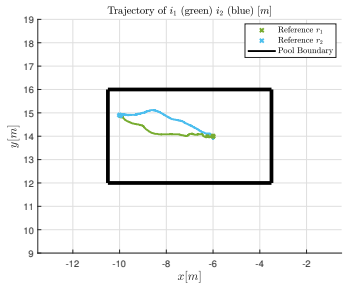
(c) Vehicles after avoidance maneuver.

Fig. 6.39 Collision Avoidance experiment Ex_1 involving two Omnidirectional Surface Vehicles.

vehicle i_2 use BLHeli_S firmware. As discussed in Remark 6.2, this difference in firmware leads to a larger deadzone in the thrusters of vehicle i_1, resulting in less smooth trajectory compared to vehicle i_2.



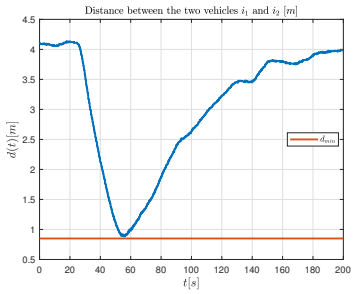
(a) Measured trajectories during experiment Ex_1.



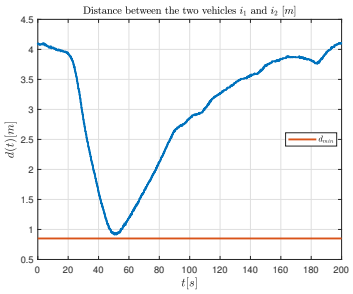
(b) Measured trajectories during experiment Ex_2.

Fig. 6.40 Measured trajectories of the two surface vehicles during the Collision Avoidance experiment in the pool.

For both experiments, the measured distance between vehicle i_1 and i_2, that never violates the imposed constraint, is depicted in Figure 6.41, while the measured velocities are reported in Figure 6.42 and Figure 6.43.

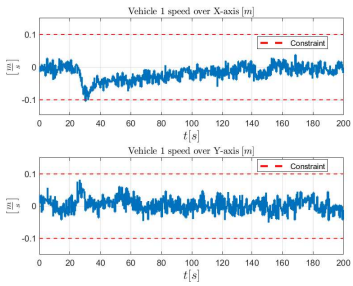


(a) Measured distance during experiment Ex_1.

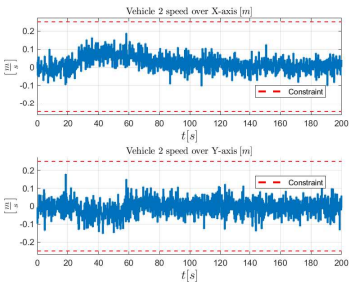


(b) Measured distance during experiment Ex_2.

Fig. 6.41 Measured distances between the two surface vehicles during the Collision Avoidance experiment in the pool.



(a) Evolution of the measured velocities of vehicle i_1 during experiment Ex_1.

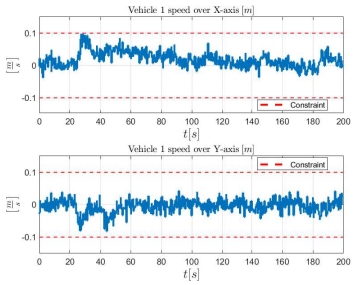


(b) Evolution of the measured velocities of vehicle i_2 during experiment Ex_1.

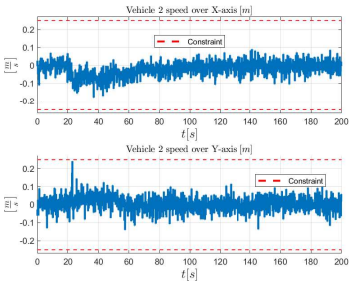
Fig. 6.42 Collision avoidance constraints validation: Plot showing the measured velocities of the two vehicles during experiment Ex_1.

For sake of completeness, the computed admissible references are shown in Figure 6.44 and Figure 6.45.

Finally, to further validate the simulation results, experiment Ex_1 was also conducted in Matlab. Figure 6.46 shows the measured distance in the simulation, which closely aligns with the real-world results, reinforcing the simulation results presented in the previous chapters.

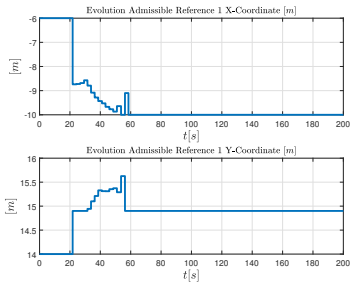


(a) Evolution of the measured velocities of vehicle i_1 during experiment Ex_2.

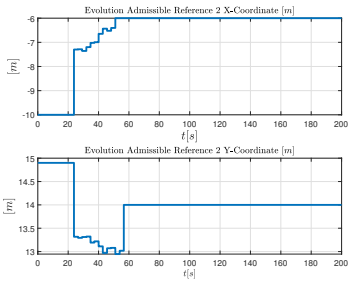


(b) Evolution of the measured velocities of vehicle i_2 during experiment Ex_2.

Fig. 6.43 Collision avoidance constraints validation: Plot showing the measured velocities of the two vehicles during experiment Ex_2.



(a) Evolution of the admissible reference of vehicle i_1 during experiment Ex_1.

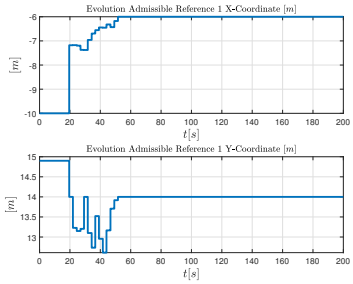


(b) Evolution of the admissible reference of vehicle i_2 during experiment Ex_1.

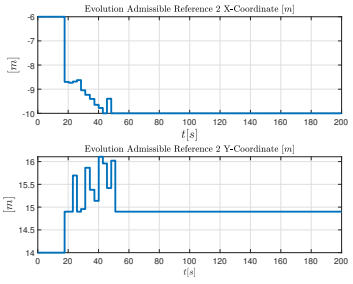
Fig. 6.44 Collision avoidance constraints validation: Plot showing the evolution of the admissible references of the two vehicles during experiment Ex_1.

6.4.3.4 Computational Details

Code optimization was taken into account. For instance, to speed up the computation, offline computation of matrices associated to predictions has been implemented. The sampling time of the inner-vehicle control loop T_s^c is equal to 0.01 $[s]$ while the sampling time of the supervision architecture T_s^{cg} is set on average to 0.5 $[s]$. More in detail, given the relatively slow dynamics of the marine environment and the low vehicle speeds, the supervision sampling time T_s^{cg} can be increased. In the experimental validation showed above, different values of the *admissibility index* k_0 and Ψ weighting matrix have been used.



(a) Evolution of the admissible reference of vehicle i_2 during experiment Ex_2.



(b) Evolution of the admissible reference of vehicle i_2 during experiment Ex_2.

Fig. 6.45 Collision avoidance constraints validation: Plot showing the evolution of the admissible references of the two vehicles during experiment Ex_2.

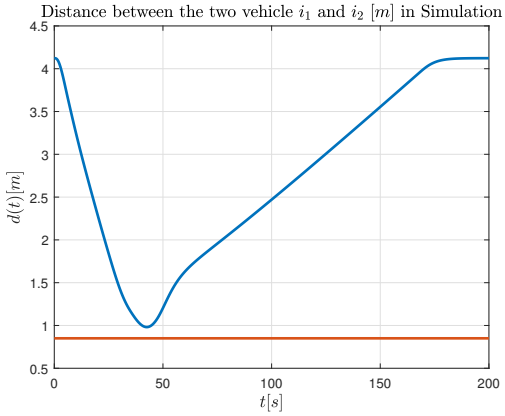


Fig. 6.46 Measured distance during experiment Ex_1 in Matlab.

Table 6.5 Computational details of the distributed CG using Gurobi.

Admissibility index k_0	Average Computational Time
$k_0^{\min} = 20$	$0.12\,[s]$
$k_0^{\max} = 60$	$0.41\,[s]$

Table 6.5 reports the average measured computational times required to solve the optimization problem in relation to both the lowest $k_0^{\min} = 20$ and highest $k_0^{\max} = 60$ values the of *admissibility index* used in the experiments. The average time is lower than the T_s^{cg} time, indicating that the optimization problem is consistently solved within the time window reserved to solve the optimization problem. In fact, the maximum value, equal to $0.41\,[s]$ is lower than $0.5\,[s]$. Please notice that the computational times reported in

Table 6.5 were measured using the specific hardware described in Section 6.2.2. These results demonstrate the feasibility of the proposed strategy on real-world robotic platforms. It is important to emphasize that these values provide general insights into the implementability of distributed CG-based strategies, and they can vary significantly based on hardware capabilities, software optimizations, and solver choices. For instance, using more powerful hardware like the UDOO Bolt v8 or different programming languages and solvers could lead to improved performance results. An interesting aspect is the scalability analysis, which provides insights into the performance of the approach with increasing numbers of vehicles and constraints. Considering that adding more vehicles would result in the addition of new constraints, it is expected that the computational performance would not degrade significantly in a distributed setting for networks with a relatively small number of communication links. This expectation is supported by simulations, as illustrated in Figure 6.47, which shows the computation time for solving the optimization problem (3.20) for networks with a relatively sparse communication topology. To highlight the difference in computational burdens and to demonstrate scalability, both centralized and distributed approaches have been used. The presented results were obtained using a CG sample time of $0.1\,[s]$ and CVXPY [106] within the Gazebo Simulator. The simulations were executed on a *Ryzen 5600X* CPU, a 6-core processor with a $3,7\,[GHz]$ clock rate.

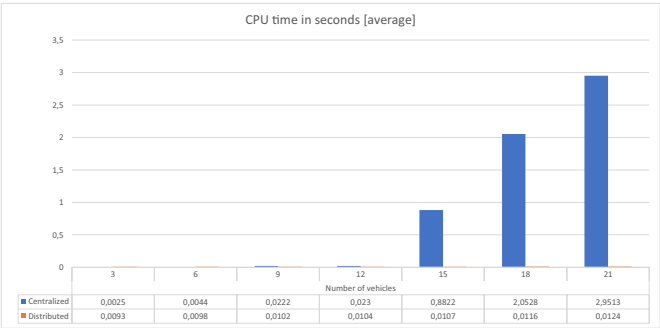


Fig. 6.47 Computational time comparison between distributed and centralized approaches.

As far as communication is concerned, vehicles communicate with each other peer-to-peer, by means of a Wireless Local Area Network (WLAN). More in detail, once coupling constraints have been enforced, vehicles publish and read messages from the *vehicle/info* topic (Figure 3.5). Due to space reasons, a detailed description of all ROS messages is omitted. However, the message exchanged between vehicles, the *Info_msg*, published at a rate of 100 messages per second, is reported in Table 6.6.

Table 6.6 Description of the custom ROS message *Info_msg*.

Data	Data Type	Components
Vehicle state	Int64	6
Controller state	Int64	3
Admissible Command	Int64	3
Vehicle ID	Int16	1

On average, each vehicle transmits and receives data at a rate of 76.5625 *kbps*: $100 \times 12 \times 64$ bit/s for both vector x (which has 9 components) and g (which has 3 components); $100 \times 1 \times 16$ bit/s for the vehicle id. More in general, the number of bit/s received by a generic agent i is equal to $|\mathcal{N}_i| \times r_c \times 784$ bit/s, where r_c is the communication rate per second. Please notice that state-related information is transmitted at a significantly higher rate than the sampling time of the CG, ensuring resilience to occasional packet loss. The reliable communication channel and close proximity of the vehicles further minimize the likelihood of packet loss. CG-based schemes usually present a certain level of robustness with respect to model uncertainty and/or communication issues non directly counted during the design phase. As a matter of fact, during the described experiments the system maintained consistent performance, with all agents successfully coordinating their actions without interruption. On the other hand, the proposed scheme can be easily extended to handle more challenging communication environments by leveraging the work of [81]. This work offers techniques for supervising networked dynamical systems under coordination constraints, which can be integrated into our architecture to handle communication delays more effectively. Finally, since the current communication setup does not prioritize minimizing data exchange rates, significant reductions can be achieved when alternative communication channels are used. For instance, by reducing data packet sizes from 64 bits to 8 bits per component and aligning the transmission rate with the sampling time of the CG, the overall data exchange can be substantially reduced.

Remark 6.4 To reduce the computational burden and overall data exchange rates associated to each agent, dynamic CG-based approaches can also be used [169].

6.5 Concluding Remarks

In this chapter, the goals to be achieved were the following

- present a detailed description of the engineering design and implementation of a small ASV with a near-zero turning radius. This design makes it particularly suitable for operating in swarms in confined spaces;

- demonstrate, through experimental validation, that a linear time-invariant mathematical model can effectively approximate the dynamics of the vehicle under specific assumptions. This model is particularly beneficial for the control design phase and for reducing the computational burden associated with developing predictive control strategies;
- Assess the practical implementability of the proposed distributed CG-based strategies, carrying out real-world field tests involving the designed and built marine surface vehicles.

Firstly, the main design criteria are discussed. Near-zero turning radius, that allows the ASV to orient itself toward desired heading angle without changing its position, is achieved by selecting the cylindrical shape for the hull design. The design prioritized practicality and real-time monitoring, balancing portability (small size, lightweight for two-person deployment) with navigation performance, using durable materials for environmental resilience. Furthermore, design prioritized precise maneuvering instead of fast maneuvering. The main computer board has been chosen to simplify hardware/-software integration, and the USB standard has been preferred to enhance modularity, facilitating the integration of scientific sensors. Component selection focused on minimizing power consumption to maximize operational endurance. Calculations indicate an estimated 5-hour endurance, which can provide sustained monitoring capabilities suitable for comprehensive environmental data collection. Furthermore, obstacle detection capabilities have been given by means of using a stereo camera.

In the second part, the parameters of the proposed linear mathematical model were identified and the estimated model was validated using a velocity prediction approach, where the prediction horizon was selected to be aligned with the requirements of model-based control. The resulting model was then augmented to design a state-feedback control law. Control law design was followed by experimental validation of the controlled system using a basic ROS-based GNC architecture. Initial validation occurred in a controlled pool environment, demonstrating its suitability for applications requiring high-precision positioning (e.g., bathymetry and confined space monitoring). Subsequent testing at a pier environment assessed its waypoint navigation capabilities in real-world environmental monitoring scenarios, showing that vehicle can be employed for diverse data collection tasks, with flexibility in sensor payloads for water quality and environmental monitoring. Beyond its multi-purpose functionality and adaptability, a significant advantage lies in the validated, computationally efficient mathematical model, which serves as a stepping stone for the development of predictive supervision strategies.

In the last part, the basic ROS-based GNC architecture has been extended to accommodate a distributed Command Governor-based strategy in order to assess the capability of this strategy to meet the typical computational requirements imposed by the real time applications. Several tests, carried out in various environments, have validated, under the stated assumptions, such as negligible currents and calm water, the capability of the supervisory

scheme to enforce local constraints on speed and thrust, to enforce obstacle avoidance constraints in presence of static obstacles and to satisfy collision avoidance constraints. The results demonstrated the computational efficiency of the distributed CG-based approach. Although the hardware and programming language choices can influence computational performance, these tests provide valuable insights into the implementability and effectiveness of the proposed strategy. Moreover, the simulation results align closely with these results, further verifying the efficacy of simulation-based validation.

Future research could focus on enhancing the controller to accommodate rougher sea conditions, which, as observed, the current system does not fully address. Moreover, given the time-consuming nature of identifying and validating the model for different environments, such as transitioning from a pool to the sea, a data-driven approach can be considered. In this case, after designing an output controller, subsequent controller adjustments could be made without requiring a complete re-identification and validation process.

Chapter 7
Conclusions and Directions for Future Research

This dissertation has introduced and investigated Command Governor-based supervision strategies applied to surface marine multi-vehicle autonomous systems, developing specialized theoretical, computational, and simulative/-experimental results in each chapter. The first approach consists of a distributed supervision scheme for a fleet of vehicles moving in 2D and subject to a comprehensive set of constraints, deemed essential for practical applications. More in detail, the hierarchy structure of a non-iterative scheme has been exploited, and the resulting scheme guarantees satisfaction of local, obstacle avoidance, collision avoidance, and connectivity maintenance, while it ensures formation compactness and induces cooperation during obstacle avoidance maneuvers, with much more moderate computational burden than traditional iterative schemes. The second approach consists of a cooperative distributed CG scheme for supervising dynamically coupled interconnected linear systems subject to local and coupling constraints. In contrast to other CG-based approaches, the proposed approach enables agents to obtain a feasible, near-optimal solution to the centralized problem in a distributed and finite-time manner. Furthermore, with the cooperative distributed CG scheme, where the RSDD algorithm has been adopted, constrained coordination problems are addressed while ensuring privacy by avoiding the exchange of information related to objective functions, constraints, or admissible commands. Motivated by the time-consuming nature of modeling, identification, and validation, particularly evident during the study of marine surface vehicles, the third approach consists of a data-driven supervision scheme, useful to supervise a linear time-invariant discrete-time system without relying on the system model. Concepts from behavioral system theory and unfalsified control have been leveraged and an alternative to model-based CG supervision schemes was specifically designed. The latter uses offline noise-free data from the plant and a representation of the controller. In contrast to existing approaches, changes in the controller do not require to run the entire process of offline data collection, making the approach more practical and paving the way for data-driven schemes with time-varying controllers. The final part

detailed comprehensive design, identification, validation, control, and supervision of omnidirectional marine surface vehicle prototypes. To fully adhere to the assumptions and setting introduced, highly maneuverable vehicles, ideal for swarm operations in confined spaces, are considered and a validated linear model is employed to represent their open-loop dynamics. Since optimization based methods are generally associated to expensive computational costs, particularly when considering requirements such as constraints on vehicle states, obstacle and collision avoidance requirements, which can limit real-time implementation, the practical feasibility of the proposed distributed CG-based strategies has been assessed through real-world field tests.

Future research endeavors could focus on: extending the range of applications by developing cooperation-inducing solutions to deal effectively with external disturbances. This would involve addressing challenges such as preserving the modularity of the strategy and avoiding overly conservative approaches; improving the computational performance of the cooperative approach, by addressing the challenge of reducing the number of iterations without compromising solution properties, potentially increasing its applicability; extending the data-driven approach to handle noisy data, time-varying controllers, and multi-agent systems. The latter would require careful consideration of aspects such as guaranteeing overall feasibility and efficient data exchange, especially for trajectory sharing.

Finally, after the recognition of the inter-dependence between human and aquatic ecosystem health, it is hoped that this research can contribute to facilitate both monitoring and gaining a deeper understanding of these ecosystem, ultimately helping the efforts that have to be made to restore and preserve the precious aquatic environments.

Appendix A
Computational Details of the Command Governor with Linear Constraints

This appendix details the computational details required to implement the CG strategy when linear constraints are considered and recalls material presented in [156].

A.1 The disturbance-free case

In this section, all computational details for the design and the implementation of the above CG strategy in the disturbance-free case $d(t) = 0,\ \forall t$, are presented. Because linear constraints for $c(t) \in \mathcal{C}$ are assumed, they can be represented as

$$c(t) \in \mathcal{C} \Longleftrightarrow \mathcal{C} := \{c \in \mathbb{R}^{n_c} : Tc \leq q\} \tag{A.1}$$

$$T = \begin{bmatrix} T_1^T \\ T_2^T \\ \cdots \\ T_z^T \end{bmatrix} \in \mathbb{R}^{z \times n_c},\ q = \begin{bmatrix} q_1 \\ q_2 \\ \cdots \\ q_z \end{bmatrix} \in \mathbb{R}^z,\ z \geq n_c, \tag{A.2}$$

$$\text{rank } (T) = n_c, \tag{A.3}$$

with T_i^T and q_i denoting respectively the i-th rows of T and q. Notice that the number z of rows of T is in general larger than the number of columns and the rank condition on T ensures that $\mathcal{C}$ is a bounded set. Closure and convexity trivially follow.

In the disturbance-free case all sets $\mathcal{C}_k$ defined in 2.14 coincide with $\mathcal{C}$. Therefore, the set $\mathcal{C}_\delta$ is given by

$$\mathcal{C}_\delta = \mathcal{C} \sim \mathcal{B}_\delta = \{c \in \mathbb{R}^{n_c} \ :\ T_i^T c \leq q_i - \delta\sqrt{T_i^T T_i},\ i = 1, \ldots, z\},$$

where the terms $\delta\sqrt{T_i^T T_i}$ is the support function of the ball $\mathcal{B}_\delta$. Then, the $\mathcal{W}_\delta$ set can be directly computed and results

$$\mathcal{W}_\delta = \{g \in \mathbb{R}^m : \ c_g \in \mathcal{C}_\delta\} \tag{A.4}$$

$$= \{g \in \mathbb{R}^m : \ T\left(H_c(I-\Phi)^{-1}G + L\right) g \leq q - \delta \begin{bmatrix} \sqrt{T_1^T T_1} \\ \sqrt{T_2^T T_2} \\ \cdots \\ \sqrt{T_z^T T_z} \end{bmatrix}\}. \tag{A.5}$$

It remains to characterize the set $\mathcal{V}(x)$ which, in the disturbance-free case, is defined as

$$\mathcal{V}(x) = \{g \in \mathcal{W}^\delta : \ c(k,x,g) \in \mathcal{C}, \ k = 0, \ldots, k_0\}, \tag{A.6}$$

with k_0 the integer referred to as *admissibility index* in Section 2.2.3 and whose computation for this disturbance-free case is specified at the end of this section. Following the definitions of evolution (2.7) and (2.8)

$$\begin{aligned} x(k,x,g) &= \Phi^k x + \left(\sum_{i=0}^{k-1} \Phi^i G\right) g \\ &= \Phi^k x + R_k^x g, \qquad (A.7) \\ c(k,x,g) &= H_c \bar{x}(k,x,g) + Lg \\ &= H_c \Phi^k x + (H_c R_k + L)\, g \\ &= H_c \Phi^k x + R_k^c g. \qquad (A.8) \end{aligned}$$

Substituting (A.8) in (A.6)

$$\mathcal{V}(x) = \{g \in \mathcal{W}^\delta : \ T H_c \Phi^k x + T R_k^c g \leq q, \ k = 0, \ldots, k_0\}. \tag{A.9}$$

Finally, following (2.21), the CG action computation consists of solving the following QP optimization problem

$$g(t) = \arg\min_g \|g - r(t)\|_\Psi^2 \tag{A.10}$$

$$\text{s.t.} \tag{A.11}$$

$$T H_c \Phi^k x(t) + T R_k^c g \leq q, \ k = 0, \ldots, k_0, \tag{A.12}$$

$$T\left(H_c(I-\Phi)^{-1}G + L\right) g \leq q - \delta[\sqrt{T_i^T T_i}]. \tag{A.13}$$

Then, the following procedure, which is ensured to converge in a finite number of steps, can be used to compute the *admissibility index* k_0

Algorithm for Admissibility Index k_0 ***Computation: Disturbance-free case***
INITIALIZE $k = 1$

1 COMPUTE $G_k(j)$ AS

$$G_k(j) := \max_{x\in\mathbb{R}^n, g\in\mathcal{W}_\delta} \quad T_j^T c(k,x,g) - q_j \qquad \text{(A.14)}$$
$$\text{s.t.}$$
$$T_j^T c(i,x,g) \le q_j, \quad i = 0,\dots,k-1.$$

2 **If** $G_k(j) \le 0,\ \forall j = 1,\dots,z$

2.1 SET $k_0 = k$
2.2 GO TO 4

3 **Else**

3.1 SET $k = k+1$
3.2 GO TO 2

4 RETURN k_0

A.2 The disturbance acting case

We assume here that the disturbances are bounded and characterizable as follows

$$d(t) \in \mathcal{D} \Longleftrightarrow \mathcal{D} := \{d \in \mathbb{R}^{n_d} \ :\ Ud \le h\} \qquad \text{(A.15)}$$

$$U = \begin{bmatrix} U_1^T \\ U_2^T \\ \cdots \\ U_u^T \end{bmatrix} \in \mathbb{R}^{u\times n_d}, \quad h = \begin{bmatrix} h_1 \\ h_2 \\ \cdots \\ h_u \end{bmatrix} \in \mathbb{R}^u, \qquad \text{(A.16)}$$

$$u \ge n_d,\ \text{rank}\ U = n_d,\ h_i \ge 0, \qquad \text{(A.17)}$$

with U_i^T and h_i denoting respectively the i-th rows of U and h. Notice that the number u of rows of U is in general larger than the number of columns. The rank condition on U ensures that $\mathcal{D}$ is a bounded set whereas $h_i \ge 0$ ensures that $0_{n_d} \in \mathcal{D}$. Closure and convexity trivially follow.

The set inclusions (2.13)-(2.14) can be computed as follows

$$\begin{aligned} \mathcal{C}_k &= \mathcal{C}_{k-1} \sim H_c\Phi^{k-1}G_d\mathcal{D} \qquad \text{(A.18)}\\ &= \{c \in \mathbb{R}^{n_c}\colon T_i^T c \le q_i^{k-1} - \sup_{d\in\mathcal{D}} T_i^T H_c\Phi^{k-1}G_d d,\ i = 1,\dots,z\} \\ &= \{c \in \mathbb{R}^{n_c} : T_i^T c \le q_i^k,\ i = 1,\dots,z\}, \end{aligned}$$

where

$$\begin{aligned}\mathcal{C}_0 = \mathcal{C} \sim L_d\mathcal{D} &= \{c \in \mathbb{R}^{n_c} : T_i^T c \leq q_i - \sup_{d\in\mathcal{D}} T_i^T L_d d,\ i = 1,\dots,z\} \\ &= \{c \in \mathbb{R}^{n_c} : T_i^T c \leq q_i^0,\ i = 1,\dots,z\}. \end{aligned} \tag{A.19}$$

In order to compute $\mathcal{W}_\delta$ one should evaluate $\mathcal{C}_\delta$ by computing $\mathcal{C}_\infty$ first. To this end, a convenient approximation of $\mathcal{C}_\infty$ can becomputed by looking for a convenient set $\mathcal{C}_\infty(\varepsilon)$ such that

$$\mathcal{C}_\infty(\varepsilon) \subset \mathcal{C}_\infty \subset \mathcal{C}_\infty(\varepsilon) + \mathcal{B}_\varepsilon. \tag{A.20}$$

Such a set, unlike $\mathcal{C}_\infty$, is computable in a finite number of steps. In fact, it can be shown that

$$\mathcal{C}_\infty = \mathcal{C}_k \sim \left(\sum_{i=k}^{\infty} H_c \Phi^i G_d \mathcal{D}\right). \tag{A.21}$$

Moreover, because of stability of Φ, there exist two constants $M > 0$ and $\lambda \in (0,1)$ such that

$$\|\Phi^k\| \leq M\lambda^k.$$

Boundedness of $\mathcal{D}$ also implies that there exists finite

$$\bar{d} := \max_{d\in\mathcal{D}} \|d\|_2. \tag{A.22}$$

The above facts are enough to ensure that, once chosen the desired accuracy ε, there exists an index k_ε such that

$$\sum_{i=k_\varepsilon}^{\infty} H_c \Phi^i G_d \mathcal{D} \subset \mathcal{B}_\varepsilon. \tag{A.23}$$

In fact, it suffices that

$$\bar{d}\bar{\sigma}(H_c)\bar{\sigma}(G_d) M \sum_{i=k_\varepsilon}^{\infty} \lambda^i \leq \varepsilon, \tag{A.24}$$

which, after direct steps, gives rise to

$$k_\varepsilon = \frac{\log\varepsilon + \log(1-\lambda) - \log[\bar{\sigma}(H_c)\bar{\sigma}(G_d)M\bar{d}]}{\log\lambda}. \tag{A.25}$$

Then, the desired approximation can be computed as

$$\mathcal{C}_\infty(\varepsilon) = \mathcal{C}_{k_\varepsilon} \sim \mathcal{B}_\varepsilon \tag{A.26}$$

and, in turn,

$$\begin{aligned}\mathcal{C}_\delta(\varepsilon) &= (\mathcal{C}_{k_\varepsilon} \sim \mathcal{B}_\varepsilon) \sim \mathcal{B}_\delta \\ &= \{c \in \mathbb{R}^{n_c} \,:\, T_i^T c \leq q^{k_\varepsilon} - (\varepsilon+\delta)[\sqrt{T_i^T T_i}]\}, \qquad \text{(A.27)} \\ \mathcal{W}_\delta &= \{g \in \mathbb{R}^m \,:\, c_g \in \mathcal{C}_\infty^\delta(\varepsilon)\} \qquad \text{(A.28)} \\ &= \{g \in \mathbb{R}^m : T\big(H_c(I-\Phi)^{-1}G+L\big)\, g \leq q^{k_\varepsilon} - (\varepsilon+\delta)[\sqrt{T_i^T T_i}]\}. \qquad \text{(A.29)}\end{aligned}$$

Of course, emptiness of $\mathcal{C}_\infty^\delta(\varepsilon)$ or $\mathcal{W}_\delta$, that is at least one component of $q^{k_\varepsilon} - (\varepsilon+\delta)[\sqrt{T_i^T T_i}]$ is strictly negative, implies that the problem is not solvable.

In this case, $\mathcal{V}(x)$ can be characterized as

$$\mathcal{V}(x) = \{g \in \mathcal{W}^\delta \,:\, c(k,x,g) \in \mathcal{C}_k,\ k = 0, \ldots, k_0\}, \qquad \text{(A.30)}$$

where the computation of the *admissibility index*, i.e. the integer k_0, for this case is specified at the end of this section. Because we work with disturbance-free predictions, we can use (A.7) and (A.8) also in this case. Hence

$$\mathcal{V}(x) = \{g \in \mathcal{W}^\delta \,:\, TH_c\Phi^k x + TR_k^c g \leq q^k,\ k = 0, \ldots, k_0\}. \qquad \text{(A.31)}$$

Finally, the CG action computation consists of solving the following QP optimization problem

$$\begin{aligned} g(t) &= \min_g (g - r(t))^T \Psi (g - r(t)) \qquad \text{(A.32)} \\ &\text{s.t.} \qquad \text{(A.33)} \\ TH_c\Phi^k x(t) + TR_k^c g &\leq q^k,\ k = 0, \ldots, k_0 \\ T\left(H_c(I-\Phi)^{-1}G + L\right) g &\leq q^{k_\varepsilon} - (\varepsilon+\delta)[\sqrt{T_i^T T_i}]. \end{aligned}$$

The computation of the *admissibility index* k_0 when considering disturbance, can be accomplished via the following procedure

Algorithm for Admissibility Index k_0 ***Computation: Disturbance acting case***
INITIALIZE $k = 1$

1 COMPUTE $G_k(j)$ AS

$$G_k(j) := \max_{x \in \mathbb{R}^n, g \in \mathcal{W}_\delta} \quad T_j^T c(k, x, g) - q_j^k \qquad \text{(A.34)}$$
$$\text{s.t.}$$
$$T_j^T c(i, x, g) \le q_j^i, \quad i = 0, \ldots, k-1.$$

2 **If** $G_k(j) \le 0, \ \forall j = 1, \ldots, z$

2.1 SET $k_0 = k$
2.2 GO TO 4

3 **Else**

3.1 SET $k = k + 1$
3.2 GO TO 2

4 RETURN k_0

Appendix B
Omnidirectional Marine Surface Vehicles Non Linear and Linear Modeling

This appendix aims at: *i*) providing kinematic and dynamic modeling for fully and/or over-actuated marine surface vehicles and *ii*) showing that, under specified assumptions, a linear model can be easily determined for this type of vehicles, for which validity has been proved through in field experiments. To this end, it is useful to introduce their kinematic and dynamic models.

B.1 Kinematics

The full 6 DOF model can be simplified to a 3 DOF model to describe planar motions in terms of surge, sway and yaw [157]. It is advantageous to define the earth-fixed frame $O_eX_eY_e$ and the body-fixed frame $O_bX_bY_b$ following the conventions of [158]. A schematic representation of the earth-fixed and body-fixed frames is provided in Figure B.1. The kinematics of the fully actuated USV, in planar motion and without the presence of disturbances, can be represented as

$$\begin{cases} \dot{p}^x = ucos(\theta) - vsin(\theta), \\ \dot{p}^y = usin(\theta) + vcos(\theta), \\ \dot{\theta} = \psi, \end{cases} \tag{B.1}$$

where $p^x(t)$ and $p^y(t)$ represent the x and y positions of the vehicle in an earth-fixed reference frame, θ is the yaw angle, u, v, and ψ are the corresponding velocities in the surge, sway and yaw in the body-fixed frame, respectively. Let $\hat{p} = [p^x, p^x, \theta]^T$ and $\nu = [u, v, \psi]^T$.

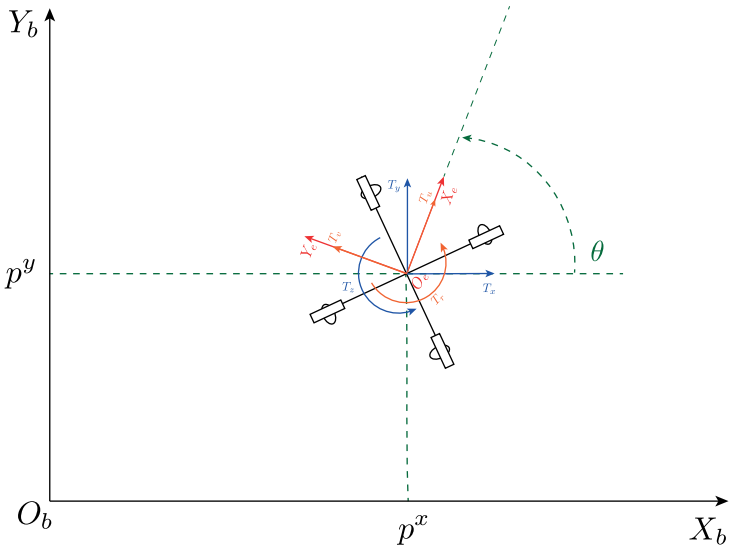


Fig. B.1 Earth-fixed and body-fixed frames and generated thrusts and moments characterization.

B.1.1 Allocation Matrix

For the considered type of vehicle, thrusters are placed under the buoy at the corners of a square inscribed in the base of the buoy itself. These are inclined at an angle α relative to the horizontal axis of the vehicle and placed at a distance l from the center. The thrust vector T_u (Figure B.2(a)) is given by the sum of the projections of the individual thrusts of the 4 thrusters onto the T_u direction

$$T_u = T_1 sin(\alpha) + T_2 sin(\alpha) - T_3 sin(\alpha) - T_4 sin(\alpha),$$

while thrust T_v (Figure B.2(b)) has the following form

$$T_v = T_1 cos(\alpha) + T_2 cos(\alpha) - T_3 cos(\alpha) - T_4 cos(\alpha).$$

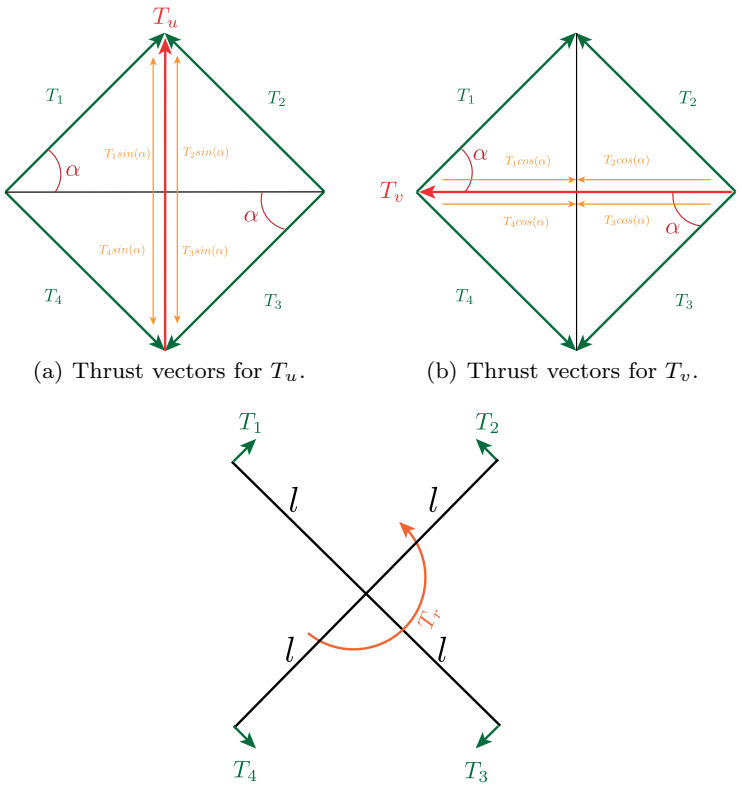
Finally, the torque T_r (Figure B.2(c)) are given by

$$T_r = -T_1 l + T_2 l - T_3 l + T_4 l.$$

In general, the relationship between thrust values expressed in the body-fixed frame and thrust values for individual thrusters can be expresssed as

$$T_{uvr} = \Sigma \cdot T_{1234}, \tag{B.2}$$

where



(a) Thrust vectors for T_u.

(b) Thrust vectors for T_v.

(c) Thrust vectors for torque T_r.

Fig. B.2 Thrust vectors in the body-fixed reference frame.

$$\Sigma = \begin{bmatrix} sin(\alpha) & sin(\alpha) & -sin(\alpha) & -sin(\alpha) \\ cos(\alpha) & -cos(\alpha) & -cos(\alpha) & -cos(\alpha) \\ -l & l & -l & l \end{bmatrix}. \tag{B.3}$$

To extract the thrusts $T_{1234} = [T_1, T_2, T_3, T_4]^T$ from the given the vector $T_{uvr} = [T_u, T_v, T_r]^T$, it is necessary to invert equation (B.2). For this allocation, individual thruster commands T_{1234} are calculated by multiplying the vector of commands T_{uvr} by the Moore–Penrose inverted thrust allocation matrix Σ^+

$$T_{1234} = \Sigma^+ \cdot T_{uvr} = (\Sigma^T \Sigma)^{-1} \Sigma^T \cdot T_{uvr}.$$

Please notice that in this case robustness in case of actuator failure is not guaranteed. In order to online redistribute the control effort among the remaining available healthy actuators after that some of them have failed,

thruster fault diagnosis and accommodation systems (FDASs) may be considered [159].

Remark B.1 More computational demanding methods, based on Mixed-Integer formulations, can be considered to describe non-linear static characterization of the actuators [168].

B.2 Dynamics

In order to better facilitate control design, existing research [8] assumes that

- USVs are moving in a horizontal plane in the ideal fluid;
- USV masses are uniformly distributed;
- The body-fixed coordinate axis coincides with the center of gravity;
- Both the center of gravity and center of buoyancy point vertically along the Z-axis;
- USVs own the port-starboard symmetry;
- Surge and sway-yaw dynamics are essentially decoupled.

Based on these assumptions and following the results of [160, 161], the dynamics of the vehicle is described by

$$M\dot{\nu} = -C(\nu)\nu - D(\nu)\nu + \tau, \tag{B.4}$$

where restoring forces and moments due to gravitation/buoyancy are neglected in (B.4) and

$$M = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix},$$

$$C(\nu) = \begin{bmatrix} 0 & 0 & -m_{22}v - \frac{1}{2}(m_{23}+m_{32})r \\ 0 & 0 & m_{11}u \\ m_{22}v + \frac{1}{2}(m_{23}+m_{32})r & -m_{11}u & 0 \end{bmatrix},$$

$$D(\nu) = \begin{bmatrix} d_{11}(u) & 0 & 0 \\ 0 & d_{22}(v,r) & d_{23}(v,r) \\ 0 & d_{32}(v,r) & d_{33}(v,r) \end{bmatrix},$$

where $D(\nu)$ is hydrodynamic non-linear damping matrix induced by skin friction, potential damping (due to the energy carried away by waves), vortex shedding damping, and wave drift damping; $C(\nu)$ is the coriolis and centripetal matrix (including added mass); and $\tau = [\tau_u, \tau_v, \tau_r]^T$ denotes the thrusts and the moments in the body frame.

B.2.1 Linear Dynamics

Although a more accurate and complete USV model that represents the physics of the real world is normally required, some common model simplifications and reductions are still inevitable in order to facilitate controller design. These phenomena are primarily due to the many practical challenges faced USV development, including hydrodynamic phenomenon (hydrodynamic forces and moments), which are still not fully understood. For the planned operative regimes of the vehicle, nonlinear dynamics are unnecessary. In fact, for this particular class of vehicles, a linear model of the vehicle's dynamic can be obtained from (B.4) through several simplification steps. First the ASV we deal with are fully actuated (actuated in surge, sway and yaw), very small in dimension and do normally operate on the surface in settings with glassy to rippling waters, such as in Radiation Monitoring of Pool Walls [145], and operative conditions for the model used throughout this thesis do not require high-speed profiles ($\leq 0.8\ m/s$). This type of model is excellent for applications where the vessel's goal is to track and maintain a desired position and orientation. Linear motion in the heave direction and angular motion in the roll and pitch directions are limited in this case. Moreover, fore/aft symmetry, fully actuation, absence of slipping (pure rolling) in the longitudinal direction and no sliding in the lateral direction creates independence between the longitudinal, lateral and angular velocities simplifying the dynamic equations [162, 163]. More specifically, the effects of nonlinear terms and the coupling between lateral, longitudinal and rotational dynamics can be neglected [164, 165]. Consequently, by considering the previous operative conditions, experimental validations have shown that the previously introduced dynamics can be approximated by mass points subject to viscous friction. Then, in the earth-fixed frame, by the following equations

$$\begin{cases} m\,\ddot{p}^x(t) + \beta\,\dot{p}^x(t) = T_x(t), \\ m\,\ddot{p}^y(t) + \beta\,\dot{p}^y(t) = T_y(t), \\ J\,\ddot{\theta}(t) + \beta_t\,\dot{\theta}(t) = T_z(t). \end{cases} \tag{B.5}$$

The parameter m is the mass of the vehicle, J is the inertia coefficient, while β is the friction coefficient and β_t rotational friction coefficient. Model (B.5) can also be expressed in the discrete state-space representation as

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = C_y x(t) + Du(t). \end{cases} \tag{B.6}$$

It is important to discuss the relationship between thrust expressed in the inertial frame and in the body frame. To compute the conversion from thrust expressed in body frame to thrust expressed in the inertial frame, and viceversa, the rotation matrix, which is a function of the orientation angle of the

vehicle θ, has to be used. The relationship can be expressed as

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = R(\theta) \begin{bmatrix} T_u \\ T_v \\ T_r \end{bmatrix}, \begin{bmatrix} T_u \\ T_v \\ T_r \end{bmatrix} = R^T(\theta) \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}, \tag{B.7}$$

where

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{B.8}$$

Appendix C
Gazebo low level physics-based Simulator

Testing algorithms for localization, navigation, and control in real-world scenarios can be both dangerous and expensive. Simulations offer a safe and controlled environment to evaluate these algorithms, providing realistic data to assess their performance in setting close to real-world deployments. Furthermore, given the significant cost and time associated with constructing a large fleet of physical vehicles, simulation provides an effective means of validating results. Gazebo Simulator has been selected due to its seamless integration with ROS, its ability to accurately simulate marine environments, and its widespread use in the robotics community. This appendix provides details on the simulation setup and details of the Gazebo simulator for marine environments.

C.1 Gazebo

The Gazebo Simulator, which can be used with ROS and ROS2, is an open-source dynamic multi-robot simulator, extensively utilized in robotic research and development. Gazebo offers a high level of realism when replicating physical interactions and environmental conditions, making it an invaluable platform for precise testing and development. Furthermore, its active community aims always at improving the environment, by enhancing its features or implementing new ones. The main components of such environment are

- *Worlds*: Simulation description format (SDF) files, creates a realistic representation of a 3D environment integrating basic geometric figures with complex meshes. They are usually composed of static and dynamic objects (buildings, natural elements, vehicles, pedestrians) and global parameters, such as the sky, ambient/artificial lights and physics properties;
- *Model files*: Models are SDF files describing entities with kinematic, dynamic and visual properties. A model is essentially a container for its descriptive elements: links, that represent fundamental rigid bodies that

composes the robot, and joints, that connect two links specifying a parent/child relationship.
- *Plugins*: A plugin is a shared library created to extend the functionalities of worlds and models. They can simulate the behavior of sensors (IMUs, cameras, encoders and more) and forces like gravity, buoyancy and wind. Plugins are also used to extend the functionalities of the Gazebo GUI to include useful tools for visualization (e.g. video recording).

These components significantly enhance simulation capabilities. *Worlds* provide a flexible environment for testing, while *Models* facilitate the inclusion and customization of the proposed prototype. The extensibility of Gazebo, through the use of plugins, further enhances its versatility." By leveraging two key packages, Gazebo can be effectively used to simulate marine surface vehicles

Remark C.1 In this thesis, the Gazebo Classic version has been used. However, newer version, such as Gazebo Sim, that offer compatibility with ROS2, are currently being released.

C.1.1 UUV Simulator

It is a Gazebo Classic extension capable of simulating multiple robots, evenly embedded of manipulators for underwater operations. Its main features are relative to sensors and actuators simulation, but also to the hydrodynamic interaction between the robot and the water, underwater worlds, vehicle models and even controllers [166].

An example of UUV Simulator features could be the *thrusters plugin*, divided in two modules: one representing the propeller dynamics, and the other one describing the steady-state relationship between the rotor's angular velocity and output thrust.

C.1.2 ASV Wave Simulator

Since accurate environmental modeling can significantly enhance the development and testing of autonomous surface vehicles, simulating realistic water interactions is of paramount importance. To accurately model the physical behavior of the robots interacting with the water and thus have reliable testing scenarios, Gazebo packages, that employ advanced water models, can be used. These achieve physical accuracy through the use of sophisticated mathematical models and computational techniques. More in detail, to simulate waves, surface vessels and their interaction forces, the ASV Wave Simula-

tor, supporting plug-ins like hydrodynamics, package is used. Thus, the main purpose of the ASV Wave Simulator package [167] is to provide a suitable environment to test algorithms and ensure their functionalities in a surface marine environment, and it is composed by a collection of plugins that support the simulation of surface vessels in the marine environment in *Gazebo*. This package offers the resources to set up complex simulations and also to observe the simulation evolution through *Gazebo*'s Graphical User Interface. The latter is possible by exploiting the UUV Simulator package for the vertex shaders used in the wave field visuals [166]. In the ASV Wave Simulator package the water surface is modeled through a *wavefield* element. By modifying the parameters that characterize a wavefield element, it is possible to simulate different environmental conditions. The parameters used in the simulation presented in this section are listed in Table C.1.

Table C.1 Wave parameters

Name	**Value**
Number	*3*
Scale	*2.5*
Angle	*0.3*
Steepness	*1.0*
Amplitude	*0.0*
Period	*8.0*
Direction	$[1, 1]^T$

Details about the machineries underlying the ASV Wave Simulator package go beyond the goals of this appendix. On the other hand, the parameters of Table C.1 are indeed self explanatory. For instance, setting the value $Amplitude = 0.0$ will result in a flat ocean surface. Moreover, in order to set up the interactions between the bodies submerged in water and water itself, the package provides the *libHydrodynamics* plugin, which has been configured with the parameters reported in Table C.2. In accordance to the *libHydrodynamics* plugin implementation, the main forces involved into ocean-body interactions are the dumping force and pressure force. These two forces are modeled by using the constants reported in Table C.2. For further details about the internal implementation of the plugin and the effect that the parameters have on their respective forces, please refer to the main web-page of the ASV Wave Simulator package [167].

When simulating the interaction between a body and the fluid, the ASV wave simulator exploits the whole geometry of the body in order to compute both the damping and pressure forces. Thus, it is important to use the correct shape of the vehicle in order to avoid asymmetries and imperfections.

To overcome this issue, particular attention has been devoted to obtain an accurate Computer-Aided design (CAD) model of the vehicle for the Gazebo environment (see Figure C.2). The vehicle structure, in the simulation envi-

Table C.2 Damping and Pressure parameters

Damping	
cDampL1	$1.0E-6$
cDampL2	$1.0E-6$
cDampR1	$1.0E-6$
cDampR2	$1.0E-6$
Pressure	
cPDrag1	$1.0E+2$
cPDrag2	$1.0E+2$
fPDrag	0.4
cSDrag1	$1.0E+2$
cSDrag2	$1.0E+2$
fSDrag	0.4
vRDrag	1

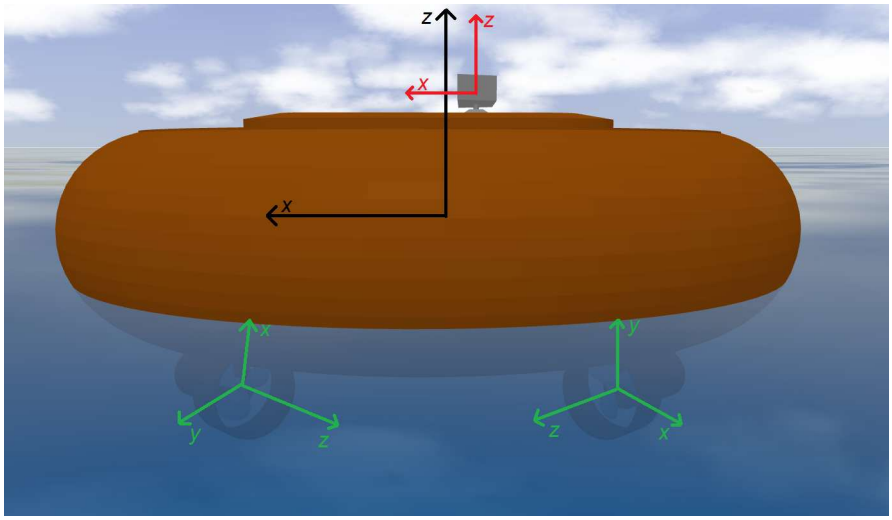


Fig. C.1 Simulated prototype of the vehicle in Gazebo.

ronment, can be logically divided into three different sections: the main body, the thrusters sections and the sensor section. The *main body* of the vehicle, representing most of its structure, is the main element exploited by the ASV Wave Sim package when computing the interaction forces. In accordance with the design principles and intended applications outlined in Section 6.2.1, the vehicle should be small, fully actuated, and operate under the assumptions specified in Section B.2.1. The cylindrical shape of the vehicle, depicted in Figure C.1, is a direct consequence of its holonomic nature. Four thrusters have been placed at the base of the vehicle, following the cross symmetrical configuration discussed in Section B.1.1.

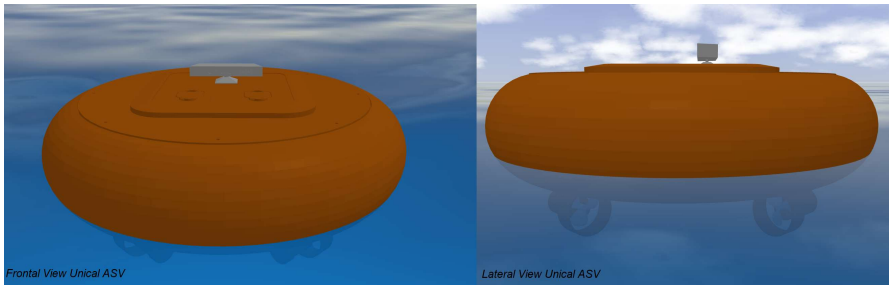


Fig. C.2 Frontal and lateral view of the ASV equipped with the *Vision Module*.

In the Gazebo platform, it is common to have a model in a "*.urdf*" or "*.urdf.xacro*" file types that are also used in *RVIZ*, a visualization tool heavily used for testing and debugging. The final "*.urdf*" model obtained included all the sensors and actuators, along with the joints and references with which the simulator has to interact with (Figure C.1). In particular, on top of the vehicle we find the *Stereolabs ZED 2i* stereo camera, essential component of the *Vision Module*, used to recognize obstacles and to compute the distance from them. Figure C.2 depicts the vehicle in the simulation environment, where robot poses are retrieved directly by the odometry topic maintained by Gazebo, while robot inputs are sent on suitable topics read by Gazebo plugins.

References

1. A. Knap, E. Dewailly, C. Furgal, J. Galvin, D. Baden, R.E. Bowen, M. Depledge, L. Duguay, L.E. Fleming, T. Ford, F. Moser, R. Owen, W.A. Suk, and U. Unluata, "Indicators of ocean health and human health: a research framework," *Environmental Health Perspectives*, vol. 110, no. 9, pp. 839-845, 2002.
2. J. Tibbetts, "Coastal cities: living on the edge," *Environmental Health Perspectives*, vol. 110, no. 11, pp. 674-681, 2002.
3. L.E. Fleming, K. Broad, A. Clement, E. Dewailly, S. Elmir, A. Knap, S.A. Pomponi, S. Smith, H. Solo Gabriele, and P. Walsh, "Oceans and human health: Emerging public health risks in the marine environment," *Marine pollution bulletin*, vol. 53, no. 10-12, pp. 545-560, 2006.
4. A.F. Molland, "Marine vehicle types," in *The Maritime Engineering Reference Book*. Oxford: Butterworth-Heinemann, ch. 2, pp. 43-74, 2008.
5. T.R. Walker, O. Adebambo, M.C. Del Aguila Feijoo, E. Elhaimer, T. Hossain, S.J. Edwards, C.E. Morrison, J. Romo, N. Sharma, S. Taylor, and S. Zomorodi, "Environmental effects of marine transportation," in *World Seas: An Environmental Evaluation*, 2nd ed. C. Sheppard, Ed. Academic Press, ch. 27, pp. 505-530, 2019.
6. J. Brown, C. Tuggle, J. MacMahan, and A. Reniers, "The use of autonomous vehicles for spatially measuring mean velocity profiles in rivers and estuaries," *Intelligent Service Robotics*, vol. 4, no. 4, pp. 233-244, 2011.
7. J.E. Manley, "Unmanned surface vehicles, 15 years of development," *In Proceedings of the OCEANS Conference*, 15-18 Sept., Quebec City, QC, Canada, pp. 1-4, 2008.
8. Z. Liu, Y. Zhang, X. Yu, and Chi Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71-93, 2016.
9. T.T. Furfaro, J. Dusek, and K. von Ellenrieder, "Design, Construction, and Initial Testing of an Autonomous Surface Vehicle for Riverine and Coastal Reconnaissance," *In Proceedings of the OCEANS Conference*, 26-29 Oct., Biloxi, MS, USA, pp. 1-6, 2009.
10. E. Desa, P. Maurya, A. Pereira, A.M. Pascoal, R.G. Prabhudesai, A. Mascarenhas, R. Madhan, S.G.P. Matondkar, G. Navelkar, S. Prabhudesai, and S. Afzulpurkar, "A Small Autonomous Surface Vehicle for Ocean Color Remote Sensing," *IEEE Journal of Oceanic Engineering*, vol. 32, no. 2, pp. 353-364, 2007.
11. S. Bhattacharya, H. Heidarsson, G.S. Sukhatme, and V. Kumar, "Cooperative control of autonomous surface vehicles for oil skimming and cleanup," *In Pro-*

ceedings of the IEEE International Conference on Robotics and Automation (ICRA), 09-13 May, Shanghai, China, pp. 2374-2379, 2011.
12. D.P. Eickstedt, M. Benjamin, and J. Curcio, "Behavior Based Adaptive Control for Autonomous Oceanographic Sampling," *In Proceedings of the IEEE International Conference on Robotics and Automation*, 10-14 April, Roma, Italy, pp. 4245-4250, 2007.
13. S. Frolov, J. Bellingham, W. Anderson, and G. Hine, "Wave Glider; A platform for persistent monitoring of algal blooms," *In Proceedings of the OCEANS MTS/IEEE KONA Conference*, 19-22 Sept., Waikoloa, HI, USA, pp. 1-5, 2011.
14. J. Melo and A. Matos, "Guidance and control of an ASV in AUV tracking operations," *In Proceedings of the OCEANS Conference*, 15-18 Sept., Quebec City, QC, Canada, pp. 1-7, 2008.
15. V. Djapic and D. Nad, "Using collaborative Autonomous Vehicles in Mine Countermeasures," *In Proceedings of the OCEANS Conference*, 24-27 May, Sydney, NSW, Australia, pp. 1-7, 2010.
16. M.H.A. Majid and M.R. Arshad, "Design of an autonomous surface vehicle (ASV) for swarming application," *In Proceedings of the IEEE/OES Autonomous Underwater Vehicles (AUV) Conference*, 06-09 Nov., Tokyo, Japan, pp. 230-235, 2016.
17. D.P. Eickstedt, M.R. Benjamin, H. Schmidt, and J.J. Leonard, "Adaptive Control of heterogeneous marine sensor platforms in an autonomous sensor network," *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 09-15 Oct., Beijing, China, pp. 5514-5521, 2006.
18. M. Jacinto, R. Cunha, and A.M. Pascoal, "Chemical Spill Encircling Using a Quadrotor and Autonomous Surface Vehicles: A Distributed Cooperative Approach," *Sensors*, vol. 22, no. 6, art. no. 2178, 2022.
19. K. Shojaei, "Leader–follower formation control of underactuated autonomous marine surface vehicles with limited torque," *Ocean Engineering*, vol. 105, pp. 196-205, 2015.
20. J.M. Soares, A.P. Aguiar, A.M. Pascoal, and A. Martinoli, "Design and Implementation of a Range-Based Formation Controller for Marine Robots," *In Proceedings of the ROBOT2013 First Iberian Robotics Conference: Advances in Robotics*, 28-29 Nov., Madrid, Spain, vol. 1, pp. 55-67, 2014.
21. L. Zuo, W. Yan, R. Cui, and J. Gao, "A coverage algorithm for multiple autonomous surface vehicles in flowing environments," *International Journal of Control, Automation and Systems*, vol. 14, no. 2, pp. 540-548, 2016.
22. N. Miskovic, D. Nad, and I. Rendulic, "Tracking Divers: An Autonomous Marine Surface Vehicle to Increase Diver Safety," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 72-84, 2015.
23. F. Velez, A. Nadziejko, A. Christensen, S. Oliveira, T. Rodrigues, V. Costa, M. Duarte, F. Silva, and J. Gomes, "Wireless Sensor and Networking Technologies for Swarms of Aquatic Surface Drones," *In Proceedings of the IEEE 82nd Vehicular Technology Conference (VTC Fall)*, 06-09 Sept., Boston, MA, USA, pp. 1-2, 2015.
24. R.R. Murphy, E. Steimle, M. Hall, M. Lindemuth, D. Trejo, S. Hurlebaus, Z. Medina-Cetina, and D. Slocum, "Robot-assisted bridge inspection," *Journal of Intelligent & Robotic Systems*, vol. 64, pp. 77-95, 2011.
25. T. Adamek, C.A. Kitts, and I. Mas, "Gradient-based cluster space navigation for autonomous surface vessels," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 506-518, 2014.
26. P. Mahacek, C.A. Kitts, and I. Mas, "Dynamic guarding of marine assets through cluster control of automated surface vessel fleets," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 1, pp. 65-75, 2012.

27. E. Raboin, P. Svec, D.S. Nau, and S.K. Gupta, "Model-predictive asset guarding by team of autonomous surface vehicles in environment with civilian boats", *Autonomous Robots*, vol. 38, pp. 1-22, 2014.
28. T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?," *Advanced Robotics*, vol. 18, no. 10, pp. 1001-1024, 2004.
29. A. Ravankar, A.A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, pp. 3170, 2018.
30. S.M. LaValle, " *Planning Algorithms*," *Cambridge Univ. Press*, 2006.
31. C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," *Springer Tracts in Advanced Robotics*, pp. 649-666, 2016.
32. J. Bellingham, M. Tillerson, A. Richards, and J.P. How, "Multi-task allocation and path planning for cooperating UAVs," *Cooperative Control: Models, Applications and Algorithms*, pp. 23-41, 2003.
33. J. Minguez and L. Montano, "Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 45-59, 2004.
34. T. Perez-Lozano, "Spatial Planning: A Configuration Space Approach,"' *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108-120, 1983.
35. C.W. Warren, "Fast Path Planning Using Modified A* Method," *In Proceedings of the IEEE International Conference on Robotics and Automation*, 02-06 May, Atlanta, GA, USA, pp. 662-667, 1993.
36. J. Cui, Y. Zhang, S. Ma, Y. Yi, J. Xin, and D. Liu, "Path planning algorithms for power transmission line inspection using unmanned aerial vehicles," *In Proceedings of the 29th IEEE Chinese Control And Decision Conference (CCDC)*, 28-30 May, Chongqing, China, pp. 2304-2309, 2017.
37. S. Ahmed, B. Qiu, F. Ahmad, C. 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, no. 6, art. no. 1069, 2021.
38. W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1287-1306, 2019.
39. J. Tordesillas, B.T. Lopez, M. Everett, and J.P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 922-938, 2022.
40. Y. Kantaros and M.M. Zavlanos, "Global planning for multi-robot communication networks in complex environments," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1045-1061, 2016.
41. G. Meng, J. Tumova, and D.V. Dimarogonas, "Hybrid control of multi-agent systems under local temporal tasks and relative-distance constraints," *In Proceedings of the 54th IEEE Conference on Decision and Control (CDC 2015)*, 15-18 Dec., Osaka, Japan, pp. 1701-1706, 2015.
42. J. Stephan, J. Fink, V. Kumar, and A. Ribeiro, "Concurrent control of mobility and communication in multirobot systems," *IEEE Transactions on Robotics*, vol 33, no. 5, pp. 1248-1254, 2017.
43. X. Li, D. Sun, and J. Yang, "A bounded controller for multirobot navigation while maintaining network connectivity in the presence of obstacles," *Automatica*, vol 49, pp. 258-292, 2013.
44. D. Panagou and V. Kumar, "Cooperative visibility maintenance for leader & follower formations in obstacle environments," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 831-844, 2014.
45. C. Sun, G. Hu, L. Xie, and M. Egerstedt, "Robust finite-time connectivity preserving coordination of second-order multi-agent systems," *Automatica*, vol. 89, pp. 21-27, 2018.

46. C. Dong, Q. Ye, and S. Dai, "Neural-network-based adaptive output-feedback formation tracking control of USVs under collision avoidance and connectivity maintenance constraints," *Neurocomputing*, vol. 401, pp. 101-112, 2020.
47. B. Seok Park, and S. Jin Yoo, "Connectivity-maintaining obstacle avoidance approach for leader-follower formation tracking of uncertain multiple nonholonomic mobile robots," *Expert Systems with Applications*, vol. 171, art. no. 114589, 2021.
48. M. Mesbahi, "On State-dependent dynamic graphs and their controllability properties," *IEEE Transactions on Automatic Control*, vol. 50, no. 3, pp. 387-392, 2005.
49. M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Distributed algorithms for environment partitioning in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1834-1848, 2011.
50. R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401-420, 2006.
51. J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243-255, 2004.
52. A. Bemporad, A. Casavola, and E. Mosca, "Nonlinear control of constrained linear systems via predictive reference management," *IEEE Transactions on Automatic Control*, vol. 42, no. 3, pp. 340-349, 1997.
53. E. Garone, S. Di Cairano, and I. Kolmanovsky, "Reference and command governors for systems with constraints: A survey on theory and applications," *Automatica*, vol. 75, pp. 306-328, 2017.
54. I. Kolmanovsky, U. Kalabic, and E. Gilbert, "Developments in constrained control using reference governors," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 282-290, 2012.
55. Z. Li and J. Sun, "Disturbance compensating model predictive control with application to ship heading control," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 257-265, 2012.
56. S.R. Oh and J. Sun, "Path following of underactuated marine surface vessels using line-of-sight based model predictive control," *Ocean Engineering*, vol. 37, no. 2-3, pp. 289-295, 2010.
57. E.G. Gilbert and I. Kolmanovsky, "Fast reference governors for systems with state and control constraints and disturbance inputs," *International Journal on Robust and Nonlinear Control*, vol. 9, pp. 1117-1141, 1999.
58. A. Casavola, E. Mosca, and D.Angeli, "Robust command governors for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 45, pp. 2071-2077, 2008.
59. A. Bemporad and E. Mosca, "Fulfilling Hard Constraints in Uncertain Linear Systems by Reference Managing," *Automatica*, vol. 34, no. 4, pp. 451-461, 1998.
60. D. Angeli, A. Casavola, and E. Mosca, "On feasible set-membership state estimators in constrained command governor control," *Automatica*, vol. 37, no. 1, pp. 151-156, 2001.
61. A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 415-419, 1998.
62. D. Angeli, A. Casavola, and E. Mosca, "Command Governors for Constrained Nonlinear Systems: Direct Nonlinear vs. Linearization-based Strategies," *International Journal of Robust and Nonlinear Control*, vol. 9, pp. 677-699, 1999.
63. D. Angeli and E. Mosca, "Command governor for nonlinear systems under constraints," *IEEE Transactions on Automatic Control*, vol. 44, no. 4, pp. 816-820, 1999.
64. A. Casavola, E. Garone, and F. Tedesco, "A distributed command governor based on graph colorability theory," *International Journal of Robust and Nonlinear Control*, vol. 28, no. 8, pp. 3056-3072, 2018.

65. F. Tedesco, A. Casavola, and E. Garone, "Distributed command governor strategies for constrained coordination of multi-agent networked systems," *In Proceedings of the IEEE American Control Conference (ACC)*, 27-29 June, Montreal, QC, Canada, pp. 6005-6010, 2012.
66. F. Tedesco and A. Casavola, "Distributed iterative command governor schemes for interconnected linear systems," *International Journal of Robust and Nonlinear Control*, vol. 27, no. 18, pp. 4788-4807, 2014.
67. F. Tedesco, "Distributed Command Governor Strategies for Multi-agent Dynamical Systems," *Ph.D. Thesis*, Department of Ingegneria Informatica Modellistica Elettronica e Sistemistica, University of Calabria, 2011.
68. A. Casavola, E. Garone, and F. Tedesco, "A distributed multi-agent command governor strategy for the coordination of networked interconnected systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 8, pp. 2099-2112, 2014.
69. F. Tedesco, A. Casavola and R. Russo, "Plug-and-Play Distributed Supervision Schemes for Decoupled Interconnected Dynamical Systems," *In Proceedings of the 58th IEEE Conference on Decision and Control (CDC)*, 11-13 Dec., Nice, France, pp. 3527-3532, 2019.
70. F. Tedesco and A. Casavola, "Turn-based command governor strategies for interconnected dynamical systems with time-varying couplings," *In Proceedings of the IEEE American Control Conference (ACC)*, 01-03 July, Denver, CO, USA, pp. 4564-4569, 2020.
71. F. Tedesco and A. Casavola, "Distributed Supervision Strategies for Cyber-Physical Systems With Varying Network Topology," *IEEE Transactions on Automatic Control*, vol. 69, no. 8, pp. 5408-5423, 2024.
72. F. Tedesco, D.M. Raimondo, and A. Casavola, "Collision avoidance command governor for multi-vehicle unmanned systems," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 16, pp. 2309-2330, 2014.
73. B.T. Stewart, A.N. Venkat, J.B. Rawlings, S.J. Wright, and G. Pannocchia, "Cooperative distributed model predictive control," *Systems & Control Letters*, vol. 59, no. 8, pp. 460-469, 2010.
74. P.C.N. Verheijen, V. Breschi, and M. Lazar, "Handbook of linear data-driven predictive control: Theory, implementation and design," *Annual Reviews in Control*, vol. 56, art. no. 100914, 2023.
75. J.C. Willems, "From time series to linear system: Part I. Finite dimensional linear time invariant systems," *Automatica*, vol. 22, no. 5, pp. 561-580, 1986.
76. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *In Proceedings of the IEEE ICRA Workshop on Open Source Software in Robotics*, 12-17 May, Kobe, Japan, vol. 3, p. 5, 2009.
77. A. Koubâa, *Robot Operating System (ROS): The complete reference. Vol. 1*, 1st ed. New York: Springer, 2017.
78. I. Peake, J. La Delfa, R. Bejarano, and J. O. Blech, "Simulation Components in Gazebo," *In Proceedings of the 22nd IEEE International Conference on Industrial Technology (ICIT)*, 10-12 March, Valencia, Spain, pp. 1169-1175, 2021.
79. E.G. Gilbert, I. Kolmanovsky, and K. Tin Tan, "Discrete-time Reference Governors and the Nonlinear Control of Systems with State and Control Constraints," *International Journal of Robust and Nonlinear Control*, vol. 5, pp. 487-504, 1995.
80. A. Bemporad, "Reference Governors: On-line Set-Point Optimization Techniques for Constraint Fulfillment," *Ph.D. Thesis*, Department of Sistemi e Informatica, University of Firenze, 1997.
81. A. Casavola, M. Papini, and G. Franzè, "Supervision of networked dynamical systems under coordination constraints," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 421-437, 2006.

82. I. Kolmanovsky and E.G. Gilbert, "Maximal output admissible sets for discrete-time systems with disturbance inputs," *In Proceedings of the IEEE American Control Conference (ACC)*, 21-23 June, Seattle, WA, USA, pp. 1995-1999, 1995.
83. A. Casavola, F. Tedesco, E. Garone, "A parallel distributed supervision strategy for multi-agent networked systems," *Systems & Control Letters*, vol. 97, pp. 115-124, 2016.
84. I. Markovsky, "Closed-loop data-driven simulation," *International Journal of Control*, vol. 83, no. 10, pp. 1946–1959, 2010.
85. J. Berberich and F. Allgöwer, "A trajectory-based framework for data-driven system analysis and control," *In Proceedings of the European Control Conference (ECC)*, 12-15 May, St. Petersburg, Russia, pp. 1365-1370, 2020.
86. J.C. Willems, P. Rapisarda, I. Markovsky, and B.L. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.
87. I. Markovsky, J. C. Willems, S. Van Huffel, and B. De Moor, "Exact and Approximate Modeling of Linear Systems: A Behavioral Approach," Number 11 in Monographs on Mathematical Modeling and Computation, *SIAM*, 2006.
88. E.H. Thyri, E.A. Basso, M. Breivik, K.Y. Pettersen, R. Skjetne and A.M. Lekkas, "Reactive collision avoidance for ASVs based on control barrier functions," *In Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*, 24-26 Aug., Montreal, QC, Canada, pp. 380-387, 2020.
89. T. Soleymani, E. Garone, and M. Dorigo, "Distributed constrained connectivity control for proximity networks based on a receding horizon scheme," *In Proceedings of the IEEE American Control Conference (ACC)*, 01-03 July, Chicago, IL, USA pp. 1369-1374, 2015.
90. K. Border, "Separating hyperplane theorems," Caltech, Pasadena, PA, USA, Tech. Rep. 2016.12.08::18.50, 2012.
91. S. Brossette and P.B. Wieber. "Collision avoidance based on separating planes for feet trajectory generation," *In Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 15-17 Nov., Birmingham, UK, pp. 509-514, 2017.
92. B. Sabetghadam, R. Cunha, and A. Pascoal, "A Distributed Algorithm for Real-Time Multi-Drone Collision-Free Trajectory Replanning," *Sensors*, vol. 22, no. 5, art. no. 1855, 2022.
93. T.R. Jensen and B. Toft, "*Graph coloring problems.*" Hoboken: John Wiley and Sons, 2010.
94. T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees", *In Proceedings of the IEEE American Control Conference (ACC)*, 30 June-02 July, Boston, MA, USA, vol 6, pp. 5576-5581, 2004.
95. A. Richards and P.G. How, "Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming," *In Proceedings of the IEEE American Control Conference (ACC)*, 08-10 May, Anchorage, AK, USA, vol. 3, pp. 1936-1941, 2002.
96. R.G. Gallager, P.A. Humblet, and P.M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Transactions on Programming Languages and systems (TOPLAS*), vol. 5, no. 1, pp. 66-77, 1983.
97. Z. Lin, M. Broucke, and B. Francis, "Local Control Strategies for Groups of Mobile Autonomous Agents," *IEEE Transactions on Automatic Control*, vol. 49, pp. 622-629, 2004.
98. M. Jalalmaab, M. Pirani, B. Fidan, and S. Jeon, "Cooperative Least Square Parameter Identification by Consensus within the Network of Autonomous Vehicles," *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, vol. 9, pp. 255-261, 2016.
99. Z. Song, S. Hutangkabodee, Y.H. Zweiri, L.D. Seneviratne, and K. Althoefer, "Identification of soil parameters for unmanned ground vehicles track-terrain

interaction dynamics," *In Proceedings of the SICE 2004 Annual Conference*, 04-06 Aug., Sapporo, Japan, vol. 3, pp. 2255-2260, 2004.
100. V. D'Angelo, P. Folino, M. Lupia, G. Gagliardi, G. Cario, F.G. Cicchello and A. Casavola, "A ROS-Based GNC Architecture for Autonomous Surface Vehicle Based on a New Multimission Management Paradigm," *Drones*, vol. 6, no. 12, art. no. 382, 2022.
101. F. Tedesco, A. Casavola and E. Garone, "A parallel distributed coordination-by-constraint strategy for multi-agent networked systems," *In Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, 10-13 Dec., Maui, HI, USA, pp. 284-289, 2012.
102. F. Tedesco, A. Casavola and E. Garone, "A distributed parallel command governor strategy for the coordination of multi-agent networked systems," *In Proceedings of the 4th IFAC Conference on Nonlinear Model Predictive Control*, 23-27 Aug., Noordwijkerhout, Netherlands, pp. 478-483, 2012.
103. A. Nedić and A. Ozdaglar, "Cooperative distributed multi-agent optimization," in *Convex Optimization in Signal Processing and Communications*, D.P. Palomar and Y.C. Eldar, Eds. Cambridge: Cambridge University Press, ch. 10, pp. 340-386, 2009.
104. M.F. Ginting, K. Otsu, J.A. Edlund, J. Gao, and A. Agha-Mohammadi, "CHORD: Distributed Data-Sharing via Hybrid ROS 1 and 2 for Multi-Robot Exploration of Large-Scale Complex Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5064-5071, 2021.
105. J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," *In Proceedings of the IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, 02-04 Sept., Taipei, Taiwan, pp. 284-289, 2004.
106. S. Diamond and S. Boyd, "CVXPY: A Python-Embedded Modeling Language for Convex Optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1-5, 2016.
107. Gurobi Optimization, LLC, "*Gurobi Optimizer Reference Manual*", https://www.gurobi.com, 2023.
108. A. Casavola, V. D'Angelo, A. El Qemmah, G. Gagliardi, F. Tedesco, and F. Angelo Torchiaro, "A Matlab-Based Toolbox for Supervising Multi-Vehicle Autonomous Systems," *IEEE Access*, vol. 12, pp. 127051-127064, 2024.
109. A. Casavola, V. D'Angelo, A. El Qemmah, F. Tedesco, and F.A. Torchiaro, "CoGoV: A Safe Motion Planning Distributed Supervision Framework for Multi-Vehicle Formations," *Proceedings of the 22nd IFAC World Congress*, Yokohama, Japan, pp. 8827-8832, 2023.
110. F.P. Kelly, A.K. Maulloo, and D.K. Tan, "Rate control for communication networks: shadow prices, proportional fairness, and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237-252, 1998.
111. S. Low and D.E. Lapsley, "Optimization flow control, I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861-874, 1999.
112. S. Shakkottai and R. Srikant, "Network optimization and control," *Foundations and Trends in Networking*, vol. 2, no. 3, pp. 271-379, 2008.
113. M. Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle, "Layering as optimization decomposition: a mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255-312, 2007.
114. H. Lakshmanan and D.P. De Farias, "Decentralized resource allocation in dynamic networks of agents," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 911-940, 2008.
115. A. Simonetto, T. Keviczky, and M. Johansson, "A regularized saddle-point algorithm for networked optimization with resource allocation constraints," *In Proceedings of the 51st IEEE Conference on Decision and Control (CDC)*, 10-13 Dec., Maui, HI, USA, pp. 7476-7481, 2012.

116. I. Necoara, "Random coordinate descent algorithms for multi-agent convex optimization over networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 2001-2012, 2013.
117. A. Cherukuri and J. Cortes, "Distributed generator coordination for initialization and anytime optimization in economic dispatch," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 226-237, 2015.
118. T. Chang, A. Nedić, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1524-1538, 2014.
119. A. Simonetto and H. Jamali-Rad, "Primal recovery from consensus-based dual decomposition for distributed convex optimization," *Journal of Optimization Theory and Applications*, vol. 168, no. 1, pp. 172-197, 2016.
120. A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, vol. 84, pp. 149-158, 2017.
121. D. Mateos-Núñez and J.Cortés, "Distributed saddle-point subgradient algorithms with laplacian averaging," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2720-2735, 2017.
122. M. Bürger, G. Notarstefano and F. Allgöwer, "A polyhedral approximation framework for convex and robust distributed optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 384-395, 2014.
123. G. Inalhan, D.M. Stipanovic, and C.J. Tomlin, "Decentralized optimization, with application to multiple aircraft coordination," *In Proceedings of the 41st IEEE Conference on Decision and Control (CDC)*, 10-13 Dec., Las Vegas, NV, USA, pp. 1147-1155, 2002.
124. I. Notarnicola and G. Notarstefano, "Constraint-Coupled Distributed Optimization: A Relaxation and Duality Approach," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 483-492, 2020.
125. M. Slater, "Lagrange Multipliers Revisited," *Commission Discussion Paper, Mathematics*, art. no. 403, University of Chicago, Chicago, 1950.
126. D.P. Bertsekas, "Nonlinear Programming." Belmont, MA, USA: Athena Sci., 2018.
127. A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48-61, 2009.
128. F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R.M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof, "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annual Reviews in Control*, vol. 43, pp. 1-64, 2017.
129. Z.S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3-35, 2013.
130. K.J. Åström and T. Hägglund, "PID Controllers: Theory, Design, and Tuning," 2nd ed. Research Triangle Park, USA: Instrument society of America, 1995.
131. M.A. Taylor and L.F. Giraldo, "Data-Driven Design of a Reference Governor Using Deep Reinforcement Learning," *In Proceedings of the 5th IEEE Conference on Control Technology and Applications (CCTA)*, 09-11 Aug., San Diego, CA, USA, pp. 956-961, 2021.
132. K. Liu, N. Li, D. Rizzo, E. Garone, I. Kolmanovsky, and A. Girard, "Model-free Learning to Avoid Constraint Violations: An Explicit Reference Governor Approach," *In Proceedings of the IEEE American Control Conference (ACC)*, 10-12 July, Philadelphia, PA, USA, pp. 934-940, 2019.
133. I. Markovsky and P. Rapisarda, "Data-driven simulation and control," *International Journal of Control*, vol. 81, no. 12, pp. 1946-1959, 2008.

134. I. Markovsky, J.C. Willems, P. Rapisarda, and B.L.M. De Moor, "Data driven simulation with applications to system identification," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 970-975, 2005.
135. J. Berberich, J. Köhler, M. Müller, and F. Allgöwer, "Data-Driven Model Predictive Control With Stability and Robustness Guarantees," *IEEE Transactions on Automatic Control*, vol. 66, pp. 1702-1717, 2019.
136. J. Coulson, J. Lygeros, and F. Dorfler, "Data-enabled predictive control: In the shallows of the deepc," *In Proceedings of the 18th IEEE European Control Conference (ECC)*, 25-28 June, Naples, Italy, pp. 307-312, 2019.
137. V.L. Syrmos, C. Abdallah, and P. Dorato, "Static output feedback: a survey," *In Proceedings of the 33rd IEEE Conference on Decision and Control (CDC)*, 14-16 Dec., Lake Buena Vista, FL, USA, vol. 1, pp. 837-842, 1994.
138. H.R. Ossareh, "A Data-Driven Formulation of the Maximal Admissible Set and the Data-Enabled Reference Governor," *IEEE Control Systems Letters*, vol. 7, pp. 3465-3470, 2023.
139. E.G. Gilbert and K.T. Tan, "Linear systems with state and control constraints: the theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control*, vol. 36, no. 9, pp. 1008-1020, 1991.
140. M.G. Safonov, "Data-driven robust control design: Unfalsified control," *In Proceedings of the NATO Lecture Series 236, Robust Integrated Control Systems Design Methods for 21st Century Military Applications*, October, Nueilly-sur-Seine Cedex, France, Educational Notes RTO-EN-SCI-166, pages 4.1-4.18, 2004.
141. H.J. van Waarde, C. De Persis, M.K. Camlibel, and P. Tesi, "Willems' Fundamental Lemma for State-Space Systems and Its Extension to Multiple Datasets," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 602-607, 2020.
142. D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-Oriented Model Learning for Data-Driven MPC Design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577-582, 2019.
143. J. Berberich, J. Köhler, M.A. Müller and F. Allgöwer, "Robust Constraint Satisfaction in Data-Driven MPC," *In Proceedings of the 59th IEEE Conference on Decision and Control (CDC)*, 14-18 Dec., Jeju, Korea (South), pp. 1260-1267, 2020.
144. J. Berberich, J. Köhler, M. A. Müller and F. Allgöwer, "Data-Driven Model Predictive Control With Stability and Robustness Guarantees," *IEEE Transactions on Automatic Control*, vol. 66, no. 4, pp. 1702-1717, 2021.
145. K. Groves, A. West, K. Gornicki, S. Watson, J. Carrasco, and B. Lennox, "MallARD: An Autonomous Aquatic Surface Vehicle for Inspection and Monitoring of Wet Nuclear Storage Facilities," *Robotics*, vol. 8, no. 2, art. no. 47, 2019.
146. A. Vasilijevic, D. Nad, and N. Miskovic, "Autonomous Surface Vehicles as Positioning and Communications Satellites for the Marine Operational Environment — Step toward Internet of Underwater Things," *In Proceedings of the 8th IEEE International Conference on Underwater System Technology: Theory and Applications (USYS)*, 01-03 Dec., Wuhan, China, pp. 1-5, 2018.
147. R. Skjetne, O.N. Smogeli, and T.I. Fossen, "A nonlinear ship manoeuvering model: Identification and adaptive control with experiments for a model ship," *Modeling, Identification and Control*, vol. 25, no. 1, pp. 3-27, 2004.
148. R.P. Selvam, S.K. Bhattacharyya, and M. Haddara, "A frequency domain system identification method for linear ship maneuvering," *International Shipbuilding Progress*, vol. 52, no. 1, pp. 5-27, 2005.
149. N. Miskovic, Z. Vukic, M. Bibuli, G. Bruzzone, and M. Caccia, "Fast in-field identification of unmanned marine vehicles," *Journal of Field Robotics*, vol. 28, no. 1, pp. 101-120, 2011.
150. C. Sonnenburg, A. Gadre, D. Horner, S. Kragelund, A. Marcus, D.J. Stilwell, and C.A. Woolsey, "Control-Oriented Planar Motion Modeling of Unmanned Surface

Vehicles," *In Proceedings of the OCEANS 2010 MTS/IEEE SEATTLE*, 20-23 Sept., Seattle, WA, USA, pp. 1-10, 2010.
151. M. Caccia, G. Bruzzone, and R. Bono, "A practical approach to modeling and identification of small autonomous surface craft," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 2, pp. 133-145, 2008.
152. K.R. Muske, H. Ashrafiuon, G. Haas, R. McCloskey, and T. Flynn, "Identification of a control oriented nonlinear dynamic USV model," *In Proceedings of the American control conference (ACC)*, 11-13 June, Seattle, WA, USA, pp. 562-567, 2008.
153. H.K. Yoon and K.P. Rhee, "Identification of hydrodynamic coefficients in ship maneuvering equations of motion by estimation-before-modeling technique," *Ocean Engineering*, vol. 30, no. 18, pp. 2379-2404, 2003.
154. G. Rajesh and S.K. Bhattacharyya, "System identification for nonlinear maneuvering of large tankers using artificial neural network," *Applied Ocean Research*, vol. 30, no. 4, pp. 256-263, 2008.
155. B.N. Biswas, S. Chatterjee, S. Mukherjee, and S. Pal, "A discussion on Euler method: A review," *Electronic Journal of Mathematical Analysis and Applications*, vol. 1, no. 2, pp. 294-317, 2013.
156. A. Casavola, "The command governor approach for dummies," *DEIS*, Tech. Rep. no. 07-2005, University of Calabria, Rende, Italy, 2005.
157. D. Mu, G. Wang, Y. Fan, X. Sun, and B. Qiu, "Modeling and Identification for Vector Propulsion of an Unmanned Surface Vehicle: Three Degrees of Freedom Model and Response model," *Sensors*, vol. 18, no. 6, art. no. 1889, 2018.
158. T. Foote and M. Purvis, "Standard Units of Measure and Coordinate Conventions," *Open Robotics*, Tech. Rep. no. 103, San Jose, California, USA, 2014.
159. E. Omerdic and G. Roberts, "Thruster fault diagnosis and accommodation for open-frame underwater vehicles," *Control Engineering Practice*, vol. 12, pp. 1575-1598, 2004.
160. T. Fossen, *Guidance and Control of Ocean Vehicles*. London, U.K.: Wiley, 1994.
161. R. Skjetne, T.I. Fossen, and P.V. Kokotovic, "Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory," *Automatica*, vol. 41, pp. 289-298, 2005.
162. B.A.H. Vicente, S.S. James, and S.R. Anderson, "Linear System Identification Versus Physical Modeling of Lateral–Longitudinal Vehicle Dynamics," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1380-1387, 2021.
163. R. Dhaouadi and A.A. Hatab, "Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework," *Advances in Robotics & Automation*, vol. 2, pp. 1-7, 2013.
164. C. Li, J. Jiang, F. Duan, W. Liu, X. Wang, L. Bu, Z. Sun, and G. Yang, "Modeling and Experimental Testing of an Unmanned Surface Vehicle with Rudderless Double Thrusters," *Sensors*, vol. 19, no. 9, art. no. 2051, 2019.
165. Z.X. Liu, C. Yuan, and Y.M. Zhang, "Linear Parameter Varying Adaptive Control of an Unmanned Surface Vehicle," *IFAC-PapersOnLine*, vol. 48, no. 16, pp. 140-145, 2015.
166. UUV Simulator, "Unmanned Underwater Vehicle (UUV) simulation with Gazebo," *GitHub, GitHub Repository*, `https://github.com/uuvsimulator/uuv_simulator`, commit c855fcd52c31c104bb19fe48137e96f2707d0ef4.
167. Srmainwaring, "ASV Wave Simulator," *GitHub, GitHub Repository*, `https://github.com/srmainwaring/asv_wave_sim`, commit 524a6bdab819ed422c85d19ccff0ceef2fddcdf5.
168. S. Grechi and A. Caiti, "Comparison between Optimal Control Allocation with Mixed Quadratic & Linear Programming Techniques," *In Proceedings of the 10th IFAC Conference on Control Applications in Marine SystemsCAMS 2016*, 13-16 Sept., Trondheim, Norway, pp. 147-152, 2016.

169. A. Casavola, A. El Qemmah, F. Tedesco, and F.A. Torchiaro, "Coordination of Marine Multi-Vehicle Autonomous Systems via the Distributed Command Governor Approach," *In Proceedings of the OCEANS Conference*, 05-08 June, Limerick, Ireland, pp. 1-8, 2023.
170. A. Casavola, V. D'Angelo, A. El Qemmah, F. Tedesco, and F. A. Torchiaro, "Distributed Constrained Connectivity-Keeping Supervision Scheme in the Presence of Static Obstacles," *In Proceedings of the 61st IEEE Conference on Decision and Control (CDC)*, 06-09 Dec., Cancun, Mexico, pp. 3933-3938, 2022.
171. A. Casavola, V. D'Angelo, A. El Qemmah, F. Tedesco, and F. A. Torchiaro, "Distributed Connectivity Keeping Supervision Scheme with Collision and Obstacle Avoidance Requirements," *In Proceedings of the IEEE American Control Conference (ACC)*, 31 May-02 June, San Diego, CA, USA, pp. 1171-1176, 2023.
172. A. El Qemmah, G. Cario, A. Casavola, M. Lupia, and F. Tedesco, "Linear System Identification and Control of a Low-Cost High-Performance Omnidirectional Marine Surface Vehicle for Swarming Applications," *Journal of Field Robots*, under review.
173. A. Casavola, V. D'Angelo, A. El Qemmah, G. Gagliardi, F. Tedesco, and F. Angelo Torchiaro, "Preserving Agents Connectivity Amidst Static Obstacles: A Distributed Predictive Supervision Approach within Robot Operating System," *IEEE Transactions on Intelligent Vehicles*, to appear.
174. A. El Qemmah, A. Casavola, F. Tedesco, and B. Sinopoli, "Data-Driven Command Governors for Discrete-Time LTI Systems with Linear Constraints," *In Proceedings of the 63rd IEEE Conference on Decision and Control (CDC)*, to appear.
175. A. El Qemmah, A. Casavola, and F. Tedesco "Privacy-Preserving Cooperative Distributed Command Governor Schemes," *2025 IEEE American Control Conference (ACC)*, to appear.