Universidade Federal de Minas Gerais
Graduate Program in Electrical Engineering
Research group MACRO - Mechatronics, Control and Robotics

DISTRIBUTED APPROACHES TO MULTI-ROBOT CURVE TRACKING WITH COLLISION AVOIDANCE

Gabriel Viana Pacheco

Belo Horizonte, Brazil

2020

Gabriel Viana Pacheco

DISTRIBUTED APPROACHES TO MULTI-ROBOT CURVE TRACKING WITH COLLISION AVOIDANCE

Thesis submitted to the Graduate Program in Electrical Engineering of Escola de Engenharia at the Universidade Federal de Minas Gerais, in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Advisor: Luciano Cunha de Araújo Pimenta
Co-Advisor: Guilherme Vianna Raffo

Belo Horizonte, Brazil
2020

P116d

Pacheco, Gabriel Viana.
Distributed approaches to multi-robot curve tracking with collision avoidance [recurso eletrônico] / Gabriel Viana Pacheco. - 2020.
1 recurso online (xxv, 104 f. : il., color.) : pdf.

Orientador: Luciano Cunha de Araújo Pimenta.
Coorientador: Guilherme Vianna Raffo.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 97-104.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Campos vetoriais - Teses. 3. Controle preditivo - Teses. I. Pimenta, Luciano Cunha de Araújo. II. Raffo, Guilherme Vianna. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo Cesar Vital dos Santos CRB/6 2165
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

"Distributed Approaches To Multi-robot Curve Tracking With Collision Avoidance"

Gabriel Viana Pacheco

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 27 de julho de 2020.

Por:

[illegible]

Prof. Dr. Luciano Cunha de Araújo Pimenta
DELT (UFMG) - Orientador

[illegible]

Prof. Dr. Guilherme Vianna Raffo
DELT (UFMG) - Coorientador

[illegible]

Prof. Dr. Antonio Ferramosca
CONICET - Argentina

Alexandre Santos Brandão

Prof. Dr. Alexandre Santos Brandão
DEL (UFV)

Vinicius Mariano

Prof. Dr. Vinicius Mariano Gonçalves
DEE (UFMG)

To Alice.

Acknowledgements

I thank my beloved parents, Eudes and Raquel, and my brother Kaio, who have always supported my study journey. I also thanks Alice, for her patience and dedicated partnership.

To my advisors Luciano and Raffo, who guided me during my masters, and to all my colleagues from MACRO, especially Marcos and Thales.

Resumo

Este trabalho trata o problema de convergência e circulação de curvas fechadas por sistemas multi-robôs a partir de estratégias baseadas em campos vetoriais. O problema é tratado sob dois olhares. Na primeira parte, considera-se o problema no qual uma curva definida no espaço tridimensional, variante no tempo, deve ser circunavegada por um conjunto de veículos aéreos não tripulados do tipo quadrotor. Para isso, é proposto um sistema em cascata, em que uma etapa de controle distribuído, em alto nível, garante a convergência e a circulação da curva através de uma lei baseada em campos vetoriais. O mesmo controlador garante o evitamento de colisão entre os robôs baseando-se em leis de prioridade que levam em conta as posições dos robôs vizinhos e que permitem modular o campo vetorial de forma a evitar colisões. Os sinais de controle de alto nível geram uma trajetória a ser seguida pelo veículo quadrotor, que é comandado por um controlador de baixo nível. O projeto do controlador de baixo nível é baseado na técnica de controle não-linear backstepping, que é incrementada com a ação integral e com uma lei de controle adicional, baseada na técnica Lyapunov redesign, em que a última torna o sistema robusto a distúrbios limitados. Na segunda parte deste trabalho, o problema de convergência e circulação de curvas é endereçado sob outro olhar. Considerando curvas definidas no espaço 2-D, têm-se por objetivo projetar uma estratégia de controle preditivo distribuído em que as leis de controle baseadas em campos vetoriais são embutidas no problema de otimização. Assim, em vez de encontrar uma sequência de controle, o problema encontra os parâmetros da lei de controle de cada robô. A partir disso, o problema de controle ótimo é projetado de forma a garantir a convergência e circulação da curva alvo e o evitamento de colisões entre robôs. Depois, o mesmo problema é distribuído pelo método das direções alternadas de multiplicadores, o que permite a negociação de trajetórias entre robôs vizinhos. A eficácia das estratégias de controle propostas neste trabalho são avaliadas com resultados de simulação.

Palavras-chave: Campos vetoriais. Controle backstepping. Controle preditivo distribuído. Evitamento de colisões. Sistemas multi-robôs.

Abstract

This work deals with the problem of convergence and circulation of closed curves by multi-robot systems, through vector field based strategies. The problem is addressed with two regards. In the first part, the problem of circumnavigating a time-varying curve in the three-dimensional space with a group of quadrotor aerial vehicles is approached. For that end, a cascaded system is proposed, in which the high-level, distributed, layer guarantees convergence and circulation of the aimed curve through a vector field based control law. The same control layer provides collision avoidance among robots by considering predefined priority laws that take into account the robots positions, which allow to modulate the vector field. The high-level control law generates a trajectory to be followed by a quadrotor vehicle, which is guided by the low-level controller. The design of the low-level controller is based on the backstepping nonlinear control, improved with an integral action and an additional control law based on the Lyapunov redesign approach, which provides robustness against bounded disturbances. In the second part of this work, the problem of convergence and circulation of curves is addressed under another perspective. Considering curves defined in the 2-D space, the objective is to design a distributed predictive control strategy in which the control laws based on vector fields are embedded in the optimization problem. Therefore, instead of finding a control sequence, the problem finds the parameters of each robot's control law. From this, the optimum control problem is designed to ensure the convergence and circulation of the target curve and avoid inter-robot collisions. Then, the same problem is distributed by the alternating directions method of multipliers, which allows the negotiation of trajectories between neighboring robots. The effectiveness of the control strategies proposed in this work are evaluated with simulation results.

Keywords: Vector fields. Backstepping Control. Distributed predictive control. Collision avoidance. Multi-robot systems.

List of Figures

1.1 Example of perimeter surveillance task. 19

1.2 Mavic Pro Platinum by Sven Teschke. 20

2.1 Two examples of vector fields for convergence and circulation. 24

2.2 Example of vector field convergence and circulation modulation. (a) Convergence prioritized over circulation. (b) Convergence and circulation field have equal priority. (c) Circulation prioritized over convergence. 25

2.3 Two surfaces intersection. 26

2.4 Quadrotor vehicle model. 27

2.5 Model Predictive Control Architectures. (a) Centralized MPC. (b) Hierarchical MPC. (c) Distributed MPC. (d) Decentralized MPC. 37

2.6 Three agents in line topology network. 40

2.7 ADMM example, $\rho = 1$. 41

2.8 ADMM example, $\rho = 5$. 41

3.1 Block diagram of the cascaded system. 45

3.2 Collision avoidance example. 47

3.3 Priority sets example. 49

3.4 Lyapunov redesign: hyperbolic tangent additional law. 61

4.1 Communication range example and local optimization problems. 67

4.2 Vector field navigation example. 69

4.3 Example of vector field modulation by the control law parameters. 72

5.1 Reference and actual trajectories. 78

5.2 Evolution of states. 78

5.3 Error evolution. 78

5.4 Control signals evolution. 79

5.5 Surfaces $\alpha_1 = 0$ and $\alpha_2 = 0$. 79

5.6 Trajectories and convergence functions W in the simulations. 82

5.7 Target curve. 83

5.8 Trajectories of the centralized problem with single-integrator dynamics. . 84
5.9 Single-integrator centralized problem comparison: exponential repulsion and set-based avoidance. 85
5.10 Trajectories of the centralized problem with double-integrator dynamics. . 86
5.11 Double-integrator centralized problem comparison: exponential repulsion and set-based avoidance. 86
5.12 Distributed problem – small curve. 88
5.13 12 robots simulation with a bigger curve – End position. 89
5.14 Distributed problem – big curve. 90
5.15 20 robots simulation – End position. 91
5.16 20 robots simulation — Convergence function evolution. 92
5.17 Star shaped curve – Trajectories. 93

List of Tables

5.1 Quadrotor parameters. 76
5.2 Saturation of the control signals. 77
5.3 External disturbances actuation. 77
5.4 External disturbances actuation. 80

List of Videos

https://youtu.be/kvshsF-cojY Cascaded control 64

https://youtu.be/6OZmkQErPhI Centralized MPC problem 68

https://youtu.be/nOdIeVToHGw DMPC with 20 robots 75

https://youtu.be/zGQfJV-OepM DMPC with localization error 75

https://youtu.be/e2ERo_HfDpA DMPC with star-shaped curve 76

Acronyms

2-D	Two-Dimensional
3-D	Three-Dimensional
ADMM	Alternating Direction Method of Multipliers
DMPC	Distributed Model Predictive Control
LQR	Linear Quadratic Regulator
MACRO	Mechatronics, Control, and Robotics
MILP	Mixed-Integer Linear Programming
NMPC	Nonlinear Model Predictive Control
MPC	Model Predictive Control
PID	Proportional Integral Derivative
UAV	Unmanned Aerial Vehicle
UFMG	Universidade Federal de Minas Gerais
UGV	Unmanned Ground Vehicle

Notation

ω_i	i-th robot.
r_i	Robot radius.
R_i	Perception range.
Ω	Set of robots.
N	Number of robots.
$\boldsymbol{\xi}$	Position vector.
$\boldsymbol{u}$	Control signal.
α	Function that defines surface.
Γ	Target curve.
$\nabla(\cdot)$	Gradient of $(\cdot)$.
μ_i, λ_i, ϱ_i, ι_i	Modulation functions.
$\mathcal{N}([\cdot])$	Null-space of $[\cdot]$.
$\boldsymbol{p}$	Feedforward term.
$\boldsymbol{\Theta}_i$	Set of perceived robots.
D_{ij}	Euclidean distance between robots ω_i and ω_j.
δ_{ij}	Collision distance.
$\tilde{r}$	Augmented radius.
Ξ^i_μ, Ξ^i_λ, Ξ^i_ϱ	Priority sets.
$W(\mathbf{x}(t))$	Convergence function.
$v_{\max}$	Maximum quadrotor linear velocity.

$\boldsymbol{\eta}$	Euler angles vector.
ϕ, θ, ψ	Roll, pitch, and yaw angles.
$\mathscr{I}$	Inertial frame.
$\mathscr{B}$	Body-fixed frame.
$\boldsymbol{R}_{\mathscr{I}}$	Rotation matrix from $\mathscr{B}$ to $\mathscr{I}$.
m	Mass of the quadrotor.
$\boldsymbol{v}_{\mathscr{I}}$	Velocity of the UAV center of mass.
$\boldsymbol{I}^{\mathscr{B}}$	Inertia tensor.
$\boldsymbol{\omega}$	Angular velocity vector.
p, q, r	Angular velocities.
T	Total thrust.
$\boldsymbol{\tau}$	Torques vector.
τ_ϕ, τ_θ, τ_ψ	Torques around axes.
$\boldsymbol{S}(\boldsymbol{\omega})$	Skew-symmetric matrix.
$\boldsymbol{b}$	Disturbance.
$\boldsymbol{\gamma}$	Additional control law.
$\boldsymbol{x}$	State variables.
$\boldsymbol{f}$	State vector field.
$\boldsymbol{\kappa}$	Predefined control law.
$\boldsymbol{\theta}$	Parameters vector.
L_g	Global cost.
L_i	Local cost.
$\bar{(\cdot)}$	Generalized $(\cdot)$ vector.
$(\cdot)_{ij}$	Variable $(\cdot)_j$ as computed by the i-th robot.
Δ	Prediction horizon.

ρ	ADMM penalty parameter.
$\boldsymbol{y}$	Lagrange multiplier vector.
$\boldsymbol{z}$	Global variable.
n_{ite}	Iteration number.
n_{max}	Maximum iteration number.
J	Jacobian matrix.
η	Cost weight.
w	Collision avoidance cost term.

Contents

1 Introduction **18**
1.1 Motivation . . . 18
1.2 Contributions . . . 21
1.2.1 Publications . . . 22
1.3 Outline . . . 22

2 Background **23**
2.1 Vector Field Navigation . . . 23
2.2 Quadrotor Model . . . 26
2.3 Backstepping Control Approach . . . 28
2.4 Lyapunov Redesign . . . 32
2.5 Model Predictive Control . . . 33
2.6 Distributed Model Predictive Control . . . 36
2.6.1 Alternating Direction Method of Multipliers . . . 38
2.7 Summary . . . 42

3 Cascaded Control Strategy **43**
3.1 Problem Statement . . . 43
3.2 High-Level Control Strategy . . . 45
3.3 Low-Level Control Strategy . . . 51
3.4 Summary . . . 62

4 DMPC Based on Vector Field Navigation **63**
4.1 Problem Statement . . . 63
4.2 The Optimal Control Problem . . . 65
4.3 ADMM-based parameterized DMPC . . . 66
4.4 Vector Field Navigation . . . 68
4.5 Problem Formulation and Collision Avoidance . . . 72
4.5.1 Scheme 1 . . . 72
4.5.2 Scheme 2 . . . 73

4.6 Summary 74

5 Results **75**

5.1 Cascaded Control 75

5.1.1 Robust Backstepping Controller with Integral Action 75

5.1.2 Multi-layer Control 77

5.2 Parameterized Distributed Model Predictive Control 81

6 Conclusion **94**

6.1 Summary 94

6.2 Future Works 95

Bibliography **97**

1 Introduction

1.1 Motivation

The control of multi-robot systems has been widely studied in the past decades. The main characteristic that brings attention to this class of systems is the increased ability to execute certain tasks in relation to a single robot. Consider for example a task of area coverage (Fazli et al., 2010), in which a set of coordinated robots can cover an area much faster than a single robot. Another interesting example is the transportation of a heavy load that cannot be held by a single robot. In this case, a set of coordinated robots can transport the cargo by distributing the weight among them (Tuci et al., 2018).

As pointed out by Yan et al. (2013), multi-robot systems can have several advantages over single-robot systems. The most obvious one is the ability to have a better spatial distribution. The limited spatial aspect of a single robot makes it impossible to accomplish tasks that require actuation in two different places, at the same time, for example. This leads to another advantage that can be attained under multi-robot systems, which is the better overall performance. In tasks in which one robot must actuate in different positions or in a large area, a multi-robot system will probably outperform a single robot, regarding, for instance, the total time to execute the task (Burgard et al., 2005). Also, in exploration tasks using multiple coordinated robots brings a faster area coverage. This is especially interesting in search and rescue operations, where one wants to find the target in minimum time (Jennings et al., 1997; Hu et al., 2013).

The multi-robot systems are a natural expansion of the single robot systems. Dealing

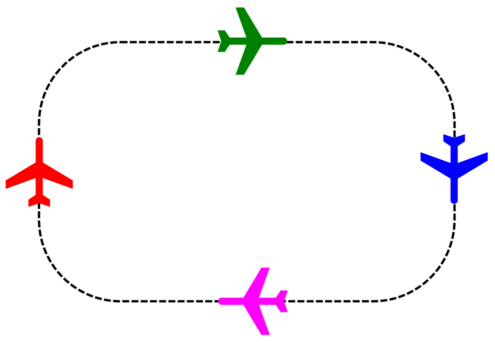


Figure 1.1: Example of perimeter surveillance task.

with several robots to accomplish a task, the overall system has better flexibility and scalability when compared to a single-robot (Cao et al., 1997; Parker, 1998), as in many situations it is possible to add or remove robots from the scheme without compromising the task accomplishment. From this, one might rightly infer that these systems have a fault-tolerance characteristic. The failure of a robot in a multi-robot system can be overcome by coordination, while in a single robot strategy the failure of the robot means the failure of the task. In addition, information sharing between robots, such as the sensed positions of each other, generally produces a positive impact in the localization efficiency (Fox et al., 2000). Also, the employ of a group of several simple robots instead of a single powerful robot to accomplish a certain task might be considerably cheaper.

Among the tasks that can be efficiently performed by multi-robot systems, certain tasks require convergence and circulation of a desired curve. An intuitive example is the perimeter surveillance task (Pimenta et al., 2013; Hsieh et al., 2008), illustrated in Figure 1.1, in which a group of robots must patrol over a boundary. Convergence and circulation approaches also arise when tracking a moving target or in source seeking (Frew et al., 2008; Briñón-Arranz et al., 2019), and in environmental monitoring problems (DeVries & Paley, 2012; Michini et al., 2014).

The problem of convergence of a group of robots to predefined curves is not new. In (Sugihara & Suzuki, 1996), the authors develop an algorithm where the robots move in the direction of the radius of a circle. Collisions among robots are avoided by considering predefined avoiding laws, in which the robot only moves in "clear" directions, that is, straight lines that do not intercept any robot. To converge to limit cycles, vector field-based control strategies have been applied (Jung et al., 2016; Frew & Lawrence, 2017; Gonçalves et al., 2010b). The main advantages of this type of strategy are the stability guarantees and the intrinsic robustness of vector field approaches. In this work, vector field-based approaches are proposed to achieve convergence and circulation of closed curves. Regarding multi-robot systems, the same vector fields are modulated to avoid collisions among robots.

For many multi-robot systems, the employment of aerial vehicles is opportune. The control of unmanned aerial vehicles (UAVs) has been extensively studied in the last

Figure 1.2: Mavic Pro Platinum by Sven Teschke [1].

years. The use of these unmanned vehicles has first emerged from the necessity of doing dangerous tasks in military applications, as reconnaissance of an enemy field, target and decoy for simulating an attack, and combat. Particularly in security missions, aerial vehicles outperform ground vehicles in various aspects, since they have better maneuverability and can move in the 3-D space. Besides that, the use of UAVs for civil and commercial applications, such as logistics, agriculture, data collection, photography, filming, and recreation has grown enormously in the last few years. The main reason for this growth is the technological development that allowed the production of small scale cheap UAVs. Despite having a smaller payload capacity, small size UAVs can execute several tasks previously performed with large UAVs, with a lower cost. Their versatility and portability are some of the advantages over standard UAVs. In addition, in some military missions, small UAVs are considered expendables, which means that recovery of the aircraft is not attempted in case of imminent risk (Valavanis & Vachtsevanos, 2015).

With regard to aerial vehicles, the control of quadrotor UAVs has been the focus of many studies. The quadrotor is a multi-rotor aircraft, that flies when propelled by the force produced by four rotors. The propellers in opposite sides in the same support shaft spin in the same direction, while one pair turns in the clockwise sense, the other pair turns in the counter-clockwise sense. This configuration of rotors allows to control the aircraft yaw angle. Figure 1.2 shows an example of a commercial quadrotor.

In comparison to a fixed-wing aircraft, the quadrotors have an interesting mobility characteristic. The fixed-wing aircraft velocity is restricted by a first-order nonholonomic Pfaffian constraint, that is, this aircraft cannot generate lateral velocities instantaneously. Despite presenting second-order (acceleration) non-holonomic constraints and being under-actuated, quadrotors have greater maneuvrability than fixed-wing aircraft, since they are not subject to Pffafian (velocity) constraints. Besides that, differently from the fixed-wing ones, the quadrotors can hover.

[1]License: CC BY-SA 3.0-de (`https://creativecommons.org/licenses/by-sa/3.0/`) via Wikimedia Commons (`https://commons.wikimedia.org/wiki/File:2019-03-23_-_Mavic_Pro_-3941.jpg`).

Therefore, quadrotors are opportune vehicles for the application of the multi-robot vector field strategies to be presented in this work, once that when modulating the vector field components, the original vector field flow is not maintained, which can lead for example to stop situations. Several strategies have been used to track trajectories with quadrotor vehicles (Lee & Kim, 2017). In this work, a robustified version of the integral backstepping whole-body controller presented by Salierno & Raffo (2017) is proposed.

Regarding multi-robot systems, the ability to optimize the system's trajectories distributively has attracted the attention of researchers to distributed model predictive control (DMPC) approaches. The major advantage of optimal control strategies is the capability of minimizing (or maximizing) a cost functional, taking into account the system dynamics and constraints. By considering a receding horizon, MPC provides feedback to the system, which increases the robustness in front of external disturbances and parametric uncertainties (Rawlings & Mayne, 2009). In complex and large systems, the employment of a centralized controller might be infeasible, as high computational burden is involved. Consequently, it is generally beneficial to implement decentralized or distributed strategies (Maestre & Negenborn, 2014). Furthermore, computing control signals in a distributed manner increases robustness in front of system failure, since the optimization does not depend on a unique central processing unit. In this work, a multi-robot system is guided by convergence and circulation vector fields linked with a distributed model predictive control scheme, in which the robots negotiate how to modulate their vector fields to accomplish the task while avoiding inter-robot collisions.

1.2 Contributions

This dissertation considers the problem of convergence and circulation of curves under two different perspectives.

In the first moment, a cascaded strategy for achieving convergence and circulation of curves in the 3-D space by quadrotor vehicles is proposed. The approach is an extension of the one presented by Pimenta et al. (2013), this time considering time-varying curves. For controlling the quadrotors based on the high-level control signal as reference, a robustified version of the backstepping controller with integral action (Salierno & Raffo, 2017) is proposed. The original controller proposed by Salierno & Raffo (2017) is capable of rejecting constant external disturbances and parameter uncertainties. The robustness of this controller is amplified by considering unknown bounded disturbances treated by an additional robustifying control law, which is based on the Lyapunov redesign procedure (Khalil, 2002).

In a second moment, the problem of modulating the convergence and circulation fields is regarded for the first time as an optimization problem, which allows us to pursue inter-robot collision avoidance through distributed optimal control. For that matter, a framework that combines predefined parameterized control laws and distributed model predictive control is developed, based on the ideas proposed by Droge & Egerstedt (2013). The framework allows us to consider vector-field based control laws, in which its parameters are modulated by the model predictive control optimization problem. The optimization problem is distributed among the neighbouring robots by the alternating direction method of multipliers (Boyd et al., 2011). Then, a negotiation process is considered to achieve consensus on the choice of each robot vector field parameters.

1.2.1 Publications

The cascaded strategy presented in this work was published as "Convergência e Circulação de Curvas Variantes no Tempo com um Conjunto de Quadrotores" and presented in the "14º Simpósio Brasileiro de Automação Inteligente – SBAI" in Ouro Preto, Brazil (Pacheco et al., 2019). The distributed model predictive control strategy was accepted for publication in the International Federation of Automation and Control (IFAC) World Congress 2020 as "Distributed Parameterized Predictive Control for Multi-robot Curve Tracking".

1.3 Outline

The remaining of this text is split as follows:

- **Chapter 2** introduces the main background of the techniques used in this work.
- **Chapter 3** shows the development of a cascaded control scheme to guide a set of quadrotors to a time-varying curve in the 3-D space.
- **Chapter 4** presents the parameterized model predictive control with distributed optimization framework, which is applied to the problem of convergence and circulation of curves in the 2-D space.
- **Chapter 5** presents and discusses the numerical simulation results of the strategies developed through the two previous chapters.
- **Chapter 6** draws the main conclusions and highlights possible future works.

2 Background

This chapter introduces the main background necessary for the development of this manuscript. First, the vector field approach to convergence and circulation of closed paths is presented. After that, the quadrotor model considered in this work is introduced, and then the control of quadrotors is reviewed, focusing on backstepping control strategies. Besides, the Lyapunov redesign approach to be incorporated in the backstepping process is also described. Thereafter, a review of model predictive control and its distributed version in multi-robot systems is presented, and the alternating direction method of multipliers is described with an application example.

2.1 Vector Field Navigation

The application of vector fields in robot navigation is not a novelty. In the seminal work (Khatib, 1986), the author presents for the first time the vector field based strategy known as artificial potential fields. The main idea of the work is to compute a virtual attraction force that attracts the robot to a desired goal, and virtual repulsion forces that emanate from the obstacles. A vector field is built by assigning a vector to each point in the space. In Figure 2.1 one can see two vector fields in the 2-D space. In the case of robotics, these vector fields generally indicate the velocity or acceleration to be impelled by a robot at a certain point of the space.

Vector fields are today a well-established tool for obtaining control actions to track paths with convergence guarantees. In (Frew et al., 2007), Lyapunov guidance vector

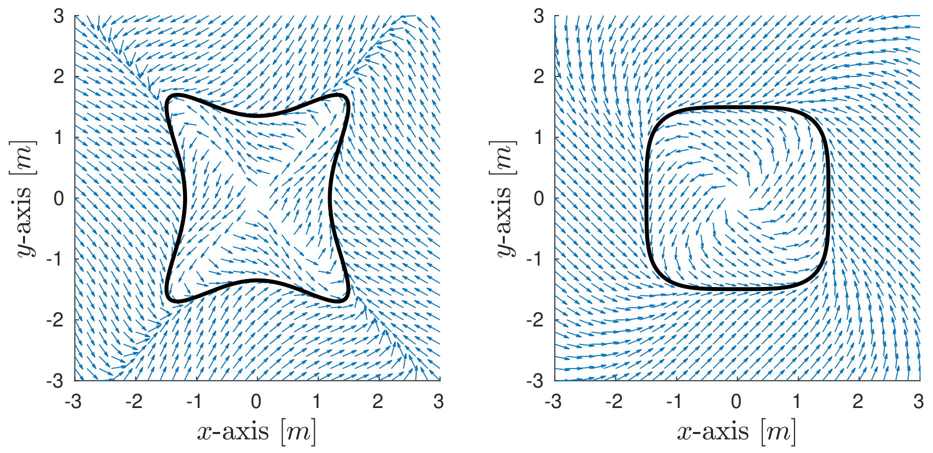


Figure 2.1: Two examples of vector fields for convergence and circulation.

fields are implemented for tracking circular loiter patterns in the 2-D space with an unmanned aircraft. In that work, the design of the vector field is based on a Lyapunov function determined by the desired distance from the loiter center and the real distance. In (Lawrence et al., 2007), the same framework is extended by considering the warping of the circular shape, preserving global stability guarantees. The framework is extended again by Frew et al. (2008), now considering a compensation term for tracking moving curves. In (Frew & Lawrence, 2012, 2017), the authors consider the construction of Lyapunov vector fields in cylindrical coordinates, which allow to track curves specified only by a set of control points.

The design of vector fields regarding convergence and circulation is also studied in (Ceccarelli et al., 2008). In this case, the authors propose a strategy to drive differential robots to a circular curve around a beacon. More general curves are not considered. Another interesting approach is presented by Wu et al. (2018), in which the vector field is encoded by a recently developed tool, namely differential Lyapunov framework for contraction analysis. The main advantage of the method is the possibility of creating vector fields in the case of curves with singularities. In addition, the framework allows a simple representation of curves obtained by interpolation.

This work considers vector fields that are constructed via implicit functions. In (Chaimowicz et al., 2005), the problem of controlling a swarm of robots to generate specific geometric patterns described by implicit functions is solved. In the same approach, repulsion forces between the agents are added into the control law, assuring that the robots spread out over the geometric formation. A similar problem is addressed by Pimenta et al. (2013), in which the interaction between robots is modeled using the Smoothed Particle Hydrodynamics (SPH). Control laws are obtained by assuming the problem as a simulation of fluids in electrostatic fields. In (Hsieh et al., 2007), the authors propose a strategy for convergence and circulation of a curve defined by implicit functions in the 2-D space. The control law relies on a convergence term, that attracts the robot to the

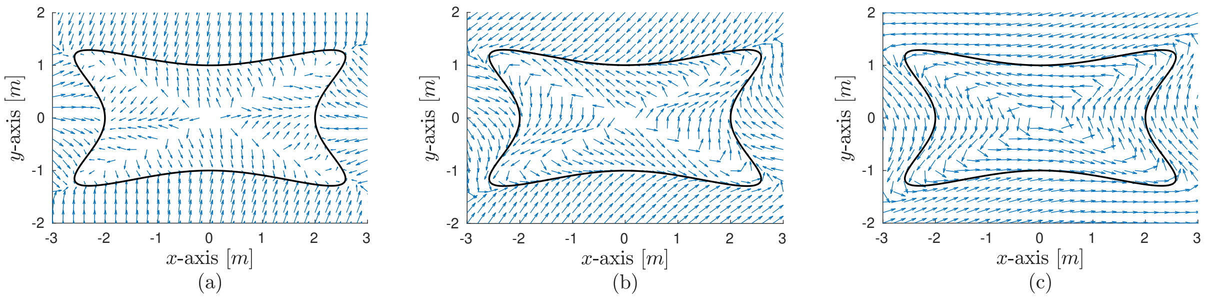


Figure 2.2: Example of vector field convergence and circulation modulation. (a) Convergence prioritized over circulation. (b) Convergence and circulation field have equal priority. (c) Circulation prioritized over convergence.

curve, and a tangential term, that impels circulation. Collision avoidance is accomplished by modulating the convergence and circulation parameters based on predefined proximity functions and priority relations. Figure 2.2 shows an example of vector field modulation.

Considering a single robot, the work (Gonçalves et al., 2010b) generalizes the convergence and circulation of generic curves specified in n-dimensional spaces. In this more general framework, the authors propose a vector field composed of three terms: (i) an attraction gradient to the curve, (ii) a tangential term, orthogonal to the first one, that guarantees circulation of the curve; and (iii) a term that compensates for the possibly time-varying characteristics of the curve. For obtaining curves in n-dimensional spaces, it is necessary to define $n-1$ surfaces, with intersection at the target curve. The surfaces are described by a set of implicit functions $\alpha_i : \mathbb{R}^{n+1} \mapsto \mathbb{R}$, such that $\alpha_i(x_1, x_2, ..., x_n, t) = 0$, for $i = 1, 2, ..., n-1$. Figure 2.3 shows an example of target curve defined by the intersection of two surfaces $\alpha_1(x_1, x_2, x_3) = 0$ and $\alpha_2(x_1, x_2, x_3) = 0$ in a 3-D space. After that, a differentiable positive definite potential function $V : \mathbb{R}^{n-1} \mapsto \mathbb{R}$ is defined. As α_i is function of $x_1, x_2, ..., x_n$ and t, $V(\alpha_1, \alpha_2, ..., \alpha_{n-1})$ is also function of the same variables. When $V = 0$, the robot is exactly at the curve. By using the potential function V, the authors derive a convergence field, based on the gradient of V, and a circulation field, orthogonal to the gradient.

By following the same reasoning from (Hsieh et al., 2007), the work (Pimenta et al., 2013) builds upon (Gonçalves et al., 2010b) and proposes a multi-robot scheme for convergence and circulation of a curve in the 3-D space while avoiding collisions by modulating convergence and circulation terms of each robot. The same work also applies the vector field modulated control signals to low-level controllers, which are responsible for guiding quadrotors vehicles in the vector field flow. In this work, vector field modulation strategies for convergence and circulation of curves are also developed, based on the strategies proposed by Pimenta et al. (2013) and Gonçalves et al. (2010b).

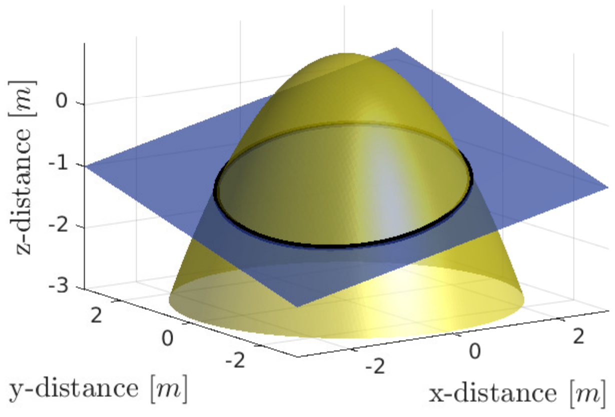


Figure 2.3: Two surfaces intersection.

2.2 Quadrotor Model

Figure 2.4 depicts the quadrotor model, where $\mathscr{I} = \{\vec{I}_1, \vec{I}_2, \vec{I}_3\}$ is the inertial frame, and $\mathscr{B} = \{\vec{B}_1, \vec{B}_2, \vec{B}_3\}$ is the body-fixed frame, attached to the body of the aircraft. The center of mass position in the inertial frame is given by $\boldsymbol{\xi} = [x\, y\, z]^T \in \mathbb{R}^3$. Considering the Euler angles parametrization, $\boldsymbol{\eta} = [\phi\, \theta\, \psi]^T \in \mathbb{R}^3$ represents the orientation of the quadrotor in the Euclidean space with respect to the body-fixed frame $\mathscr{B}$, where $\boldsymbol{R}_{\mathscr{I}} \in SO(3)$ is the rotation matrix from $\mathscr{B}$ to $\mathscr{I}$. The rotation matrix is obtained by applying three consecutive rotations to the body-fixed frame. First, assume that the body-fixed frame coincides with the inertial frame. Now, it is necessary to apply a rotation of ψ around $\vec{B}_3$, a rotation of θ around the new $\vec{B}_2$, and finally a rotation around the new $\vec{B}_1$. The three movements can be described by the following rotation matrices

$$\boldsymbol{R}(\vec{B}_3, \psi) \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ \boldsymbol{R}(\vec{B}_2, \theta) \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \ \boldsymbol{R}(\vec{B}_1, \phi) \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix},$$

which describe the rotations starting from the inertial frame to the body-fixed frame. When applying the three rotations, one can obtain

$$\boldsymbol{R}_{\mathscr{B}} = {}^{\mathscr{B}}\boldsymbol{R}_{\mathscr{I}} = \boldsymbol{R}(\vec{B}_1, \phi) \cdot \boldsymbol{R}(\vec{B}_2, \theta) \cdot \boldsymbol{R}(\vec{B}_3, \psi),$$

$$\boldsymbol{R}_{\mathscr{B}} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix}.$$

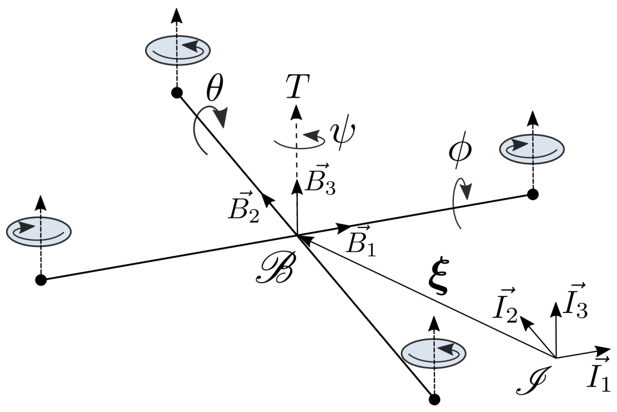


Figure 2.4: Quadrotor vehicle model.

The orientation of the body-fixed frame with respect to the inertial frame is obtained by transposing the $\boldsymbol{R}_{\mathscr{B}}$ matrix, and is given by

$$^{\mathscr{I}}\boldsymbol{R}_{\mathscr{B}} = \boldsymbol{R}_{\mathscr{I}} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}.$$

Based on the Newton-Euler formulation, the quadrotor model can be given by (Raffo, 2011)

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v}_{\mathscr{I}} \\ m\dot{\boldsymbol{v}}_{\mathscr{I}} &= -mg\boldsymbol{e}_3 + \boldsymbol{R}_{\mathscr{I}}\boldsymbol{e}_3 T + \boldsymbol{b} \\ \dot{\boldsymbol{R}}_{\mathscr{I}} &= \boldsymbol{R}_{\mathscr{I}}\boldsymbol{S}(\boldsymbol{\omega}) \\ \boldsymbol{I}^{\mathscr{B}}\dot{\boldsymbol{\omega}} &= \boldsymbol{\omega} \times \boldsymbol{I}^{\mathscr{B}}\boldsymbol{\omega} + \boldsymbol{\tau}_a, \end{aligned} \tag{2.1}$$

in which $\boldsymbol{v}_{\mathscr{I}}$ is the velocity of the UAV center of mass in the inertial frame $\mathscr{I}$, m is the mass of the quadrotor, $\boldsymbol{I}^{\mathscr{B}}$ is the inertia tensor, $\boldsymbol{b}$ represents the disturbances acting in the system, and $\boldsymbol{e}_3$ is the column vector $[0\ 0\ 1]^T$. The absolute angular velocity vector of the rigid body expressed in the body-fixed frame $\mathscr{B}$, is defined as $\boldsymbol{\omega} = [p\ q\ r]^T$. The system inputs are the total thrust T and $\boldsymbol{\tau}_a = [\tau_\phi\ \tau_\theta\ \tau_\psi]$, which are the applied torques around the axes in $\mathscr{B}$. Furthermore, $\boldsymbol{S}(\boldsymbol{\omega})$ is the skew-symmetric matrix (Spong et al., 2006), given by

$$\boldsymbol{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}.$$

Besides, the relationship between the angular rates $\boldsymbol{\omega}$ of the body-fixed frame and the Euler angles can be described by

$$\boldsymbol{\omega} = \boldsymbol{W}_\eta \dot{\boldsymbol{\eta}},$$

where $\boldsymbol{W}_\eta$ is the Euler matrix, given by

$$\boldsymbol{W}_\eta = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}.$$

2.3 Backstepping Control Approach

Since the beginning of this century, several strategies have been proposed for controlling quadrotor vehicles (Özbek et al., 2016). The design of controllers for quadrotor vehicles is complex, due to some characteristics of these aircraft. First, the quadrotor dynamics are highly nonlinear and subject to external disturbances, such as wind and turbulence. Furthermore, this system is also subject to parametric uncertainties and unmodelled dynamics, which makes it necessary to design robust controllers (Luque-Vega et al., 2012). As this kind of UAV can translate in three directions and rotate around three axes, its motion has six degrees of freedom. On the other side, it has only four control inputs, being configured as an underactuated system (Dierks & Jagannathan, 2009).

Regarding path following, Mistler et al. (2001) show how to achieve input-output decoupling by using a dynamic feedback controller, which renders the closed-loop system linear and controllable. In (Bouabdallah et al., 2004a), the system is decoupled into two subsystems: one for the angular rotations and the other one for the linear translations. A controller based on a Lyapunov function to control attitude and altitude is proposed. In (Bouabdallah et al., 2004b), the PID and LQR control techniques are applied for controlling the quadrotor attitude. The authors conclude that both techniques give good results, each one having its limitations. In (Bouabdallah & Siegwart, 2005), attitude control is achieved by using the same decoupling of the system and by applying the backstepping and sliding mode nonlinear control techniques. While the sliding mode controller had an acceptable performance, the backstepping controller handled relatively high perturbations.

In (Madani & Benallegue, 2006), the quadrotor is regarded as three subsystems. The first subsystem express the horizontal dynamics (x, y) in relation to roll and pitch angles. The second subsystem gives the dynamics of the height z and the yaw angle. The last subsystem gives the dynamics of the propellers forces. Using this architecture, the authors propose a backstepping controller capable of stabilizing the quadrotor position and yaw angle. The work (Bouabdallah & Siegwart, 2007) presents an integral backstepping strategy for altitude, attitude, and position control. In this case, the integral action allows the closed-loop system to reject constant disturbances. For achieving that, the system is again divided into two subsystems. The work (Raffo et al., 2008) presents a nonlinear robust path tracking controller for a quadrotor, in which a nonlinear $\mathscr{H}_\infty$ controller stabilizes the rotational movements and a control law based on the backstepping approach tracks the reference trajectory.

Backstepping approaches are very popular for controlling quadrotor UAVs. In front of possible external disturbances, as wind and turbulence, the insertion of integral action on the first step of the backstepping procedure is applied in (Bouabdallah & Siegwart, 2007; Colorado et al., 2010; Mian & Daobo, 2008; Jasim & Gu, 2015). In Raffo et al. (2015), a combination between a $\mathscr{H}_\infty$ controller and a backstepping control law is presented. This time, an integral action is added in the second step of the backstepping, which means that the velocity dynamics is integrated. As discussed by Skjetne & Fossen (2004), by considering an integral action in the first step of the backstepping procedure in a generic plant, it is possible to guarantee reference stabilization. However, to achieve reference tracking, it is necessary to introduce the integral action in the second step.

The application of backstepping generally relies on dividing the system dynamics into subsystems, which can lead to stability issues. The works (Salierno & Raffo, 2017; Salierno, 2018) address the path tracking problem by applying a backstepping strategy with integral action in the second step. In addition, rotation matrix properties are used in the backstepping procedure, so that subdivision of the system is not anymore necessary. The integral action provides robustness in the presence of parametric uncertainties, unmodelled dynamics, and constant external disturbances. In Chapter 3, an extension of this technique is presented by considering also an additional control law, which is responsible for compensating unknown but bounded disturbances, possibly time-varying.

The basic backstepping procedure can be explained with the help of a simple example, which is presented here just as described by Khalil (2002). Consider the case of the integrator backstepping, which the initial system is given by

$$\dot{\eta} = f(\eta) + g(\eta)\xi, \tag{2.2}$$
$$\dot{\xi} = u, \tag{2.3}$$

in which $[\eta^T \ \xi]^T \in \mathbb{R}^{n+1}$ is the state vector, and $u \in \mathbb{R}$ is the control input. Also, the functions $f : D \to \mathbb{R}^n$ and $g : D \to \mathbb{R}^n$ are smooth in a domain $D \subseteq \mathbb{R}^n$ that contains $\eta = 0$ and $\xi = 0$. Suppose that it is possible to stabilize (2.2) can be stabilized by choosing a smooth state feedback control law $\xi = \phi(\eta)$ with $\phi(0) = 0$. Suppose also that we know a smooth and positive definite Lyapunov function $V(\eta)$ such that

$$\frac{\partial V}{\partial \eta}[f(\eta) + g(\eta)\phi(\eta)] \leq -W(\eta), \quad \forall \eta \in D, \tag{2.4}$$

in which $W(\eta)$ is positive definite. If we add and subtract $g(\eta)\phi(\eta)$ in (2.2), it follows that

$$\dot{\eta} = [f(\eta) + g(\eta)\phi(\eta)] + g(\eta)[\xi - \phi(\eta)], \tag{2.5}$$
$$\dot{\xi} = u. \tag{2.6}$$

After that, by applying the change of variables $z = \xi - \phi(\eta)$ yields

$$\dot{\eta} = [f(\eta) + g(\eta)\phi(\eta)] + g(\eta)z, \tag{2.7}$$

$$\dot{z} = u - \dot{\phi}, \tag{2.8}$$

and by considering $v = u - \dot{\phi}$, the system is given by

$$\dot{\eta} = [f(\eta) + g(\eta)\phi(\eta)] + g(\eta)z, \tag{2.9}$$

$$\dot{z} = v. \tag{2.10}$$

Note that as $z \to 0$, $\xi \to \phi(\eta)$. Therefore, (2.9) has an asymptotically stable origin when $z = 0$. To stabilize the overall system, consider the following Lyapunov candidate function

$$V_c(\eta, \xi) = V(\eta) + \frac{1}{2}z^2, \tag{2.11}$$

and its time derivative, given by

$$\dot{V}_c = \frac{\partial V}{\partial \eta}[f(\eta) + g(\eta)\phi(\eta)] + \frac{\partial V}{\partial \eta}g(\eta)z + zv \leq -W(\eta) + \frac{\partial V}{\partial \eta}g(\eta)z + zv. \tag{2.12}$$

In this step, one must choose v properly in order to stabilize the overall system. By taking

$$v = -\frac{\partial V}{\partial \eta}g(\eta) - kz, \qquad k > 0, \tag{2.13}$$

the time derivative $\dot{V}_c$ gives

$$\dot{V}_c \leq -W(\eta) - kz^2, \tag{2.14}$$

which shows that the origin is asymptotically stable. After that, it is possible to obtain a control law $u = v + \dot{\phi}$ to be applied to the system, which is given by

$$u = \frac{\partial \phi}{\partial \eta}[f(\eta) + g(\eta)\xi] - \frac{\partial V}{\partial \eta}g(\eta) - k[\xi - \phi(\eta)]. \tag{2.15}$$

The same approach can be applied to more general systems. Consider

$$\dot{\eta} = f(\eta) + g(\eta)\xi, \tag{2.16}$$

$$\dot{\xi} = f_a(\eta, \xi) + g_a(\eta, \xi)u, \tag{2.17}$$

in which f_a and g_a are smooth. If $g_a(\eta, \xi) \neq 0$ over the domain of interest, one can design a feedback linearization law (Slotine et al., 1991), as follows

$$u = \frac{1}{g_a(\eta, \xi)}[u_a - f_a(\eta, \xi)], \tag{2.18}$$

which reduces (2.17) to $\dot{\xi} = u_a$. From this step, it is possible to design the control law u with the same approach previously presented.

If we apply the backstepping recursively, it is possible to stabilize strict-feedback systems

$$\begin{aligned}
\dot{x} &= f_0(x) + g_0(x)z_1 \\
\dot{z}_1 &= f_1(x, z_1) + g_1(x, z_1)z_2 \\
\dot{z}_2 &= f_2(x, z_1, z_2) + g_2(x, z_1, z_2)z_3 \\
&\vdots \\
\dot{z}_k &= f_k(x, z_1, \cdots, z_k) + g_k(x, z_1, \cdots, z_k)u
\end{aligned}$$

in which $x \in \mathbb{R}^n$, z_1, $z_2 \ldots$, z_k are scalars, the functions f_0, f_1, $\ldots$, f_k vanish at the origin, and we assume that

$$g_i(x, z_1, \ldots, z_i) \neq 0, \quad \text{for } 1 \leq i \leq k \tag{2.19}$$

over the domain of interest. The procedure starts by considering the system $\dot{x} = f_0(x) + g_0(x)z_1$, with z_1 as control input. First, we determine a stabilizing state feedback control law $z_1 = \phi_0(x)$ and a Lyapunov function $V_0(x)$ such that

$$\frac{\partial V_0}{\partial x}\left[f_0(x) + g_0(x)\phi_0(x)\right] \leq -W(x) \tag{2.20}$$

in which $W(x)$ is positive definite. After that, we proceed the backstepping design by considering the following system

$$\begin{aligned}
\dot{x} &= f_0(x) + g_0(x)z_1, \\
\dot{z}_1 &= f_1(x, z_1) + g_1(x, z_1)z_2,
\end{aligned}$$

and by applying the feedback linearization we can obtain the control law

$$\phi_1(x, z_1) = \frac{1}{g_1}\left[\frac{\partial \phi_0}{\partial x}(f_0 + g_0 z_1) - \frac{\partial V_0}{\partial x} g_0 - k_1(z_1 - \phi_0) - f_1\right], \quad k_1 > 0,$$

and the Lyapunov function

$$V_1(x, z_1) = V_0(x) + \frac{1}{2}\left[z_1 - \phi_0(x)\right]^2.$$

Then, we consider the system

$$\begin{aligned}
\dot{x} &= f_0(x) + g_0(x)z_1 \\
\dot{z}_1 &= f_1(x, z_1) + g_1(x, z_1)z_2 \\
\dot{z}_2 &= f_2(x, z_1, z_2) + g_2(x, z_1, z_2)z_3
\end{aligned}$$

and the recursive procedure is applied until we obtain the overall stabilizing state feedback control law $u = \phi_k(x, z_1, \dots, z_k)$ and the Lyapunov function $V_k(x, z_1, \dots, z_k)$. The time derivative of the final Lyapunov function yields

$$\dot{V}_k \leq -W(x_1) - k_1 z_1^2 - k_2 z_2^2 - \dots - k_k z_k^2 < 0, \quad k_1, k_2, \dots, k_k > 0, \tag{2.21}$$

which guarantees closed-loop asymptotic stability.

2.4 Lyapunov Redesign

Lyapunov stability analysis is a standard framework for evaluating the stability of nonlinear systems (Khalil, 2002). As described in (Levine, 2010), the Lyapunov-based control design is a strategy for robust control that consists of two main steps:

1. A candidate Lyapunov function $V(x)$ is proposed for a closed-loop system.
2. A control law u is designed so that $\dot{V}(x) < 0$ considering all uncertainties.

If the selection of the candidate Lyapunov is made with success, that is, the second step is performed, the function is called control Lyapunov function. In the Lyapunov redesign technique (Freeman & Kokotovic, 2008; Khalil, 2002), a known Lyapunov function for the nominal system is used as a control Lyapunov function for the uncertain system. From this, an additional control law is designed to obtain $\dot{V}(x) < 0$.

The Lyapunov redesign is a well-established strategy for robustifying control laws. In (Hwang et al., 2013) for example, Lyapunov redesign is applied together with a backstepping controller to achieve path tracking with skid-steer mobile robots. In that case, the parameter uncertainties and the skidding are regarded as disturbances. In (Raffo & de Almeida, 2016), a task of load transportation by a quadrotor UAV is proposed. A Lyapunov redesign control law is added to a $\mathscr{H}_\infty$ controller so that the swing of the load is reduced. The work (Soorki et al., 2011) presents a strategy to perform leader-follower robot formations robustified by Lyapunov redesign. As the velocity of the leader robot is difficult to measure, its value is considered a model uncertainty.

The classical Lyapunov redesign procedure is described as follows (Khalil, 2002). Consider the system

$$\dot{x} = f(x) + g(x)u + b(x, t), \tag{2.22}$$

in which x is the state vector, u is the control input, $f(x)$ and $g(x)$ are known functions that describe the nominal system, and $b(x, t)$ is an uncertainty term known to be between bounds, such as $||b(x, t)||_\infty \leq \rho(x)$. The uncertainty b can also depend on u, but it is maintained as $b(x, t)$ to simplify the understanding of the procedure. Assume that the nominal control system is stabilized by a nominal control law $u_{nom}(x)$. The nominal closed-loop system is given as follows

$$\dot{x} = f(x) + g(x)u_{nom}(x), \tag{2.23}$$

in which $x = 0$ is a globally asymptotically stable equilibrium point. Assume that the nominal system has a known Lyapunov function $V(x)$ such that

$$\nabla V(x)^T \left[f(x) + g(x)u_{nom}(x)\right] < 0 \tag{2.24}$$

for $x \neq 0$, in which $\nabla V(x)$ is the gradient of $V(x)$. The main idea is to propose an additional law $u_{rob}(x)$, so that $u = u_{nom} + u_{rob}$ stabilizes the uncertain system (2.22). Therefore, it is necessary to guarantee that the Lyapunov function derivative be negative for all uncertainty, that is

$$\dot{V} = \nabla V(x)^T \left[f(x) + g(x)u_{nom}(x)\right] + \nabla V(x)^T \left[g(x)u_{rob}(x) + b(x,t)\right]. \tag{2.25}$$

Note that the first part of the Equation (2.25) is negative. By considering that the disturbance respects the matching condition, expressed by

$$b(x,t) = g(x) \cdot \bar{b}(x,t) \tag{2.26}$$

for some uncertain $\bar{b}(x,t)$, and by obtaining the new bound $||\bar{b}(x,t)||_\infty \leq \bar{\rho}(x)$, one can choose the additional control law as

$$u_{rob}(x) = -\bar{\rho}(x)\,\mathrm{sgn}(g(x)^T \nabla V(x)), \tag{2.27}$$

in which $\mathrm{sgn}(\cdot)$ is the element-wise signal function. By substituting the control law (2.27) in (2.25) we obtain

$$\dot{V} = \nabla V(x)^T \left[f(x) + g(x)u_{nom}(x)\right] + \nabla V(x)^T g(x) \left[-\bar{\rho}(x)\,\mathrm{sgn}(g(x)^T \nabla V(x)) + \bar{b}(x,t)\right], \tag{2.28}$$

where the second term is lower or equal 0, and therefore $\dot{V} \leq 0$.

In the backstepping approach, the control law provided by the Lyapunov redesign procedure is added between the steps of the backstepping. As the backstepping technique constructs the control law by using control Lyapunov functions, an interesting match is obtained between both techniques.

2.5 Model Predictive Control

Model predictive control (MPC) strategies emerged as a natural extension of the classic optimal control ideas. In optimal control, the main idea is to define an optimization problem capable of providing a control law or a control sequence. For doing this, the dynamics of the process to be controlled is embedded in the optimization problem as a constraint (Kirk, 2004). Model predictive control uses the same ideas to construct optimal control problems, but with a receding time horizon. As described by Camacho & Bordons Alba (2013), three main ideas characterize MPC strategies:

- use of the explicit model of the system;
- obtaining a control law or sequence for a given optimal control problem; and
- receding horizon, that is, each control problem is solved considering a window of time, that is displaced in each sampling instant after applying the first control signal.

With this regard, MPC is one way of closing the loop in optimal control, as the optimal control problem is recomputed after each sampling step considering the new states, which increases robustness in front of external disturbances and uncertainties (Rawlings & Mayne, 2009). Besides that, the main advantages of the MPC strategies are the easy application in multivariable systems, the explicit definition of model constraints, and the prediction characteristics (Camacho & Bordons Alba, 2013). A general MPC strategy is described in Algorithm 2.1, in which $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the control input vector, Δ is the prediction horizon, $L(\boldsymbol{x}, \boldsymbol{u})$ is the stage cost, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ is the system dynamics, $\boldsymbol{x}\{:\}$ is the state trajectory, $\boldsymbol{u}\{:\}$ is the control input trajectory, and $\boldsymbol{x} \in \mathbb{X}$ and $\boldsymbol{u} \in \mathbb{U}$ are constraints on the state and control input, respectively.

Algorithm 2.1 General MPC Strategy

```
1: repeat
2:     $t := t + dt$;
3:     $t_0 := t$;
4:     $\boldsymbol{x}_0 := \boldsymbol{x}$;
5:     $t_f := t_0 + \Delta$;
6:     Solve the optimal control problem
       $\min_{\boldsymbol{x}\{:\},\boldsymbol{u}\{:\}} \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u})dt$
       subject to
       $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$,
       $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$,
       $\boldsymbol{x} \in \mathbb{X}$, $\boldsymbol{u} \in \mathbb{U}$
7:     Apply the first control signal computed.
8: until reaches the final time
```

At first, the MPC strategies were generally applied in systems with slow dynamics, in which the computation time for solving the optimization problems was reasonable. As the technological advances enable the production of faster and faster computers, the application of MPC strategies to control systems with fast dynamics becomes plausible. Besides that, several low-level techniques for solving optimization problems faster have been proposed regarding application in MPC problems, see (Patrinos & Bemporad, 2013; Jerez et al., 2014; Stella et al., 2017; Alamo et al., 2019).

In robotic systems, the application of MPC strategies has been much studied. Especially when dealing with navigation problems, MPC strategies provide the possibility of treating the collision avoidance problem in the optimization problem formulation. In (Kang & Hedrick, 2006), the authors propose a nonlinear MPC (NMPC) strategy using a kinematic model of a fixed-wing UAV for tracking lines and avoid obstacles. Regarding obstacle avoidance, the authors add to the optimization cost functional a repulsion term, written using exponential functions of the distance between the UAV and the obstacle. Similar strategies for collision avoidance have been used in (Lee et al., 2011) and (Droge & Egerstedt, 2013). In (Santos, 2018), the classic potential functions strategy is applied, in which attractive and repulsive potentials are included in an Economic MPC cost function. The same work also considers a hard constraint which prevents the generated trajectory from passing through an obstacle. In (Yoon et al., 2009), an MPC approach for autonomous ground vehicles (UGVs) is proposed. Two different strategies are considered for collision avoidance. In the first one, a repulsion term proportional to the velocity and inversely proportional to minimum distance (from the UGV to the obstacle) is added to the cost functional. A second strategy uses parallax information from the robot about the obstacle to construct a repulsion term, which leads to better collision avoidance characteristics since there is a notion of the form of the UGV. A similar parallax-based repulsion term is used in (Park et al., 2009) to avoid collisions. In (Richards & How, 2004; Zhao & Go, 2014), Mixed-Integer Linear Programming based collision avoidance constraints are used in MPC schemes. In (Pereira et al., 2019), collision avoidance is treated by including a hard constraint on the MPC problem that imposes a minimum distance between the quadrotor UAV predicted trajectory and the obstacle. In (Nascimento et al., 2019), a quadrotor equipped with a camera or laser sensor in a unknown environment is considered. In the proposed strategy, collision avoidance is achieved by decomposing the safe region into convex polyhedra, which are included as constraints into the MPC problem. In (Van Parys & Pipeleers, 2017), obstacle avoidance constraints are modeled as linear constraints by dividing the space using hyperplanes between the robot and the obstacles. Besides constraining the optimization problem in a simple way, the strategy is very conservative. Another interesting strategy for collision and obstacle avoidance in MPC schemes is proposed by Sathya et al. (2018). In that work, the obstacles are described by the intersection of open sets (a set of nonlinear inequalities), which allows to separate nonconvex shapes. The same strategy proposed by Sathya et al. (2018) is applied in (Small et al., 2019) to accomplish trajectory tracking with an inverted quadrotor in obstructed environments.

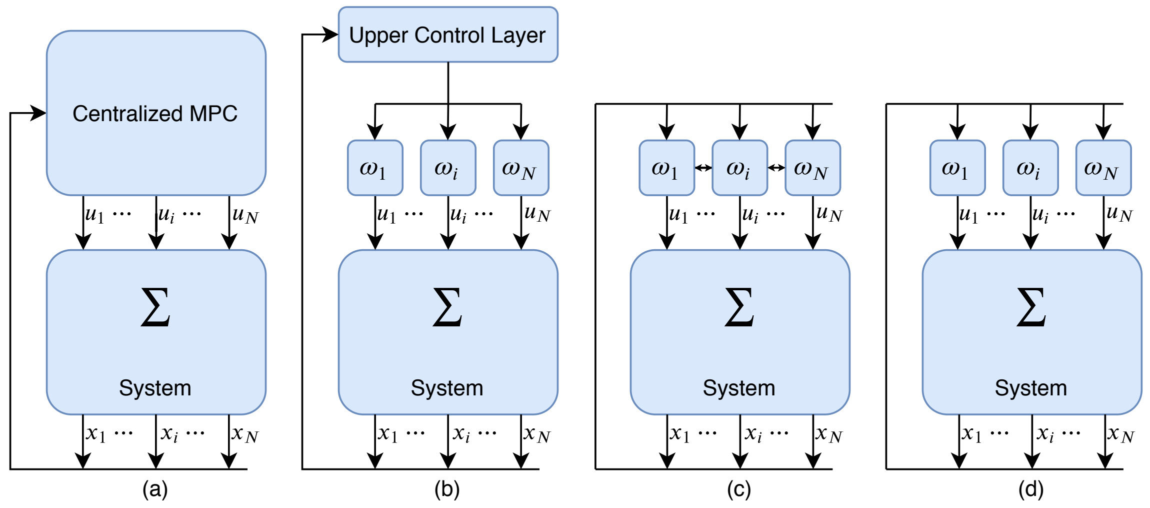
In Chapter 4, an MPC scheme is used to guide a set of robots to converge and circulate a curve in the 2-D space. To achieve that, a vector field based control law with convergence and circulation terms is embedded in the optimal control problem, so that we do not find a control sequence, but the parameters of convergence and circulation for each robot. As proposed in (Droge & Egerstedt, 2013) and (Droge, 2014), the embedding of control laws with parameters in the MPC problem has been called parameterized MPC. In this case, as fewer decision variables are considered, we expect to obtain simpler optimization problems. Besides, less ability to minimize the cost functional is also expected. In order to achieve collision avoidance, two methods are implemented in the proposed MPC scheme, one based in a repulsion exponential term added in the cost function, and the other based in the strategy proposed in (Sathya et al., 2018).

2.6 Distributed Model Predictive Control

MPC strategies can be implemented in different architectures when applied to multi-agent systems. Figure 2.5 shows some classical approaches. The first one is the centralized MPC, in which the MPC controller has knowledge of all the states in the system and can compute all the control signals. The second architecture is called Hierarchical MPC. In this architecture, there is an upper control layer that feeds all the agents ω_1 to ω_N with a reference signal. The next one is a Distributed MPC architecture. In this architecture, there is no centralized layer. Each agent computes an MPC problem based on its states and on the states of some other agents. To achieve global optimization, the agents must negotiate to achieve consensus about the control inputs to be chosen. The fourth architecture is the decentralized MPC. In this case, each agent is capable of computing a control law without a central planner or negotiation among agents. Therefore, each agent finds a control sequence by considering only its own states and other agents states, without considering the prediction. This characteristic generally causes loss of performance in the minimization of a global cost, once that the agents might be competing against each other (Negenborn & Maestre, 2014).

The main advantage of using distributed architectures in multi-robot systems is the intrinsic modularity and flexibility of this control structure, which allows for reconfiguration of the group of robots. Besides, as pointed out by Maestre & Negenborn (2014), some centralized problems do not scale well with the number of agents, which makes the distribution of the problem essential for obtaining a reasonable solution regarding the processing time. Furthermore, when a centralized controller involves communication over large distances, the distribution of the problem with local coordination can be of great benefit.

Several strategies have been proposed for providing a negotiation process in distributed MPC problems. The review book (Maestre & Negenborn, 2014) features 35 approaches for DMPC. In (Maestre et al., 2011) for example, the authors propose a negotiation scheme based on a cooperative game. In that approach, the global cost increment of a set of agents


Centralized MPC
Upper Control Layer
ω_1 ω_i ω_N
$u_1 \cdots$ $u_i \cdots$ u_N
Σ
System
$x_1 \cdots$ $x_i \cdots$ x_N
(a)
(b)
(c)
(d)
Figure 2.5: Model Predictive Control Architectures. (a) Centralized MPC. (b) Hierarchical MPC. (c) Distributed MPC. (d) Decentralized MPC.

is used to define the best control action in a scheme of negotiation proposals. Another strategy is to formulate DMPC by using distributed optimization. In these cases, coupling constraints are added to each agent problem so that equality is guaranteed at the end of a negotiation process. In (Farokhi et al., 2014) for example, the authors propose a DMPC by using the dual decomposition and the alternating direction method of multipliers (ADMM) distributed optimization methods.

Droge & Egerstedt (2013) propose a distributed parameterized MPC scheme that uses the dual decomposition technique. In the same work, the MPC problem is simplified by embedding a parameterized control law in the optimization problem. The optimization is carried out using a gradient-based approach, known as Uzawa's method. A multi-robot coordination problem also treated with distributed MPC is developed in (Van Parys & Pipeleers, 2017). The vehicle's trajectories are parameterized as splines. Besides, the agents' dynamics are considered to be differentially flat, which allows enforcing formation and input constraints via splines. To distribute the optimization, the authors apply one iteration of the alternating direction method of multipliers per MPC cycle, which lowers the computational cost. Zheng et al. (2016) address a path following problem for multiple waterborne automated vessels using DMPC. The local optimization problems are coupled with the collision avoidance constraints, and the distributed problem is also solved via ADMM.

In Chapter 4, a multi-robot control system based on a distributed predictive control strategy is proposed. The strategy considers parameterized control laws, i.e., a control law with tunable parameters. As the distributed predictive control needs to predict and agree on the trajectories of the neighboring agents, the use of parameterized control laws is convenient, assuming that the trajectory may be predicted with the initial state, the control law parameters, and the agent dynamics.

To solve the optimization problem, the ADMM distributed optimization method is adopted. In this technique, each agent has a version of the variables of the neighboring agents. To aim equality between these variables, the technique uses Lagrange multipliers and a penalty term that provides more robustness to the distributed optimization. As described previously, this technique has been implemented in multi-robot distributed MPC in (Van Parys & Pipeleers, 2017), but instead of spline-based trajectories, we use parameterized control laws as in (Droge & Egerstedt, 2013). The ADMM has been also implemented in other works such as (Rey et al., 2018) and (Ferranti et al., 2018) to solve optimal control problems regarding unmanned vehicle navigation with collision avoidance constraints. In the next section, a brief description of the ADMM technique with an elucidative example is given.

2.6.1 Alternating Direction Method of Multipliers

Distributed optimization allows to divide the computational burden among units. In multi-agent systems, a distributed optimal control scheme renders more robustness, since the decision is no longer associated with a central processing unit, but with a set of local negotiations between agents.

Several methods for distributed optimization have been studied in the last decades. Among the classical distributed optimization methods, the alternating direction method of multipliers is the one that provides better robustness and generally converges faster (Boyd et al., 2011). In distributed optimization, a global cost $f(x)$ is split among N processing units

$$f(x) = \sum_{i=1}^{N} f_i(x). \tag{2.29}$$

If all the processing units decide about the same decision variable, that is, the same decision variable is present at all the local cost functionals, a case of global consensus is set. Before properly dividing the optimization burden between agents, it is necessary to ensure that the solution of each local optimization problem reaches the same global solution. Hence, the equality constraint must be satisfied, as following

$$\begin{array}{ll} \min\limits_{\forall x_i} & \sum_{i=1}^{N} f_i\left(x_i\right) \\ \text{subject to} & x_i - z = 0, \quad i = 1, \dots, N, \end{array} \tag{2.30}$$

in which x_i is the local value of the decision variable and z is the global value of the decision variable. When the constraint $x_i - z = 0$ holds for each agent, each sub-problem returns the same value for the decision variable.

In ADMM the coupling constraints are relaxed by adding Lagrange multipliers and an additional penalty term. This additional term provides more robustness to the distributed optimization (Boyd et al., 2011). Henceforth, the cost functional (2.29) is augmented as follows

$$f_\rho\left(x, z, y\right) = \sum_{i=1}^{N} \left(f_i\left(x_i\right) + y_i^T\left(x_i - z\right) + \frac{\rho}{2}\left\|x_i - z\right\|_2^2 \right), \tag{2.31}$$

in which y_i is the Lagrange multiplier associated to the i-th processing unit, and $\rho > 0$ is a penalty constant to be chosen.

Once defined the augmented cost in (2.31), one can apply the ADMM algorithm. Three basic steps compose the algorithm: i) an x-minimization step; ii) a z-minimization step; and iii) a dual variable update. The set of iterations for the global consensus problem are defined as follows (Boyd et al., 2011)

$$x_i^{k+1} = \arg\min_{x_i} \left(f_i\left(x_i\right) + (y_i^k)^T\left(x_i - z^k\right) + \frac{\rho}{2}\left\|x_i - z^k\right\|_2^2 \right), \tag{2.32}$$

$$z^{k+1} = \frac{1}{N}\sum_{i=1}^{N}\left(x_i^{k+1} + (1/\rho)y_i^k\right), \tag{2.33}$$

$$y_i^{k+1} = y_i^k + \rho\left(x_i^{k+1} - z^{k+1}\right). \tag{2.34}$$

Each agent handles only its local objective function, whereas the dual variable update gathers global information and impels the local problems to obtain the same solution, achieving consensus. In the first step, the variables z and y are fixed and x is updated. After that, x and y are fixed and z is updated. In the third step, y is updated while the other variables are fixed. As pointed out by Boyd et al. (2011), the sequential fashion in the update of x_i and z inspires the name of the method.

From the iteration set (2.32 – 2.34), one can observe that ADMM requires two communication phases at each cycle. First, each agent computes the x-minimization step, obtaining x_i^{k+1}. After that, the computed value is transmitted to a central collector (also named fusion unit) that gathers all the information and computes z^{k+1}. Thereafter, the z^{k+1} value is transmitted by the centralizer unit to all the agents, which compute the Lagrange multipliers y_i^{k+1}.

To make clearer the presentation of the ADMM method, a simple example of distributed optimization is proposed. Consider an optimization problem given by the minimization of the cost functional

$$f(x) = (\chi_1 - 1)^2 + (\chi_2 - 3)^2 + (\chi_1 - 6)^2 + (\chi_1 - \chi_2)^2 \tag{2.35}$$

in which $x = [\chi_1\ \chi_2]^T$. This example is the same presented in (Droge, 2014) to exemplify the dual decomposition distributed optimization method.

Consider that the optimization problem must be solved in a distributed manner by three agents connected in a line network, as presented in Figure 2.6. If each agent has a

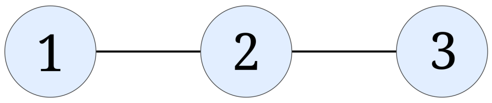


Figure 2.6: Three agents in line topology network.

version of the decision variables x, the optimization problem can be described by

$$\min_{x_i,\forall i} \quad \sum_{i=1}^{3} f_i(x_i),$$
$$\text{subject to}$$
$$x_i - z = 0 \quad \forall i \in \{1,2,3\},$$

in which $x_i = [\chi_{i1}\ \chi_{i2}]$ is the local variable, and $f_i(x_i)$ is the local cost functional, $\forall i \in \{1,2,3\}$. The cost functional might be split as follows

$$\begin{aligned} f_1(x_1) &= (\chi_{11}-1)^2 + \frac{1}{3}(\chi_{11}-\chi_{12})^2, \\ f_2(x_2) &= (\chi_{22}-3)^2 + \frac{1}{3}(\chi_{21}-\chi_{22})^2, \\ f_3(x_3) &= (\chi_{31}-6)^2 + \frac{1}{3}(\chi_{31}-\chi_{32})^2, \end{aligned} \tag{2.36}$$

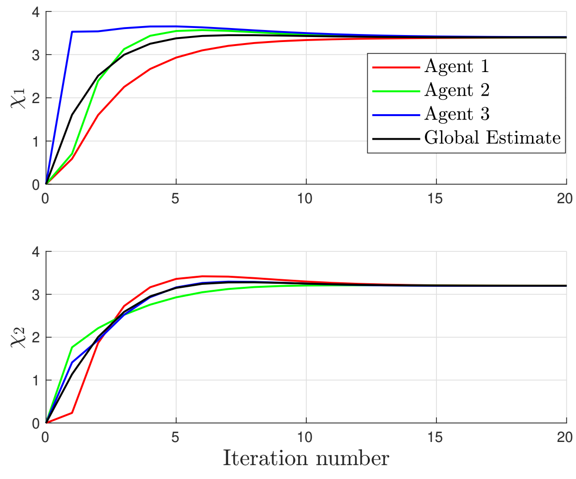
which are augmented with Lagrange multipliers and with a penalty term, yielding

$$\begin{aligned} f_{\rho 1}(x_1) &= (\chi_{11}-1)^2 + \frac{1}{3}(\chi_{11}-\chi_{12})^2 + y_1^T(x_1 - z) + \frac{\rho}{2}\|x_1 - z\|_2^2, \\ f_{\rho 2}(x_2) &= (\chi_{22}-3)^2 + \frac{1}{3}(\chi_{21}-\chi_{22})^2 + y_2^T(x_2 - z) + \frac{\rho}{2}\|x_2 - z\|_2^2, \\ f_{\rho 3}(x_3) &= (\chi_{31}-6)^2 + \frac{1}{3}(\chi_{31}-\chi_{32})^2 + y_3^T(x_3 - z) + \frac{\rho}{2}\|x_3 - z\|_2^2, \end{aligned} \tag{2.37}$$

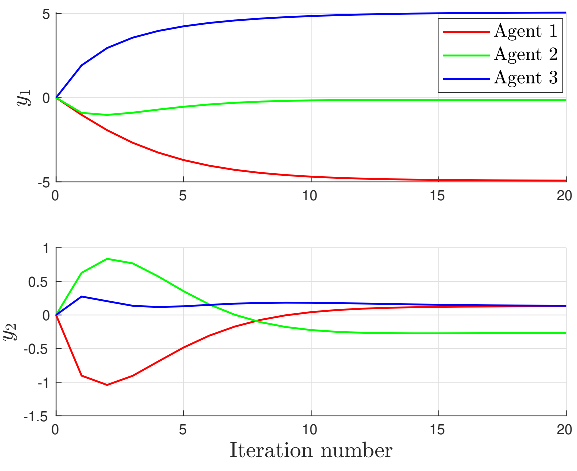
where $f_{\rho i}(x_i)$ stands for the augmented cost functional of the i-th agent.

Note that in this topology (Figure 2.6), the agent 2 is the fusion center, once that this agent is the only that can reunite the information about the local variables x_i and compute the global solution z. The optimization scheme is described by following the ADMM steps (2.32 – 2.34). In the first step, each agent computes the local solution x_i by minimizing the augmented cost $f_{\rho i}(x_i)$. Between the first and the second steps, the agents 1 and 3 communicate the local values x_i and the Lagrange multipliers y_i to the fusion center, the agent 2. In the second step, the agent 2 computes the estimate of the global solution z, which is transmitted to the other agents. In the last step, each agent computes its new Lagrange multipliers vector based on the difference between the local solution and the global solution. The set of iterations is repeated until convergence.

For solving the example here described, the penalty term is initially set to $\rho = 1$ and every value is initialized as zero. Besides that, 20 iterations of the ADMM method are executed and the optimization problems are implemented via the Casadi framework

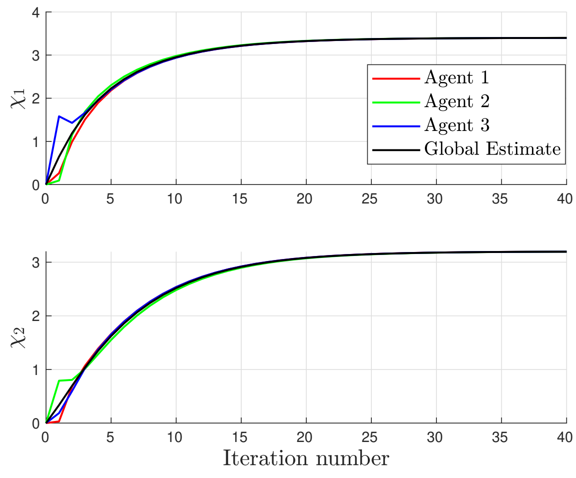


(a) Decision variable evolution.

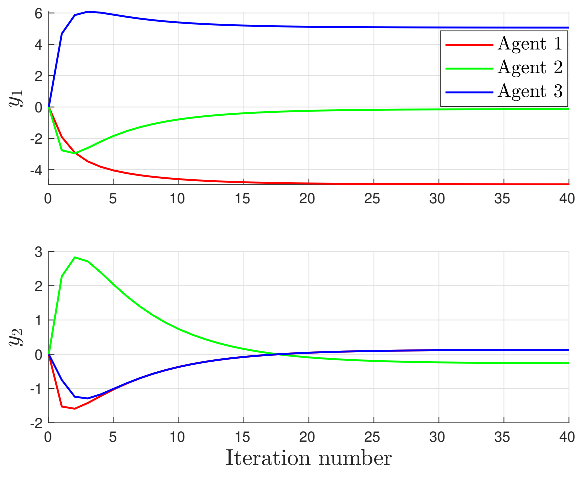


(b) Lagrange multipliers evolution.

Figure 2.7: ADMM example, $\rho = 1$.



(a) Decision variable evolution.



(b) Lagrange multipliers evolution.

Figure 2.8: ADMM example, $\rho = 5$.

(Andersson et al., 2019) and the solver Ipopt (Wächter & Biegler, 2006). Figure 2.7a shows the evolution of the local and global decision variables, and Figure 2.7b shows the evolution of the Lagrange multipliers. One can note that the variables tend to convergence and stabilization. As pointed out in (Boyd et al., 2011), the performance of the ADMM is highly dependent on the choice of the penalty parameter ρ. Figure 2.8 shows the evolution of the same distributed optimization, but this time considering $\rho = 5$ and 40 iterations. As one can abstract, a higher value of ρ forces the local problems to achieve consensus faster. However, the evolution of the global solution is slow, when compared to the previous example.

2.7 Summary

This chapter presented the main tools that compose the strategies proposed in this thesis. Vector field strategies are applied in the problems dealt in Chapter 3 and Chapter 4 to obtain control laws for convergence and circulation of closed paths. In addition, the nonlinear control techniques backstepping and Lyapunov redesign are applied in Chapter 3 to obtain a robust controller for a quadrotor vehicle. Finally, a distributed MPC approach based on the ADMM is developed to solve the problem of Chapter 4.

3 Cascaded Control Strategy

This chapter deals with the problem of convergence and circulation of a given curve by a group of quadrotors while avoiding inter-robot collisions. To achieve this, a cascaded control strategy is proposed. A high-level controller based on vector fields guides the robots, considered as single integrators, to the curve. Collision avoidance is achieved by modulating the components of the vector field in the presence of collision risk. In the low-level strategy, quadrotor UAVs are guided by a robust integral backstepping controller that takes as reference the vector field controller output. In the next section we state the problem addressed in this chapter.

3.1 Problem Statement

In this chapter, the problem of convergence and circulation of a group of robots to a curve embedded in the 3-D space while avoiding inter-robot collisions is addressed. To this end, a vector field approach based on (Gonçalves et al., 2010b) and (Pimenta et al., 2013) is proposed.

Consider a set Ω containing N robots ω_i represented as spheres, each one with configuration defined by $\boldsymbol{\xi}_i = [x_i(t)\ y_i(t)\ z_i(t)]^T$ in an instant of time t, where x_i, y_i, and z_i are the coordinates of the center of the sphere ω_i in relation to a global fixed reference frame. Each robot has a radius r_i that covers its shape, and a visibility range radius R_i that delimits its sphere of perception. The high-level coordination strategy is developed assuming single-integrator dynamics, given by

$$\dot{\boldsymbol{\xi}}_i = \boldsymbol{u}_i, \tag{3.1}$$

where $\boldsymbol{u}_i$ is the control input.

To obtain the target curve to which the group of robots must converge and circulate, a composition of two surfaces is considered. The intersection between these two surfaces results in the target curve. To represent both surfaces, consider two functions described by $\alpha_i(x, y, z, t) : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R},\ i \in \{1, 2\}$, which have continuous second partial derivatives. The parameter t is time, which allows for representing time-varying curves. The function α_1 depends only on x, y, and t, and must produce a closed, simple, and continuous curve when projected onto any plane parallel to the xy plane for all t. Therefore, any level surface of α_1 ($\alpha_1(\boldsymbol{\xi}, t^*) = C$) at the time instant t^* and with C constant is cylindric. The function α_2 is defined as

$$\alpha_2(x, y, z, t) = \sigma z - \Phi(x, y, t), \tag{3.2}$$

in which $\Phi(x, y, t)$ is a function with continuous second partial derivatives, and σ is a constant parameter such that $\sigma \neq 0$. From the functions α_1 and α_2, a curve Γ can be implicitly defined as

$$\Gamma : \{\boldsymbol{\xi} \in \mathbb{R}^3, t \in \mathbb{R} |\, \{\alpha_1(\boldsymbol{\xi}, t) = 0\} \cap \{\alpha_2(\boldsymbol{\xi}, t) = 0\}\}\,. \tag{3.3}$$

As $\alpha_1(\boldsymbol{\xi}, t) = 0$ is cylindric and every point of the plane xy at time t is a projection of only one point of $\alpha_2 = 0$ then the curve Γ is closed, continuous, and has a single projection onto the plane xy for all of its points for all t.

Remark 3.1. *It is assumed that $\nabla\alpha_1$ is different from zero at the curve and almost everywhere, being equal to zero at specific points named singular points. It is also assumed that the partial derivatives of α_1 and α_2 in relation to time are bounded.*

The problem addressed in this chapter is stated below:

Problem statement 3.1. *Design a control strategy that guides a group of quadrotors to converge and circulate an implicitly defined curve Γ as described in (3.3) without inter-robot collisions.*

Note that to ensure collision avoidance it is necessary to ensure that

$$\|\boldsymbol{\xi}_i(t) - \boldsymbol{\xi}_j(t)\|_2 - (r_i + r_j) > 0, \quad \forall t \geq 0, \quad \forall \omega_i, \omega_j \in \Omega, \omega_i \neq \omega_j. \tag{3.4}$$

In order to solve the stated problem, a cascaded control strategy is proposed, which is depicted in Figure 3.1. First, a vector field based high-level control law $\boldsymbol{u}_i$ is proposed to ensure the coordination of the group of robots Ω assuming dynamics (3.1), providing

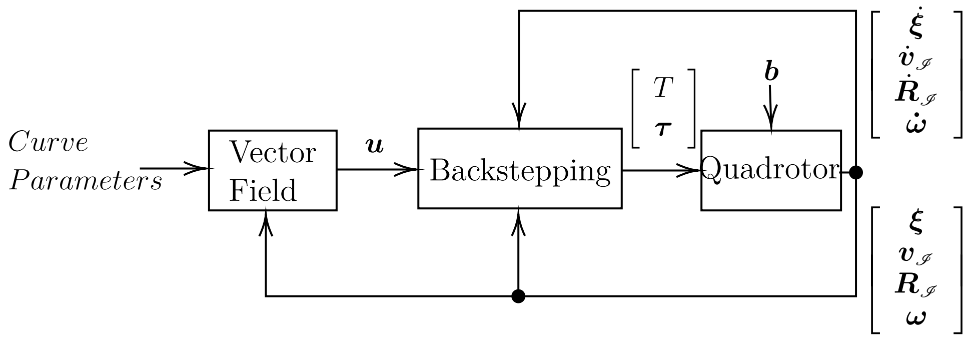

Curve Parameters
Vector Field
$\boldsymbol{u}$
Backstepping
$\begin{bmatrix} T \\ \boldsymbol{\tau} \end{bmatrix}$
$\boldsymbol{b}$
Quadrotor
$\begin{bmatrix} \dot{\boldsymbol{\xi}} \\ \dot{\boldsymbol{v}}_{\mathscr{I}} \\ \dot{\boldsymbol{R}}_{\mathscr{I}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix}$
$\begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{v}_{\mathscr{I}} \\ \boldsymbol{R}_{\mathscr{I}} \\ \boldsymbol{\omega} \end{bmatrix}$

Figure 3.1: Block diagram of the cascaded system.

convergence and circulation to the target curve Γ and collision avoidance. The control input provided by the high-level control law is then used to generate trajectories for each quadrotor, which must be followed by robust low-level controllers. These low-level controllers are designed via backstepping with integral action and robustified with a Lyapunov redesign additional control law. In Figure 3.1, the high-level controller takes as input the parameters of the target curve and the robots positions, and outputs a control signal. This signal serves as input to the low-level controller, that generates the thrust and torques $[T, \boldsymbol{\tau}^T]^T$ to be applied by the quadrotor, subject to disturbances $\boldsymbol{b}$. The controllers depend on the quadrotor state vector $[\boldsymbol{\xi}^T, \boldsymbol{v}_{\mathscr{I}}^T, (vec(\boldsymbol{R}_{\mathscr{I}}))^T, \boldsymbol{\omega}^T]^T$, composed of the position, the linear velocity, the rotation matrix, and the angular velocity, respectively.

3.2 High-Level Control Strategy

The high-level control strategy must guarantee convergence and circulation of a time-varying curve while avoiding inter-robot collisions. To accomplish this behaviour, the proposed vector field based control law results from the sum of four terms. The first two terms are convergence ones that cause the robot to be attracted to the target curve, the third one is a circulation term, orthogonal to the convergence terms, which impels circulation, and the last one is a feedforward term considered in time-varying curves, which allows a robot to follow the evolution of the curve. Then, for each robot ω_i, the guidance control law is given by

$$\mathbf{u}_i = -\mu_i f_{\alpha_2}\left(\nabla\left(\alpha_1^2\right)\right) - \lambda_i f_{\alpha_1}\left(\nabla\left(\alpha_2^2\right)\right) + \varrho_i \nabla\alpha_1 \times \nabla\alpha_2 + \iota_i \boldsymbol{p}, \tag{3.5}$$

in which $\nabla\left(\alpha_k^2\right)$ is the gradient of the square of α_k without considering t, i.e. $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]^T$, and μ_i, λ_i, ϱ_i and ι_i are the modulation functions, which are responsible for regulating the components of the control law to avoid collisions. The control law (3.5) is a combination of the ones proposed in (Pimenta et al., 2013) and (Gonçalves et al., 2010b), as in the former the feedforward term is not considered. More details about each term of

(3.5) can be found on the aforementioned works.

In Equation (3.5), the terms $-\nabla\left(\alpha_1^2\right)$ and $-\nabla\left(\alpha_2^2\right)$ are vectors that point to the surfaces $\alpha_1\left(\boldsymbol{\xi}, t\right)=0$ and $\alpha_2(\boldsymbol{\xi}, t)=0$, respectively. If the two vectors are competitive, i.e., if they tend to cancel each other, the function $f_{\alpha_k}(\mathbf{q})$ removes the competitive component by projecting a vector in the null-space of the other. The function $f_{\alpha_k}(\mathbf{q}): \mathbb{R}^3 \mapsto \mathbb{R}^3$ is defined as

$$f_{\alpha_k}(\mathbf{q})=\begin{cases}\mathbf{q} & \text{if } \left[\nabla\left(\alpha_{k}^2\right)\right]^T\mathbf{q}\geq 0\\ \pi\left(\mathbf{q},\mathcal{N}\left(\left[\nabla\left(\alpha_{k}^2\right)^T\right]\right)\right) & \text{otherwise}\end{cases}, \tag{3.6}$$

in which the operator π computes the orthogonal projection from $\boldsymbol{q}$ onto the null space of $\nabla(\alpha_k^2)^T$.

In addition, the third term of (3.5) has the same direction of the vector $\nabla\alpha_1 \times \nabla\alpha_2$, thus orthogonal to the two first terms and tangent to the surfaces $\alpha_1=0$ and $\alpha_2=0$. As shown by Gonçalves et al. (2010b), the vector corresponding to the feedforward $\boldsymbol{p}$ may be obtained by the matrix operation

$$\boldsymbol{p}=-M^{-1}\boldsymbol{a}. \tag{3.7}$$

The definitions of the matrix M and the vector $\boldsymbol{a}$ are based on M_* and $\boldsymbol{a}_*$, respectively, which are presented next:

Definition 3.1. *(Gonçalves et al., 2010b) Let M_* be a matrix in $\mathbb{R}^{2\times3}$ such that its i-th row is given by the vector $\nabla\alpha_i^T$ for $i=1,2$. Let $\boldsymbol{a}_*$ be a column vector in $\mathbb{R}^2$ such that its i-th row is given by $\partial\alpha_i/\partial t$ for $i=1,2$.*

In order to compute $\boldsymbol{p}$ as described in Equation (3.7), it is necessary to complete M_* so that the matrix becomes square, and in the same step complete $\boldsymbol{a}_*$. This is shown in the next definition.

Definition 3.2. *(Gonçalves et al., 2010b) Let M be a matrix $\mathbb{R}^{3\times3}$ such that the first two rows correspond to the matrix M_* and its last row is the vector $(\nabla\alpha_1\times\nabla\alpha_2)^T$. Let $\boldsymbol{a}$ be the vector in $\mathbb{R}^3$ such that its first two rows correspond to the vector $\boldsymbol{a}_*$ and its last row element is 0.*

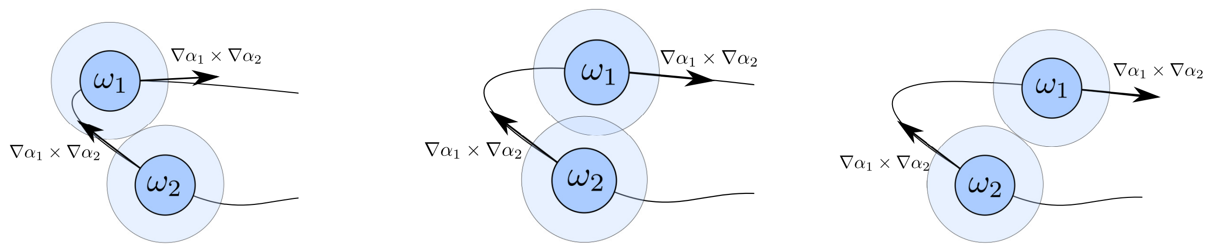
The modulation functions ϱ_i, μ_i, and λ_i in (3.5) are defined as in (Pimenta et al., 2013). The function ι_i is an extension of the previous functions and is used to modulate the feedforward term. These functions are defined as

$$\varrho_i\left(\boldsymbol{\xi}_i,\boldsymbol{\xi}_{\Theta_i}\right):\mathbb{R}^{3(\Upsilon+1)}\mapsto[0,1],$$

$$\mu_i\left(\boldsymbol{\xi}_i,\boldsymbol{\xi}_{\Theta_i}\right):\mathbb{R}^{3(\Upsilon+1)}\mapsto[0,1],$$

$$\lambda_i\left(\boldsymbol{\xi}_i,\boldsymbol{\xi}_{\Theta_i}\right):\mathbb{R}^{3(\Upsilon+1)}\mapsto[0,1],$$

$$\iota_i\left(\boldsymbol{\xi}_i,\boldsymbol{\xi}_{\Theta_i}\right):\mathbb{R}^{3(\Upsilon+1)}\mapsto[0,1],$$



$\nabla\alpha_1 \times \nabla\alpha_2$
ω_1
$\nabla\alpha_1 \times \nabla\alpha_2$
ω_2
$\nabla\alpha_1 \times \nabla\alpha_2$
ω_1
$\nabla\alpha_1 \times \nabla\alpha_2$
ω_2
$\nabla\alpha_1 \times \nabla\alpha_2$
ω_1
$\nabla\alpha_1 \times \nabla\alpha_2$
ω_2

(a) First instant. (b) Second instant. (c) Third instant.

Figure 3.2: Collision avoidance example.

in which $\boldsymbol{\xi}_{\boldsymbol{\Theta}_i}$ is the set of vectors that indicates the positions of the robots in the perception range of ω_i, and Υ is the cardinality of the set of perceived robots $\boldsymbol{\Theta}_i$. The set of robots perceived by ω_i is defined as

$$\boldsymbol{\Theta}_i \triangleq \{\omega_j \in \Omega | \{D_{ij} \leq R_i\}, j \neq i\},$$

with $D_{ij} \triangleq \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|$, that is, the Euclidean distance between robots ω_i and ω_j. Also note that as each function ϱ_i, μ_i and λ_i assumes values in $[0, 1]$, the robots cannot move away from the target curve or alternate the circulation direction.

Some further definitions are necessary to describe the modulation functions.

Definition 3.3. *(Pimenta et al., 2013) For every robot $\omega_i \in \Omega$, the augmented radius $\tilde{r}$ is given by $\tilde{r} \triangleq r_i + \tau_i$, such that τ_i is positive and greater than the largest r_i among the robots.*

In our strategy, robots with different sizes, i.e., heterogeneous robots, might be considered. The augmented radius defines a region around the robot that can be occupied by another robot in a collision risk situation, allowing collision avoidance maneuvers. Note that the augmented radius is the same for all the robots in Ω. Figure 3.2 displays an example in which two robots circulate over a curve following the tangent component of the vector field, where the augmented radius is shown in light blue. In the second instant the augmented radius is invaded, but as the robots keep moving, the intersection between the augmented spheres is reduced.

Definition 3.4. *(Pimenta et al., 2013) The collision distance δ_{ij} between two robots is given by*

$$\delta_{ij} \triangleq \begin{cases} \frac{D_{ij}^2 - (2\tilde{r}+\epsilon)^2}{R_i^2 - (2\tilde{r}+\epsilon)^2} & \textit{if } D_{ij} \geq 2\tilde{r} + \epsilon \\ 0 & \textit{otherwise} \end{cases}, \tag{3.8}$$

in which ϵ is a small positive safety value.

The calculation of the collision distance δ_{ij} is intended to reduce the velocity of a robot as two robots approach each other, if the distance between them is greater or equal $2\tilde{r} + \epsilon$ and is within the perception range. Note that the value resulting from (3.8) tends to be unitary when the distance between ω_i and ω_j approaches the limit of the perception radius.

Definition 3.5. *(Pimenta et al., 2013) The emergency stop function, defined for a pair of robots, is given by*

$$\delta_{ij}^{*} \triangleq \begin{cases} 1 & \textit{if } D_{ij} > (2\tilde{r}+\epsilon) \\ \frac{D_{ij}^2-(2\tilde{r})^2}{(2\tilde{r}+\epsilon)^2-(2\tilde{r})^2} & \textit{if } 0 \leq D_{ij}-2\tilde{r} \leq \epsilon \\ 0 & \textit{otherwise} \end{cases} . \tag{3.9}$$

The emergency stop occurs when the distance between ω_i and ω_j is equal to $2\tilde{r}$, which indicates imminent collision risk. As will be clear later, the emergency stop will allow one robot to stop a given motion direction to ensure collision avoidance.

In situations in which a pair of robots have to maneuver and reduce speed, it is necessary to define movement priorities so that the collision risk situation is solved. In this scenario, it is assumed that a robot is always capable of avoiding a collision by moving in the tangential direction of the current level curve at the instant of time t. This assumption brings the necessity to define the priority to move in the tangent direction.

Definition 3.6. *(Pimenta et al., 2013) The tangential priority Λ_{ij} between a pair of robots is given by*

$$\Lambda_{ij} = \Lambda_{\boldsymbol{\xi}_i\boldsymbol{\xi}_j} \triangleq \left(\boldsymbol{\xi}_i(t)-\boldsymbol{\xi}_j(t)\right)^T \left[\nabla\alpha_1\left(\boldsymbol{\xi}_i(t)\right) \times \nabla\alpha_2\left(\boldsymbol{\xi}_i(t)\right)\right]. \tag{3.10}$$

Regarding convergence, the robot that is closer to the $\alpha_i = 0$ surface is considered the one with higher priority to move in that direction. After defining the priority measurement methods, each robot builds priority sets considering the robots in its perception range, as defined below.

Definition 3.7. *The priority sets concerning each component of the vector field are given by*

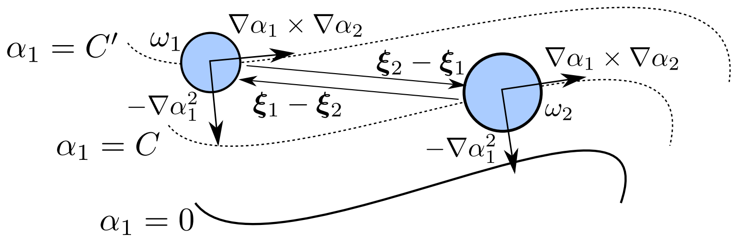
$$\Xi_\mu^i : \{\omega_j \in \Theta_i |\ |\alpha_1\left(\boldsymbol{\xi}_i(t),t\right)| > |\alpha_1\left(\boldsymbol{\xi}_j(t),t\right)|\ \textit{if}\ \alpha_1(\boldsymbol{\xi}_i(t),t) \neq \alpha_1(\boldsymbol{\xi}_i(t),t),\ j<i\ \textit{otherwise}\},$$

$$\Xi_\lambda^i : \{\omega_j \in \Theta_i |\ |\alpha_2\left(\boldsymbol{\xi}_i(t),t\right)| > |\alpha_2\left(\boldsymbol{\xi}_j(t),t\right)|\ \textit{if}\ \alpha_2(\boldsymbol{\xi}_i(t),t) \neq \alpha_2(\boldsymbol{\xi}_i(t),t),\ j<i\ \textit{otherwise}\},$$

$$\Xi_\varrho^i : \{\omega_j \in \Theta_i |\ \Lambda_{ij} < \Lambda_{ji}\ \textit{if}\ \Lambda_{ij} \neq \Lambda_{ji},\ j<i\ \textit{otherwise}\}.$$

Note that in the cases in which the priority measure is equal, the robot with a smaller index is assigned with higher priority.

Figure 3.3 shows a 2-D example, in which the function $\alpha_1 = 0$ defines the curve to be circumnavigated. Consider that the robots are in the perceived range of each other and that they move to the right over their level curves, in which $C' > C$. As ω_1 is farther from the curve $\alpha_1 = 0$ than the robot ω_2, the second has priority to move in the direction of $-\nabla(\alpha_1^2)$, that is, $\Xi_\mu^1 = \{\omega_2\}$ and $\Xi_\mu^2 = \{\ \}$. Regarding the circulation, the distance between the robots increases when ω_2 has tangential priority over ω_1, that is $\Xi_\varrho^1 = \{\omega_2\}$ and $\Xi_\varrho^2 = \{\ \}$.



$\alpha_1 = C'$
ω_1
$\nabla\alpha_1 \times \nabla\alpha_2$
$\boldsymbol{\xi}_2 - \boldsymbol{\xi}_1$
$\nabla\alpha_1 \times \nabla\alpha_2$
$-\nabla\alpha_1^2$
$\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2$
ω_2
$\alpha_1 = C$
$-\nabla\alpha_1^2$
$\alpha_1 = 0$

Figure 3.3: Priority sets example.

Note that for each pair of robots in perception range there exists only one robot with movement priority to move in a certain direction, so that there are no deadlock situations in this sense. Besides that, it is necessary to define the following functions

$$\xi_l^i\left(\Theta_i\right) = \begin{cases} \prod_{\omega_j \in \Xi_l^i} |\delta_{ij}| & \text{if } \Xi_l^i \neq \varnothing \\ 1 & \text{otherwise} \end{cases},$$

$$\beta^i\left(\Theta_i\right) = \begin{cases} \prod_{\omega_j \in \Theta_i} |\delta_{ij}^*| & \text{if } \Theta_i \neq \varnothing \\ 1 & \text{otherwise} \end{cases},$$

in which l stands for μ, λ, or ϱ, and $\prod$ is the product.

Finally, the modulation functions can be evaluated by computing

$$\mu_i = \xi_\mu^i \beta^i, \quad \lambda_i = \xi_\lambda^i \beta^i, \quad \varrho_i = \xi_\varrho^i, \quad \iota_i = \beta^i. \tag{3.11}$$

From (3.11), one can note that the two modulation functions relative to convergence terms take into account the collision distance and the emergency stop function. By the evaluation of the collision distance, a robot stops moving in a certain direction in the presence of collision risk with a higher priority robot. Besides, the emergency stop function causes a robot to stop moving in a certain direction if a critical collision distance is reached, without considering priorities. To provide collision avoidance, a robot stops circulation by computing the collision distance, that takes into account only the robots with higher priority. To avoid undesired possibly dangerous behaviour caused by following the evolution of the time-varying curve, the feedforward term is disregarded in the presence of collision risk. It is important to stress that the collision avoidance scheme is effective assuming that a robot can always avoid collisions by moving in the direction tangent to the current level curve, when both gradient components are set to zero, i.e. $\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2 = 2\tilde{r}$. The satisfaction of this assumption depends on the region in which the robots start, the curvature and torsion of the integral curves $\nabla\alpha_1 \times \nabla\alpha_2$, and the radius of the robots (Pimenta et al., 2013).

To evaluate the convergence of the robots to the curve, the function $W(\mathbf{x}(t), t)$ is used

$$W(\mathbf{x}(t), t) = \sum_{i=1}^{N} V_i\left(\boldsymbol{\xi}_i(t), t\right), \tag{3.12}$$

in which $\mathbf{x}(t) = [\boldsymbol{\xi}_1^T(t) \,...\, \boldsymbol{\xi}_n^T(t)]^T$, and $V_i(\boldsymbol{\xi}_i(t), t) = \alpha_1^2(\boldsymbol{\xi}_i(t), t) + \alpha_2^2(\boldsymbol{\xi}_i(t), t)$. Therefore, $W = 0$ indicates the convergence of the group of robots to the curve.

Next we show a theorem that guarantees that a single robot following the proposed control law will be able to converge and circulate the curve as desired. Thus, if no deadlock situation is verified and the curve is large enough to fit all the robots, the whole group will be able to converge and circulate the target. In order to support the proof of the theorem to be presented, we borrow one lemma from (Pimenta et al., 2013).

Lemma 3.1. *(Pimenta et al., 2013) Vectors $\nabla\alpha_1(\boldsymbol{\xi}, t)$ and $\nabla\alpha_2(\boldsymbol{\xi}, t)$ are linearly independent for every $\boldsymbol{\xi} \in \mathbb{R}^3$ and $t \in \mathbb{R}$ where $\nabla\alpha_1(\boldsymbol{\xi}, t) \neq \mathbf{0}$. Also, if $\nabla\alpha_1(\boldsymbol{\xi}, t) \neq \mathbf{0}$ then $\alpha_1(\boldsymbol{\xi}, t)\nabla\alpha_1(\boldsymbol{\xi}, t) + \alpha_2(\boldsymbol{\xi}, t)\nabla\alpha_2(\boldsymbol{\xi}, t) \neq \mathbf{0}$ for every point $(\boldsymbol{\xi}, t) \notin \Gamma$.*

Proof. From (3.2), we can obtain

$$\frac{\partial}{\partial z}\alpha_2(x, y, z, t) = \frac{\partial}{\partial z}\left[\sigma z - \Phi(x, y, t)\right] = \sigma.$$

As $\sigma \neq 0$, it is impossible to have $\nabla\alpha_2 = \mathbf{0}$. In addition, as α_1 does not depend on z, we know that

$$\frac{\partial}{\partial z}\alpha_1(x, y, t) = 0.$$

Therefore, the vectors $\nabla\alpha_1$ and $\nabla\alpha_2$ are linearly independent, since the z-axis component of α_2 is identically null for all $(\boldsymbol{\xi}, t)$, the same z-axis component of $\nabla\alpha_2$ is non-null for all $(\boldsymbol{\xi}, t)$ and $\nabla\alpha_1(\boldsymbol{\xi}, t) \neq \mathbf{0}$ by hypothesis. Also, for $(\boldsymbol{\xi}, t) \notin \Gamma$

$$\alpha_1(\boldsymbol{\xi}, t)\nabla\alpha_1(\boldsymbol{\xi}, t) + \alpha_2(\boldsymbol{\xi}, t)\nabla\alpha_2(\boldsymbol{\xi}, t) \neq \mathbf{0}.$$

□

Theorem 3.1. *A single robot system with model given by (3.1) under the control input in (3.5) converges to and circulate the target curve as long as singular points are never reached.*

Proof. Consider the following positive semi-definite Lyapunov candidate function

$$V = \alpha_1^2 + \alpha_2^2, \tag{3.13}$$

which time-derivative yields

$$
\begin{aligned}
\dot{V} &= 2\alpha_1 \nabla\alpha_1^T \dot{\boldsymbol{\xi}} + 2\alpha_2 \nabla\alpha_2^T \dot{\boldsymbol{\xi}} + 2\alpha_1 \frac{\partial \alpha_1}{\partial t} + 2\alpha_2 \frac{\partial \alpha_2}{\partial t} \\
\dot{V} &= 2\alpha_1 \nabla\alpha_1^T(-f_{\alpha_2}(\nabla(\alpha_1^2)) - f_{\alpha_1}(\nabla(\alpha_2^2)) + \nabla\alpha_1 \times \nabla\alpha_2 - \boldsymbol{M}^{-1}\boldsymbol{a}) \\
&+ 2\alpha_2 \nabla\alpha_2^T(-f_{\alpha_2}(\nabla(\alpha_1^2)) - f_{\alpha_1}(\nabla(\alpha_2^2)) + \nabla\alpha_1 \times \nabla\alpha_2 - \boldsymbol{M}^{-1}\boldsymbol{a}) \\
&+ 2\alpha_1 \frac{\partial \alpha_1}{\partial t} + 2\alpha_2 \frac{\partial \alpha_2}{\partial t} \\
\dot{V} &= -2\alpha_1 \nabla\alpha_1^T(-f_{\alpha_2}(\nabla(\alpha_1^2))) - 2\alpha_1 \nabla\alpha_1^T(f_{\alpha_1}(\nabla(\alpha_2^2))) - 2\alpha_2 \nabla\alpha_2^T(f_{\alpha_2}(\nabla(\alpha_1^2))) \\
&- 2\alpha_2 \nabla\alpha_2^T(f_{\alpha_1}(\nabla(\alpha_2^2))) + 2\alpha_1 \frac{\partial \alpha_1}{\partial t} + 2\alpha_2 \frac{\partial \alpha_2}{\partial t} - (2\alpha_1 \nabla\alpha_1^T + 2\alpha_2 \nabla\alpha_2^T)(\boldsymbol{M}^{-1}\boldsymbol{a}) \\
\dot{V} &= -2\alpha_1 \nabla\alpha_1^T(-f_{\alpha_2}(\nabla(\alpha_1^2))) - 2\alpha_1 \nabla\alpha_1^T(f_{\alpha_1}(\nabla(\alpha_2^2))) - 2\alpha_2 \nabla\alpha_2^T(f_{\alpha_2}(\nabla(\alpha_1^2))) \\
&- 2\alpha_2 \nabla\alpha_2^T(f_{\alpha_1}(\nabla(\alpha_2^2))) + 2[\alpha_1 \ \alpha_2 \ 0] \cdot \begin{bmatrix} \frac{\partial \alpha_1}{\partial t} \\ \frac{\partial \alpha_2}{\partial t} \\ 0 \end{bmatrix} - 2[\alpha_1 \ \alpha_2 \ 0] \cdot \begin{bmatrix} \nabla\alpha_1^T \\ \nabla\alpha_2^T \\ (\nabla\alpha_1 \times \nabla\alpha_2)^T \end{bmatrix} \boldsymbol{M}^{-1} \begin{bmatrix} \frac{\partial \alpha_1}{\partial t} \\ \frac{\partial \alpha_2}{\partial t} \\ 0 \end{bmatrix} \\
\dot{V} &= -4\alpha_1 \nabla\alpha_1^T f_{\alpha_2}(\alpha_1 \nabla\alpha_1) - 4\alpha_1 \nabla\alpha_1^T f_{\alpha_1}(\alpha_2 \nabla\alpha_2) - 4\alpha_2 \nabla\alpha_2^T f_{\alpha_2}(\alpha_1 \nabla\alpha_1) - 4\alpha_2 \nabla\alpha_2^T f_{\alpha_1}(\alpha_2 \nabla\alpha_2) \\
\dot{V} &\leq 0.
\end{aligned}
$$

From the definition of $f_{\alpha_k}(\boldsymbol{q})$, the projections $\alpha_1 \nabla\alpha_1^T f_{\alpha_1}(\alpha_2 \nabla\alpha_2)$ and $\alpha_2 \nabla\alpha_2^T f_{\alpha_2}(\alpha_1 \nabla\alpha_1)$ are non-negative, and null whenever $[\nabla(\alpha_1^2)]^T \nabla(\alpha_2^2) \leq 0$. As described in Lemma 3.1, $\nabla\alpha_1$ and $\nabla\alpha_2$ are linearly independent. Therefore, the projections $\alpha_1 \nabla\alpha_1^T f_{\alpha_2}(\alpha_1 \nabla\alpha_1)$ and $\alpha_2 \nabla\alpha_2^T f_{\alpha_1}(\alpha_2 \nabla\alpha_2)$ are non-negative, and null if and only if $\alpha_1 = 0$ or $\alpha_2 = 0$, respectively, which yields $\dot{V} \leq 0$.

Note that $\dot{V} = 0$ if and only if $\alpha_1 = 0$ and $\alpha_2 = 0$, i.e. if the robot is over the curve, assuming that singular points are not reached. As $\dot{V} \leq 0$, α_1 and α_2 are bounded, that is $|\alpha_1| \leq C_1$ and $|\alpha_2| \leq C_2$. The second time derivative $\ddot{V}$ is composed of the terms α_1, $\nabla\alpha_1$, $\dot{\alpha}_1$, $\dot{\nabla\alpha_1}$, α_2, $\nabla\alpha_2$, $\dot{\alpha}_2$, $\dot{\nabla\alpha_2}$, which are bounded, since the second partial derivatives are continuous and the domain is bounded. Therefore, $\dot{V}$ is uniformly continuous in time. As V is lower bounded, $\dot{V}$ is negative semi-definite and uniformly continuous in time, then from the Barbalat's "Lyapunov-like Lemma" (Slotine et al., 1991) $\dot{V} \to 0$ as $t \to \infty$, and therefore $\alpha_1 \to 0$ and $\alpha_2 \to 0$.

As $\nabla\alpha_1 \times \nabla\alpha_2$ is tangent to the curve without change in its orientation, circulation is also ensured. □

3.3 Low-Level Control Strategy

In this work, we consider a team of quadrotor UAVs to accomplish the task of convergence and circulation of a given curve. These vehicles have no first-order non-holonomic constraints, which allows better maneuverability in comparison with non-holonomic systems, such as fixed-wing UAVs and differential robots. These UAVs have also the ability to hover, which is essential to follow the high-level control strategy proposed in this chapter. To enforce the vector field to each quadrotor, consider the vector produced by the control law presented in Equation (3.5). As in (Pimenta et al., 2013), the high-level control vector is used to generate a reference trajectory for each quadrotor. The output signal generated by the vector field is used to produce a reference for the low-level controller, which is obtained by computing

$$\begin{aligned}
&\boldsymbol{\xi}_r(t) = \boldsymbol{\xi}(t_0) + \boldsymbol{u}\delta_t, \\
&\dot{\boldsymbol{\xi}}_r = \boldsymbol{u},\ \ddot{\boldsymbol{\xi}}_r = \dddot{\boldsymbol{\xi}}_r = \mathbf{0}_{3,1}, \\
&\psi_r = \dot{\psi}_r = \ddot{\psi}_r = 0,
\end{aligned} \tag{3.14}$$

in which $\boldsymbol{\xi}_r(t) = [x_r(t)\ y_r(t)\ z_r(t)]^T$ is the reference position, ψ_r is the reference yaw angle, $\boldsymbol{u}$ is the high-level control signal, $\delta_t = t - t_0$, and t_0 is the initial time instant. As the control laws are implemented in a discrete manner, we consider that the low-level controller runs much faster than the high-level one and that the vector field is constant between the high-level controller updates, that is, $\ddot{\boldsymbol{\xi}}_r = \dddot{\boldsymbol{\xi}}_r = \mathbf{0}_{3,1}$. In addition, as the velocity of the UAV is bounded, the control signal from the vector field is saturated as

$$\bar{\mathbf{u}}_i = \begin{cases} \mathbf{u}_i, & \text{if } ||\mathbf{u}_i|| \leq v_{\max}, \\ v_{\max}\frac{\mathbf{u}_i}{||\mathbf{u}_i||}, & \text{otherwise}, \end{cases} \tag{3.15}$$

in which $v_{\max}$ is the maximum linear velocity of a vehicle.

For a quadrotor to follow the trajectory references described in Equation (3.14), a robust integral backstepping controller is designed. The controller is based on the integral backstepping approach proposed by Salierno & Raffo (2017), which is a controller capable of tracking time-varying trajectories of position and yaw angle considering the whole-body dynamics, that is, without decoupling between translation and rotation dynamics. This is achieved by using rotation matrices to represent the body rotations in the controller.

The conception of the controller by the backstepping technique allows to constructively design the control law. As shown by Salierno & Raffo (2017), an integral backstepping controller may be designed to control the states x, y, z and ψ (position and yaw angle) of a quadrotor by using four control inputs (T, τ_ϕ, τ_θ, and τ_ψ). The insertion of the integral step brings robustness to the system in front of modeling errors and constant external disturbances.

In this work, besides considering constant disturbances, unknown but bounded disturbances are also taken into account. To deal with this, a control law based on Lyapunov redesign is added to the backstepping process. The goal of the additional control law is to compensate for the unknown but bounded disturbances and to ensure system stability.

In summary, the design of the backstepping controller in this work follows the same steps presented in (Salierno & Raffo, 2017), but this time applying an additional control law. The steps are presented as follows:

- First Step

One of the control objectives is to follow position references. To that end, the translation error is defined as

$$\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\xi}(t) - \boldsymbol{\xi}_r(t). \tag{3.16}$$

To drive the translation error to zero, it is necessary to analyse its dynamics, given by

$$\dot{\boldsymbol{\mathcal{E}}}_1 = \dot{\boldsymbol{\xi}}(t) - \dot{\boldsymbol{\xi}}_r(t). \tag{3.17}$$

Hereafter the temporal dependence will be suppressed.

To stabilize (3.17), the backstepping procedure can be applied. For this purpose, one can choose $\dot{\boldsymbol{\xi}}$ as virtual control input for the system, i.e., $\boldsymbol{\phi}_1(\boldsymbol{\xi}) = (\dot{\boldsymbol{\xi}})_d$, where the subscript d means desired. Therefore, the dynamics (3.17) can be regarded as

$$\dot{\boldsymbol{\mathcal{E}}}_1 = \boldsymbol{\phi}_1(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}_r. \tag{3.18}$$

After that, the following control Lyapunov function is proposed

$$V_1(\boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{\mathcal{E}}_1^T\boldsymbol{\mathcal{E}}_1, \tag{3.19}$$

which time derivative yields

$$\begin{aligned} \dot{V}_1(\boldsymbol{\xi}) &= \boldsymbol{\mathcal{E}}_1^T\left(\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_r\right) \\ &= \boldsymbol{\mathcal{E}}_1^T\left(\boldsymbol{\phi}_1(\boldsymbol{\xi}) - \dot{\boldsymbol{\xi}}_r\right). \end{aligned} \tag{3.20}$$

If the virtual control law $\boldsymbol{\phi}_1(\boldsymbol{\xi})$ is chosen as

$$\boldsymbol{\phi}_1(\boldsymbol{\xi}) = \dot{\boldsymbol{\xi}}_r - \boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1, \tag{3.21}$$

in which $\boldsymbol{k}_1$ is a constant positive definite diagonal matrix ($\boldsymbol{k}_1 > 0$), substituting (3.21) in (3.20) yields to

$$\dot{V}_1 = \boldsymbol{\mathcal{E}}_1^T\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1\right) = -\boldsymbol{\mathcal{E}}_1^T\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 < 0, \tag{3.22}$$

which is negative definite. As in the standard backstepping procedure, a system equivalent to (3.17) can be obtained summing and subtracting (3.21), as follows

$$\begin{aligned} \dot{\boldsymbol{\mathcal{E}}}_1 &= -\dot{\boldsymbol{\xi}}_r + \boldsymbol{\phi}_1(\boldsymbol{\xi}) + \left[\dot{\boldsymbol{\xi}} - \boldsymbol{\phi}_1(\boldsymbol{\xi})\right] \\ &= -\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \left[\dot{\boldsymbol{\xi}} - \boldsymbol{\phi}_1(\boldsymbol{\xi})\right]. \end{aligned} \tag{3.23}$$

In (3.23), the difference between $\dot{\boldsymbol{\xi}}$ and $\boldsymbol{\phi}_1(\boldsymbol{\xi})$ is highlighted. This difference must be zero to stabilize (3.17). Therefore, the new state vector $\boldsymbol{z}_1 = \left[\dot{\boldsymbol{\xi}} - \boldsymbol{\phi}_1(\boldsymbol{\xi})\right]$ is included into the system and its dynamics derived from (3.21) and (3.23), as follows

$$\begin{aligned} \dot{\boldsymbol{\mathcal{E}}}_1 &= -\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1, \\ \dot{\boldsymbol{z}}_1 &= \dot{\boldsymbol{v}}_{\mathscr{I}} - \ddot{\boldsymbol{\xi}}_r + \boldsymbol{k}_1\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1\right), \end{aligned} \tag{3.24}$$

in which the acceleration vector $\dot{\boldsymbol{v}}_{\mathscr{I}}$ comes from the aircraft model.

- Second step

To stabilize $\boldsymbol{z}_1$ in the origin, the error between the real and desired velocities $\dot{\boldsymbol{\xi}} - \boldsymbol{\phi}_1(\boldsymbol{\xi})$ is evaluated. In this step, the integral action $\mathcal{X}_{\boldsymbol{\xi}} = \int_0^t \boldsymbol{z}_1(\tau)d\tau$ is included in the system. This action guarantees the convergence to time-varying references, besides guaranteeing the rejection of constant disturbances (Skjetne & Fossen, 2004; Raffo et al., 2015).

By substituting the equations that describe the UAV dynamics (2.1) and by inserting the integral term, system (3.24) is rewritten as

$$\begin{aligned} \dot{\boldsymbol{\mathcal{E}}}_1 &= -\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1, \\ \dot{\mathcal{X}}_{\boldsymbol{\xi}} &= \boldsymbol{z}_1, \\ \dot{\boldsymbol{z}}_1 &= -g\boldsymbol{e}_3 + \frac{1}{m}\boldsymbol{R}\boldsymbol{e}_3 T + \frac{\boldsymbol{b}}{m} - \ddot{\boldsymbol{\xi}}_r + \boldsymbol{k}_1\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1\right). \end{aligned} \tag{3.25}$$

In this step, the term $(\boldsymbol{R}\boldsymbol{e}_3T)_d = \boldsymbol{\phi}_2(\boldsymbol{R}, T)$ can be chosen as virtual control input. To find $\boldsymbol{\phi}_2(\boldsymbol{R}, T)$, consider the following control Lyapunov function

$$V_2\left(\boldsymbol{\mathcal{E}}_1, \boldsymbol{z}_1, \mathcal{X}_{\boldsymbol{\xi}}\right) = V_1 + \frac{1}{2}\boldsymbol{z}_1^T\boldsymbol{z}_1 + \frac{1}{2}\mathcal{X}_{\boldsymbol{\xi}}^T\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\mathcal{X}_{\boldsymbol{\xi}}, \tag{3.26}$$

in which $\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}$ is a positive definite diagonal matrix ($\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}} > 0$). The time derivative $\dot{V}_2\left(\boldsymbol{\mathcal{E}}_1, \boldsymbol{z}_1, \mathcal{X}_{\boldsymbol{\xi}}\right)$ is given by

$$\dot{V}_2 = \boldsymbol{\mathcal{E}}_1^T\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1\right) + \boldsymbol{z}_1^T\left(-g\boldsymbol{e}_3 + \frac{1}{m}\boldsymbol{\phi}_2 + \frac{\boldsymbol{b}}{m} - \ddot{\boldsymbol{\xi}}_r + \boldsymbol{k}_1\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1\right)\right) + \mathcal{X}_{\boldsymbol{\xi}}^T\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\boldsymbol{z}_1. \tag{3.27}$$

Therefore, the virtual control law $\boldsymbol{\phi}_2(\boldsymbol{R}, T)$ is chosen as

$$\boldsymbol{\phi}_2(\boldsymbol{R}, T) = m\Big(g\boldsymbol{e}_3 + \ddot{\boldsymbol{\xi}}_r - \boldsymbol{k}_1\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1\right) - \boldsymbol{\mathcal{E}}_1 - \boldsymbol{k}_2\boldsymbol{z}_1 - \boldsymbol{k}_{\chi\boldsymbol{\xi}}\mathcal{X}_{\boldsymbol{\xi}}\Big) + \boldsymbol{\gamma}, \tag{3.28}$$

with $\boldsymbol{k}_2 = \boldsymbol{k}_2^T > 0$, $\boldsymbol{k}_{\chi\boldsymbol{\xi}} = \boldsymbol{k}_{\chi\boldsymbol{\xi}}^T > 0$, and $\boldsymbol{\gamma} := \boldsymbol{\gamma}(\boldsymbol{z}_1)$ being an additional control law to robustify the system in front of unknown but bounded disturbances. The choice of such additional control law will be discussed later in this chapter.

Substituting (3.28) in (3.27) yields to

$$\dot{V}_2 = -\boldsymbol{\mathcal{E}}_1^T \boldsymbol{k}_1 \boldsymbol{\mathcal{E}}_1 - \boldsymbol{z}_1^T \boldsymbol{k}_2 \boldsymbol{z}_1 + \boldsymbol{z}_1^T \left(\frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m} \right). \tag{3.29}$$

Thus, by choosing an appropriate $\boldsymbol{\gamma}(\boldsymbol{z}_1)$ function the system can be stabilized (Levine, 2010). Remember that as shown in (Salierno, 2018), the states $\boldsymbol{\mathcal{E}}_1$ and $\boldsymbol{z}_1$ converge to zero if there are no time-varying unknown disturbances, while $\mathcal{X}_{\boldsymbol{\xi}}$ converges to a value (Skjetne & Fossen, 2004; Loria et al., 2002).

By summing and subtracting $\frac{1}{m}\boldsymbol{\phi}_2$ from (3.25), one can obtain

$$\begin{aligned}
\dot{\boldsymbol{\mathcal{E}}}_1 &= -\boldsymbol{k}_1 \boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1, \\
\dot{\mathcal{X}}_{\boldsymbol{\xi}} &= \boldsymbol{z}_1, \\
\dot{\boldsymbol{z}}_1 &= -\boldsymbol{\mathcal{E}}_1 - \boldsymbol{k}_2 \boldsymbol{z}_1 - \boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}} \mathcal{X}_{\boldsymbol{\xi}} + \frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m} + \frac{1}{m} \left[\boldsymbol{R} \boldsymbol{e}_3 T - \boldsymbol{\phi}_2 \right],
\end{aligned} \tag{3.30}$$

in which the change of variables $\boldsymbol{z}_2 = [\boldsymbol{R}\boldsymbol{e}_3 T - \boldsymbol{\phi}_2]$ is made.

- Third step

In this step, it is desired to stabilize $\boldsymbol{z}_2$ in zero, that is, to drive the difference between $\boldsymbol{R}\boldsymbol{e}_3 T$ and $\boldsymbol{\phi}_2 = (\boldsymbol{R}\boldsymbol{e}_3 T)_d$ to zero. By computing $\dot{\boldsymbol{z}}_2$, one can obtain

$$\begin{aligned}
\dot{\boldsymbol{z}}_2 &= d\frac{\boldsymbol{R}\boldsymbol{e}_3 T}{dt} - d\frac{\boldsymbol{\phi}_2}{dt} \\
&= \boldsymbol{R}\boldsymbol{e}_3 \dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T - m \dddot{\boldsymbol{\xi}}_r + m\boldsymbol{k}_1 \ddot{\boldsymbol{\mathcal{E}}}_1 + m\dot{\boldsymbol{\mathcal{E}}}_1 + m\boldsymbol{k}_2 \dot{\boldsymbol{z}}_1 + m\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}} \boldsymbol{z}_1 - \dot{\boldsymbol{\gamma}}.
\end{aligned} \tag{3.31}$$

After that, the term $(\boldsymbol{R}\boldsymbol{e}_3 \dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T)_d = \boldsymbol{\phi}_3(\boldsymbol{R}, T, \dot{T}, \boldsymbol{\omega})$ is chosen as virtual control input. The following control Lyapunov function is proposed

$$V_3 \left(\boldsymbol{\mathcal{E}}_1, \boldsymbol{z}_1, \mathcal{X}_{\boldsymbol{\xi}}, \boldsymbol{z}_2 \right) = V_2 + \frac{1}{2} \boldsymbol{z}_2^T \boldsymbol{z}_2, \tag{3.32}$$

which time derivative yields

$$\begin{aligned}
\dot{V}_3 = {} & \boldsymbol{\mathcal{E}}_1 \left(-\boldsymbol{k}_1 \boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1 \right) + \boldsymbol{z}_1^T \left(-\boldsymbol{\mathcal{E}}_1 - \boldsymbol{k}_2 \boldsymbol{z}_1 - k_{\mathcal{X}\boldsymbol{\xi}} X_{\boldsymbol{\xi}} + \frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m} + \frac{1}{m} \boldsymbol{z}_2 \right) + \mathcal{X}_{\boldsymbol{\xi}}^T \boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}} \boldsymbol{z}_1 \\
& + \boldsymbol{z}_2^T \left(\boldsymbol{\phi}_3 - m \dddot{\boldsymbol{\xi}}_r + m\boldsymbol{k}_1 \ddot{\boldsymbol{\mathcal{E}}}_1 + m \left(-\boldsymbol{k}_1 \boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1 \right) + m\boldsymbol{k}_2 \dot{\boldsymbol{z}}_1 + m k_{x\boldsymbol{\xi}} \boldsymbol{z}_1 - \dot{\boldsymbol{\gamma}} \right).
\end{aligned} \tag{3.33}$$

Thus, the virtual control law can be chosen as

$$\boldsymbol{\phi}_3 = m \dddot{\boldsymbol{\xi}}_r - m\boldsymbol{k}_1 \ddot{\boldsymbol{\mathcal{E}}}_1 - m \left(-\boldsymbol{k}_1 \boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1 \right) - m\boldsymbol{k}_2 \dot{\boldsymbol{z}}_1 - m\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}} \boldsymbol{z}_1 - \frac{1}{m} \boldsymbol{z}_1 - \boldsymbol{k}_3 \boldsymbol{z}_2 + \dot{\boldsymbol{\gamma}}, \tag{3.34}$$

with $\boldsymbol{k}_3 = \boldsymbol{k}_3^T > 0$. By substituting (3.34) in (3.33) one can obtain

$$\dot{V}_3 = -\boldsymbol{\mathcal{E}}_1^T \boldsymbol{k}_1 \boldsymbol{\mathcal{E}}_1 - \boldsymbol{z}_1^T \boldsymbol{k}_2 \boldsymbol{z}_1 - \boldsymbol{z}_2^T \boldsymbol{k}_3 \boldsymbol{z}_2 + \boldsymbol{z}_1^T \left(\frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m} \right), \tag{3.35}$$

which is negative definite if an appropriate additional control law $\boldsymbol{\gamma}(\boldsymbol{z}_1)$ is applied.

Thereafter, summing and subtracting (3.34) from (3.31) yields to

$$\dot{\boldsymbol{z}}_2 = -\frac{1}{m}\boldsymbol{z}_1 - \boldsymbol{k}_3\boldsymbol{z}_2 + \left[\boldsymbol{R}\boldsymbol{e}_3\dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T - \boldsymbol{\phi}_3\right], \tag{3.36}$$

in which one can change the variable $\boldsymbol{z}_3 = [\boldsymbol{R}\boldsymbol{e}_3\dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T - \boldsymbol{\phi}_3]$, as in the previous steps.

- Fourth step

The difference $(\boldsymbol{R}\boldsymbol{e}_3\dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{w})\boldsymbol{e}_3 T - \boldsymbol{\phi}_3)$ must converge to zero for the convergence of $\boldsymbol{z}_2$. The dynamics of $\boldsymbol{z}_3$ are given by

$$\begin{aligned}\dot{\boldsymbol{z}}_3 &= d\frac{\boldsymbol{R}\boldsymbol{e}_3\dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T}{dt} - d\frac{\boldsymbol{\phi}_3}{dt}\\ &= \boldsymbol{R}\boldsymbol{e}_3\ddot{T} - \boldsymbol{R}\boldsymbol{S}\left(\boldsymbol{e}_3\right)\dot{\boldsymbol{\omega}}T + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T + 2\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3\dot{T} - m\ddddot{\boldsymbol{\xi}}_r + m\boldsymbol{k}_1\dddot{\boldsymbol{\mathcal{E}}}_1 + m\ddot{\mathcal{E}}_1\\ &\quad + m\boldsymbol{k}_2\ddot{\boldsymbol{z}}_1 + m\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\dot{\boldsymbol{z}}_1 + \frac{1}{m}\dot{\boldsymbol{z}}_1 + \boldsymbol{k}_3\dot{\boldsymbol{z}}_2 - \ddot{\boldsymbol{\gamma}}.\end{aligned} \tag{3.37}$$

In this case, the term $\left(\boldsymbol{R}\boldsymbol{e}_3\ddot{T} - \boldsymbol{R}S\left(\boldsymbol{e}_3\right)\dot{\boldsymbol{\omega}}T\right)$ from Equation (3.37) can be simplified by substituting $\dot{\boldsymbol{\omega}} = [\dot{p}\;\dot{q}\;\dot{r}]^T$ and by applying the properties of the skew-symmetric matrices (Spong et al., 2006), which leads to

$$\boldsymbol{R}\boldsymbol{e}_3\ddot{T} - \boldsymbol{R}S\left(\boldsymbol{e}_3\right)\dot{\boldsymbol{\omega}}T = \boldsymbol{R}\begin{bmatrix}0 & T & 0\\ -T & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \ddot{T}\end{bmatrix}. \tag{3.38}$$

Substituting (3.38) in (3.37) yields to

$$\begin{aligned}\dot{\boldsymbol{z}}_3 = \boldsymbol{R}&\begin{bmatrix}0 & T & 0\\ -T & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}\dot{p}\\ \dot{q}\\ \ddot{T}\end{bmatrix} + 2\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3\dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T - m\ddddot{\boldsymbol{\xi}}_r + m\boldsymbol{k}_1\dddot{\boldsymbol{\mathcal{E}}}_1 + m\ddot{\boldsymbol{\mathcal{E}}}_1\\ &+ m\boldsymbol{k}_2\ddot{\boldsymbol{z}}_1 + m\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\dot{\boldsymbol{z}}_1 + \frac{1}{m}\dot{\boldsymbol{z}}_1 + \boldsymbol{k}_3\dot{\boldsymbol{z}}_2 - \ddot{\boldsymbol{\gamma}}.\end{aligned} \tag{3.39}$$

From Equation (3.39), one can choose $\boldsymbol{\phi}_4 = ([\dot{p} \quad \dot{q} \quad \ddot{T}]^T)_d$ as virtual control input. Consider the control Lyapunov function

$$V_4(\boldsymbol{\mathcal{E}}_1, \boldsymbol{z}_1, \mathcal{X}_{\boldsymbol{\xi}}, \boldsymbol{z}_2, \boldsymbol{z}_3) = V_3 + \frac{1}{2}\boldsymbol{z}_3^T\boldsymbol{z}_3, \tag{3.40}$$

which time derivative is given by

$$
\begin{aligned}
\dot{V}_4 = {} & \boldsymbol{\mathcal{E}}_1\left(-\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1+\boldsymbol{z}_1\right)+\boldsymbol{z}_1^T\left(-\boldsymbol{\mathcal{E}}_1-\boldsymbol{k}_2\boldsymbol{z}_1-k_{\mathcal{X}_{\boldsymbol{\xi}}}\mathcal{X}_{\boldsymbol{\xi}}+\frac{\boldsymbol{b}}{m}+\frac{\gamma}{m}+\frac{1}{m}\boldsymbol{z}_2\right)+\mathcal{X}_{\boldsymbol{\xi}}^T\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\boldsymbol{z}_1+\boldsymbol{z}_2^T\left(-\frac{1}{m}\boldsymbol{z}_1\right. \\
& \left.-\boldsymbol{k}_3\boldsymbol{z}_2+\boldsymbol{z}_3\right)+\boldsymbol{z}_3^T\left(\boldsymbol{R}\begin{bmatrix}0 & T & 0\\ -T & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \ddot{T}\end{bmatrix}+2\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3\dot{T}+\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3T-m\dddot{\boldsymbol{\xi}}_r+m\boldsymbol{k}_1\ddot{\boldsymbol{\mathcal{E}}}_1\right. \\
& \left.+m\ddot{\boldsymbol{\mathcal{E}}}_1+m\boldsymbol{k}_2\ddot{\boldsymbol{z}}_1+m\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\dot{\boldsymbol{z}}_1+\frac{1}{m}\dot{\boldsymbol{z}}_1+\boldsymbol{k}_3\dot{\boldsymbol{z}}_2-\ddot{\gamma}\right).
\end{aligned}
\tag{3.41}
$$

In order to cancel the undesired terms, the virtual control input $\boldsymbol{\phi}_4$ is chosen as

$$
\begin{aligned}
\boldsymbol{\phi}_4 = {} & \begin{bmatrix}0 & -\frac{1}{T} & 0\\ \frac{1}{T} & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\boldsymbol{R}^T\left(-2\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3\dot{T}-\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3T+m\dddot{\boldsymbol{\xi}}_r-m\boldsymbol{k}_1\ddot{\boldsymbol{\mathcal{E}}}_1-m\ddot{\boldsymbol{\mathcal{E}}}_1\right. \\
& \left.-m\boldsymbol{k}_2\ddot{\boldsymbol{z}}_1-m\boldsymbol{k}_{\mathcal{X}_{\boldsymbol{\xi}}}\dot{\boldsymbol{z}}_1-\frac{1}{m}\dot{\boldsymbol{z}}_1-\boldsymbol{k}_3\dot{\boldsymbol{z}}_2-\boldsymbol{z}_2-\boldsymbol{k}_4\boldsymbol{z}_3+\ddot{\gamma}\right).
\end{aligned}
\tag{3.42}
$$

By substituting (3.42) in (3.41), the time derivative of the control Lyapunov function results in

$$
\dot{V}_4=-\boldsymbol{\mathcal{E}}_1^T\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1-\boldsymbol{z}_1^T\boldsymbol{k}_2\boldsymbol{z}_1-\boldsymbol{z}_2^T\boldsymbol{k}_3\boldsymbol{z}_2-\boldsymbol{z}_3^T\boldsymbol{k}_4\boldsymbol{z}_3+\boldsymbol{z}_1^T\left(\frac{\boldsymbol{b}}{m}+\frac{\gamma}{m}\right). \tag{3.43}
$$

Note that $\boldsymbol{\phi}_4$ is composed of elements that can be transformed in inputs of the real system. By substituting the control law $\boldsymbol{\phi}_4$ in Equation (3.39), it is possible to simplify the dynamics of $\boldsymbol{z}_3$ in the form of

$$
\dot{\boldsymbol{z}}_3=-\boldsymbol{z}_2-\boldsymbol{k}_4\boldsymbol{z}_3. \tag{3.44}
$$

- Fifth step

To follow the yaw angle reference ψ_r, consider the difference between real yaw angle and its reference, given by

$$
\mathcal{E}_\psi=\psi-\psi_r, \tag{3.45}
$$

which time derivative leads to

$$
\dot{\mathcal{E}}_\psi=\dot{\psi}-\dot{\psi}_r. \tag{3.46}
$$

By choosing $(\dot{\psi})_d=\phi_5\left(\mathcal{E}_\psi\right)$ as virtual control input, it is possible to design ϕ_5 from the control Lyapunov function

$$
V_5\left(\mathcal{E}_\psi\right)=\frac{1}{2}\mathcal{E}_\psi^T\mathcal{E}_\psi. \tag{3.47}
$$

By differentiating $V_5\left(\mathcal{E}_\psi\right)$, it is easy to find that

$$
\dot{V}_5=\mathcal{E}_\psi^T\left(\phi_5-\dot{\psi}_r\right). \tag{3.48}
$$

Subsequently, to guarantee the asymptotic convergence of $\mathcal{E}_\psi$ to zero, the virtual control input ϕ_5 is chosen as

$$\phi_5\left(\mathcal{E}_\psi\right) = \dot{\psi}_r - k_5\mathcal{E}_\psi, \tag{3.49}$$

with $k_5 > 0$. Note that in this case the constant gain k_5 has unitary dimension.

Substituting the virtual control input (3.49) in (3.48) yields to

$$\dot{V}_5 = \mathcal{E}_\psi^T\left(-k_5\mathcal{E}_\psi\right) = -\mathcal{E}_\psi^T k_5\mathcal{E}_\psi < 0, \tag{3.50}$$

which is negative definite. By summing and subtracting the virtual control law ϕ_5 in the yaw error dynamics $\dot{\mathcal{E}}_\psi$, it follows that

$$\dot{\mathcal{E}}_\psi = -\dot{\psi}_r + \phi_5 + \left[\dot{\psi} - \phi_5\right], \tag{3.51}$$

in which the change of variables $z_4 = [\dot{\psi} - \phi_5]$ is applied.

- Sixth step

In this step the objective is to stabilize the error related to the derivative of the yaw angle error. An integral term in the form of $\mathcal{X}_\psi = \int_0^t z_4(\tau)d\tau$ is also added in order to ensure tracking of time-varying trajectories and constant disturbance rejection. Like this, the system can be written as

$$\begin{aligned}\dot{\mathcal{E}}_\psi &= -k_5\mathcal{E}_\psi + z_4\\ \dot{\mathcal{X}}_\psi &= z_4\\ \dot{z}_4 &= \ddot{\psi} - \ddot{\psi}_r + k_5\dot{\mathcal{E}}_\psi,\end{aligned} \tag{3.52}$$

in which $(\ddot{\psi})_d = \phi_6$ is a control input.

Consider the following control Lyapunov function

$$V_6\left(\mathcal{E}_\psi, \mathcal{X}_\psi, z_4\right) = V_5 + \frac{1}{2}z_4^T z_4 + \frac{1}{2}\mathcal{X}_\psi^T k_{\mathcal{X}_\psi}\mathcal{X}_\psi, \tag{3.53}$$

which time derivative results in

$$\dot{V}_6 = \mathcal{E}_\psi^T\left(-k_5\mathcal{E}_\psi + z_4\right) + z_4^T\left(\phi_6 - \ddot{\psi}_r + k_5(-k_5\mathcal{E}_\psi + z_4)\right) + \mathcal{X}_\psi^T k_{\mathcal{X}_\psi} z_4, \tag{3.54}$$

where $k_{\mathcal{X}\psi} > 0$. By designing ϕ_6 from the time derivative of the Lyapunov candidate function (3.54), one can obtain

$$\phi_6 = \ddot{\psi}_r - k_5(-k_5\mathcal{E}_\psi + z_4) - k_6 z_4 - \mathcal{E}_\psi - k_{\mathcal{X}\psi}\mathcal{X}_\psi, \tag{3.55}$$

in which $k_6 > 0$. The substitution of (3.55) in (3.54) yields

$$\dot{V}_6 = -\mathcal{E}_\psi^T k_5 \mathcal{E}_\psi - z_4^T k_6 z_4 \leq 0. \tag{3.56}$$

As shown by Salierno (2018), the origin $(\mathcal{E}_\psi, z_4) = (0,0)$ is uniformly globally asymptotically stable, while $\mathcal{X}_\psi$ remains with a residual value.

To obtain the quadrotor closed-loop dynamics it is necessary to map the desired yaw angle acceleration $\phi_6 = (\ddot{\psi})_d$ to angular acceleration $\dot{r}_d$. As shown in (Salierno & Raffo, 2017), assuming that the roll and pitch angles are always in the interval $(-\pi/2, \pi/2)$, one can obtain

$$\ddot{\psi} = -[0\ 0\ 1]\boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\dot{\boldsymbol{W}}_{\boldsymbol{\eta}}\boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\boldsymbol{\omega} + \dot{q}\frac{\sin(\phi)}{\cos(\theta)} + \dot{r}\frac{\cos(\phi)}{\cos(\theta)}, \tag{3.57}$$

in which $\boldsymbol{W}_{\boldsymbol{\eta}}$ is the Euler matrix.

Assuming $\ddot{\psi} = (\ddot{\psi})_d$ and substituting ϕ_6, it follows that

$$\dot{r}_d = \frac{\cos(\theta)}{\cos(\phi)}\left(\ddot{\psi}_r - k_5(-k_5\mathcal{E}_\psi + z_4) - k_6 z_4 - \mathcal{E}_\psi - k_{\mathcal{X}\psi}\mathcal{X}_\psi + [0\ 0\ 1]\boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\dot{\boldsymbol{W}}_{\boldsymbol{\eta}}\boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\boldsymbol{\omega} - \dot{q}_d\frac{\sin(\phi)}{\cos(\theta)}\right). \tag{3.58}$$

Finally, the closed-loop system dynamics is described by the following set of equations

$$\begin{aligned}
\dot{\boldsymbol{\mathcal{E}}}_1 &= -\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 + \boldsymbol{z}_1\\
\dot{\mathcal{X}}_{\boldsymbol{\xi}} &= \boldsymbol{z}_1\\
\dot{\boldsymbol{z}}_1 &= -\boldsymbol{\mathcal{E}}_1 - \boldsymbol{k}_2\boldsymbol{z}_1 - \boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\mathcal{X}_{\boldsymbol{z}_1} + \frac{1}{m}\boldsymbol{z}_2 + \frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m}\\
\dot{\boldsymbol{z}}_2 &= -\frac{1}{m}\boldsymbol{z}_1 - \boldsymbol{k}_3\boldsymbol{z}_2 + \boldsymbol{z}_3\\
\dot{\boldsymbol{z}}_3 &= \boldsymbol{R}\begin{bmatrix} 0 & T & 0\\ -T & 0 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{p}\\ \dot{q}\\ \ddot{T}\end{bmatrix}_d + 2\boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3\dot{T} + \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{e}_3 T - m\ddddot{\boldsymbol{\xi}}_r + m\boldsymbol{k}_1\dddot{\boldsymbol{\mathcal{E}}}_1 + m\ddot{\boldsymbol{\mathcal{E}}}_1\\
&\quad + m\boldsymbol{k}_2\ddot{\boldsymbol{z}}_1 + m\boldsymbol{k}_{\mathcal{X}\boldsymbol{\xi}}\dot{\boldsymbol{z}}_1 + \frac{1}{m}\dot{\boldsymbol{z}}_1 + \boldsymbol{k}_3\dot{\boldsymbol{z}}_2 - \ddot{\boldsymbol{\gamma}}\\
\dot{\mathcal{E}}_\psi &= -k_5\mathcal{E}_\psi + z_4\\
\dot{\mathcal{X}}_\psi &= z_4\\
\dot{z}_4 &= -[0\ 0\ 1]\boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\boldsymbol{W}_{\boldsymbol{\eta}}\boldsymbol{W}_{\boldsymbol{\eta}}^{-1}\boldsymbol{\omega} + \dot{q}\frac{\sin(\phi)}{\cos(\theta)} + \dot{r}_d\frac{\cos(\phi)}{\cos(\theta)} - \ddot{\psi}_r + k_5\dot{\mathcal{E}}_\psi
\end{aligned} \tag{3.59}$$

Remark 3.2. *In this work, all the states are available and can be measured or estimated. In addition, in a practical implementation, the linear and angular positions, velocities and accelerations are measurable variables, with appropriate sensors, and the jerk* $\dddot{\boldsymbol{\xi}}$ *can be estimated based on the model and on the measures above.*

From the desired control laws, the input signals to be applied to the system are obtained by computing

$$\boldsymbol{\tau}_a = I\dot{\boldsymbol{\omega}}_d + \boldsymbol{\omega} \times I\boldsymbol{\omega}, \tag{3.60}$$

$$T = \int \left(\int \ddot{T}_d dt \right) dt, \tag{3.61}$$

in which $\boldsymbol{\omega} = [p \; q \; r]^T$ is measured. The desired inputs for (3.60) and (3.61) are computed with (3.42) and (3.58), which give $[\dot{p}_d \; \dot{q}_d \; \ddot{T}_d]^T$ and $\dot{r}_d$, respectively.

To analyze the closed-loop system stability, the sum of all the control Lyapunov functions chosen during the backstepping process is considered:

$$V = \frac{1}{2}\boldsymbol{\mathcal{E}}_1^T\boldsymbol{\mathcal{E}}_1 + \frac{1}{2}\boldsymbol{z}_1^T\boldsymbol{z}_1 + \frac{1}{2}\mathcal{X}_{\boldsymbol{\xi}}^T k_{\mathcal{X}_{\boldsymbol{\xi}}}\mathcal{X}_{\boldsymbol{\xi}} + \frac{1}{2}\boldsymbol{z}_2^T\boldsymbol{z}_2 + \frac{1}{2}\boldsymbol{z}_3^T\boldsymbol{z}_3 + \frac{1}{2}\mathcal{E}_\psi^T\mathcal{E}_\psi + \frac{1}{2}z_4^T z_4 + \frac{1}{2}\mathcal{X}_\psi k_{\mathcal{X}_\psi}\mathcal{X}_\psi. \tag{3.62}$$

By differentiating V and applying the designed control laws, it follows that

$$\dot{V} = -\boldsymbol{\mathcal{E}}_1^T\boldsymbol{k}_1\boldsymbol{\mathcal{E}}_1 - \boldsymbol{z}_1^T\boldsymbol{k}_2\boldsymbol{z}_1 - \boldsymbol{z}_2^T\boldsymbol{k}_3\boldsymbol{z}_2 - \boldsymbol{z}_3^T\boldsymbol{k}_4\boldsymbol{z}_3 - k_5\mathcal{E}_\psi^T\mathcal{E}_\psi - k_6 z_4^T z_4 + \boldsymbol{z}_1^T\left(\frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m}\right). \tag{3.63}$$

As $\boldsymbol{k}_i > 0$ for $i = 1, \cdots, 6$, the following inequality must be ensured to stabilize the system

$$\boldsymbol{z}_1^T\left(\frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m}\right) \leq 0. \tag{3.64}$$

Consider that the disturbance $\boldsymbol{b}$ is bounded with known bounds $|b_i| \leq \varpi$, for $i = 1, 2, 3$. The largest value of $\boldsymbol{z}_1^T\boldsymbol{b}$ occurs when $\boldsymbol{z}_1^T$ and $\boldsymbol{b}$ have the same signal and $\boldsymbol{b}$ has maximum magnitude ϖ in all its elements. Hence, applying the Lyapunov redesign technique, the additional control law is used to reject the term $\boldsymbol{z}_1^T\boldsymbol{b}$ in the worst case, given by

$$\boldsymbol{\gamma}(\boldsymbol{z}_1) = -\zeta \cdot \mathrm{sgn}(\boldsymbol{z}_1), \tag{3.65}$$

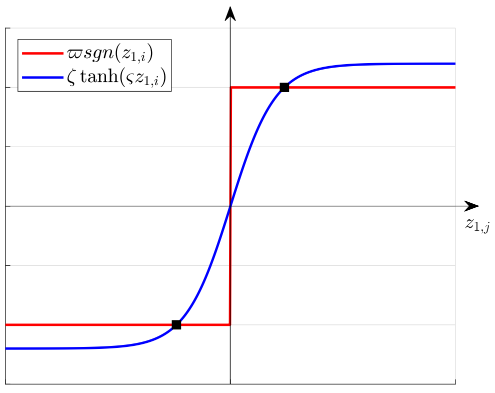
in which $\mathrm{sgn}(\cdot)$ is the element-wise signal function, and ζ is a constant such that $\zeta \geq \varpi$.

In spite of rendering the inequality (3.64) true for all value of $\boldsymbol{z}_1$, the additional control law (3.65) is not continuous and can generate chattering behavior (Khalil, 2002). In addition, its derivatives are not continuous, what brings impossible the computation of the virtual control laws that depend on the time derivatives of $\boldsymbol{\gamma}(\boldsymbol{z}_1)$. Therefore, it makes sense to choose the following continuous approximation of the signal function

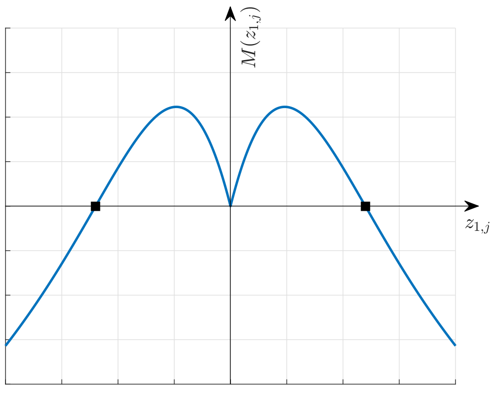
$$\boldsymbol{\gamma}(\boldsymbol{z}_1) = -\zeta \cdot \tanh(\varsigma\boldsymbol{z}_1), \tag{3.66}$$

in which ς is a positive constant that allows to adjust the slope of the hyperbolic tangent function, and $\tanh(\cdot)$ is the element-wise hyperbolic tangent function. Note that the higher the value of ς, the closer the function gets to the signal function.

As the hyperbolic tangent function is bounded between 1 and -1, it is strictly necessary to consider $\zeta > \varpi$ to reject all the disturbances, once that the maximum and the minimum



(a) Signal versus hyperbolic tangent function.



(b) Plot of $M(z_{1,i})$.

Figure 3.4: Lyapunov redesign: hyperbolic tangent additional law.

occurs only when the argument tends to $+\infty$ or $-\infty$, respectively. This way, ζ must be chosen to cancel the maximum magnitude of the disturbance by considering the magnitude of $\varsigma \boldsymbol{z}_1$ finite. Figure 3.4a shows a comparison between the signal function and the proposed hyperbolic tangent function, in which the points where the functions are coincident are highlighted. In the same figure, we consider that $\zeta_{sgn} = \varpi$ and $\zeta_{tanh} > \varpi$. By substituting (3.66) in (3.63), one can find that, in the worst case,

$$\dot{V} \leq \boldsymbol{z}_1^T \left(\frac{\boldsymbol{b}}{m} + \frac{\boldsymbol{\gamma}}{m} \right) = \boldsymbol{z}_1^T \left(\frac{\boldsymbol{b}}{m} - \frac{\zeta \cdot \tanh(\varsigma \boldsymbol{z}_1)}{m} \right) \leq \frac{1}{m} \sum_{j=1}^{3} \left(|z_{1,j}| \varpi - z_{1,j} \zeta \tanh(\varsigma z_{1,j}) \right), \tag{3.67}$$

in which $z_{1,j}$ is the j-th element of the $\boldsymbol{z}_1$ vector. The behavior of $M(z_{1,j}) = |z_{1,j}|\varpi - z_{1,j}\zeta \tanh(\varsigma z_{1,j})$ is depicted on the Figure 3.4b, where the points in which $M(z_{1,j}) = 0$ for $z_{1,j} \neq 0$ are highlighted. One can compute the maximum value of $M(z_{1,j})$ by doing

$$\frac{\partial M(z_{1,j}^*)}{\partial z_{1,j}} = 0, \tag{3.68}$$

and by considering $z_{1,i} > 0$, one can find that

$$e^{2\varsigma z_{1,j}^*}(\varpi - \zeta) + e^{-2\varsigma z_{1,j}^*}(\varpi + \zeta) + 2\varpi - 4\zeta\varsigma z_{1,j}^* = 0, \tag{3.69}$$

where the solution $z_{1,j}^*$ leads to the maximum value $M(z_{1,j}^*)$ (Farrell & Polycarpou, 2006).

Due to the smoothing in the central range of the $\boldsymbol{\gamma}(\boldsymbol{z}_1)$ function, the effect of the disturbance is not always completely compensated. Therefore, the additional control law establishes an attraction region, which is regulated by the choice of ς. Consequently, the system will be ultimately uniformly bounded, since for large values of $\boldsymbol{z}_1$ the system converges asymptotically to a region around the origin in which the system remains bounded. Indeed, the system converges asymptotically to the origin when the inequality

(3.64) is true and stays bounded in an attraction region, regulated by the slope of the hyperbolic tangent function.

Chapter 5 presents numerical simulations that compare the performance of the robustified control law with the former one (Salierno & Raffo, 2017). Besides, simulation results also attest the functioning of the cascaded control system. Simulation videos of the cascaded strategy can be watched in `https://youtu.be/kvshsF-cojY`.

3.4 Summary

This chapter presented a strategy capable of guiding a set of aerial robots to a time-varying curve in the 3-D space, in which the robots converge and circumnavigate to it. In this same strategy, a scheme based on modulation functions is applied over the components of the vector field to accomplish collision avoidance among robots.

Also, the backstepping control with integral action for a quadrotor vehicle was extended by considering unknown but bounded disturbances, by the addition of a robustifying control law.

4 DMPC Based on Vector Field Navigation

This chapter proposes a distributed model predictive control (DMPC) framework for systems with predefined control laws with parameters to be tuned. Once formalized the optimization problem to solve a specific task with a multi-robot system, the problem is distributed considering the communication range of each robot and solved by the alternating direction method of multipliers (ADMM). The task considered in this work is the one of circulation and convergence to a curve in the 2-D space taking into account inter-robot collision avoidance.

4.1 Problem Statement

As in Chapter 3, consider a set Ω that contains N robots ω_i, in which $i = 1, ..., N$. The state vector of each robot is defined as $\boldsymbol{x}_i(t)$. At first, its dynamical model is given by

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}_i(\boldsymbol{x}_i, \boldsymbol{u}_i), \tag{4.1}$$

in which $\boldsymbol{u}_i$ is the control signal. Then, consider the control law for each robot predefined as follows

$$\boldsymbol{u}_i = \boldsymbol{\kappa}_i(\boldsymbol{x}_i, \boldsymbol{\theta}_i), \tag{4.2}$$

where $\boldsymbol{\theta}_i$ is a set of parameters of the control law.

Each robot ω_i has a communication range R_i and position $\boldsymbol{\xi}_i = [x_i\ y_i]^T$. As in Chapter

3, the set of agents within the perceived sphere of ω_i is given by

$$\boldsymbol{\Theta}_i \triangleq \{j \in \{1, ..., N\} | \{D_{ij} \leq R_i\}, j \neq i\},$$

with $D_{ij} \triangleq \|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2$, that is, the Euclidean distance between robots ω_i and ω_j. Consequently, if ω_i is in the perception range of ω_j, ω_j is in the perception range of ω_i:

$$i \in \boldsymbol{\Theta}_j \Leftrightarrow j \in \boldsymbol{\Theta}_i. \tag{4.3}$$

The first goal of this chapter is given as following:

Problem statement 4.1. *Given a set of robots Ω, with dynamics (4.1) and control law (4.2), design a distributed optimal control framework capable of adapting the parameterized controllers to solve multi-robot coordination tasks.*

The second goal is to apply the framework in a task similar to the one in Chapter 3. In this case, a group of robots navigating in a plane must converge and circulate a curve implicitly defined by the function $\alpha(x, y) = 0$. Further assumptions on the choice of $\alpha(x, y)$ are made in Section 4.4. Besides, it is assumed that the robots are described either by single integrators, that is

$$\dot{\boldsymbol{\xi}}_i = \boldsymbol{u}_i, \tag{4.4}$$

in which the states are $\boldsymbol{x} = \boldsymbol{\xi}$, or double integrators

$$\ddot{\boldsymbol{\xi}}_i = \boldsymbol{u}_i, \tag{4.5}$$

in which the states are $\boldsymbol{x} = [\boldsymbol{\xi}^T \; \dot{\boldsymbol{\xi}}^T]^T$. Note that if the control system of a robot expects acceleration references, the double integrator must be considered. On the other hand, if the control system of a robot expects velocity references, the single integrator is used. The problem to be solved might be stated as following.

Problem statement 4.2. *Given a set of robots $\omega_i \in \Omega$ for $i = 1, ..., N$ with dynamics (4.4) or (4.5) and communication range R_i, design a control strategy capable of providing convergence and circulation of a curve $\alpha(x, y) = 0$ while avoiding collisions among robots.*

One can note that differently from Chapter 3, the curve to which the group of robots must converge and circulate is not time-varying and is embedded in the 2-D space. This simplification is considered in order to simplify the solution of the formulated optimization problem. We stress that the proposed strategy can be directly extended to time-varying curves in the 3-D space, which would be an interesting future work.

4.2 The Optimal Control Problem

In order to solve the first problem described in Section 4.1, a parameterized MPC scheme distributed by the ADMM is proposed. The MPC strategy consists basically in computing the optimal control sequence that minimizes a cost functional in a prediction horizon Δ, then applying the first control signal and restarting the cycle (Rawlings & Mayne, 2009). In parameterized MPC (Droge & Egerstedt, 2013), parameterized control laws are embedded in the optimal control problem such that the parameters are decision variables, constant in a given time horizon.

By implementing parameterized control laws, the number of decision variables in the optimal control problem is reduced from a control sequence to a small set of parameters. Besides that, since the problem is solved with a lesser degree of freedom, less ability to minimize the cost is expected. Regarding distributed multi-robot systems, the parameterized predictive control strategy is convenient, since it allows the computation of agents' trajectories by their neighboring agents with limited amount of data: the initial state, the control law parameters, and the dynamics of the agents. In this case, it is not necessary to compute and communicate the entire control sequence (Droge & Egerstedt, 2013).

With regard to the problem previously stated, an optimal control problem is set. Considering the entire set Ω, the global optimization problem is given by

$$\begin{aligned}
&\min_{\bar{\boldsymbol{x}},\bar{\boldsymbol{\theta}}} \int_{t_0}^{t_f} L_g(\bar{\boldsymbol{x}},\bar{\boldsymbol{\theta}})dt \\
&\text{subject to} \\
&\dot{\bar{\boldsymbol{x}}} = \bar{\boldsymbol{f}}(\bar{\boldsymbol{x}},\bar{\boldsymbol{\kappa}}(\bar{\boldsymbol{x}},\bar{\boldsymbol{\theta}})), \\
&\boldsymbol{x}_i(t_0) = \boldsymbol{x}_{i,t_0}, \\
&\boldsymbol{\theta}_i \in \mathbb{P}, \quad \forall i \in \{1,...,N\},
\end{aligned} \tag{4.6}$$

in which $\bar{\boldsymbol{x}} = [\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_N^T]^T$ is the generalized state vector containing all agents states, $\bar{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T]^T$ is the generalized parameters vector, $\bar{\boldsymbol{\kappa}}$ is the generalized control law, $\boldsymbol{x}_{i,t_0}$ is the i-th robot initial state, and $\boldsymbol{\theta}_i \in \mathbb{P}$ is a constraint in each parameter vector, where $\mathbb{P}$ is a set that must be defined according to the specific task to be solved. The term $L_g(\bar{\boldsymbol{x}},\bar{\boldsymbol{\theta}})$ refers to the global stage cost, that must be designed regarding the accomplishment of the desired task. In the task considered in this chapter, the cost must drive the solution of the optimization problem to obtain convergence and circulation of a target curve, and collision avoidance among the agents. Concerning predictive control, the final time is set to be $t_f = t_0 + \Delta$, in which Δ is the continuous time prediction horizon.

To solve the optimization problem (4.6) in a distributed manner, the global cost must be split among agents. Considering in this split of (4.6) the communication ranges, one can obtain

$$\begin{aligned}
&\min_{\bar{\boldsymbol{x}},\bar{\boldsymbol{\theta}}} \sum_{i=1}^{N} \int_{t_0}^{t_f} L_i(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{\theta}}_i)dt \\
&\text{subject to} \\
&\dot{\bar{\boldsymbol{x}}}_i = \bar{\boldsymbol{f}}_i(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{\kappa}}_i(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{\theta}}_i)), \quad \forall i \in \{1, ..., N\}, \\
&\bar{\boldsymbol{\theta}}_{ij} = \bar{\boldsymbol{\theta}}_{jj}, \quad \forall j \in \boldsymbol{\Theta}_i, \\
&\bar{\boldsymbol{x}}_{ij}(t_0) = \bar{\boldsymbol{x}}_{jj,t_0}, \\
&\bar{\boldsymbol{\theta}}_{ij} \in \mathbb{P}, \quad \forall j \in \boldsymbol{\Theta}_i \cup \{i\},
\end{aligned} \tag{4.7}$$

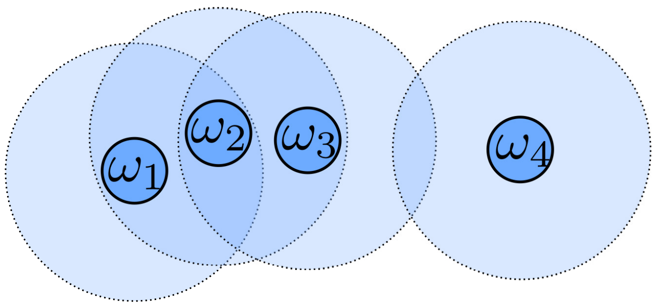
in which the generalized vectors $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{\theta}}_i$ contain the positions and parameters of the agents j such that $j \in \boldsymbol{\Theta}_i \cup \{i\}$ as computed by the i-th agent. Furthermore, $\bar{\boldsymbol{\theta}}_{ij}$ is the parameter vector of the j-th agent as computed by the i-th agent. Remember that the i-th agent computes its own parameter and trajectory and the parameters and trajectories of the agents in the perception set $\boldsymbol{\Theta}_i$.

To split the global cost considering the communication range, the global cost, and, consequently, the individual costs must be varying. At each MPC cycle, the sets of agents inside the communication range $\boldsymbol{\Theta}_i$ $\forall i$ may change. Indeed, if the i-th agent does not communicate with an agent j, there must be no coupling between these agents in the optimal control problem.

Section 4.3 describes the distributed optimization scheme used to solve the optimal control problem (4.7). Section 4.4 provides a set of vector field based control laws that are used in the framework to accomplish the task of convergence and circulation of a curve.

4.3 ADMM-based parameterized DMPC

In this work, an agent can only communicate with agents in a certain range. Therefore, there is no global consensus of variables, instead a set of local consensus problems in which the i-th agent is the fusion center of the $\boldsymbol{\theta}_i$ variable is executed. This problem is named as general consensus (Boyd et al., 2011), and the solution is similar to the one of the global consensus. As the decision variables of the i-th agent are determined by the agent network defined by the communication range, each local consensus problem is equivalent to a global one, considering only the shared decision variables and the agents in the communication range. Figure 4.1 illustrates the optimization scheme with a four agents example, in which the agents and their communication ranges are drawn. Agent ω_1 connects only with ω_2; therefore, only ω_1 and ω_2 influence the choice of $\bar{\boldsymbol{\theta}}_{11}$. As ω_2 connects to ω_1 and ω_3, the three agents negotiate to achieve consensus on $\bar{\boldsymbol{\theta}}_{22}$ value. Similarly, as ω_4 does not connect to any agent, there is no influence of other agents.



$$\min_{\bar{\boldsymbol{x}}_1,\bar{\boldsymbol{\theta}}_1} \int_{t_0}^{t_f} L_1(\bar{\boldsymbol{x}}_1,\bar{\boldsymbol{\theta}}_1)dt$$
subject to
$$\dot{\bar{\boldsymbol{x}}}_1 = \bar{\boldsymbol{f}}_1(\bar{\boldsymbol{x}}_1,\bar{\boldsymbol{\kappa}}_1(\bar{\boldsymbol{x}}_1,\bar{\boldsymbol{\theta}}_1)),$$
$$\bar{\boldsymbol{\theta}}_{12} = \bar{\boldsymbol{\theta}}_{22},\ \bar{\boldsymbol{\theta}}_{11} = \bar{\boldsymbol{\theta}}_{21},$$
$$\bar{\boldsymbol{\theta}}_{11} \in \mathbb{P},\ \bar{\boldsymbol{\theta}}_{12} \in \mathbb{P},$$
$$\bar{\boldsymbol{x}}_{11}(t_0) = \bar{\boldsymbol{x}}_{11,t_0},$$
$$\bar{\boldsymbol{x}}_{12}(t_0) = \bar{\boldsymbol{x}}_{22,t_0}$$

$$\min_{\bar{\boldsymbol{x}}_2,\bar{\boldsymbol{\theta}}_2} \int_{t_0}^{t_f} L_2(\bar{\boldsymbol{x}}_2,\bar{\boldsymbol{\theta}}_2)dt$$
subject to
$$\dot{\bar{\boldsymbol{x}}}_2 = \bar{\boldsymbol{f}}_2(\bar{\boldsymbol{x}}_2,\bar{\boldsymbol{\kappa}}_2(\bar{\boldsymbol{x}}_2,\bar{\boldsymbol{\theta}}_2)),$$
$$\bar{\boldsymbol{\theta}}_{21} = \bar{\boldsymbol{\theta}}_{11},\ \bar{\boldsymbol{\theta}}_{23} = \bar{\boldsymbol{\theta}}_{33},$$
$$\bar{\boldsymbol{\theta}}_{22} = \bar{\boldsymbol{\theta}}_{12},\ \bar{\boldsymbol{\theta}}_{22} = \bar{\boldsymbol{\theta}}_{32},$$
$$\bar{\boldsymbol{\theta}}_{21} \in \mathbb{P},\ \bar{\boldsymbol{\theta}}_{22} \in \mathbb{P},\ \bar{\boldsymbol{\theta}}_{23} \in \mathbb{P},$$
$$\bar{\boldsymbol{x}}_{21}(t_0) = \bar{\boldsymbol{x}}_{11,t_0},$$
$$\bar{\boldsymbol{x}}_{22}(t_0) = \bar{\boldsymbol{x}}_{22,t_0},$$
$$\bar{\boldsymbol{x}}_{23}(t_0) = \bar{\boldsymbol{x}}_{33,t_0}$$

$$\min_{\bar{\boldsymbol{x}}_3,\bar{\boldsymbol{\theta}}_3} \int_{t_0}^{t_f} L_3(\bar{\boldsymbol{x}}_3,\bar{\boldsymbol{\theta}}_3)dt$$
subject to
$$\dot{\bar{\boldsymbol{x}}}_3 = \bar{\boldsymbol{f}}_3(\bar{\boldsymbol{x}}_3,\bar{\boldsymbol{\kappa}}_3(\bar{\boldsymbol{x}}_3,\bar{\boldsymbol{\theta}}_3)),$$
$$\bar{\boldsymbol{\theta}}_{32} = \bar{\boldsymbol{\theta}}_{22},\ \bar{\boldsymbol{\theta}}_{33} = \bar{\boldsymbol{\theta}}_{23},$$
$$\bar{\boldsymbol{\theta}}_{32} \in \mathbb{P},\ \bar{\boldsymbol{\theta}}_{33} \in \mathbb{P},$$
$$\bar{\boldsymbol{x}}_{32}(t_0) = \bar{\boldsymbol{x}}_{22,t_0},$$
$$\bar{\boldsymbol{x}}_{33}(t_0) = \bar{\boldsymbol{x}}_{33,t_0}$$

$$\min_{\bar{\boldsymbol{x}}_4,\bar{\boldsymbol{\theta}}_4} \int_{t_0}^{t_f} L_4(\bar{\boldsymbol{x}}_4,\bar{\boldsymbol{\theta}}_4)dt$$
subject to
$$\dot{\bar{\boldsymbol{x}}}_4 = \bar{\boldsymbol{f}}_4(\bar{\boldsymbol{x}}_4,\bar{\boldsymbol{\kappa}}_4(\bar{\boldsymbol{x}}_4,\bar{\boldsymbol{\theta}}_4)),$$
$$\bar{\boldsymbol{\theta}}_{44} \in \mathbb{P},$$
$$\bar{\boldsymbol{x}}_{44}(t_0) = \bar{\boldsymbol{x}}_{44,t_0}$$

Figure 4.1: Communication range example and local optimization problems.

As shown in Section 4.2, the optimal control problem is given by

$$\begin{aligned}
&\min_{\bar{\boldsymbol{x}},\bar{\boldsymbol{\theta}}} \sum_{i=1}^{N} \int_{t_0}^{t_f} L_i(\bar{\boldsymbol{x}}_i(t),\bar{\boldsymbol{\theta}}_i)dt \\
&\text{subject to} \\
&\dot{\bar{\boldsymbol{x}}}_i = \bar{\boldsymbol{f}}_i(\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{\kappa}}_i(\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{\theta}}_i)), \quad \forall i \in \{1,...,N\}, \\
&\bar{\boldsymbol{\theta}}_{ij} = \bar{\boldsymbol{\theta}}_{jj}, \quad \forall j \in \boldsymbol{\Theta}_i, \\
&\bar{\boldsymbol{x}}_{ij}(t_0) = \bar{\boldsymbol{x}}_{jj,t_0}, \\
&\bar{\boldsymbol{\theta}}_{ij} \in \mathbb{P}, \quad \forall j \in \boldsymbol{\Theta}_i \cup \{i\},
\end{aligned} \tag{4.8}$$

where the generalized vectors $\bar{\boldsymbol{x}}_i$ and $\bar{\boldsymbol{\theta}}_i$ are formed accordingly to $\boldsymbol{\Theta}_i$. Remember that the generalized vectors $\bar{\boldsymbol{x}}_i$ are formed by stacking the states $\bar{\boldsymbol{x}}_{ij}$ such that $j \in \boldsymbol{\Theta}_i \cup \{i\}$, in which $\bar{\boldsymbol{x}}_{ij}$ is the state vector of the j-th agent as computed by the i-th agent. Likewise, the $\bar{\boldsymbol{\theta}}_i$ vectors are constructed by stacking the parameters $\bar{\boldsymbol{\theta}}_{ij}$ such that $j \in \boldsymbol{\Theta}_i \cup \{i\}$. Note that the global problem (4.8) can be split by the ADMM technique already described in Chapter 2.

The costs $L_i(\bar{\boldsymbol{x}}_i(t),\bar{\boldsymbol{\theta}}_i)$ are augmented as in the ADMM distributed optimization, yielding

$$L_{\rho,i}(\bar{\boldsymbol{x}}_i(t),\bar{\boldsymbol{\theta}}_i,\bar{\boldsymbol{y}}_i,\bar{\boldsymbol{z}}_i) = \int_{t_0}^{t_f} L_i(\bar{\boldsymbol{x}}_i(t),\bar{\boldsymbol{\theta}}_i)dt + \sum_{j\in\boldsymbol{\Theta}_i\cup\{i\}} \left(\bar{\boldsymbol{y}}_{ij}^T(\bar{\boldsymbol{\theta}}_{ij} - \bar{\boldsymbol{z}}_{ij}) + \frac{\rho}{2}\left\|\bar{\boldsymbol{\theta}}_{ij} - \bar{\boldsymbol{z}}_{ij}\right\|_2^2\right), \tag{4.9}$$

in which $\bar{\boldsymbol{z}}_i$ is the generalized vector that contains $\bar{\boldsymbol{z}}_{ij} = \boldsymbol{z}_j\ \forall j \in \boldsymbol{\Theta}_i \cup \{i\}$, $\bar{\boldsymbol{y}}_i$ is the generalized vector that contains the Lagrange multipliers $\bar{\boldsymbol{y}}_{ij}$, $\forall j \in \boldsymbol{\Theta}_i \cup \{i\}$ as computed by the i-th agent, and ρ is the ADMM penalty parameter.

Algorithm 4.1 aims to compute the control laws' parameters. In that algorithm, an external loop is responsible for incrementing the MPC horizon, while an internal loop is responsible for finding the optimal problem solution in a distributed manner. One can note that in the internal loop the sets of agents in range $\boldsymbol{\Theta}_i$ are taken into account. In addition, a constraint on the parameters values is also applied, which will be discussed in the next sections. As in (Van Parys & Pipeleers, 2017) and (Ferranti et al., 2018), the number of ADMM iterations per MPC cycle n_{ite} is limited by n_{max}, aiming to reduce the communication exchange and the computational burden. In addition, after the first ADMM iteration in each MPC cycle the optimization problems are warm started, i.e., the problems are initialized with the previous computed solutions. With this approach, we start the optimization problem from a solution that we expect to be near to the current optimization problem solution, which makes the optimization process faster. Furthermore, one can note that the ADMM requires synchronization among the agents, since each of the three steps occurs in parallel and must be complete before the next step. Therefore, the ADMM iterations are subject to the "straggler" problem, in which the update velocity depends on the processing time of the slowest worker (Zhang & Kwok, 2014). Besides, during each MPC cycle, we assume that the communication network is maintained.

Convergence of the ADMM regarding convex problems is a well established result (Boyd et al., 2011). Besides that, a number of nonconvex problems have been successfully approached by ADMM, and recently its convergence has been shown for consensus and sharing problems (Hong et al., 2016) and for more general problems (Wang et al., 2019).

4.4 Vector Field Navigation

The strategy used to guide multiple robots to converge and circulate a curve is based on those presented by Gonçalves et al. (2010a) and Gonçalves et al. (2010b), with some slight modifications. At first, consider that the i-th robot is described by the single integrator

$$\dot{\boldsymbol{\xi}}_i = \boldsymbol{u}_i, \tag{4.10}$$

in which $\boldsymbol{\xi}_i \in \mathbb{R}^2$ is the position. Note that we use $\boldsymbol{x}$ to represent general state vectors and $\boldsymbol{\xi}$ to represent positions.

In this chapter, consider simple planar curves that can be implicitly described by a function $\alpha(x, y) : \mathbb{R}^2 \to \mathbb{R}$ as the constraint $\alpha(x, y) = 0$, in which x and y are coordinates in a Cartesian system $\boldsymbol{\xi} \in \mathbb{R}^2$ such that $\boldsymbol{\xi} = [x\ y]^T$. To produce the desired behavior, the function $\alpha(x, y)$ must be positive when evaluated outside of the desired curve, and negative inside. Furthermore, consider the constants C_1 and C_2 such that $C_1 > C_2$. The contour lines $\alpha = C_1$ and $\alpha = C_2$ must be closed and the contour line $\alpha = C_2$ must be inside the

Algorithm 4.1 Distributed Parameterized Model Predictive Control

```
1: repeat
2:     $t := t + dt$;
3:     $t_0 := t$;
4:     $t_f := t_0 + \Delta$;
5:     $n_{ite} := 0$;
6:     $\bar{\boldsymbol{y}}_i^k = \mathbf{0}, \forall i$;
7:     $\bar{\boldsymbol{z}}_i^k = \mathbf{0}, \forall i$;
8:     Each agent communicates $\bar{\boldsymbol{\theta}}_{ii}$ and $\boldsymbol{x}_{ii}(t_0)$ to the agents in range.
9:     repeat
10:        In parallel, each agent solves an optimization problem
           $\bar{\boldsymbol{\theta}}_i^{k+1} = \arg\min_{\bar{\boldsymbol{\theta}}_i} \quad L_{\rho,i}(\bar{\boldsymbol{x}}_i(t), \bar{\boldsymbol{\theta}}_i, \bar{\boldsymbol{y}}_i^k, \bar{\boldsymbol{z}}_i^k)$
           subject to
           $\dot{\bar{\boldsymbol{x}}}_i(t) = \bar{\boldsymbol{f}}_i(\bar{\boldsymbol{x}}_i, \bar{\boldsymbol{\kappa}}_i(\bar{\boldsymbol{x}}_i(t), \bar{\boldsymbol{\theta}}_i))$
           $\bar{\boldsymbol{x}}_{ij}(t_0) = \bar{\boldsymbol{x}}_{jj,t_0}$,
           $\bar{\boldsymbol{\theta}}_{ij} \in \mathbb{P}, \quad \forall j \in \boldsymbol{\Theta}_i \cup \{i\}$.
11:        Each agent communicates $\bar{\boldsymbol{\theta}}_{ii}$, $\bar{\boldsymbol{\theta}}_{ij}$ and $\bar{\boldsymbol{y}}_{ij}$.
12:        In parallel, the $i$-th agent computes $\boldsymbol{z}_i^{k+1}$
           $\boldsymbol{z}_i^{k+1} := \frac{1}{|\Theta_i|+1} \sum_{j \in \Theta_i \cup \{i\}} \left( \bar{\boldsymbol{\theta}}_{ji}^{k+1} + (1/\rho) \bar{\boldsymbol{y}}_{ji}^k \right)$ .
13:        Each $i$-th agent transmits the $\boldsymbol{z}_i^{k+1}$ value.
14:        In parallel, each agent computes the Lagrange multipliers
           $\bar{\boldsymbol{y}}_i^{k+1} := \bar{\boldsymbol{y}}_i^k + \rho \left( \bar{\boldsymbol{\theta}}_i^{k+1} - \bar{\boldsymbol{z}}_i^{k+1} \right)$ .
15:        $n_{ite} = n_{ite} + 1$
16:    until $n_{ite} == n_{max}$
17:    Apply the first control signal computed.
18: until reaches the final time
```

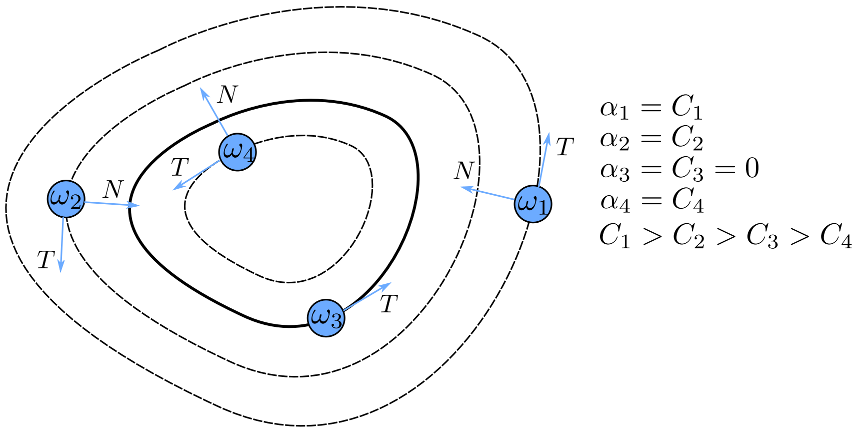


Figure 4.2: Vector field navigation example.

contour line $\alpha = C_1$. Figure 4.2 shows an example with 4 robots, in which the solid curve is the target and $\alpha(x_i, y_i) = \alpha_i$.

The task of convergence and circulation can be solved separately. After that, the solutions are joined, generating a vector field in the form of

$$\boldsymbol{v}(x,y) = \boldsymbol{N}(x,y) + \boldsymbol{T}(x,y), \tag{4.11}$$

in which $\boldsymbol{N}(x,y)$ is a field component normal to the contour curve at (x,y), and $\boldsymbol{T}(x,y)$ is a field component tangent to the same contour curve. The first term is responsible for providing convergence to the aimed curve, while the second is responsible for driving the circulation of the curve. In Figure 4.2 one can note the direction of the described fields for four robots in different positions.

To achieve convergence, one can observe the contour lines. In a given position $[x\ y]^T$, the agent is in the contour line $\alpha(x,y) = C_1$. To guide the agent to the desired curve ($\alpha(x,y) = 0$), one can consider a velocity component in the direction of the gradient $\nabla\alpha(x,y)$. Therefore, if an agent is inside the curve, that is $\alpha(x,y) < 0$, the gradient points in the direction of the desired curve. On the other hand, if the agent is outside the curve, that is, $\alpha(x,y) > 0$, the gradient points in the opposite sense. Moreover, if the agent is on the curve, the convergence component must be zero. To obtain the desired behavior from $\boldsymbol{N}(x,y)$, a function that describes the change in the velocity sense must be designed. Thus, the convergence term can be obtained by computing

$$\boldsymbol{N}\left(x,y\right) = \vartheta\left(x,y\right)\nabla\alpha\left(x,y\right), \tag{4.12}$$

where the function $\vartheta(x,y)$ is multiplied to produce the desired behavior of $\boldsymbol{N}(x,y)$. The function $\vartheta(x,y)$ can be defined as

$$\vartheta(x,y) = G(\alpha(x,y)), \tag{4.13}$$

such that $G(b) > 0$ for $b < 0$, $G(b) < 0$ for $b > 0$, and $G(0) = 0$. The simplest function that presents this behavior is given by $G(b) = -b$.

The circulation term can be designed to point tangentially to the contour line in which the agent is located. Thus, the tangential term can be obtained as follows

$$\boldsymbol{T}\left(x,y\right) = \beta\left(x,y\right)\left[-\frac{\partial\alpha}{\partial y}\ \frac{\partial\alpha}{\partial x}\right]^T, \tag{4.14}$$

in which $\beta(x,y)$ is a continuous function non-null at $\alpha(x,y) = 0$, and $\left[-\frac{\partial\alpha}{\partial y}\ \frac{\partial\alpha}{\partial x}\right]^T$ is the vector orthogonal to the gradient, denominated Hamiltonian vector field $\nabla_{H\alpha}$. If $\beta(x,y)$ is defined as

$$\beta(x,y) = H(\alpha(x,y)), \tag{4.15}$$

then $H(\alpha)$ is constant in a contour line and must be non-null in $H(0)$, so that the circulation is guaranteed when the agent is on the curve. A trivial example function is $H = 1$.

Finally, a vector field-based control law for the i-th agent can be defined as

$$\boldsymbol{\kappa}_i(\boldsymbol{\xi}_i) = G(\alpha)\nabla\alpha + H(\alpha)\nabla_H\alpha, \tag{4.16}$$

in which α is evaluated in $\boldsymbol{\xi}_i$.

Similarly to (Pimenta et al., 2013) and to Chapter 3, it is desired to avoid collisions with other robots, and this can be done by modulating the parameters of the vector field. In this work, a parameter vector $\boldsymbol{\theta}_i \in \mathbb{R}^2$ such that $\boldsymbol{\theta}_i = [\boldsymbol{\theta}_{i,1}\ \boldsymbol{\theta}_{i,2}]$ is considered, in which the first is multiplied by the convergence term and the second is multiplied by the circulation term. To achieve convergence and circulation of the curve in the desired sense, the $\boldsymbol{\theta}_i$ elements must be positive. However, if we allow only positive values in $\boldsymbol{\theta}_i$, the maneuverability is restricted, since the robot can circulate the curve in just one sense, and cannot move away from the target curve. To guarantee better maneuverability in front of collision risk situations, we allow negative values for the $\boldsymbol{\theta}_i$ elements. Besides that, the same values are limited by $\pm\theta_{max}$, so that the maximum velocity is limited. With the introduced parameters, a parameterized control law is given by

$$\boldsymbol{\kappa}_i(\boldsymbol{\xi}_i, \boldsymbol{\theta}_i) = \boldsymbol{\theta}_{i,1}G(\alpha)\nabla\alpha + \boldsymbol{\theta}_{i,2}H(\alpha)\nabla_H\alpha. \tag{4.17}$$

As one can observe in (4.17), the vector field magnitude in each point (x, y) depends on the choice of the function α. This characteristic can generate huge control signals through optimal control solved discretely, since it can cause large displacements between two samples when assuming unbounded velocities, making collision detection difficult. In front of this, the vector field components are normalized

$$\boldsymbol{\kappa}_i(\boldsymbol{\xi}_i, \boldsymbol{\theta}_i) = \boldsymbol{\theta}_{i,1}G(\alpha)\frac{\nabla\alpha}{||\nabla\alpha||} + \boldsymbol{\theta}_{i,2}H(\alpha)\frac{\nabla_H\alpha}{||\nabla_H\alpha||}. \tag{4.18}$$

Using (4.18) in (4.10) we obtain the system dynamics considered in the optimization problem in the case of single integrators.

As in (4.18) the gradient is zero only on singular points and we are assuming those points to be at the center of the curve, there are no divisions by zero if the agents' initial position is considered to be different from that point, since the center of the curve is repulsive. For $\boldsymbol{\theta}_{i,1} \geq 0$ and $\boldsymbol{\theta}_{i,2} \geq 0$, the proof of convergence and circulation for this field is done as presented in (Gonçalves et al., 2010a). To illustrate the influence of tuning $\boldsymbol{\theta}_{i,1}$ and $\boldsymbol{\theta}_{i,2}$ on the vector field, Figure 4.3 shows three tuning situations for a given vector field. As expected, when the convergence parameter dominates the vector field, the vectors point almost directly to the curve. Conversely, when the circulation parameter dominates, the vectors are disposed almost tangentially to the curve.

Now, the control law just described can be modified to suit in double integrator systems

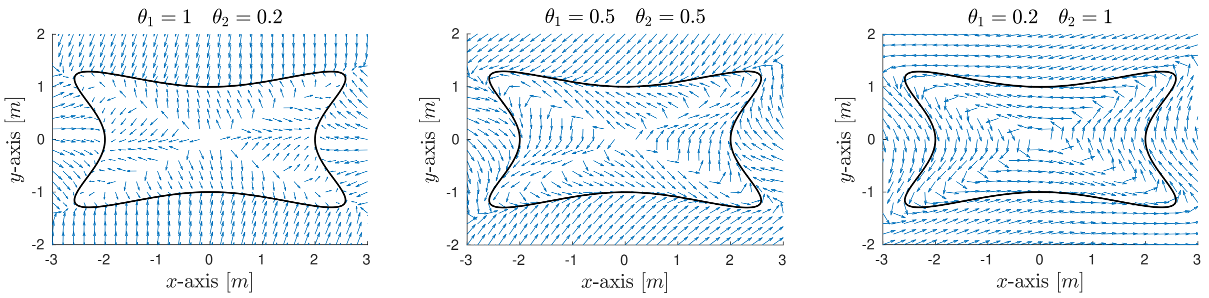


Figure 4.3: Example of vector field modulation by the control law parameters.

(Gonçalves et al., 2010b). In this case, it is also desired to obtain

$$\dot{\boldsymbol{\xi}}_i = \boldsymbol{\kappa}_i(\boldsymbol{\xi}_i, \boldsymbol{\theta}_i), \tag{4.19}$$

where $\boldsymbol{\kappa}_i(\boldsymbol{\xi}_i, \boldsymbol{\theta}_i)$ is given by (4.18). To guarantee (4.19), consider the following Lyapunov candidate function

$$V = \frac{1}{2}(\dot{\boldsymbol{\xi}}_i - \boldsymbol{\kappa}_i)^T(\dot{\boldsymbol{\xi}}_i - \boldsymbol{\kappa}_i), \tag{4.20}$$

in which the arguments of $\boldsymbol{\kappa}_i$ are omitted. By disregarding the $\boldsymbol{\theta}_i$ variation, the time derivative of the Lyapunov candidate function yields

$$\dot{V} = (\dot{\boldsymbol{\xi}}_i - \boldsymbol{\kappa}_i)^T \left(\ddot{\boldsymbol{\xi}} - J_{\boldsymbol{\xi}_i}(\boldsymbol{\kappa}_i)\dot{\boldsymbol{\xi}}_i\right), \tag{4.21}$$

where $J_{\boldsymbol{\xi}_i}(\boldsymbol{\kappa}_i)$ is the Jacobian matrix of $\boldsymbol{\kappa}_i$ in relation to $\boldsymbol{\xi}_i$.

Therefore, the following control law can be applied

$$\ddot{\boldsymbol{\xi}}_i = J_{\boldsymbol{\xi}_i}\dot{\boldsymbol{\xi}}_i + \boldsymbol{\sigma}(\dot{\boldsymbol{\xi}}_i - \boldsymbol{\kappa}_i), \tag{4.22}$$

in which $\boldsymbol{\sigma}$ is a control law to be chosen, capable of stabilizing systems in the form of $\dot{\boldsymbol{p}} = \boldsymbol{\sigma}(\boldsymbol{p})$.

Equation (4.22) is the system dynamics to be considered in the optimization problem when a double integrator model is assumed.

4.5 Problem Formulation and Collision Avoidance

4.5.1 Scheme 1

To aim at convergence and circulation of the curve while avoiding collisions, a natural proposition is to consider desired parameters $\boldsymbol{\theta}_d = [\boldsymbol{\theta}_{d,1}\ \boldsymbol{\theta}_{d,2}]$ for convergence and circulation, and to add a repulsion term between agents. For the control law (4.18), a robot converges to the curve if $\boldsymbol{\theta}_{i,1} > 0$, and maintains a certain circulation sense if $\boldsymbol{\theta}_{i,2} > 0$. Therefore, it makes sense to choose positive $\boldsymbol{\theta}_{d,1}$ and $\boldsymbol{\theta}_{d,2}$ desired parameters. Besides, if $G(\alpha)$ and

$H(\alpha)$ are limited, the maximum vector field magnitude is regulated by the same desired parameters.

Consider the position vectors present in the state variables given by $\boldsymbol{\xi}_{ij}$, that is, the j-th robot position as computed by the i-th robot, $\forall j \in \boldsymbol{\Theta}_i$. Therefore, the i-th agent stage cost might be chosen as

$$L_i(\bar{\boldsymbol{x}}_i(t), \bar{\boldsymbol{\theta}}_i) = \eta_1(\bar{\boldsymbol{\theta}}_{ii,1} - \boldsymbol{\theta}_{d,1})^2 + \eta_2(\bar{\boldsymbol{\theta}}_{ii,2} - \boldsymbol{\theta}_{d,2})^2 + \eta_3 \sum_{j \in \boldsymbol{\Theta}_i} \exp\left(-\zeta\left(\|\boldsymbol{\xi}_{ii}(t) - \boldsymbol{\xi}_{ij}(t)\|_2^2 - \epsilon_e\right)\right), \tag{4.23}$$

in which η_1, η_2 and η_3 are weighting parameters, $\zeta > 0$ is a constant, $\epsilon_e > 0$ is a safety distance, and $\bar{\boldsymbol{\theta}}_{ii} = [\bar{\boldsymbol{\theta}}_{ii,1}\ \bar{\boldsymbol{\theta}}_{ii,2}]^T$ is the parameter vector of the i-th agent computed by itself. The terms $(\bar{\boldsymbol{\theta}}_{ii,1} - \boldsymbol{\theta}_{d,1})^2$ and $(\bar{\boldsymbol{\theta}}_{ii,2} - \boldsymbol{\theta}_{d,2})^2$ penalize the cost based on the desired convergence and circulation parameters, $\boldsymbol{\theta}_{d,1}$ and $\boldsymbol{\theta}_{d,2}$, respectively. The sum of exponential functions provides repulsion between agents as its value increases when agents get closer to each other.

As the elements of the parameter vectors are limited between $-\theta_{max}$ and θ_{max}, the three terms in (4.23) are bounded. Therefore, collision avoidance can be aimed by choosing η_1, η_2, η_3, ζ and θ_{max} properly.

4.5.2 Scheme 2

An interesting way of describing obstacle avoidance constraints in optimization problems is presented in (Sathya et al., 2018). By this approach, an obstacle can be described by an intersection of nonlinear inequalities

$$O = \{\boldsymbol{\xi} \in \mathbb{R}^{n_d} : h^i(\boldsymbol{\xi}) > 0, i \in \mathbb{N}_{[1,m]}\}, \tag{4.24}$$

where n_d is the number of position variables, m is the number of nonlinear inequalities, and $h^i : \mathbb{R}^{n_d} \to \mathbb{R}$ are continuously differentiable functions with Lipschitz-continuous gradient ($C^{1,1}$). The hard-constraint for collision avoidance $\boldsymbol{\xi} \notin O$ is satisfied if and only if

$$h^{i_0}(\boldsymbol{\xi}) \leq 0, \text{for some } i_0 \in \mathbb{N}_{[1,m]}, \tag{4.25}$$

that is, there exists at least one function $h^{i_0}(\boldsymbol{\xi})$ that indicates that the position $\boldsymbol{\xi}$ is out of the obstacle. The same condition can be expressed by using the operator $[v]_+ = \max\{v, 0\}$, for $v \in \mathbb{R}$. If $[h^{i_0}(\boldsymbol{\xi})]_+^2 = 0$, then $h^{i_0}(\boldsymbol{\xi}) \leq 0$. Therefore, the constraint (4.25) can be formulated as

$$\alpha_O(\boldsymbol{\xi}) = \frac{1}{2}\prod_{i=1}^{m}[h^i(\boldsymbol{\xi})]_+^2 = 0. \tag{4.26}$$

In order to simplify the optimization problem, the hard constraints in the form of

(4.26) can transformed in soft constraints by using the quadratic penalty method, yielding

$$\tilde{h}(\boldsymbol{\xi}) = \eta \prod_{i=1}^{m} [h^i(\boldsymbol{\xi})]_+^2. \tag{4.27}$$

In the case proposed in this work, a robot is considered to be an obstacle to another robot. If each robot is an obstacle in the form of a sphere with radius r, the i-th robot can be described by

$$O_i = \{\boldsymbol{\xi} \in \mathbb{R}^{n_d} : (2r + \epsilon_s)^2 - \|\boldsymbol{\xi} - \boldsymbol{\xi}_i\|_2^2 > 0\} \tag{4.28}$$

in which r is the robot radius, and ϵ_s is safety distance. Note that only one inequality is necessary to represent a ball.

By considering each robot as an obstacle, the optimization problem to be solved by the i-th robot is given by

$$L_i(\bar{\boldsymbol{x}}_i(t), \bar{\boldsymbol{\theta}}_i) = \eta_1(\bar{\boldsymbol{\theta}}_{ii,1} - \boldsymbol{\theta}_{d,1})^2 + \eta_2(\bar{\boldsymbol{\theta}}_{ii,2} - \boldsymbol{\theta}_{d,2})^2 + \eta_3 \sum_{j \in \Theta_i} \left[(2r + \epsilon_s)^2 - \|\boldsymbol{\xi}_{ii}(t) - \boldsymbol{\xi}_{ij}(t)\|_2^2\right]_+^2. \tag{4.29}$$

Chapter 5 presents several numerical results obtained with the framework just described, in which single and double integrator dynamics are implemented with the two obstacle avoidance schemes. One of the proposed simulations can be watched in https://youtu.be/e2ERo_HfDpA.

4.6 Summary

This chapter proposes an MPC based strategy to drive a set of agents to converge and circulate a curve while avoiding inter-robot collisions. The strategy is based on embedding a control law with parameters in a predictive optimal control problem. Hence, the agent dynamics are restricted to the control law behavior, in which the control law parameters are the decision variables. A similar strategy is presented in (Droge & Egerstedt, 2013), in which a dual decomposition scheme is applied to distribute the optimization. In this Chapter, the optimal control problem was distributed using the ADMM method, which generally converges faster and is more robust than the dual decomposition. Besides, this chapter presents an extension of the vector field navigation framework by regarding the problem of modulating the convergence and circulation fields as an optimization problem, which allows us to pursue collision avoidance through distributed optimal control.

5 Results

This chapter presents the numerical simulation results used to validate the strategies proposed in this dissertation. Section 5.1 presents a comparison between a backstepping controller with integral action and a robust backstepping controller, also with integral action, regarding trajectory tracking by a quadrotor vehicle. The same section presents numerical results of the cascaded system, in which the high level strategy is given by a vector field modulated law and the low level strategy is given by the proposed robust backstepping control law. Section 5.2 presents several simulations of the parameterized DMPC framework considering single and double-integrator dynamics, two collision avoidance approaches, and some variation of the prediction horizon of the optimal control problem.

5.1 Cascaded Control

5.1.1 Robust Backstepping Controller with Integral Action

In Chapter 3 (Section 3.3), a trajectory tracking controller for a quadrotor subject to unknown but bounded disturbances is proposed. The strategy is based on a backstepping controller with integral action, in which a robustification law is added in the backstepping procedure. In this section, the proposed strategy is compared with a previous strategy, given in (Salierno & Raffo, 2017), in which the additional control law is not considered.

To analyze the proposed controller, consider an eight-shaped reference trajectory

Table 5.1: Quadrotor parameters.

Parameter	**Symbol**	**Value**
Mass of the quadrotor UAV	m	$2.24\ kg$
Distance between the rotors and the vehicle's center of gravity	l	$0.332\ m$
Thrust coefficient of the rotors	b	$9.5e-6\ Ns^2$
Drag coefficient of the rotors	k_τ	$1.7e-7\ Nms^2$
Maximum propeller force	$f_{i_{max}}$	$12.2\ N$
Gravitational acceleration	g	$9.81\ m/s^2$
Moment of inertia around the x-axis	I_{xx}	$0.0363\ kg.m^2$
Moment of inertia around the y-axis	I_{yy}	$0.0363\ kg.m^2$
Moment of inertia around the z-axis	I_{zz}	$0.0615\ kg.m^2$

parameterized in time as

$$\begin{aligned} x_r &= \frac{1}{2}\cos\left(\frac{\pi t}{40}\right) m, \\ y_r &= \frac{1}{2}\sin\left(\frac{\pi t}{20}\right) m, \\ z_r &= 2 - \frac{1}{2}\cos\left(\frac{\pi t}{40}\right) m, \end{aligned}$$

in which the yaw angle follows the direction of the curve

$$\psi_r = \arctan\left(\frac{\dot{y}_r}{\dot{x}_r}\right) rad.$$

The quadrotor parameters are the same used in (Raffo, 2011) and are given in Table 5.1. A saturation of the control signals is applied, according to Table 5.2, and the force produced from each propeller in the aircraft model is also saturated between the limits 0 and $12.2\ N$. Furthermore, the controller of each vehicle is adjusted by try and error in the same sequence that the parameters appear on the backstepping process, while aiming at fast convergence to the target trajectory and robustness to disturbances. The parameters are adjusted to the following values:

$$\begin{aligned} &\boldsymbol{k}_1 = \text{diag}(3,3,4.6), \quad \boldsymbol{k}_2 = \text{diag}(5,5,5), \quad \boldsymbol{k}_{\mathcal{X}\xi} = \text{diag}(10,10,10), \quad \boldsymbol{k}_3 = \text{diag}(6,6,6), \\ &\boldsymbol{k}_4 = \text{diag}(10,10,10), \quad k_5 = 10, \quad k_6 = 6, \quad k_{\mathcal{X}\psi} = 20, \quad \varsigma = 20, \quad \zeta = 4.5, \end{aligned} \tag{5.1}$$

in which diag$(\cdot)$ is a diagonal matrix populated by the elements $(\cdot)$.

The simulations in this section are implemented in a computer equipped with an intel i5-3210M processor and Ubuntu 16.04 Xenial operating system. We emphasize that the same simulations are not performed in real time, and the computer can take as long as needed to compute the solution.

The numerical simulation is carried out by integrating the quadrotor dynamics in $1000\,Hz$, while the backstepping controller runs at $100\,Hz$. Besides that, the system is

Table 5.2: Saturation of the control signals.

Input	Minimum value	Maximum value
T	$0.01N$	$48.0N$
τ_ϕ	$-4.0N.m$	$+4.0N.m$
τ_θ	$-4.0N.m$	$+4.0N.m$
τ_ψ	$-0.425N.m$	$+0.425N.m$

Table 5.3: External disturbances actuation.

Coordinate	Shape	Time (s)
x	$4\sin(2\pi t)\ N$	$20 < t \leq 50$
y	$4\sin(2\pi t)\ N$	$40 < t \leq 70$
z	$4\sin(2\pi t)\ N$	$60 < t \leq 90$

affected by time-varying disturbances described in Table 5.3. The total simulation time is $80s$, and the initial position is defined as $\boldsymbol{\xi} = [0.5\ 0.5\ 0.5]^T$ and $\psi = 0$.

Figure 5.1 shows the reference trajectory and the actual trajectory performed by the quadrotor, applying the standard backstepping approach and the robust one. One can note in Figure 5.1a that the effect of the disturbances in the trajectories of the quadrotor is much more relevant than in Figure 5.1b. If we observe separately at the evolution of each controlled variable, in Figure 5.2, it is possible to note the direct influence of disturbances in the trajectory oscillations. Once again, one can observe the better performance achieved by the robust backstepping controller. Next, Figure 5.3 presents the evolution of tracking error for both controllers during the simulations, in which the superiority of the robust controller is confirmed. Besides, note in Figure 5.3a that the error obtained in the z-axis is lesser than the ones in the x and y-axis. Indeed, as the quadrotor vehicle can control more easily its position in the z-axis, due to its construction characteristics, the integral action is fast enough to soften the effects of the disturbance in this direction.

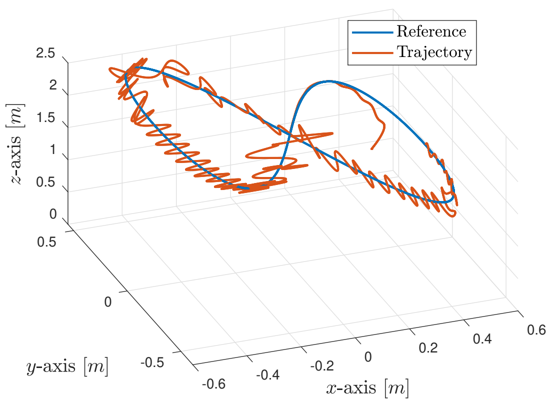
Finally, Figure 5.4 shows the control signals evolution for both simulations. One can note that the control signal is more aggressive when considering an additional Lyapunov redesign control law, which allows better trajectory tracking in front of time-varying disturbances.

5.1.2 Multi-layer Control

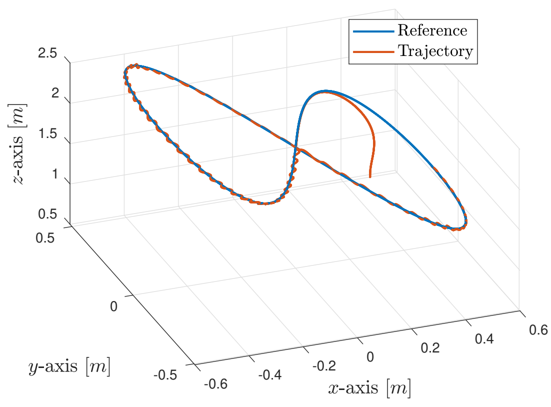
To attest the functioning of the cascaded control strategy proposed in Chapter 3, it is presented a simulation with a group of quadrotors. The group of vehicles must converge and circulate a time-varying curve in the three-dimensional space, given by

$$\alpha_1 = a(x+\nu t)^4 - b(x+\nu t)^2y^2 + cy^4 - 1 = 0, \tag{5.2}$$

$$\alpha_2 = z - d\left(y^2 + (x+\nu t)^2\right) + e = 0, \tag{5.3}$$

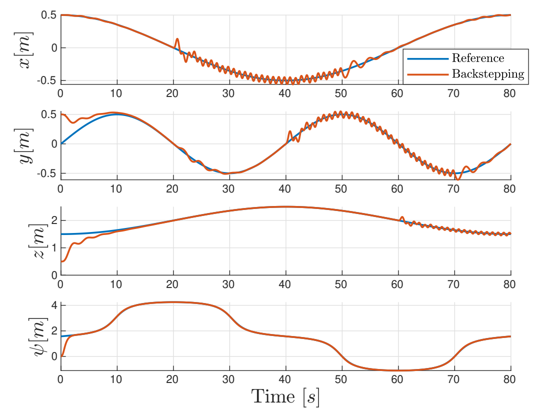


(a) Backstepping control with integral action.

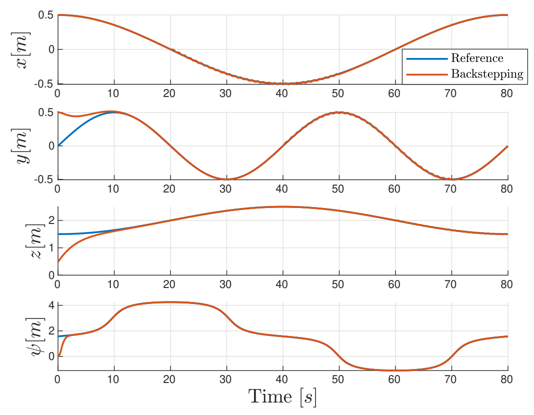


(b) Robust backstepping control with integral action.

Figure 5.1: Reference and actual trajectories.

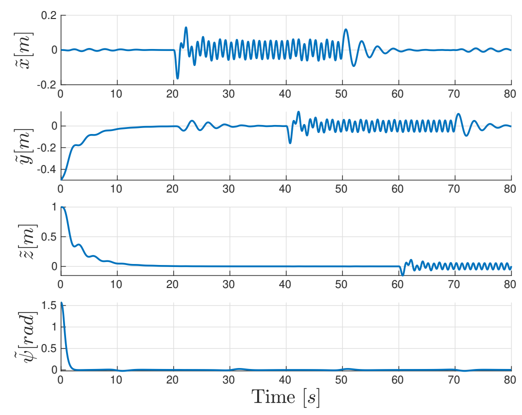


(a) Backstepping control with integral action.

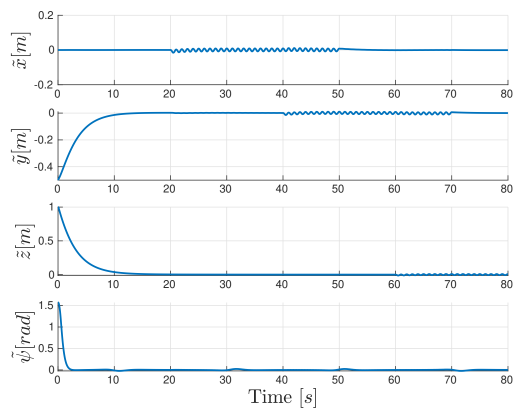


(b) Robust backstepping control with integral action.

Figure 5.2: Evolution of states.

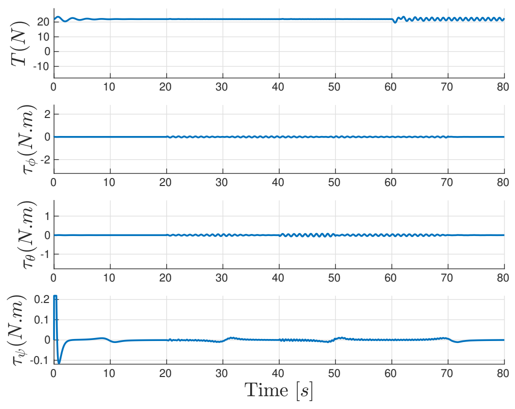


(a) Backstepping control with integral action.

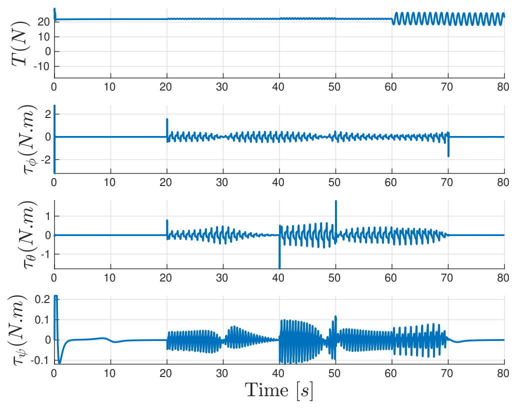


(b) Robust backstepping control with integral action.

Figure 5.3: Error evolution.



(a) Backstepping control with integral action.



(b) Robust backstepping control with integral action.

Figure 5.4: Control signals evolution.

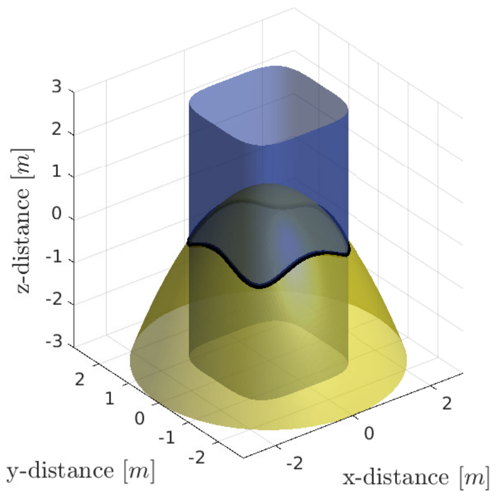


Figure 5.5: Surfaces $\alpha_1 = 0$ and $\alpha_2 = 0$.

in which α_1 defines a cylindrical surface and α_2 defines a slightly modified paraboloid. Initially, the constants are chosen as: $a = 0.3$, $b = 0.03$, $c = 0.3$, $d = -0.5$ and $e = -1$, with a velocity of $0.04\,m/s$ in the positive sense of the x-axis, in this case, $\nu = -0.04\,m/s$. Regarding the collision avoidance strategy, $r = 0.3\,m$, $\tilde{r} = 0.6\,m$, $R = 1.4\,m$ and $\epsilon = 0.1\,m$ are considered. Figure 5.5 shows the surfaces defined by α_1 and α_2 when $t = 0$. In the same figure, the target curve Γ is highlighted, given by the intersection of both surfaces.

To compose the cascaded system, the loops are executed at different frequencies. The outermost loop, which corresponds to the high-level control, runs at the frequency of $10\,Hz$. The innermost loop, which corresponds to the low-level controller, runs at the frequency of $100\,Hz$. The dynamics of each aerial vehicle is integrated in $1\,ms$ steps.

For the simulated aircraft, a maximum speed of $v_{max} = 0.4\,m/s$ is considered. As in the Section 5.1.1, the low-level controller of each vehicle is adjusted by try and error in the

Table 5.4: External disturbances actuation.

Coordinate	Shape	Time (s)
x	$4\sin(2\pi t)\ N$	$20 < t \leq 30$
y	$4\sin(2\pi t)\ N$	$30 < t \leq 40$
z	$4\sin(2\pi t)\ N$	$40 < t \leq 50$

sequence in which the parameters appear on the backstepping process, while regarding fast convergence to the curve and robustness to disturbances. The controller is adjusted to the following values:

$$\begin{aligned} &\boldsymbol{k}_1 = \text{diag}(3, 2, 4.5), \quad \boldsymbol{k}_2 = \text{diag}(5, 5, 5), \quad \boldsymbol{k}_{\mathcal{X}\xi} = \text{diag}(25, 25, 25), \quad \boldsymbol{k}_3 = \text{diag}(5, 5, 5), \\ &\boldsymbol{k}_4 = \text{diag}(35, 35, 35), \quad k_5 = 15, \quad k_6 = 60, \quad k_{\mathcal{X}\psi} = 25, \quad \varsigma = 20, \quad \zeta = 4.5. \end{aligned} \tag{5.4}$$

Besides, the UAVs have the same parameters presented in Table 5.1, and the control signals are saturated as described in Table 5.2.

In the first simulation, presented in Figures 5.6a and 5.6b, a set of 4 quadrotors is guided by the vector field produced from the functions presented in Equations (5.2) and (5.3), with the parameters already mentioned. In this particular simulation, the backstepping controller actuates without the additional control law designed. In a second simulation, presented in Figures 5.6c and 5.6d, the same vector field is considered, but the robustification control law is activated. In all the simulations the system is affected by time-varying force disturbances with maximum magnitudes of $4\,N$, as shown in Table 5.4. The simulations performed can be watched in `https://youtu.be/kvshsF-cojY`.

Note that the trajectories performed in Simulation 2 (Figures 5.6c and 5.6d) present fewer oscillations than the trajectories performed in Simulation 1 (Figures 5.6a and 5.6b). Also, the oscillations introduced in the convergence function due to external disturbances are reduced by using the robust controller. It is also worth noting that in both simulations the system behaves similarly to oscillations introduced in the z-axis (between 40 and $50s$). As remarked previously, altitude control of a quadrotor vehicle is more easily accomplished due to the arrangement of thrusters.

Even with the addition of the robustification control law, it can be seen from Figure 5.6d that spikes in the convergence function periodically occur. Such peaks are caused by changes of direction in the reference trajectories and are accentuated by the disturbances.

In a second moment, a third simulation with a set of 8 robots is addressed, and its simulation results are presented in Figures 5.6e and 5.6f. For the curve to support a larger number of robots, the parameters of the α functions are adjusted as: $a = 0.15$, $b = 0.03$, $c = 0.15$, $d = -0.3$ e $e = -1.25$. In addition to the disturbances shown in Table 5.4, a variation of the quadrotor mass and inertia parameters of +20% is included. In this simulation, the robots are started in close positions to verify the effectiveness of the collision avoidance strategy.

It can be seen from Figure 5.6e that the robots perform irregular trajectories at the beginning of the simulation, converging to the desired curve over time. Figure 5.6f shows an increase in convergence function in the first few seconds of simulation. The proximity between robots does not allow most robots to move, causing the distance to the curve to grow, since the curve moves in the positive direction of the x-axis. However, as the collision avoidance situations are solved, the robot set converges to the curve.

5.2 Parameterized Distributed Model Predictive Control

In Chapter 4, a distributed parameterized MPC strategy is proposed. In this strategy, a parameterized control is embedded in an optimal control problem, which is solved in a distributed way with the ADMM. In the same chapter, an application of the developed control strategy is proposed, in which a group of robots must converge and circulate a simple curve while avoiding collisions. This section presents numerical results for the proposed application with the distributed parameterized MPC framework.

In order to observe the complete system operation, a simulation with twelve agents is proposed. As target, consider the quartic plane curve described implicitly by the function

$$\alpha_1(x, y) = ax^4 + bx^2y^2 + cy^4 - 1 = 0, \tag{5.5}$$

in which $a = 0.5$, $b = -0.6$ and $c = 0.3$. Figure 5.7 displays the target curve.

The agents are guided by the control law presented in (4.18) in the form of

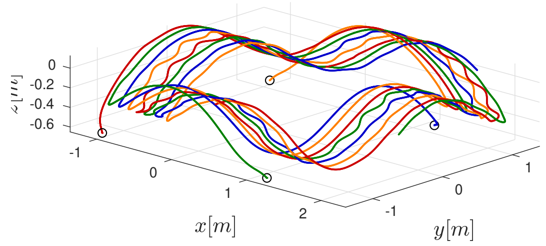
$$\boldsymbol{\kappa}_i(\boldsymbol{\xi}_i, \boldsymbol{\theta}_i) = \boldsymbol{\theta}_{i,1} \tanh(-\alpha_1)\frac{\nabla\alpha_1}{||\nabla\alpha_1||} + \boldsymbol{\theta}_{i,2}\frac{\nabla_H\alpha_1}{||\nabla_H\alpha_1||}, \tag{5.6}$$

in which the parameters $\boldsymbol{\theta}_{i,1}$ and $\boldsymbol{\theta}_{i,2}$ are chosen according to the optimization problem to be set. The control laws' parameters are bounded by $-1 \leq \boldsymbol{\theta}_{i,1} \leq 1$ and $-1 \leq \boldsymbol{\theta}_{i,2} \leq 1$, $\forall i$, which allows the definition of a maximum velocity. Remember that by allowing negative values for $\boldsymbol{\theta}_{i,1}$ and $\boldsymbol{\theta}_{i,2}$, a robot can get away from the curve and alternate the circulation sense in the presence of collision risk. In order to measure the convergence of the robots to the curve, consider $W(\mathbf{x}(t))$

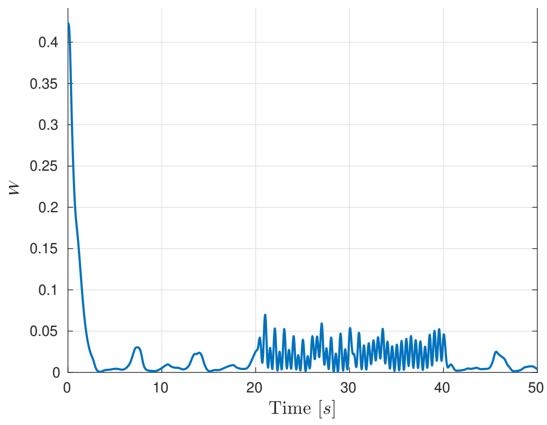
$$W(\mathbf{x}(t)) = \sum_{i=1}^{N} \alpha_1^2(\boldsymbol{\xi}_i(t)), \tag{5.7}$$

in which $\mathbf{x}(t) = [\boldsymbol{\xi}_1^T(t) \dots \boldsymbol{\xi}_n^T(t)]^T$.

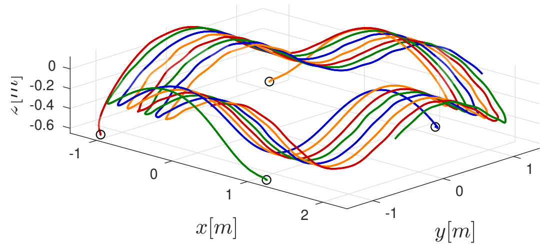
In the first moment, the MPC problem is solved in a centralized way, in order to validate the task formulation as an optimal control problem. Consider the OCP described



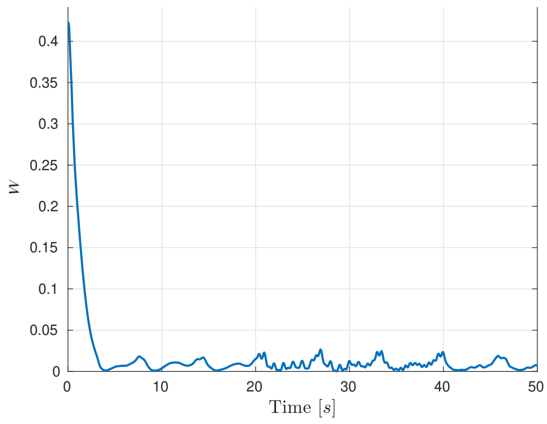
(a) Simulation 1 – Trajectories.



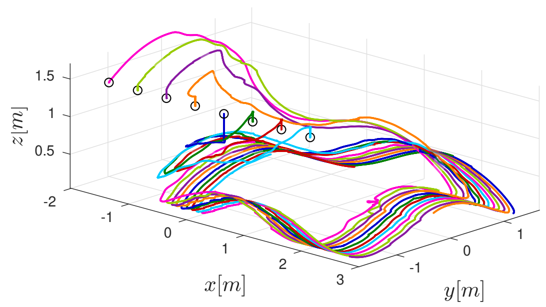
(b) Simulation 1 – Function W.



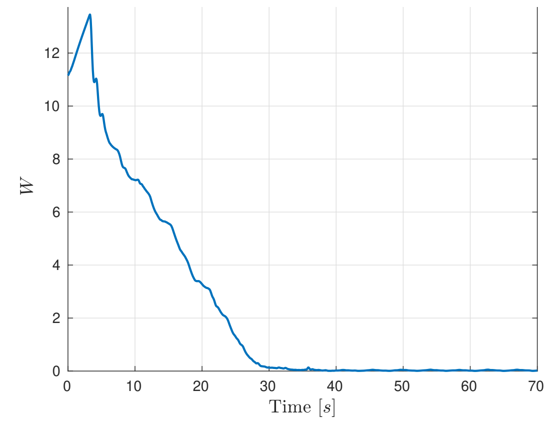
(c) Simulation 2 – Trajectories.



(d) Simulation 2 – Function W.



(e) Simulation 3 – Trajectories.



(f) Simulation 3 – Function W.

Figure 5.6: Trajectories and convergence functions W in the simulations.

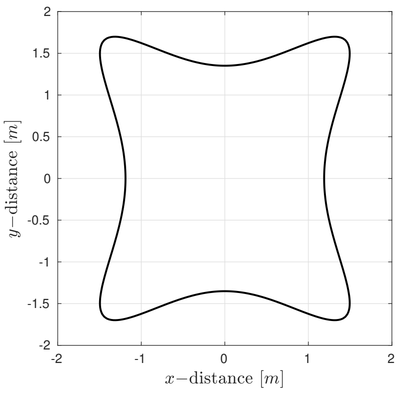


Figure 5.7: Target curve.

in the Section 4.5, in which the error between the real and the desired parameters $\boldsymbol{\theta}_d$ and a collision avoidance term are taken into account. For a set of robots Ω, the global optimization problem is given by

$$\begin{aligned}
&\min_{\boldsymbol{x}_i, \boldsymbol{\theta}_i \forall i} \sum_{i=1}^{N} \int_{t_0}^{t_f} \left(\eta_1 (\boldsymbol{\theta}_{i,1} - \boldsymbol{\theta}_{d,1})^2 + \eta_2 (\boldsymbol{\theta}_{i,2} - \boldsymbol{\theta}_{d,2})^2 + \eta_3 \sum_{j \in \Theta_i} w(\boldsymbol{\xi}_i(t), \boldsymbol{\xi}_j(t)) \right) dt \\
&\text{subject to} \\
&\dot{\boldsymbol{\xi}}_i = \boldsymbol{\kappa}_i(\boldsymbol{\xi}_i, \boldsymbol{\theta}_i), \\
&\bar{\boldsymbol{x}}_i(t_0) = \bar{\boldsymbol{x}}_{i,t_0}, \\
&\begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{\theta}_{i,1} \\ \boldsymbol{\theta}_{i,2} \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \forall i \in \{1, ..., N\}.
\end{aligned} \tag{5.8}$$

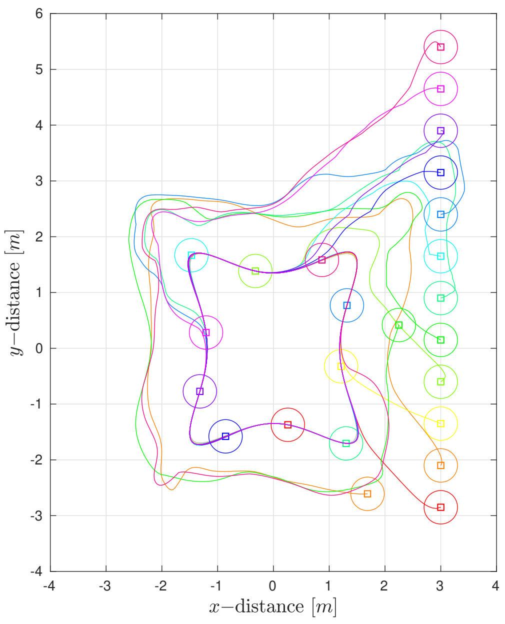
in which the i-th state vector is given by $\boldsymbol{x}_i = \boldsymbol{\xi}_i$ and $t_f = t_0 + \Delta$. The collision avoidance term $w(\boldsymbol{\xi}_i(t), \boldsymbol{\xi}_j(t))$ might be given by

$$\text{Scheme 1: } w_e(\boldsymbol{\xi}_i(t), \boldsymbol{\xi}_j(t)) = \exp\left(-\zeta \left((\boldsymbol{\xi}_i(t) - \boldsymbol{\xi}_j(t))^T (\boldsymbol{\xi}_i(t) - \boldsymbol{\xi}_j(t)) + \epsilon_e \right)\right), \tag{5.9}$$

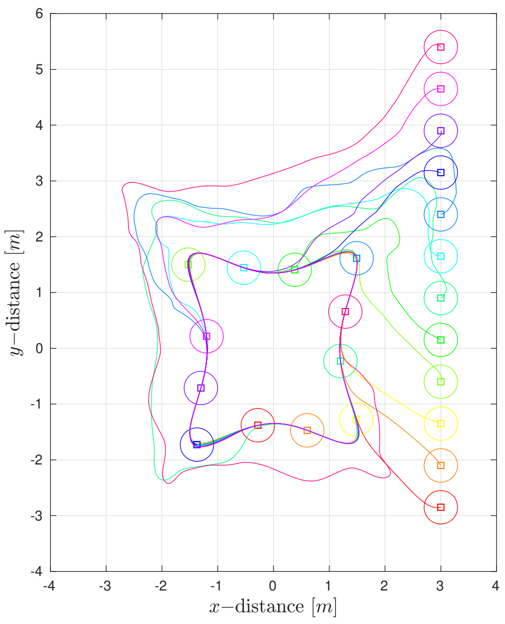
or by

$$\text{Scheme 2: } w_s(\boldsymbol{\xi}_i(t), \boldsymbol{\xi}_j(t)) = [(2r + \epsilon_s)^2 - (\boldsymbol{\xi}_i(t) - \boldsymbol{\xi}_j(t))^T (\boldsymbol{\xi}_i(t) - \boldsymbol{\xi}_j(t))]_+^2. \tag{5.10}$$

The total simulation time is set $40\,s$ with step size $dt = 0.1\,s$, and the prediction horizon is set $\Delta = 1\,s$. The optimization problems are implemented with the symbolic framework CasADi (Andersson et al., 2019) and solved with the Ipopt package (Wächter & Biegler, 2006), an interior-point solver. In this case, the optimization problem is solved in its discrete form given by the Euler approximation. Each robot has radius $r = 0.3\,m$ and perception range of $R = 1.5\,m$. In addition, the optimization weights and variables are chosen as $\eta_1 = 1$, $\eta_2 = 2$, $\eta_3 = 50$, $\zeta = 10$, $\epsilon_e = 4r^2$, $\epsilon_s = 0.2\,m$, and $\boldsymbol{\theta}_{d,1} = \boldsymbol{\theta}_{d,2} = 0.5$. It is



(a) Exponential repulsion.



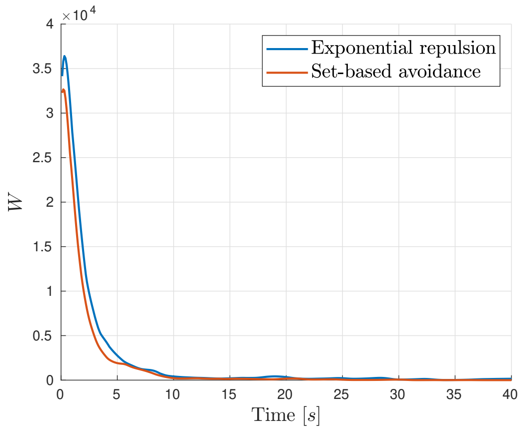
(b) Set-based avoidance.

Figure 5.8: Trajectories of the centralized problem with single-integrator dynamics.

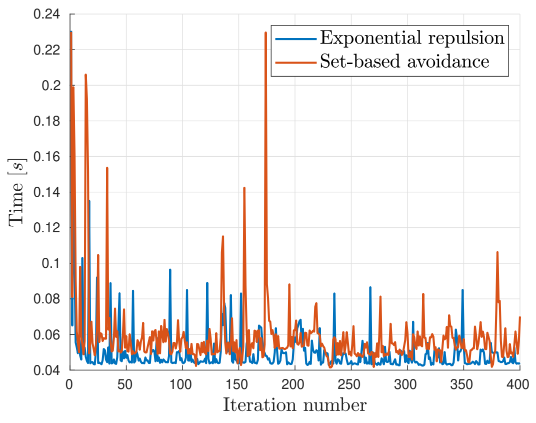
important to remark that the weights and parameters were tuned by choosing the maximum weight on the collision avoidance that does not compromise the solver convergence. The centralized problem simulations can be watched in `https://youtu.be/6OZmkQErPhI`.

The first two simulations are presented in Figure 5.8, in which Figure 5.8a shows the trajectories performed using the exponential-based repulsion and Figure 5.8b the trajectories performed using the set-based collision avoidance. These figures also show the initial and ending position of each robot. The trajectory evolution indicates that the convergence with the set-based avoidance might be slightly faster, since all robots end over the curve. Besides, a better distribution over the curve is expected from the exponential method, since a repulsion force appears when a robot sense another one. Note that the exponential method tends to be harder to tune, because it has one additional parameter.

Regarding the evolution of the convergence of the set of robots to the curve in both simulations, Figure 5.9a shows that convergence with the set-based method is slightly faster, but it is mostly equivalent from $10\,s$ to the end time. A comparison between the solving time of the optimization problems for both collision avoidance schemes is shown in Figure 5.9b. The first MPC cycle optimization problem is the one that takes longer to be solved. After that, the problems are warm started with the solution of the previous optimization problem. As one can note, there is no much difference between the processing time using both techniques. Disregarding the time to solve the first OCP and computing the average of the remaining values, the problem with exponential repulsion is solved in $0.0493s$ and the set-based is solved in $0.0585s$, which corresponds to a difference of 18.5%.



(a) Convergence function evolution.



(b) Processing time per optimization problem.

Figure 5.9: Single-integrator centralized problem comparison: exponential repulsion and set-based avoidance.

Furthermore, in the worst case, the problem with exponential repulsion took $0.1456s$ to be solved and the set-based took $0.2296s$. Remember that in the implementation of the centralized problem simulation, a unique optimization problem is solved per MPC cycle.

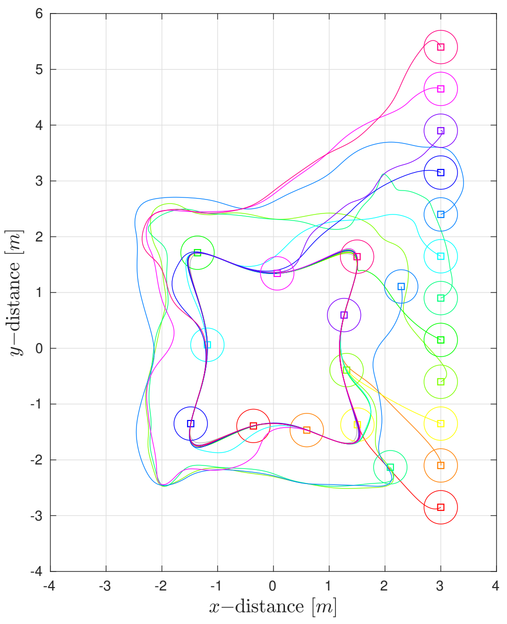
Next, the double-integrator dynamics are considered. As discussed in Section 4.4, the vector field control law might be chosen as

$$\ddot{\boldsymbol{\xi}}_i = J_{\boldsymbol{\xi}_i}\dot{\boldsymbol{\xi}}_i + \boldsymbol{\sigma}(\dot{\boldsymbol{\xi}}_i - \boldsymbol{\kappa}_i). \tag{5.11}$$

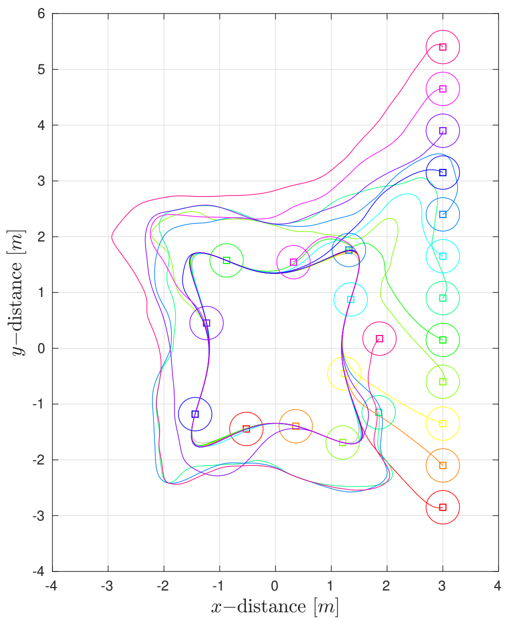
Therefore, the optimal control problem can be written as

$$\begin{aligned}
&\min_{\boldsymbol{x}_i,\boldsymbol{\theta}_i\forall i} \sum_{i=1}^{N} \int_{t_0}^{t_f} \left(\eta_1(\boldsymbol{\theta}_{i,1} - \boldsymbol{\theta}_{d,1})^2 + \eta_2(\boldsymbol{\theta}_{i,2} - \boldsymbol{\theta}_{d,2})^2 + \eta_3 \sum_{j\in\Theta_i} w(\boldsymbol{\xi}_i(t), \boldsymbol{\xi}_j(t)) \right) dt \\
&\text{subject to} \\
&\ddot{\boldsymbol{\xi}}_i = J_{\boldsymbol{\xi}_i}\dot{\boldsymbol{\xi}}_i - k_v(\dot{\boldsymbol{\xi}}_i - \boldsymbol{\kappa}_i), \qquad\qquad (5.12)\\
&\bar{\boldsymbol{x}}_i(t_0) = \bar{\boldsymbol{x}}_{i,t_0}, \\
&\begin{bmatrix}-1\\-1\end{bmatrix} \leq \begin{bmatrix}\boldsymbol{\theta}_{i,1}\\ \boldsymbol{\theta}_{i,2}\end{bmatrix} \leq \begin{bmatrix}1\\1\end{bmatrix} \qquad \forall i \in \{1, ..., N\}.
\end{aligned}$$

in which $\boldsymbol{x}_i = [\boldsymbol{\xi}_i^T\ \dot{\boldsymbol{\xi}}_i^T]^T$, $k_v > 0$ is a constant, and $w(\boldsymbol{\xi}_i(t), \boldsymbol{\xi}_j(t))$ is given by (5.9) or (5.10). For the problem with double-integrator dynamics, consider $k_v = 2$ and the same parameters of the single-integrator problem. Figure 5.10 shows the centralized problem trajectories using the exponential repulsion and the set-based avoidance. One can note that for each simulation, two robots are out of the curve in the end time. Figure 5.11a shows that the set-based method is again slightly faster in convergence. Regarding processing time, presented in Figure 5.11b, the two collision avoidance approaches do not present visually

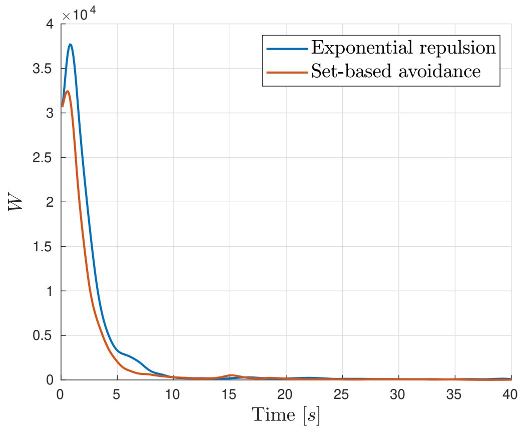


(a) Exponential repulsion.

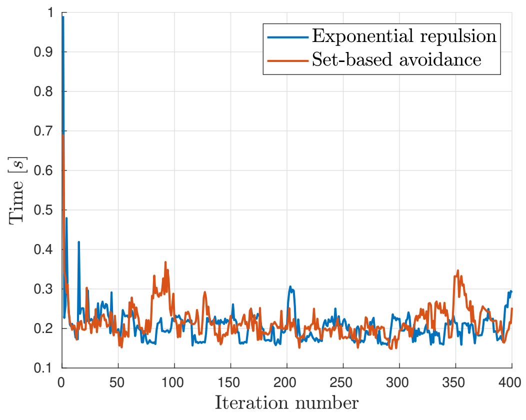


(b) Set-Based Avoidance.

Figure 5.10: Trajectories of the centralized problem with double-integrator dynamics.



(a) Convergence function evolution.



(b) Processing time per optimization problem.

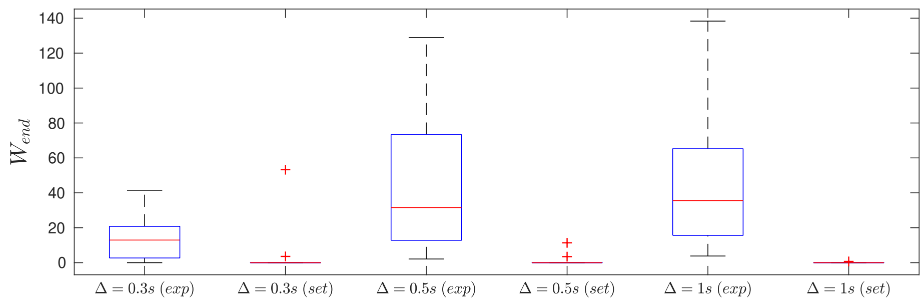
Figure 5.11: Double-integrator centralized problem comparison: exponential repulsion and set-based avoidance.

much difference. If we look at the averages, disregarding the first optimization problem, the problem with exponential repulsion takes $0.2049s$ to be solved, and the problem with set-based avoidance takes $0.2162s$ to be solved, which is a 5.5% difference. Looking for the worst cases, the problem with exponential repulsion took $0.4793s$ to be solved and the problem with set-based avoidance took $0.3682s$. Note that, in general, the double-integrator problem requires more processing time to be solved in relation to the single-integrator, due to the increased number of variables in the problem.

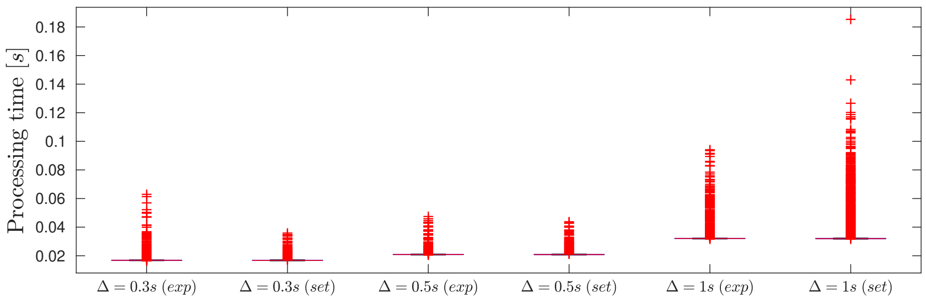
Next, the global problems are distributed using the ADMM strategy described in Section 4.3. To obtain a better perception of the system performance, a set of simulations is proposed. In each simulation, twelve robots are placed in random initial positions in the space limited by $-4 < x < 4$ and $-4 < y < 6$, in which a small safe distance between the robots is respected. We test the system using the exponential repulsion and the set-based avoidance, while choosing the prediction horizon as $\Delta = 0.3s$, $\Delta = 0.5s$, and $\Delta = 1s$. The parameters of each simulation are the same described previously, and ADMM penalty parameter is set $\rho = 4$ with the number of iterations per MPC cycle $n_{\max} = 5$. For the single-integrator dynamics, 30 simulation trials are executed for each combination of strategies, and for the double-integrator dynamics, 15 trials are performed for each combination. The value of the convergence function in the final time $t_f = 40s$ and the time to solve each local optimization problem are stored and presented in boxplots. In the first case, the same curve previously presented is considered (Equation (5.5)), with $a = 0.5$, $b = -0.6$ and $c = 0.3$.

Figure 5.12 presents the stored data for the single and double-integrator dynamics simulations. When considering single-integrator dynamics, no collision happened. In Figure 5.12a, the value of the convergence function at the end time for each set of simulations is presented with boxplots. One can note that the set-based avoidance performs considerably better in these cases. Regarding the processing time, Figure 5.12b shows that its values grow with the prediction horizon. No visually relevant difference is observed between exponential and set-based avoidance processing time. When solving the problem with double-integrator dynamics, collisions happened with the set-based avoidance in 15 out of 15 simulations with $\Delta = 0.3s$, and in 5 out of 15 simulations with $\Delta = 0.5s$. Figures 5.12c and 5.12d show the stored data of the successful simulations with double-integrator dynamics. One can note that the end value of the convergence function with $\Delta = 1s$ with exponential repulsion is the greatest among the presented groups of simulations, which indicates that the parameters could be possibly better tuned regarding such horizon. As one could expect, more time is necessary to compute the double-integrator problem in relation to the single-integrator one.

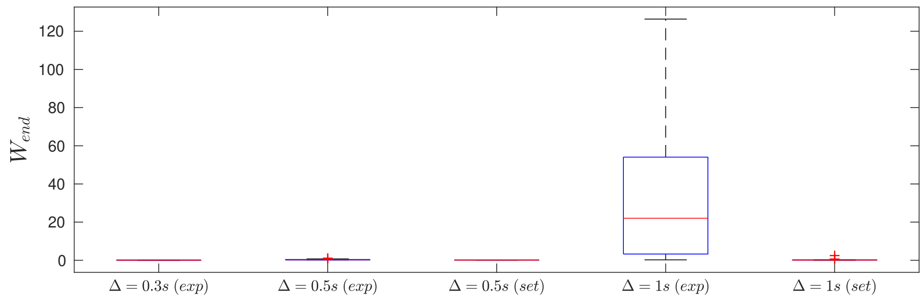
The problem in which twelve robots must converge and circulate the curve previously described is considerably hard to solve, since the robots are tight over the curve. To observe the system operation in an easier to solve condition, a curve with $a = 0.125$, $b = -0.15$ and $c = 0.075$ is proposed. Figure 5.13 shows the end time positions of a 12 robots simulation in the new, bigger curve, where the robots are more widely spaced than in the previous case. Besides, Figure 5.14 shows the results obtained with this curve. For the single-integrator dynamics, once more no collision happened. In Figure 5.14a, one can note that in all simulations the convergence value at the end time is close to zero. For the double-integrator dynamics, which convergence end values are shown in Figure 5.14c, collisions happened in 14 out of 15 simulations with $\Delta = 0.3s$ with set-based avoidance.



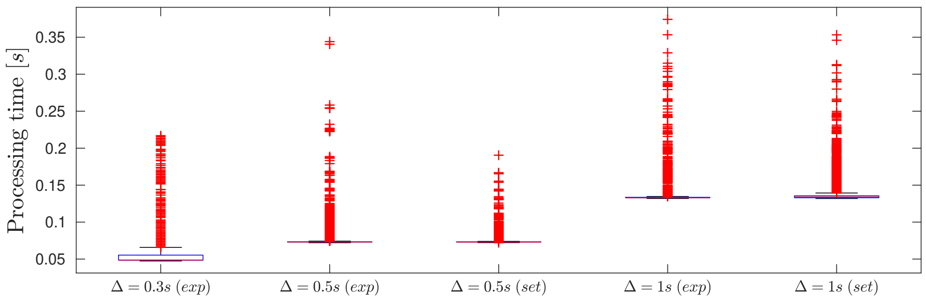
(a) Single-integrator – Convergence function.



(b) Single-integrator – Processing time of local problems.



(c) Double-integrator – Convergence function.



(d) Double-integrator – Processing time of local problems.

Figure 5.12: Distributed problem – small curve.

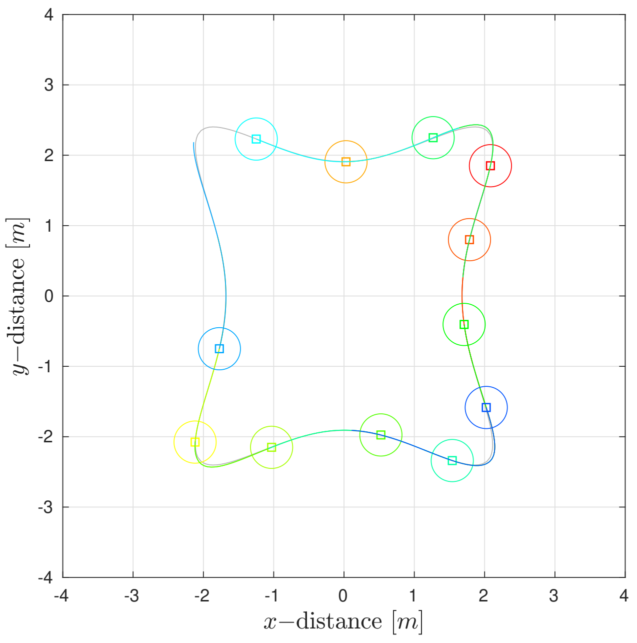
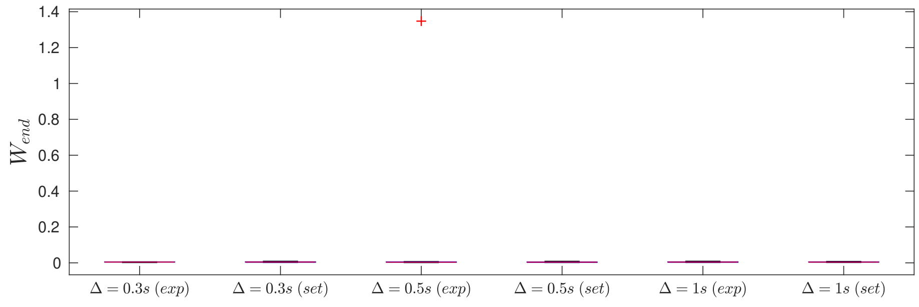


Figure 5.13: 12 robots simulation with a bigger curve – End position.

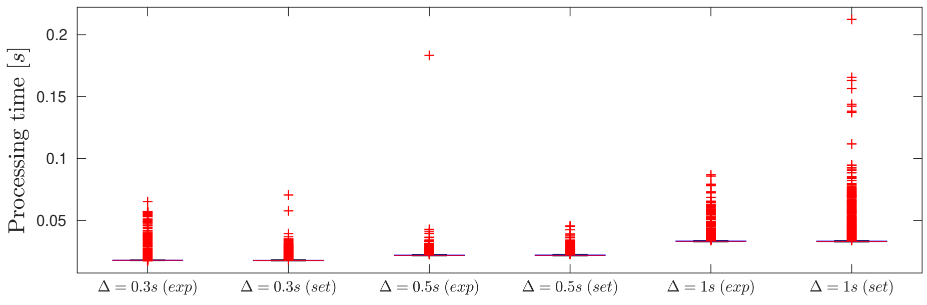
The processing time, presented in Figures 5.14b and 5.14d, scales with the prediction horizon. Note that for both curves, the set-based avoidance did not perform well for $\Delta = 0.3s$ when considering double-integrator dynamics.

In front of the presented results, one can note that the exponential repulsion presents a more robust behavior for collision avoidance. As the repulsion force starts to actuate when a robot enters the perception range of another robot, the method is more conservative than the set-based avoidance, that only deals with imminent collision risk. Therefore, the tuning of the cost functional weights when considering set-based avoidance must be done carefully. Another important aspect to note is that the processing time of the optimization problems must be reduced in order to make practical experiments with the scheme. As in the cases presented $n_{max} = 5$, for a practical implementation each agent must compute five optimization problems and carry out ten communication steps per control cycle. Among the simulations presented, the minimum average processing time is about $0.02s$, which would sum in average $0.1s$ of processing time per control cycle, without considering the communication time. Therefore, the scheme is not fast enough for considering $0.1s$ control cycles in practical problems. Note that this study did not take into account the worst case scenario for the solving time of each optimization problem, since no real time simulation was considered and no care was taken in order to guarantee maximum priority for solving the optimization problem in the machine. However, by gathering the processing time of each optimization problem in box plots, we expect to obtain a notion of the applicability of the DMPC scheme in a real time simulation or experiment.

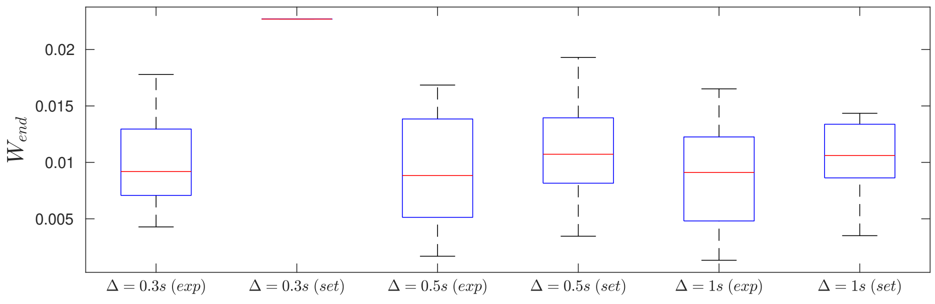
The next simulation is an attempt of stressing the system with a number of robots that do not fit over the curve. For doing that, consider 20 robots and the same small curve



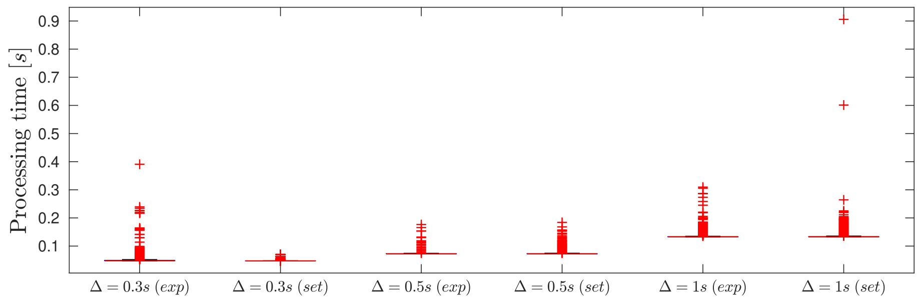
(a) Single-integrator – Convergence function.



(b) Single-integrator – Processing time of local problems.

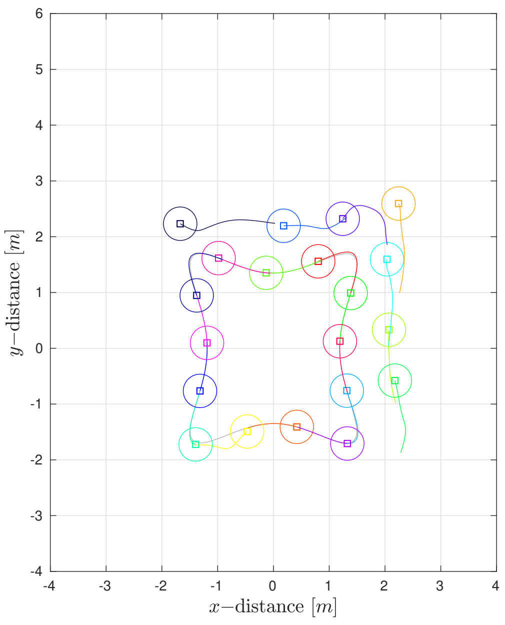


(c) Double-integrator – Convergence function.

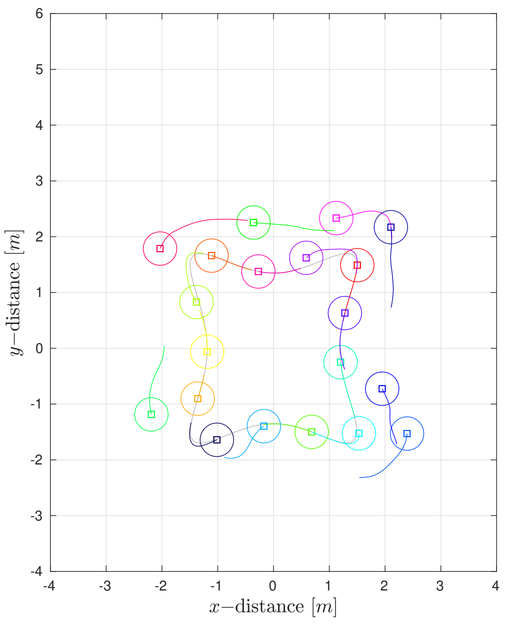


(d) Double-integrator – Processing time of local problems.

Figure 5.14: Distributed problem – big curve.



(a) Single-integrator dynamics.

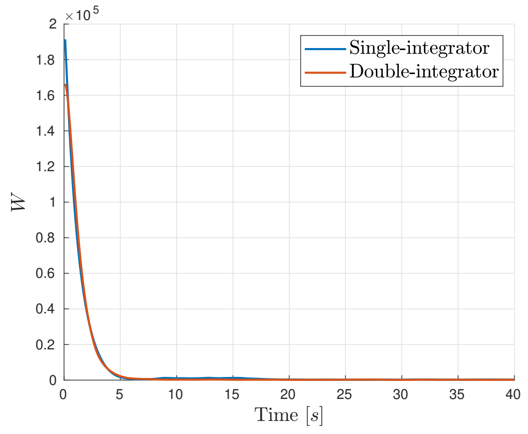


(b) Double-integrator dynamics.

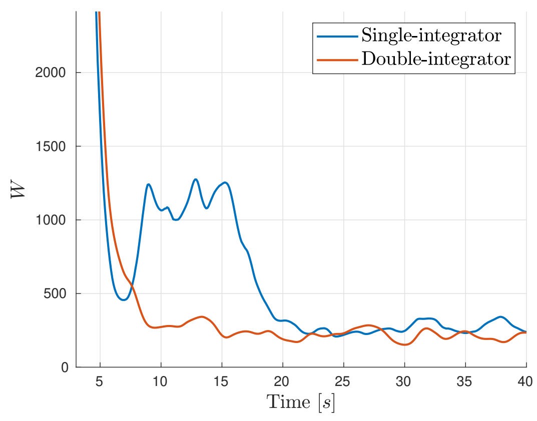
Figure 5.15: 20 robots simulation – End position.

previously proposed, with $a = 0.5$, $b = -0.6$ and $c = 0.3$. Note that number of robots that fit over a curve depends on the radius of the robots and on the geometric properties of the curve. The robot initial positions are random and the set-based avoidance is applied. Figure 5.15 shows the end position of each robot for single and double-integrator dynamics. One can note that in both simulations a set of robots do not fit in the curve and keep circulating around the layer of robots over the curve. In Figure 5.16 one can observe the evolution of the convergence function. As presented in Figure 5.16a, the convergence function values rapidly decays. Figure 5.16b shows that after $20s$ of simulation, the convergence function oscillates around a value. As one can note in the same figure, the convergence function value is not monotonic decreasing, since negative values for the parameters are allowed, so that a robot can move away from the target curve in a collision risk situation. The simulation video can be watched in https://youtu.be/nOdIeVToHGw.

Numerical simulations are also performed considering localization errors. For that, at each time step the real position of a robot is disturbed by a white Gaussian noise, with power of $-23\,dB$, $-18\,dB$ and $-13dB$. The simulation video can be watched in https://youtu.be/zGQfJV-OepM. Collisions and optimization errors occurred when disturbing the positions with the greatest noise. One can also note that the trajectories performed by the double-integrator agents are smoother than the ones performed by the single-integrators, once that the actuation on the acceleration level works as a filter in this case.



(a) Overview.



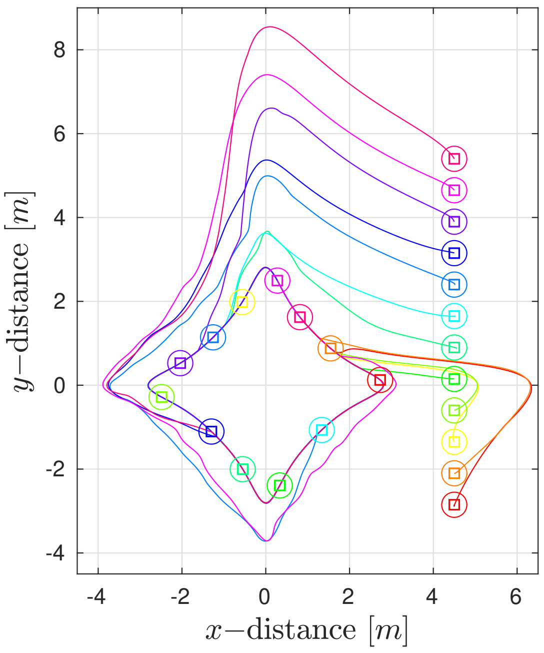
(b) Zoom.

Figure 5.16: 20 robots simulation — Convergence function evolution.

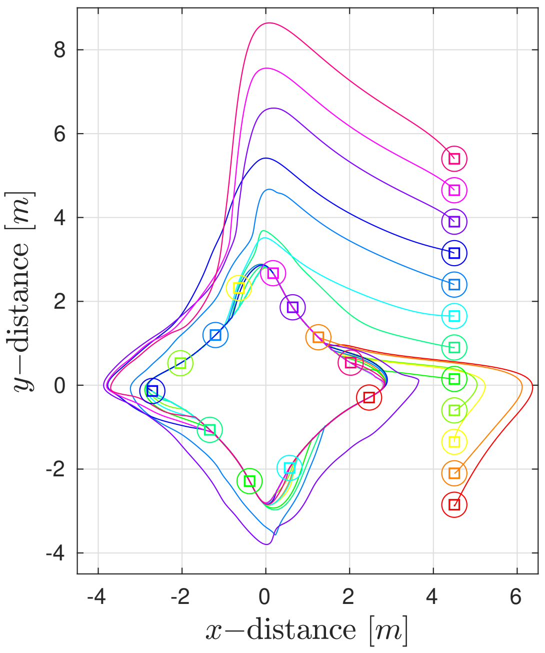
The same system is also tested with a star-shaped curve, defined implicitly by

$$\alpha_2 (x, y) = a_0 y^6 + a_1 x^2 y^4 + a_2 x^4 y^2 + a_3 x^6 - c^6 = 0, \tag{5.13}$$

in which $a_0 = 0.05$, $a_1 = 4$, $a_2 = 4$, $a_3 = 0.05$ and $c = 1.7$. Figure 5.17 shows the trajectory performed by the robots considering single and double-integrator dynamics. In this case, an exponential based repulsion is applied. An important point to note is that as the complexity of the curve increases, the optimization problem might become harder, therefore taking longer time to solve. The simulation video can be watched in https://youtu.be/e2ERo_HfDpA.



(a) Single-integrator dynamics.



(b) Double-integrator dynamics.

Figure 5.17: Star shaped curve – Trajectories.

6
Conclusion

6.1 Summary

This dissertation addressed the problem of multi-robot coordination applied to convergence and circulation of closed paths by two different strategies. In the first part, a decentralized strategy for coordination of multiple agents with predefined collision avoidance laws was studied. The parameters of the vector field control law are tuned in each time step in order to avoid inter-robot collisions, based on predefined priority rules known by every agent. Besides considering closed curves in the 3-D space, the proposed approach considers also time-varying curves. To implement the high-level strategy in quadrotor UAVs, a low level controller based on the backstepping technique is proposed. By applying the Lyapunov redesign technique, the backstepping controller robustifies the closed loop system in front of unknown but bounded disturbances.

The second part of this work presents the development of a distributed and parameterized model predictive control framework. The main characteristic of this framework is to take advantage of predefined, well established control laws for certain systems. Besides, the distribution of the optimal control problem allows trajectory negotiation between agents, with the aim of minimizing a global cost. Numerical results show the applicability of the framework in a convergence and circulation problem of curves in 2-D spaces considering inter-robot collision avoidance with two different schemes.

The strategy discussed in the first part of this work is very efficient in providing collision avoidance, and as the problem structure is decentralized, minimum communication is required. However, the same strategy requires the strong assumption that a robot can always avoid collisions by moving in the direction tangent to the aimed curve. Besides, no notion of optimality on performing the task is taken into account.

In the strategy presented in the second part, collision avoidance and task accomplishment are measured with a cost functional and the dynamics of each robot are taken into account as constraints. Besides presenting interesting results, the optimal control strategy also suffer of some drawbacks. First, to guarantee stability, it is necessary to assure that every optimization problem converges to a solution, which is not treated in this work. The vanilla ADMM procedure also requires synchronous communication between the agents. Therefore, a distributed optimization problem is subject to the response time of the slower processor in the distributed scheme, which can complicate practical implementations and brings the necessity of fast optimization methods. In fact, when dealing with real time systems, we ideally need solvers with good worst case estimates of the execution time (Bemporad & Patrinos, 2012), which is not treated in the present work. In view of this, there are many possibilities for extending the current work, and the field of distributed MPC for multi-robot systems in general.

6.2 Future Works

To corroborate the proposed strategies in this work, experimentation with real robots would add great value on their evaluation. In both approaches, we assume perfect communication, that is, the communication is synchronous and there is no packet loss. Implementing robuster strategies regarding communication might be important for the realization of real experiments.

To perform experiments with the cascaded strategy, it is first necessary to implement the backstepping controller and high-order state estimators in a processing board to be embedded in the quadrotor. To experiment with the complete system, it is also necessary to guarantee that each robot updates the position of the neighboring robots in reasonable frequency for avoiding collisions.

Regarding the parameterized DMPC framework, future works might also focus on the stability proof of the scheme and in the collision avoidance approach, looking forward to establish clear directions to define cost weights to achieve collision avoidance via optimization. Besides, the work can also be extended by implementing low-level optimization approaches that can render the system faster, as well as by exploring asynchronous solutions to distributed optimization problems, which would make practical implementations simpler.

Furthermore, both strategies can be extended by considering collision avoidance with obstacles. In the case of the cascaded system, this could be implemented by augmenting the high-level control law with repulsion fields emanating from the obstacles. For the DMPC scheme, obstacle avoidance can be achieved by augmenting the stage cost the same way as we did for inter-robot collision avoidance. Another approach would be to consider hard-constraints on the optimal control problem.

Also, it would be interesting to investigate distributed strategies for tracking curves with groups of fixed-wing aircraft. During the development of this work, we implemented a minimum velocity constraint on the DMPC strategy, in order to obtain a trajectory suitable for a fixed-wing aircraft. However, in most situations the optimization solver was not able to find a solution for the OCP.

Bibliography

Alamo, T., Limon, D., & Krupa, P. (2019). Restart FISTA with Global Linear Convergence. In 2019 18th European Control Conference (ECC) (pp. 1969–1974).: IEEE.

Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi: a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation, 11(1), 1–36.

Bemporad, A. & Patrinos, P. (2012). Simple and certifiable quadratic programming algorithms for embedded linear model predictive control. IFAC Proceedings Volumes, 45(17), 14–20.

Bouabdallah, S., Murrieri, P., & Siegwart, R. (2004a). Design and control of an indoor micro quadrotor. In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, volume 5 (pp. 4393–4398).: IEEE.

Bouabdallah, S., Noth, A., & Siegwart, R. (2004b). PID vs LQ control techniques applied to an indoor micro quadrotor. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 3 (pp. 2451–2456).: IEEE.

Bouabdallah, S. & Siegwart, R. (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In Proceedings of the 2005 IEEE international conference on robotics and automation (pp. 2247–2252).: IEEE.

Bouabdallah, S. & Siegwart, R. (2007). Full control of a quadrotor. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 153–158).: IEEE.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning, 3(1), 1–122.

Briñón-Arranz, L., Seuret, A., & Pascoal, A. (2019). Circular formation control for cooperative target tracking with limited information. Journal of the Franklin Institute, 356(4), 1771 – 1788.

Burgard, W., Moors, M., Stachniss, C., & Schneider, F. E. (2005). Coordinated multi-robot exploration. IEEE Transactions on robotics, 21(3), 376–386.

Camacho, E. F. & Bordons Alba, C. (2013). Model predictive control. Springer Science & Business Media.

Cao, Y. U., Fukunaga, A. S., & Kahng, A. (1997). Cooperative mobile robotics: Antecedents and directions. Autonomous robots, 4(1), 7–27.

Ceccarelli, N., Di Marco, M., Garulli, A., & Giannitrapani, A. (2008). Collective circular motion of multi-vehicle systems. Automatica, 44(12), 3025–3035.

Chaimowicz, L., Michael, N., & Kumar, V. (2005). Controlling swarms of robots using interpolated implicit functions. In Proceedings of the 2005 IEEE international conference on robotics and automation (pp. 2487–2492).: IEEE.

Colorado, J., Barrientos, A., Martinez, A., Lafaverges, B., & Valente, J. (2010). Mini-quadrotor attitude control based on hybrid backstepping & Frenet-Serret theory. In 2010 IEEE International Conference on Robotics and Automation (pp. 1617–1622).: IEEE.

DeVries, L. & Paley, D. A. (2012). Multivehicle control in a strong flowfield with application to hurricane sampling. Journal of Guidance, Control, and Dynamics, 35(3), 794–806.

Dierks, T. & Jagannathan, S. (2009). Output feedback control of a quadrotor UAV using neural networks. IEEE transactions on neural networks, 21(1), 50–66.

Droge, G. & Egerstedt, M. (2013). Distributed parameterized model predictive control of networked multi-agent systems. In 2013 American Control Conference (pp. 1332–1337).: IEEE.

Droge, G. N. (2014). Behavior-Based Model Predictive Control for Networked Multi-Agent Systems. PhD thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology.

Farokhi, F., Shames, I., & Johansson, K. H. (2014). Distributed MPC via dual decomposition and alternative direction method of multipliers. In Distributed model predictive control made easy (pp. 115–131). Springer.

Farrell, J. A. & Polycarpou, M. M. (2006). Adaptive approximation based control: unifying neural, fuzzy and traditional adaptive approximation approaches, volume 48. John Wiley & Sons.

Fazli, P., Davoodi, A., Pasquier, P., & Mackworth, A. K. (2010). Complete and robust cooperative robot area coverage with limited range. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 5577–5582).: IEEE.

Ferranti, L., Negenborn, R. R., Keviczky, T., & Alonso-Mora, J. (2018). Coordination of multiple vessels via distributed nonlinear model predictive control. In 2018 European Control Conference (ECC) (pp. 2523–2528).: IEEE.

Fox, D., Burgard, W., Kruppa, H., & Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. Autonomous robots, 8(3), 325–344.

Freeman, R. & Kokotovic, P. V. (2008). Robust nonlinear control design: state-space and Lyapunov techniques. Springer Science & Business Media.

Frew, E. W. & Lawrence, D. (2012). Tracking expanding star curves using guidance vector fields. In 2012 American Control Conference (ACC) (pp. 1749–1754).: IEEE.

Frew, E. W. & Lawrence, D. (2017). Tracking dynamic star curves using guidance vector fields. Journal of Guidance, Control, and Dynamics, 40(6), 1488–1495.

Frew, E. W., Lawrence, D. A., Dixon, C., Elston, J., & Pisano, W. J. (2007). Lyapunov guidance vector fields for unmanned aircraft applications. In 2007 American control conference (pp. 371–376).: IEEE.

Frew, E. W., Lawrence, D. A., & Morris, S. (2008). Coordinated standoff tracking of moving targets using Lyapunov guidance vector fields. Journal of guidance, control, and dynamics, 31(2), 290–306.

Gonçalves, V. M., Maia, C. A., Pereira, G. A. S., & Pimenta, L. C. d. A. (2010a). Navegação de robôs utilizando curvas implícitas. Sba: Controle & Automação Sociedade Brasileira de Automatica, 21, 43 – 57.

Gonçalves, V. M., Pimenta, L. C., Maia, C. A., Dutra, B. C., & Pereira, G. A. (2010b). Vector fields for robot navigation along time-varying curves in n-dimensions. IEEE Transactions on Robotics, 26(4), 647–659.

Hong, M., Luo, Z.-Q., & Razaviyayn, M. (2016). Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. SIAM Journal on Optimization, 26(1), 337–364.

Hsieh, M. A., Kumar, V., & Chaimowicz, L. (2008). Decentralized controllers for shape generation with robotic swarms. Robotica, 26(5), 691–701.

Hsieh, M. A., Loizou, S., & Kumar, V. (2007). Stabilization of multiple robots on stable orbits via local sensing. In Proceedings 2007 IEEE International Conference on Robotics and Automation (pp. 2312–2317).: IEEE.

Hu, J., Xu, J., & Xie, L. (2013). Cooperative search and exploration in robotic networks. Unmanned Systems, 1(01), 121–142.

Hwang, E.-J., Kang, H.-S., Hyun, C.-H., & Park, M. (2013). Robust backstepping control based on a Lyapunov redesign for skid-steered wheeled mobile robots. International Journal of Advanced Robotic Systems, 10(1), 26.

Jasim, W. & Gu, D. (2015). Integral backstepping controller for quadrotor path tracking. In 2015 International Conference on Advanced Robotics (ICAR) (pp. 593–598).: IEEE.

Jennings, J. S., Whelan, G., & Evans, W. F. (1997). Cooperative search and rescue with a team of mobile robots. In 1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97 (pp. 193–200).: IEEE.

Jerez, J. L., Goulart, P. J., Richter, S., Constantinides, G. A., Kerrigan, E. C., & Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. IEEE Transactions on Automatic Control, 59(12), 3238–3251.

Jung, W., Lim, S., Lee, D., & Bang, H. (2016). Unmanned aircraft vector field path following with arrival angle control. Journal of Intelligent & Robotic Systems, 84(1-4), 311–325.

Kang, Y. & Hedrick, J. (2006). Design of nonlinear model predictive controller for a small fixed-wing unmanned aerial vehicle. In AIAA Guidance, Navigation, and Control Conference and Exhibit (pp. 6685).

Khalil, H. K. (2002). Nonlinear Systems. Prentice-Hall, New Jersey, 3th edition.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In Autonomous robot vehicles (pp. 396–404). Springer.

Kirk, D. E. (2004). Optimal control theory: an introduction. Courier Corporation.

Lawrence, D., Frew, E., & Pisano, W. (2007). Lyapunov vector fields for autonomous UAV flight control. In AIAA Guidance, Navigation and Control Conference and Exhibit (pp. 6317).

Lee, H. & Kim, H. J. (2017). Trajectory tracking control of multirotors from modelling to experiments: A survey. International Journal of Control, Automation and Systems, 15(1), 281–292.

Lee, J.-W., Walker, B., & Cohen, K. (2011). Path planning of unmanned aerial vehicles in a dynamic environment. In Infotech@ Aerospace 2011 (pp. 1654).

Levine, W. S. (2010). The Control Systems Handbook: Control System Advanced Methods. CRC press.

Loria, A., Panteley, E., Popovic, D., & Teel, A. R. (2002). An extension of Matrosov's theorem with application to stabilization of nonholonomic control systems. In Proceedings of the 41st IEEE Conference on Decision and Control, 2002., volume 2 (pp. 1528–1533 vol.2).

Luque-Vega, L., Castillo-Toledo, B., & Loukianov, A. G. (2012). Robust block second order sliding mode control for a quadrotor. Journal of the Franklin Institute, 349(2), 719–739.

Madani, T. & Benallegue, A. (2006). Backstepping control for a quadrotor helicopter. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3255–3260).: IEEE.

Maestre, J. M., Munoz De La Pena, D., & Camacho, E. F. (2011). Distributed model predictive control based on a cooperative game. Optimal Control Applications and Methods, 32(2), 153–176.

Maestre, J. M. & Negenborn, R. R. (2014). Distributed model predictive control made easy, volume 69. Springer.

Mian, A. A. & Daobo, W. (2008). Modeling and backstepping-based nonlinear control strategy for a 6DOF quadrotor helicopter. Chinese Journal of Aeronautics, 21(3), 261–268.

Michini, M., Rastgoftar, H., Hsieh, M. A., & Jayasuriya, S. (2014). Distributed formation control for collaborative tracking of manifolds in flows. In 2014 American Control Conference (pp. 3874–3880).: IEEE.

Mistler, V., Benallegue, A., & M'sirdi, N. (2001). Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In Proceedings 10th IEEE International Workshop on Robot and Human Interactive Communication. ROMAN 2001 (Cat. No. 01TH8591) (pp. 586–593).: IEEE.

Nascimento, I. B. P., Ferramosca, A., Pimenta, L. C. A., & Raffo, G. V. (2019). Nmpc strategy for a quadrotor uav in a 3d unknown environment. In 2019 19th International Conference on Advanced Robotics (ICAR) (pp. 179–184).

Negenborn, R. R. & Maestre, J. M. (2014). Distributed model predictive control: An overview and roadmap of future research opportunities. IEEE Control Systems Magazine, 34(4), 87–97.

Özbek, N. S., Önkol, M., & Efe, M. Ö. (2016). Feedback control strategies for quadrotor-type aerial robots: a survey. Transactions of the Institute of Measurement and Control, 38(5), 529–554.

Pacheco, G. V., Pimenta, L. C. A., & Raffo, G. V. (2019). Convergência e circulação de curvas variantes no tempo com um conjunto de quadrotores. In 14º Simpósio Brasileiro de Automação Inteligente Ouro Preto - MG.

Park, J., Kim, D., Yoon, Y., Kim, H., & Yi, K. (2009). Obstacle avoidance of autonomous vehicles based on model predictive control. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, 223(12), 1499–1516.

Parker, L. E. (1998). ALLIANCE: An architecture for fault tolerant multirobot cooperation. IEEE transactions on robotics and automation, 14(2), 220–240.

Patrinos, P. & Bemporad, A. (2013). An accelerated dual gradient-projection algorithm for embedded linear model predictive control. IEEE Transactions on Automatic Control, 59(1), 18–33.

Pereira, J. C., Leite, V. J. S., & Raffo, G. V. (2019). Nonlinear model predictive control on se(3) for quadrotor trajectory tracking and obstacle avoidance. In 2019 19th International Conference on Advanced Robotics (ICAR) (pp. 155–160).

Pimenta, L. C., Pereira, G. A., Gonçalves, M. M., Michael, N., Turpin, M., & Kumar, V. (2013). Decentralized controllers for perimeter surveillance with teams of aerial robots. Advanced Robotics, 27(9), 697–709.

Pimenta, L. C. A., Pereira, G. A. S., Michael, N., Mesquita, R. C., Bosque, M. M., Chaimowicz, L., & Kumar, V. (2013). Swarm coordination based on smoothed particle hydrodynamics technique. IEEE Transactions on Robotics, 29(2), 383–399.

Raffo, G. V. (2011). Robust control strategies for a QuadRotor helicopter: an underactuated mechanical system. PhD thesis, Universidad de Sevilla.

Raffo, G. V. & de Almeida, M. M. (2016). Nonlinear robust control of a quadrotor UAV for load transportation with swing improvement. In 2016 American Control Conference (ACC) (pp. 3156–3162).: IEEE.

Raffo, G. V., Ortega, M. G., & Rubio, F. R. (2008). Backstepping/nonlinear H∞ control for path tracking of a quadrotor unmanned aerial vehicle. In 2008 American Control Conference (pp. 3356–3361).: IEEE.

Raffo, G. V., Ortega, M. G., & Rubio, F. R. (2015). Robust nonlinear control for path tracking of a quad-rotor helicopter. Asian Journal of Control, 17(1), 142–156.

Rawlings, J. B. & Mayne, D. Q. (2009). Model predictive control: Theory and design. Nob Hill Pub.

Rey, F., Pan, Z., Hauswirth, A., & Lygeros, J. (2018). Fully decentralized ADMM for coordination and collision avoidance. In 2018 European Control Conference (ECC) (pp. 825–830).

Richards, A. & How, J. (2004). Decentralized model predictive control of cooperating UAVs. In 2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601), volume 4 (pp. 4286–4291).: IEEE.

Salierno, G. F. (2018). Whole-Body Backstepping Control of a Quadrotor UAV for Trajectory Tracking. Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte.

Salierno, G. F. & Raffo, G. V. (2017). Whole-body backstepping control with integral action of a quadrotor UAV. In XIII Simpósio Brasileiro de Automação Inteligente (pp. 2157 – 2164). Porto Alegre - RS.

Santos, M. A. (2018). Tube-based mpc with economical criteria for load transportation tasks using tilt-rotor uavs. Master's thesis, Universidade Federal de Minas Gerais.

Sathya, A., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., & Patrinos, P. (2018). Embedded nonlinear model predictive control for obstacle avoidance using PANOC. In 2018 European Control Conference (ECC) (pp. 1523–1528).: IEEE.

Skjetne, R. & Fossen, T. I. (2004). On integral control in backstepping: Analysis of different techniques. In American Control Conference, 2004. Proceedings of the 2004, volume 2 (pp. 1899–1904).: IEEE.

Slotine, J.-J. E., Li, W., et al. (1991). Applied nonlinear control, volume 199. Prentice hall Englewood Cliffs, NJ.

Small, E., Sopasakis, P., Fresk, E., Patrinos, P., & Nikolakopoulos, G. (2019). Aerial navigation in obstructed environments with embedded nonlinear model predictive control. In 2019 18th European Control Conference (ECC) (pp. 3556–3563).: IEEE.

Soorki, M. N., Talebi, H., & Nikravesh, S. (2011). Robust leader-following formation control of multiple mobile robots using Lyapunov redesign. In IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society (pp. 277–282).: IEEE.

Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). Robot modeling and control, volume 3. Wiley New York.

Stella, L., Themelis, A., Sopasakis, P., & Patrinos, P. (2017). A simple and efficient algorithm for nonlinear model predictive control. In 2017 IEEE 56th Annual Conference on Decision and Control (CDC) (pp. 1939–1944).: IEEE.

Sugihara, K. & Suzuki, I. (1996). Distributed algorithms for formation of geometric patterns with many mobile robots. Journal of robotic systems, 13(3), 127–139.

Tuci, E., Alkilabi, M. H., & Akanyeti, O. (2018). Cooperative object transport in multi-robot systems: A review of the state-of-the-art. Frontiers in Robotics and AI, 5, 59.

Valavanis, K. P. & Vachtsevanos, G. J. (2015). Handbook of unmanned aerial vehicles, volume 1. Springer.

Van Parys, R. & Pipeleers, G. (2017). Distributed MPC for multi-vehicle systems moving in formation. Robotics and Autonomous Systems, 97, 144–152.

Wächter, A. & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming, 106(1), 25–57.

Wang, Y., Yin, W., & Zeng, J. (2019). Global convergence of ADMM in nonconvex nonsmooth optimization. Journal of Scientific Computing, 78(1), 29–63.

Wu, C., Chen, J., Jeltsema, D., & Dai, C. (2018). Guidance vector field encoding based on contraction analysis. In 2018 European Control Conference (ECC) (pp. 282–287).: IEEE.

Yan, Z., Jouandeau, N., & Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems, 10(12), 399.

Yoon, Y., Shin, J., Kim, H. J., Park, Y., & Sastry, S. (2009). Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. Control Engineering Practice, 17(7), 741–750.

Zhang, R. & Kwok, J. (2014). Asynchronous distributed admm for consensus optimization. In International conference on machine learning (pp. 1701–1709).

Zhao, W. & Go, T. H. (2014). Quadcopter formation flight control combining MPC and robust feedback linearization. Journal of the Franklin Institute, 351(3), 1335–1355.

Zheng, H., Negenborn, R. R., & Lodewijks, G. (2016). Fast ADMM for distributed model predictive control of cooperative waterborne AGVs. IEEE Transactions on Control Systems Technology, 25(4), 1406–1413.