UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Adriano Martins da Costa Rezende

Vector field based guidance and control strategies for robot navigation

Belo Horizonte
2022

Adriano Martins da Costa Rezende

Vector field based guidance and control strategies for robot navigation

Final Version

Dissertation presented to the Graduate Program in Electrical Engineering of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

Advisor: Luciano Cunha de Araújo Pimenta.
Co-Advisor: Vinicius Mariano Gonçalves
Co-Advisor: Gustavo Medeiros Freitas

Belo Horizonte
2022

Rezende, Adriano Martins da Costa.
R467e Estratégias de guiagem e controle de navegação de robôs baseadas em campos vetoriais [recurso eletrônico] / Adriano Martins da Costa Rezende. - 2022.
1 recurso online (xxiii, 132 f. : il., color.) : pdf.

Orientador: Luciano Cunha de Araújo Pimenta.
Coorientadores: Vinicius Mariano Gonçalves, Gustavo Medeiros Freitas.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f.115-124.

Bibliografia: f.125-132.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Robôs - Sistemas de controle - Teses. 3. Robótica - Teses. I. Pimenta, Luciano Cunha de Araújo. II. Gonçalves, Vinícius Mariano. III. Freitas, Gustavo Medeiros. IV. Universidade Federal de Minas Gerais. Escola de Engenharia. V. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela Bibliotecária Roseli Alves de Oliveira CRB/6 2121
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
COLEGIADO DO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"Vector Field Based Guidance And Control Strategies For Robot Navigation"

ADRIANO MARTINS DA COSTA REZENDE

Tese de Doutorado defendida e aprovada, no dia 25 de março de 2022 , pela Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais constituída pelos seguintes professores:

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

Prof. Dr. Gustavo Medeiros Freitas - DEE (UFMG)

Prof. Dr. Vinícius Mariano Gonçalves - DEE (UFMG)

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

Prof. Dr. Leonardo Antônio Borges Tôrres - DELT (UFMG)

Prof. Dr. Guilherme Augusto Silva Pereira - Department of Mechanical and Aerospace Engineering (West Virginia University – WVU)

Prof. Dr. Marco Henrique Terra - Departamento de Engenharia Elétrica e de Computação (EESC/USP)

Belo Horizonte, 25 de março de 2022.

Documento assinado eletronicamente por **Luciano Cunha de Araujo Pimenta**, **Professor do Magistério Superior**, em 27/03/2022, às 14:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1327767** e o código CRC **D3BD3327**.

Referência: Processo nº 23072.215676/2022-71 SEI nº 1327767

Acknowledgments

First, I should thank my entire family for all the support during the years of this Ph.D. course and for the incentives to the studies since I was a child.

I would like to acknowledge professors Luciano Pimenta and Vinicius Gonçalves for the orientation during my academic life. I also acknowledge professor Gustavo Freitas for the co-orientation in the last years and for the opportunity to get involved with projects from the ITV. I also acknowledge the other professors of the program for all the knowledge they have transmitted to me.

I should also say thanks to several undergrad, master and Ph.D. students: those from the Coro Laboratory for the company and friendship; those involved with the EspeleoRobô project, we did a good work together; and those from Verlab, who shared a lot of knowledge during the drone racing competition.

Finally, I am thankful for the financial support provided by the ITV through the CNPq, and for the availability of the EspeleoRobô, a robotic platform used for many experiments. Also, I am grateful for the support of the InSAC, which financed some expensive robots and equipment used in the experiments presented in this thesis, specially, the DJI drone and the OptiTrack motion capture system. During the year 2019 I had the opportunity to travel abroad several times due to the drone racing competition, thanks to all supporters of this journey: DTI Digital, Vale, Programa de Pós Graduação em Engenharia Elétrica, InSac, Departamento de Engenharia Elétrica, Escola de Engenharia, Departamento de Ciência da computação and Programa de Pos Graduação em Ciência da Computação.

Resumo

Neste trabalho, dois problemas na área de navegação de robôs são abordados: guiagem e controle. Para resolver o problema de guiagem, propõe-se um campo vetorial artificial, que pode ser aplicado em várias plataformas robóticas. O problema de controle é abordado no contexto de um modelo de quadrirrotor, que precisa seguir o campo vetorial proposto.

Após introdução e revisão da literatura, apresenta-se a metodologia para calcular um campo vetorial artificial variante no tempo em n dimensões, que admite uma determinada curva como ciclo limite estável. A função de distância Euclidiana constrói o campo, facilmente calculado a partir de uma representação paramétrica da curva. A norma da componente de *feedforward* que compensa a dependência temporal é limitada pela velocidade máxima da curva, permitindo uma normalização do campo vetorial variante no tempo sem afetar a convergência. Provas de estabilidade do campo normalizado variante no tempo são apresentadas e a existência de *ultimate bounds* é demonstrada para o caso de distúrbios limitados. Duas modificações no campo vetorial são apresentadas: a primeira para curvas com auto interseção e a segunda para desvio de obstáculos.

Com o campo vetorial definido, apresenta-se uma estratégia de controle para um quadrirrotor seguir uma curva variante no tempo usando o campo vetorial proposto. Primeiramente, leis de controle são desenvolvidas para impor o comportamento das curvas integrais do campo vetorial ao modelo do integrador de segunda ordem. Depois, leis de controle impõem a dinâmica do integrador de segunda ordem controlado a um modelo de quadrirrotor, que assume o impulso total e as velocidades angulares como comandos de entrada. A convergência assintótica do sistema completo é provada ao mostrar que as camadas individuais são estáveis no sentido entrada-estado. Uma análise da influência de distúrbios limitados nas entradas de controle avalia a robustez do controlador, mostrando que distúrbios limitados causam desvios limitados da curva alvo. Várias simulações ilustram as metodologias propostas para guiagem e controle. Experimentos com diferentes quadrirrotores e um robô terrestre em ambientes externos não estruturados mostram que o arcabouço proposto é adequado para aplicações no mundo real. Uma simulação com um manipulador de 6 graus de liberdade também exemplifica o uso do campo vetorial em dimensões maiores que 3.

Palavras-chave: Campo Vetorial Artificial; Controle de Quadrirrotor; Guiagem e Navegação; Estabilidade e Robustez.

Abstract

In this work we consider two problems in robot navigation: guidance and control. To solve the guidance problem we propose an artificial vector field, applicable to various robot platforms. The control problem is addressed by using a quadcopter model that follows the proposed vector field.

After a brief introduction and literature review, we present a methodology to compute an artificial time-varying vector field in n dimensions with a given curve as its stable limit cycle. The Euclidean distance function constructs the field, easily computed from a parametric curve representation. The time feedforward term's norm is limited by the curve's maximum velocity, allowing normalization of the vector field without negatively affecting convergence. We provide stability proofs for the normalized time-varying vector field and demonstrate ultimate bounds with bounded disturbances. We also present two modifications: one for self-intersecting curves and another for obstacle avoidance.

Given the vector field, we propose a control strategy for a quadcopter to follow a time-varying curve. First, control laws are developed to impose the behavior induced by the integral curves of the vector field to a second order integrator model. After that, control laws are developed to impose the dynamics of the controlled second order integrator to a quadcopter model, which assumes the total thrust and angular rates as input commands. Asymptotic convergence of the system is proved by showing input-to-state stability of individual layers. We analyze the influence of norm-bounded disturbances on control inputs to evaluate robustness, showing limited deviations from the target curve. Simulations illustrate the guidance and control methodologies, while experiments with different quadcopters and a ground robot in outdoor environments demonstrate the frameworks' real-world applicability. A simulation with a six-degree-of-freedom manipulator exemplifies the vector field's use in higher dimensions.

Keywords: Artificial Vector Field; Quadcopter Control; Guidance and Navigation; Stability and Robustness.

List of Figures

List of abbreviations and acronyms

UFMG	Universidade Federal de Minas Gerais
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
ITV	Instituto Tecnológico Vale
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
InSac	Instituto Nacional de Ciência e Tecnologia para Sistemas Autônomos Cooperativos
ISS	Input-to-State Stability
SLAM	Simultaneous Localization and Mapping
UAV	Unmanned Aerial Vehicle
AIRR	Artificial Intelligence Robotic Racing
ROS	Robot Operating System
IMU	Inertial Measurement Unit
PWM	Pulse-Width Modulation
LiDAR	Light Detection And Ranging
LeGO-LOAN	Lightweight and Ground-Optimized Lidar Odometry and Mapping
EKF	Extended Kalman Filter
GPS	Global Positioning System
SGC	Separated Guidance and Control
IGC	Integrated Guidance and Control
LOS	Line-of-sight
VTP	Virtual Target Point
NLGL	Non-Linear Guidance Law
RRT	Rapidly-exploring Random Tree

RRT*	Optimum Rapidly-exploring Random Tree
VFRRT	Vector Field Rapidly-exploring Random Tree
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
DOF	Degrees Of Freedom
PD	Proportional Derivative
PID	Proportional Integral Derivative

List of Symbols

$\mathbb{R}$	Set of the real numbers
$\mathbb{N}$	Set of the natural numbers
$\mathbb{I}$	Set of the integer numbers
n	Element of $\mathbb{N}$ representing the dimension of the Euclidean space
t	Time
t_0	Initial time of a system's evolution
$\mathcal{C}(\mathrm{t})$	Time varying curve to be followed
$\mathbf{x}$	Position vector
$\mathbf{r}(\mathrm{s},\mathrm{t})$	Parametrization of the curve $\mathcal{C}(\mathrm{t})$
s	Parameter of the curve used in $\mathbf{r}(\mathrm{s},\mathrm{t})$
$\mathrm{s}^*(\mathbf{x},\mathrm{t})$	Parameter associated to the point on $\mathbf{r}(\mathrm{s},\mathrm{t})$ that is the closest to $\mathbf{x}$ at instant t
$\mathbf{x}^*(\mathbf{x},\mathrm{t})$	Point on $\mathcal{C}(\mathrm{t})$ that is the closest to $\mathbf{x}$ at instant t. May not be uniquely defined
$\mathbf{\Phi}(\mathbf{x},\mathrm{t})$	Euclidean distance time varying vector field
$\mathbf{D}(\mathbf{x},\mathrm{t})$	Distance vector that goes from $\mathbf{x}^*(\mathbf{x},\mathrm{t})$ to $\mathbf{x}$
$\mathrm{D}(\mathbf{x},\mathrm{t})$	Euclidean norm of $\mathbf{D}(\mathbf{x},\mathrm{t})$
$\mathcal{U}$	Subset of $\mathbb{R}^{\mathrm{n}} \times \mathbb{R}^{+}$ of the points $(\mathbf{x},\mathrm{t})$ such that $\mathbf{x}^*(\mathbf{x},\mathrm{t})$ is unique
∂	Symbol for partial derivative
$\mathbf{T}(\mathrm{s},\mathrm{t})$	Unity vector tangent to the curve at point $\mathbf{r}(\mathrm{s},\mathrm{t})$
$\mathbf{T}^*(\mathbf{x},\mathrm{t})$	Unity vector tangent to the curve at point $\mathbf{x}^*(\mathbf{x},\mathrm{t})$
$\mathbf{N}(\mathrm{s},\mathrm{t})$	Unity vector normal to the curve at point $\mathbf{r}(\mathrm{s},\mathrm{t})$
$\kappa(\mathrm{s},\mathrm{t})$	Curvature of the curve at point $\mathbf{r}(\mathrm{s},\mathrm{t})$
v_{r}	Norm of the vector field

$\mathrm{I}_{\mathrm{n}\times\mathrm{n}}$	Identity matrix of order n
$\Pi_{\mathbf{T}}(\mathbf{x},\mathrm{s})$	Projector on the null space of $\mathbf{T}^*(\mathbf{x},\mathrm{s})$
$\mathrm{v_m}$	Maximum "local speed" of the curve $\mathcal{C}(\mathrm{t})$
$\Delta\mathrm{t}$	Small amount of time
$\mathrm{G(D)}$	Function that defines the weight of the convergent component
$\mathrm{H(D)}$	Function that defines the weight of the traversal component
$\mathrm{k_f}$	Gain of the vector field controller
$\hat{\mathrm{G}}(\mathbf{x},\mathrm{t})$	Function G(D) mapped to the domain of $(\mathbf{x},\mathrm{t})$
$\hat{\mathrm{H}}(\mathbf{x},\mathrm{t})$	Function H(D) mapped to the domain of $(\mathbf{x},\mathrm{t})$
$\eta(\mathbf{x},\mathrm{t})$	normalization factor of the vector field $\mathbf{\Phi}$
$\Omega(\mathbf{x},\mathrm{t})$	Conditioning factor associated with the curvature of $\mathcal{C}(\mathrm{t})$ at point $\mathbf{r}(\mathbf{x},\mathrm{t})$
∇	Gradient operator with respect to $\mathbf{x}$
P	Lyapunov function for the vector field
$\mathbf{\Phi}_{\mathrm{G}}(\mathbf{x},\mathrm{t})$	Convergent component of the vector field
$\mathbf{\Phi}_{\mathrm{H}}(\mathbf{x},\mathrm{t})$	Tangential component of the vector field
$\mathbf{\Phi}_{\mathrm{T}}(\mathbf{x},\mathrm{t})$	Time feedforward component of the vector field
$\mathbf{\Phi}_{\mathrm{S}}$	Scalable component of the vector field
$\mathrm{E}(\mathbf{x})$	Distance from $\mathbf{x}$ to the complementary set of $\mathcal{U}$
D_0	Distance between $\mathbf{x}$ and $\mathcal{C}(\mathrm{t})$ at instant t_0
$\mathbf{u}$	Control signal for the simple integrator model
δ_{v}	Uncertainty vector in the input control $\mathbf{u}$
Δ_{v}	Maximum norm of δ_{v}
$\mathcal{I}(\mathrm{t})$	Positive invariant set associated to the distance from $\mathbf{x}$ to the curve $\mathcal{C}(\mathrm{t})$
δ_0	Smallest distance $\mathrm{D}(\mathbf{x}_{\mathrm{P}},\mathrm{t})$ for any $(\mathbf{x}_{\mathrm{P}},\mathrm{t}) \notin \mathcal{U}$
$\mathcal{X}_0$	Subset of $\mathbb{R}^{\mathrm{n}}$ in which any trajectory of the field $\mathbf{\Phi}$ is singularity free
$\mathrm{s_v}$	Virtual variable representing a parameter of the curve

$\mathbf{x}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}})$	Point in the curve associated to the parameter s_{v}
$\mathbf{D}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}}, \mathbf{x})$	Distance vector that goes from $\mathbf{x}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}})$ to $\mathbf{x}$
$\mathrm{D}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}}, \mathbf{x})$	Euclidean norm of $\mathbf{D}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}}, \mathbf{x})$
$\boldsymbol{\Phi}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}}, \mathbf{x})$	Euclidean distance vector field in the augmented space
$\Omega_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}}, \mathbf{x})$	Conditioning factor associated to the curvature of $\mathcal{C}$ at point $\mathbf{x}_{\mathrm{v}}(\mathrm{s}_{\mathrm{v}})$
k_{s}	Gain of the controller for s_{v}
$\kappa_{\max}$	Maximum curvature of the curve $\mathcal{C}$
$\mathcal{X}_{\mathrm{v}}$	Subset of $\mathbb{R}^{\mathrm{n}}$ in which any trajectory of the field $\boldsymbol{\Phi}_{\mathrm{v}}$ is singularity free
$\mathcal{O}(\mathrm{t})$	Set of points that belong to an obstacle
$\mathbf{x}_{\mathrm{o}}$	A point that belongs to the set $\mathcal{O}(\mathrm{t})$
$\mathbf{x}_{\mathrm{o}}^{*}(\mathbf{x}, \mathrm{t})$	Point $\mathbf{x}_{\mathrm{o}} \in \mathcal{O}(\mathrm{t})$ that is the closest to $\mathbf{x}$
$\mathbf{D}_{\mathrm{o}}(\mathbf{x}, \mathrm{t})$	Distance vector that goes from $\mathbf{x}_{\mathrm{o}}^{*}(\mathbf{x}, \mathrm{t})$ to $\mathbf{x}$
$\mathrm{D}_{\mathrm{o}}(\mathbf{x}, \mathrm{t})$	Euclidean norm of $\mathbf{D}_{\mathrm{o}}(\mathbf{x}, \mathrm{t})$
λ	Distance that an obstacle is contoured
$\mathcal{S}_{\mathrm{o}}(\mathrm{t})$	Set of points that dist λ from the obstacle set $\mathcal{O}(\mathrm{t})$
$\mathbf{D}_{\lambda}(\mathbf{x}, \mathrm{t})$	Shortest distance vector that goes from $\mathcal{S}_{\mathrm{o}}$ to $\mathbf{x}$
$\mathrm{D}_{\lambda}(\mathbf{x}, \mathrm{t})$	Euclidean norm of $\mathbf{D}_{\lambda}(\mathbf{x}, \mathrm{t})$
$\mathcal{A}(\mathbf{x}, \mathrm{t})$	Auxiliary vector field to define the tangent component of the contouring field
$\mathbf{T}_{\lambda}(\mathbf{x}, \mathrm{t})$	Unity vector tangent to the set $\mathcal{S}_{\mathrm{o}}(\mathrm{t})$
$\boldsymbol{\Psi}_{\mathrm{G}}(\mathbf{x}, \mathrm{t})$	Convergent component of the contouring vector field
$\boldsymbol{\Psi}_{\mathrm{H}}(\mathbf{x}, \mathrm{t})$	Traversal component of the contouring vector field
$\boldsymbol{\Psi}_{\mathrm{T}}(\mathbf{x}, \mathrm{t})$	Time feedforward component of the contouring vector field
$\boldsymbol{\Psi}(\mathbf{x}, \mathrm{t})$	Contouring vector field
$\Pi_{\nabla \mathrm{D}_{\lambda}}$	Projector on the null space of $\nabla \mathrm{D}_{\lambda}(\mathbf{x}, \mathrm{t})$
$\Pi_{\mathbf{T}_{\lambda}}$	Projector on the null space of $\mathbf{T}_{\lambda}(\mathbf{x}, \mathrm{t})$
$\mathbf{F}$	Composite vector field that incorporates $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$

$\mathrm{D_{in}}$	Distance from which the vector field $\mathbf{\Psi}$ is followed completely
$\mathrm{D_{in}^0}$	Distance in which the obstacle starts to influence the vector field
$\theta(\mathbf{x}, \mathrm{t})$	Parameter for the convex combination of $\mathbf{\Phi}$ and $\mathbf{\Psi}$
$\mathbf{p}$	Position of the quadcopter
$\mathbf{v}$	Velocity of the quadcopter in the inertial frame
$\mathrm{R_b^w}$	Matrix representing the quadcopter's orientation with respect to the inertial frame
τ	Total thrust force, input of the quadcopter model
ω	Angular velocity vector in the body frame, input of the quadcopter model
$\mathrm{S}(\cdot)$	Skey-symetric matrix associated to a given vector
$\omega_\mathrm{x},\ \omega_\mathrm{y},\ \omega_\mathrm{z}$	Componnents of ω
δ_τ	Uncertainty in the control input τ
δ_ω	Uncertainty vector in the control input ω
Δ_τ	Maximum norm of the uncertainty δ_τ
Δ_ω	Maximum norm of the uncertainty δ_ω
$\hat{\mathbf{z}}$	Canonical vector in the z direction of the inertial frame
m	Mass of the quadcopter
g	Acceleration of gravity
$\mathrm{f_d}(\mathbf{v})$	Drag force represented in the world frame
$\mathbf{a}_\mathrm{d}$	Desired acceleration to be imposed to the quadcopter
$\mathrm{J}_{\mathbf{\Phi}}$	Jacobian matrix of $\mathbf{\Phi}(\mathbf{p}, \mathrm{t})$ with respect to $\mathbf{p}$
$\mathrm{k_v}$	Gain of the velocity controller
$\mathrm{V_v}$	Lyapunov function for the velocity error
δ_a	Disturbance of acceleration
Δ_a	Maximum norm of δ_a
$\mathcal{I}_\mathrm{v}$	Positive invariant set associated to the velocity error
$\hat{\mathbf{z}}_\mathrm{b}$	Canonical vector in the z direction of the body (quadcopter) frame

$\mathbf{a}_\mathrm{r}$ Acceleration to be achieved by the quadcopter's thrust force

$\mathrm{R}_\mathrm{r}^\mathrm{w}$ Reference orientation for the quadcopter

ψ_r Heading reference for the quadcopter

w_ψ Vector representing the orientation ψ_r

$\hat{\mathbf{x}}_\mathrm{r},\ \hat{\mathbf{y}}_\mathrm{r},\ \hat{\mathbf{z}}_\mathrm{r}$ Vectors that compose the matrix $\mathrm{R}_\mathrm{r}^\mathrm{w}$

R_e Matrix representing the orientation error of the quadcopter

ω_r Angular velocity of the reference orientation given by $\mathrm{R}_\mathrm{r}^\mathrm{w}$, written in the body frame

ω_e Rotation velocity of the matrix R_e, written in the body frame

$\hat{\mathbf{n}}$ Unity vector representing the axis of the axis-angle representation of the matrix R_e

β Angle of the axis-angle representation of the matrix R_e

k_β Gain of the orientation controller

V_β Lyapunov function for the orientation error

$\mathcal{I}_\beta$ Positive invariant set associated to the orientation error

Δ_β Maximum orientation error, ultimate bound

τ_max Maximum value of the thrust τ

$\mathcal{I}_\mathrm{D}$ Positive invariant set associated to the distance of the drone to the curve

$\gamma_1(\Delta_\mathrm{v})$ Function that returns the ultimate bound for the distance D

$\gamma_2(\Delta_\mathrm{a})$ Function that returns the ultimate bound for the velocity error δ_v

$\gamma_3(\Delta_\tau, \Delta_\omega)$ Function that returns the ultimate bound for the acceleration error δ_a

$\mathbf{f}(\mathbf{x})$ Generic normalized vector field

$\mathbf{d}$ Unit vector indicating a direction

ϵ Infinitesimal positive number

$\mathbf{x}_\mathrm{P}$ Point $\mathbf{x}$ not in $\mathcal{U}$

$\mathbf{m}$ One of the multiple choices of $\mathbf{x}^*$ when it is not unique

$\boldsymbol{\Phi}(\mathbf{x}_\mathrm{P};\ \mathbf{m}))$	Vector field $\boldsymbol{\Phi}(\mathbf{x}_\mathrm{P})$ computed by choosing $\mathbf{m}$ as the closest point on the curve
$\mu(\mathbf{x}_\mathrm{P};\mathbf{m})$	Closest point on the curve if the $\mathbf{x}_\mathrm{P}$ is reached by following the field $\boldsymbol{\Phi}(\mathbf{x}_\mathrm{P};\ \mathbf{m}))$
$\mathcal{R}_\mathbf{f}(\mathbf{x})$	Set of all values the vector field $\mathbf{f}(\mathbf{x})$ can have when we approach $\mathbf{x}$ from all possible directions
$\Gamma(\mathbf{x}_\mathrm{P};\ \mathbf{y})$	Auxiliary function to analyse the approachability of points $\mathbf{x}_\mathrm{P}$
$\mathrm{K}(\mathbf{x}_\mathrm{P})$	Maximum value of the function $\Gamma(\mathbf{x}_\mathrm{P};\ \mathbf{y})$
$\mathbb{S}$	Space of the circle
s_end	Ending parametrization of a curve in the Torus space
k_i	Integers used to replicate the curve $\mathcal{C}$
T_i	"Period" of the i-th component of a curve in $\mathbb{S}^\mathrm{n}$
$\mathcal{C}_\mathrm{rep}$	Curve in $\mathbb{S}^\mathrm{n}$ replicated in the Euclidean space
$\mathrm{D}_\mathbb{S}$	Distance to a replicated curve in a given dimension
$\mathrm{k}^*_\mathbb{S}$	Index of the closest replicated curve

Contents

Chapter 1

Introduction

1.1 Robot navigation

The robot navigation problem is a complex and interdisciplinary challenge in the area of robotics. Solutions to this problem aim to make an autonomous robot move safely in a given environment. In general, this complex task can be broken into several different sub-problems, for instance: perception, localization, mapping, planning, guidance, and control. Perception consists of the extraction of exteroceptive data of the environment[1]. Localization involves the use of this information and additional proprioceptive sensor data to estimate the robot states. Mapping focuses on the obtainment of a representation of the environment. Planning consists of the definition of a target place, path or trajectory the robot should pursue. The guidance stage computes a reference velocity, or a motion direction, to make the robot achieve its planned objective. Finally, in the control stage, signals for the robot's actuators are computed so that it moves according to the guiding reference.

Many times, some strategies tackle more than one sub-problem without making an exact distinction between the parts. For instance, it is common to observe perception together with localization, planning with guidance, and localization with mapping, the so-called Simultaneous Localization and Mapping (SLAM). The guidance and control problems are also addressed together in many works. Some authors do not distinguish these tasks, a common approach in trajectory tracking, for example. In fact, one may consider that guidance and control are simply different layers of the same task. From this point of view, the guidance is a higher-level controller. For instance, if we consider the simple integrator model for a robot, the guidance is equivalent to control. The distinction between these sub-problems is usually made according to the utility it has in the description of the adopted approach.

In this work, we address two of the mentioned sub-problems in robot navigation: guidance and control. The guidance solution can be used in the navigation of seveval

[1] http://www.cs.cmu.edu/~rasc/Download/AMRobots4.pdf

robotic platforms. The control solution is focused on quadcopter aerial vehicles and is integrated with the proposed guidance strategy. Our main contribution is a novel vector field based guidance strategy to allow path following. In addition, for the specific case of quadcopters, we also propose a nonlinear controller to follow the guidance vector field. The proposed nonlinear controller, designed for a quadcopter robot, controls the vehicle to follow the guidance vector field. Here we will adopt the term *guidance* for the task of defining a reference velocity to the robot. It can be seen as a high-level control. On the other hand, the term *control* will be mostly used for the computation of control signals that, given a specific robot architecture, make the vehicle follow the guidance velocity.

1.1.1 Guidance vector field

The guidance problem in this work is solved with the use of artificial vector fields. Let $\mathbf{x} \in \mathbb{R}^n$ be the robot state vector and $\mathbf{\Phi}(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^n$ be a vector field that returns a velocity reference for the robot. The field is designed so that it leads the robot to converge to and follow a predefined path, possibly time-varying. In the case of a static closed path, and a robot with single integrator dynamics, we have that $\mathcal{C}$ corresponds to the limit cycle of the dynamical system $\dot{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x})$.

Strategies based on artificial vector fields have been widely used to guide the high-level control of several types of robots. For instance, one of the most frequent uses of such a strategy is associated with the guidance of fixed-wing UAVs (Unmanned Aerial Vehicles) [56, 64, 86]. Such approaches have already been discussed in scientific books, such as [7]. Vector fields are also used to guide quadcopters, [87, 63, 66]. Other types of robots that can be controlled with vector fields include wheeled robots [34, 40], AUVs (Autonomous Underwater Vehicles) [1], manipulators [46], and even humanoid-like robots [29, 6, 47].

In [27], a strategy to generate time-varying vector fields for converging to and circulating a curve in n dimensions is proposed. There, the curve to which the integral lines of the vector field converge is defined by the intersection of n–1 zero level sets given by $\alpha_i(\mathbf{x}, t) = 0,\ i = 1, ..., n-1$, in which $\alpha_i : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}$. This approach has the drawback of requiring the design of these scalar functions α_i that implicitly define the target curve. For simple curves, such as circles and ellipses, it is easy to obtain these functions, however, for general curves it is not the case. Methodologies for generating these functions were presented, but the obtained α_i functions are prone to induce spurious undesirable stable equilibrium points in the vector field, a problem that is often a source of criticism when [27] is referred. Now, we propose a simpler and improved methodology to solve the same

problem. Thus, our contribution may be considered as an improvement of [27].

Chapter 3 of this work is dedicated to the description of this simpler methodology that can be easily applied given a parametric representation of the curve, instead of using a representation with an intersection of zero level surfaces. The main feature of this field is that it is computed by using the Euclidean distance between the robot and the curve as an error measurement function. The use of a parametric representation of the curve and the consideration of the distance function make our method simple to implement and allow for easy geometric interpretation of our results. Another achieved property is that the proposed field does not have spurious stable equilibrium points, which is a problem that often appears in other vector field methodologies. Nonetheless, as we will discuss in Chapter 3, there might be singular points in the field. We also present a novel normalization strategy for time-varying fields, allowing for fields with a fixed norm, a useful feature for practical applications. Figure 1.1 shows an example of the vector field generated by the methodology described in this work.

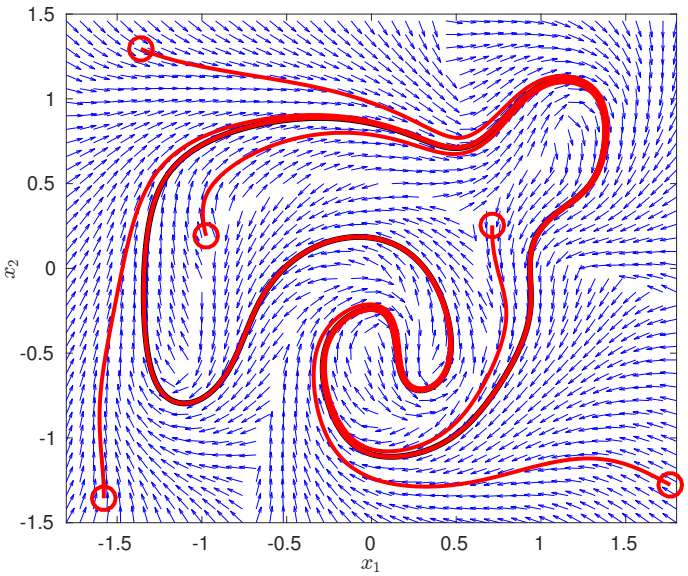

Figure 1.1: Paths generated by following the vector field described in this work for a given target curve. The vector field, also shown, is devoid of any stable equilibrium point.

Two extensions of the proposed vector field are also presented to deal with some further issues. The first is an adaptation that tackles the singularities of the original field. The second one incorporates the ability to deviate from obstacles by contouring them. Rigorous mathematical proofs for these extended vector field methodologies are left for a future research.

1.1.2 Quadcopter control

In the last years, the presence of robots in everyday life grew quickly. Among several types of robots, quadcopters are receiving special attention due to their peculiar capabilities. They have a high maneuverability and are used in indoor and outdoor environments. Several researchers address the use of these flying robots for mapping [30], surveillance [79, 42], inspections [21, 5], package delivery [62, 80], among other tasks. Such applications motivate the development of the control strategy that will be presented in Chapter 4 of this thesis.

The control problem in this work aims to make a quadcopter follow the vector field designed in Chapter 3. We assume a robot model that has states representing position, linear velocities, and orientation. Given the vector field $\mathbf{\Phi}(\mathbf{x}, \mathrm{t})$, the controller will compute signals for total thrust force and angular velocities such that the quadcopter follows the velocity reference given by a vector field and then converge to a given path. The developments in this thesis were motivated by the participation of our research group in the Autonomous Drone Racing (Alpha Pilot[2]) [66]. In the race, the vehicle needs to flight through a sequence of gates. The XQuad team[3], constituted by members of our research group, was ranked 8th among 400 teams worldwide in the qualifying stage, which classified us to the Artificial Intelligence Robotic Racing (AIRR[4]). The theory proposed in this work is the basis for the controllers used in the competition.

In [64], the authors develop a controller to make a fixed-wing UAV follow a curve by using an artificial vector field. The considered airplane model has a second-order position dynamics. The approach considers the derivatives of the vector field in order to control the UAV and ensure asymptotic convergence. The approach considered in this work takes the same guidelines of [64], thus it can be considered as an extension of that work for a different type of robot, the quadcopter.

1.2 Robotic platforms

Besides the computational simulations provided to exemplify the methodologies presented in Chapters 3 and 4, which include a quadcopter and a 6 DOF (degrees of freedom) manipulator, real robot experiments were also performed. It is important to

[2] https://www.herox.com/alphapilot/teams
[3] https://xquadufmg.com
[4] https://thedroneracingleague.com/airr

emphasize that the vector field methodology presented in Chapter 3 can be applied to a variety of robotic platforms. The controller developed in Chapter 4 is, in fact, destined for quadcopters. In order to exemplify the use of such methodologies and evaluate their behavior in real scenarios, three robots were used for physical experiments. In the following, we describe the aerial and ground vehicles used for validation.

1.2.1 DJI M100

DJI Matrice 100, or just M100, is a commercial drone sold by DJI[5] and was used to perform the outdoor quadcopter experiments presented in this work. Besides being equipped with an IMU (Inertial Measurement Unit), the robot also counts with a GPS (Global Positioning System) for localization, which allows for outdoor experiments. We used the localization packages provided by DJI available online[6], which provides reliable information of the drone's position, orientation, and velocity. The robot is equipped with an NVIDIA® Jetson Nano computer running Ubuntu 18.04 operational system. Figure 1.2 shows a picture of the platform.

Figure 1.2: Picture of the DJI M100 drone used in the experiments.

[5] https://www.dji.com
[6] https://github.com/dji-sdk/Onboard-SDK-ROS

1.2.2 ESPcopter

The ESPcopter is a lighweigth, low-cost, commercial drone[7] and was used to perform the indoor quadcopter experiments presented in this work.

The robot is equipped with an 9-axis IMU MPU9250, with gyro, accelerometer and magnetometer. The propellers are driven by four DC motors controlled via PWM (Pulse-Width Modulation) signals. The vehicle counts with a 32-bit microcontroller ESP8266 running at 160MHz. Figure 1.3 shows a picture of the used platform. During the experiments, the quadcopter pose was captured with an OptiTrack[8] motion capture system. It counts with 8 Prime 41 cameras placed in a space of 36 m^2, providing measurements at 180Hz with a precision of less than 1 mm.

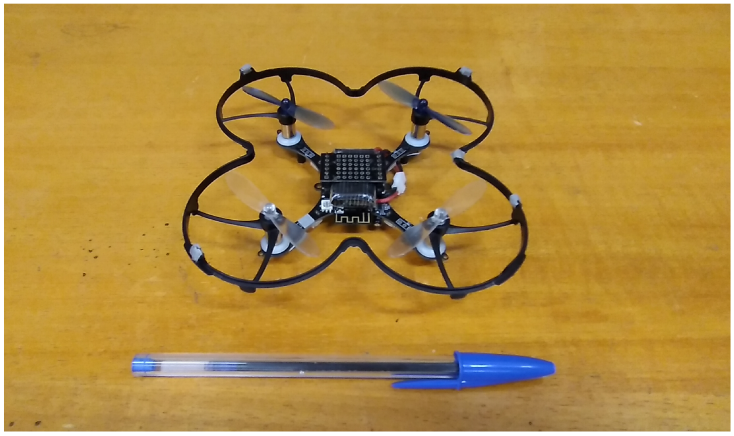

Figure 1.3: Picture of the ESPcopter drone used in the experiments.

1.2.3 EspeleoRobô

The EspeleoRobô is a robotic platform Developed by the Instituto Tecnológico Vale (ITV). It has six independent motors mounted on the skid-steering configuration. Different wheel types can be coupled to the motor shafts. The configuration used in the experiments used only 4 wheels. The robot is also equipped with an Intel NUC computer running Ubuntu 16.04 and ROS (Robot Operating System) Kinetic. Among the sensors of the platform, there are cameras, an IMU, and a LiDAR Ouster OS1-16, which has a vertical resolution with 16 set of beams. The LiDAR Ouster OS1 can also be replaced,

[7] https://espcopter.com/
[8] https://optitrack.com/

in a second configuration, by a Hokuyo UTM-30LX-EW, which has a single set of beans. A better description of the robot and the Ouster based localization system can be found in [65]. Figure 1.4 shows on the left the components of the platform and on the right a picture of the robot used in the experiments.

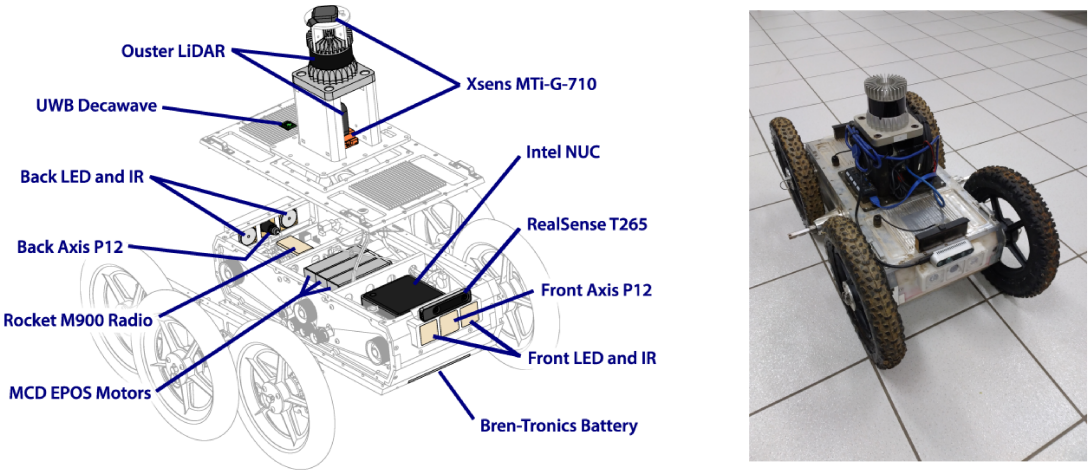

Figure 1.4: Equipment of the EspeleoRobô (left) and picture of the robot (right).

In the experiments with the EspeleoRobô, three localization methods were considered. The first considers the Ouster LiDAR together with the LeGO-LOAM[9] algorithm, [74], to provide the pose of the robot. This pose is used in an Extended Kalman Filter (EKF) that incorporates the information of the IMU and wheel odometry. The second considers the Hokuyo sensor with the gmapping[10] algorithm to provide the pose of the robot in the plane. These two localization algorithms were executed on the embedded computer. The third method considers the OptiTrack mentioned in Section 1.2.2.

The controller algorithms also run in the EspeleoRobô's embedded computer.

1.3 Contributions

Both the guidance and control approaches presented in this work have some peculiarities and advantages with respect to related works.

[9]LeGO-LOAM is a lightweight and ground optimized lidar odometry and mapping algorithm. It takes in a point cloud and an optional IMU data as inputs. It outputs 6D pose estimation in real-time. It is available online at https://github.com/RobustFieldAutonomyLab/LeGO-LOAM

[10]OpenSlam's Gmapping is a laser-based SLAM algorithm, provided as the ROS package gmapping. It creates a 2D occupancy grid map from laser and pose data collected by a mobile robot. The planar pose is provided as an output. The code is available online at https://github.com/ros-perception/slam_gmapping.

The first property of the vector field proposed in Chapter 3 is that it can be easily computed from the parametric representation of the curve. This contrasts with previous methods that require an implicit representation with intersection of zero-level sets [27]. These implicit functions may be hard to find analytically. Some numerical methods are available, however, despite being valid only locally, they may induce undesirable equilibrium points. Having a field defined from a parametric representation allows for an easy consideration of a wider variety of curve shapes. In addition, the strategy can be easily applied if the curve is represented by a sequence of points.

The function that measures the position error of the robot is the Euclidean distance to the curve. This feature gives the vector field a homogeneous convergence behavior, meaning that aggressiveness depends only on the distance to the curve, not on the direction nor on the part of the curve the robot is. When general analytic functions are used, such as in [27], the convergence properties will be tied to the behavior of these functions. For instance, the field may be more aggressive in some parts of the workspace while less aggressive in others. In practice, a field with a homogeneous convergence behavior makes it easier to perform the calibration of the controller gains.

The proposed vector field methodology is also applicable for curves embedded in Euclidean spaces of any finite dimension and possibly time-varying. This contrasts with other methodologies that are usually limited to $\mathbb{R}^2$ and $\mathbb{R}^3$. The method proposed in [27] also considers n dimensions and time dependence. However, as the number of dimensions grows, the harder is to obtain the functions that implicitly define the curve. In Section 5.3.3, a simulation with a 6 DOF manipulator exemplifies the use of the proposed vector field in higher dimensions.

Another important achievement of the theory that we present here is that the designed field has a constant norm even for time-varying cases. Thus, the robot can converge to and traverse the curve with a constant speed, which is a useful characteristic in practical applications. In the literature, it is common to normalize the field in order to obtain a constant norm over the whole workspace. However, when the curve is time-varying and the field has an associated time feedforward component, the standard normalization strategy can not be directly applied, since it will jeopardize the convergence. We propose a novel normalization technique that enables the obtainment of a constant norm while keeping asymptotic convergence. This new strategy can be applied to the Euclidean distance vector field since the norm of the feedforward component is limited by the maximum speed of the moving curve, which is not the case for other methodologies in the literature [27].

An attribute of the guidance vector field is that it is classified as a controller for paths, not for trajectories. To follow a path means that, given a specific time, there is no specific point in the curve the robot needs to be. A better discussion on this topic is made in Chapter 2.

We also show how the Euclidean distance field we propose can be easily applied to make a robot contour an obstacle. The only additional requirement is the robot's ability to detect the closest point on the obstacle. A LiDAR sensor, for instance, can be used for this task. We then propose a vector field that switches between the behavior of following a desired curve and contouring obstacles. With that, we incorporate a reactive feature to the navigation methodology that allows it to deviate from obstacles by contouring them.

Experiments with the DJI drone, the ESPcopter, and the EspeleoRobô, and simulations with a robotic manipulator, were performed to illustrate the proposed methodology. The features described here, which are formally described over the text, are corroborated by the results observed in practice.

The quadcopter control designed in Chapter 4 has also some positive characteristics. Vector fields have already been used to control quadcopters. The most direct use consists of the consideration of lower-level controllers that make the vehicle respond to linear velocity commands. In other words, the closed-loop systems behave as a simple integrator. Evidently, this assumption is reasonably valid only for small velocities. Our methodology assumes a more complex quadcopter model, which enables the consideration of the vector field in the design of the inner layers of the controller. This fact is responsible for better performance of the controller and enables higher velocities with aggressive flights. We consider that a drone has an aggressive flight when it is moving in high speeds and the orientation dynamics cannot be disregarded without leading to a poor representation of the system. In such case, the vector field, which is originally designed for a first order system, cannot be successfully applied by considering independent lower-level controllers. In the literature, the consideration of the vector field in the inner layers of the quadcopter control has already been studied in [87]. However, only static curves are considered. Our approach considers time-varying curves as well.

Another feature of the proposed controller is that it inherits the property of being a controller for paths, not for trajectories. As it will be better discussed in Chapter 2, following a path is less restrictive than tracking a trajectory. Besides, path controllers can possibly deal better with common practical issues such as some types of motion failure and saturation. For a variety of navigation problems where no synchronization is required, a path following approach may be a better choice given these advantages. On the other hand, a trajectory tracking approach may still be necessary when there are specific position/time constraints in the problem, *i.e.* the robot should be in a specified configuration at each instant.

An additional contribution of this work is the execution of experiments with two real quadcopters, the DJI M100 and the ESPcopter. Additional aggressive flights, with higher speeds, were tested only on simulations. Nevertheless, simulators that consider the full dynamics of the quadcopter were used to illustrate these flights with high velocities.

1.4 Publications

In this section, we list the academic works published during the Ph.D. studies.

1.4.1 International journals

- **Adriano M. C. Rezende**; Vinicius M. Gonçalves; Luciano C. A. Pimenta. *Constructive time-varying vector fields for robot navigation.* **IEEE Transactions on Robotics (T-RO)**, 2021.
- **Adriano M. C. Rezende**; Vinicius M. Gonçalves; Luciano C. A. Pimenta. *Safe coordination of robots in cyclic paths.* **Elsevier ISA Transactions**, Volume 109, March 2021, Pages 126-140.
- **Adriano M. C. Rezende**; Victor R. F. Miranda; Vinicius M. Gonçalves; Gustavo M. Freitas. *An integrated solution for an autonomous drone racing in indoor environments.* **Springer Intelligent Service Robotics (JISR)**, 2021.
- Victor R. F. Miranda; **Adriano M. C. Rezende**; Thiago L. Rocha; Héctor Azpúrua; Luciano C. A. Pimenta; Gustavo M. Freitas. *Autonomous Navigation System for a Delivery Drone.* **Springer Journal of Control Automation and Electrical Systems**, 2021.
- Gilmar P. Cruz Júunior; **Adriano M. C. Rezende**; Victor R. F. Miranda; Rafael Fernandes; Héctor Azpúrua; Armando A. Neto; Gustavo Pessin; Gustavo M. Freitas. *EKF-LOAM: an Adaptive Fusion of LiDAR SLAM with Wheel Odometry and Inertial Data for Confined Spaces with Few Geometric Features.* **IEEE Transactions on Automation Science and Engineering (T-ASE)**, 2022.
- Héctor Azpúrua; **Adriano Rezende**; Guilherme Potje; Gilmar Júnior; Rafael Fernandes; Victor Miranda; Levi Filho; Jacó Domingues; Filipe Rocha; Frederico Sousa; Luiz Barros; Erickson Nascimento; Douglas Macharet; Gustavo Pessin; Gustavo Freitas. *Towards semi-autonomous robotic inspection and mapping in confined spaces with the EspeleoRobô.* **Springer Journal of Intelligent and Robotic Systems (JINT)**, 2021.

1.4.2 International conferences

- **Adriano M. C. Rezende**; Vinicius M. Gonçalves; Arthur H. D, Nunes; Luciano C. A. Pimenta. *Robust quadcopter control with artificial vector fields.* **IEEE International Conference on Robotics and Automation (ICRA)**, Paris, 2020.

- **Adriano M. C. Rezende**; Gilmar P. C. Junior; Rafael F. G. da Silva; Victor R. F. Miranda; Héctor I. Azpúrua; Gustavo Pessin; Gustavo M. Freitas. *Indoor Localization and Navigation Control Strategies for a Mobile Robot Designed to Inspect Confined Environments.* **IEEE 16th International Conference on Automation Science and Engineering (CASE)**, Hong Kong, 2020.
(*This publication was a finalist for the Best Student Paper Award*)

- **Adriano M. C. Rezende**; Victor R. F. Miranda; Henrique N. Machado; Antonio C. B. Chiella; Vinicius M. Gonçalves; Gustavo M. Freitas. *Autonomous System for a Racing Quadcopter.* **IEEE 19th International Conference on Advanced Robotics (ICAR)**, Belo Horizonte, MG, 2019.

- Leonardo A. A. Pereira; Arthur H. D. Nunes; **Adriano M. C. Rezende**; Vinicius M. Gonçalves; Guilherme V. Raffo; Luciano C. A. Pimenta. *Collision-free vector field guidance and MPC for a fixed-wing UAV.* **IEEE International Conference on Robotics and Automation (ICRA)**, China, 2021.

- André Cid; Mateus Nazário; Mauricio Sathler; Frederico Martins; Jaco Domingues; Mário Delunardo; Paulo Alves; Rodrigo Teotônio; Luiz Guilherme Barros; **Adriano Rezende**; Victor Miranda; Gustavo Freitas; Gustavo Pessin; Héctor Azpúrua. *A Simulated Environment for the Development and Validation of an Inspection Robot for Confined Spaces.* **2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)**.

1.4.3 National conferences

- Douglas Coutinho; Israel Amaral; **Adriano M. C. Rezende**; Hector Azpúrua; Gustavo Pessin; Gustavo Freitas. *Análise e Comparação dos Algoritmos RRT* e Dijkstra para Planejamento de Caminhos em Terrenos Acidentados.* **Simpósio Brasileiro de Automação Inteligente - SBAI**, Online, 2021.

- Lucca Leão; Israel Amaral; Hector Azpúrua; **Adriano M. C. Rezende**; Gustavo Pessin; Gustavo Freitas. *Exploração autônoma de ambientes planares utilizando um dispositivo robótico móvel.* **Simpósio Brasileiro de Automação Inteligente - SBAI**, Online, 2021.

- Alvaro Araujo; Gilmar Júnior; David Marques; Victor Miranda; **Adriano M. C. Rezende**; Paulo Alves; Hector Azpúrua; Gustavo Pessin; Gustavo Freitas. *Comando de direção de antenas de rádio para teleoperação de robô móvel.* **Simpósio Brasileiro de Automação Inteligente - SBAI**, Online, 2021.

- Arthur H. D. Nunes; Álvaro Araújo; Maria Luiza A. Alves; **Adriano M. C. Rezende**; Isabela Braga; Luciano C. A. Pimenta. *Controle de robô para serviços de inspeção industrial utilizando campos vetoriais artificias variantes no tempo.* **XXIII Congresso Brasileiro de Automática (CBA)**, Porto Alegre, RS, 2020.

- Israel F. S. Amaral; Carolina F. A. Duarte; David S. Marques; Lucas Matos; Álvaro Araújo; Lucca G. Leão; **Adriano M. C. Rezende**; Héctor I. Azpúrua; Gustavo Pessin; Gustavo M. Freitas. *Sistema de Alertas e Operação Assistida de um Robô para a Inspeção de Ambientes Confinados - EspeleoRobô.* **XXIII Congresso Brasileiro de Automática (CBA)**, Porto Alegre, RS, 2020.

- Thiago L. Rocha; **Adriano M. C. Rezende**; Victor R. F. Miranda; Héctor I. Azpúrua; Gustavo M. Freitas. *Desenvolvimento de um drone autônomo para tarefas de entrega de carga.* **XXIII Congresso Brasileiro de Automática (CBA)**, Porto Alegre, RS, 2020.

- **Adriano M. C. Rezende**; Henrique N. Machado; Victor R. F. Miranda; Antonio C. B. Chiella; Vinicius M. Gonçalves; Gustavo M. Freitas. *Planejamento de trajetória, localização e controle aplicados a um quadrirrotor autônomo de corrida.* **14º Simpósio Brasileiro de Automação Inteligente (SBAI)**, Ouro Preto, MG, 2019.

1.5 Available library

An implementation of the theoretical methods presented in this thesis is available online at https://github.com/adrianomcr/vectorfield_stack. The software is implemented in Python and is already compatible with ROS. In fact, the library called *vectorfield_stack* is a collection of ROS packages. It focuses on vector field control codes

designed for different robotic platforms: holonomic integrator, differential drive, skid-steering drive, and quadcopter. One main package, called *distancefield* contains the main implementation of the novel vector field methodology. Additional packages provide the additional controllers to use the vector field to control the different robots. The packages are *ground_robot*, and *quad_robot*.

The code of the library vectorfield_stack was registered along with The Federal University of Minas Gerais (UFMG) and the Vale Institute of Technology (ITV). In fact, the algorithms are not only used in experiments performed in the UFMG campus, but also in experiments with the EspeleoRobô in gallery inspection missions performed by the ITV's robotics group[11]. Appendix C shows the Certificate of the software registration.

The *vectorfield_stack* also counts with lightweight simulators for different robotic platforms, which provide an easy way to execute and test the control codes. The available robots are depicted in Figure 1.5. The rviz[12] is used as the graphic interface for the robot simulators. Figure 1.6 illustrates a simulation performed with the library control codes. There, a quadcopter is controlled to follow a circular path while deviating from spherical obstacles.

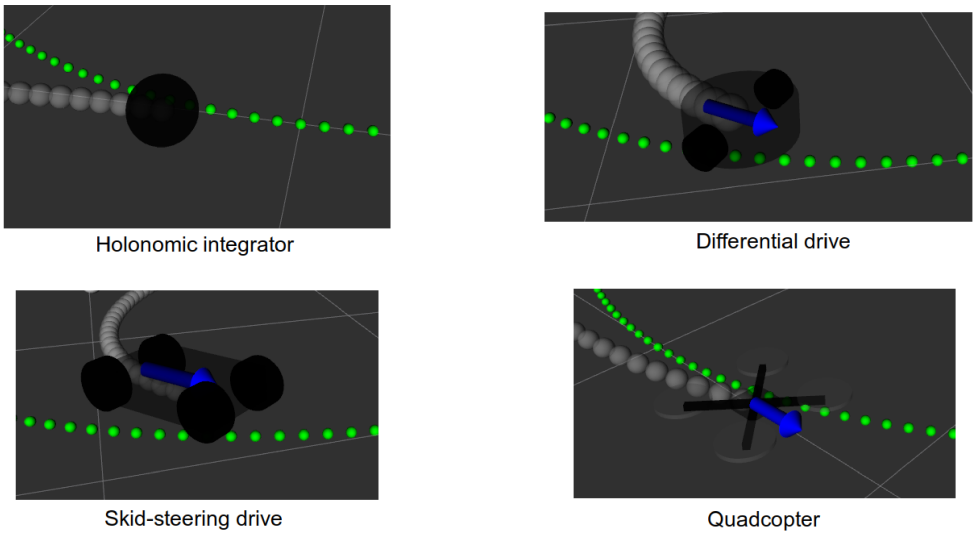

Figure 1.5: Robots implemented in the package *robotsim* of the *vectorfield_stack*.

[11] https://www.itv.org/en/itv-mining/research-groups-and-partnerships
[12] http://wiki.ros.org/rviz

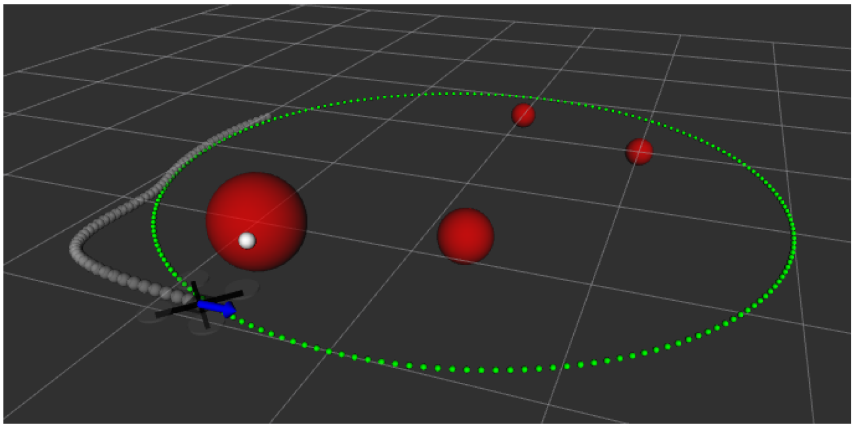

Figure 1.6: Frame of the simulation performed with the *vectorfield_stack*. A drone follows a circular path while deviating from the red obstacles.

1.6 Text structure

Following this introduction, Chapter 2 contains a review of the works in the literature that tackle the guidance and control problems in the robot navigation scope. In Chapter 3, the novel Euclidean distance based vector field methodology is introduced. In Chapter 4, a controller able to make a quadcopter follow the proposed vector field is presented. In Chapter 5, several simulation and experimental results are presented to exemplify the methodologies presented in Chapters 3 and 4 and show how they can be applied in several experimental scenarios. Finally, in Chapter 6, the work is concluded and future research topics are suggested.

Chapter 2

Related works

Instead of focusing on very basic theoretical fundamentals, in this chapter we concentrate on strongly related and state-of-the-art works. Section 2.1 focuses on the high-level guidance problem, while Section 2.2 on the quadcopter control problem.

2.1 Guidance

In this Section, we first review some works that specify the difference between the path following and the trajectory tracking problems. Then, we present a literature review on solutions for the path following problem and focus on vector field methodologies. After that, we discuss how the vector field proposed in this work relates to the discussed references.

2.1.1 Path following and trajectory tracking

In the literature, many authors consider a subtle difference between a *path* and a *trajectory*. By *path*, we should understand one-dimensional continuous manifold $\mathcal{C}(\mathrm{t})$ that can be described by a possibly time-varying parametrization. By *trajectory*, we mean a time-dependent reference point (or state) $\mathbf{x}_{\mathrm{r}}(\mathrm{t}) \in \mathbb{R}^{\mathrm{n}}$. In [70], a trajectory tracking problem is defined as a specified reference in time that should be tracked, in which the references are given by a temporal evolution of each spatial coordinate. In the path following problem, the authors of [70] assume that there is no preassigned timing information, thus, any time-dependence of the problem is removed. According to our more generic definition of the path following problem, the path may be time-varying, thus, some time dependence is allowed. The important detail in the definition of a path following problem

is that, given a specific time, no point in this path is a special reference. Thus, although the path varies on time, there is no specific point on this path where the robot should be in a given instant. Another common misconception in the definition of path following problems is that the vehicle's speed must be constant. Actually, a path can also be followed with a given velocity profile [32].

Some works that point to the advantages of a path controller over the trajectory tracking are available in the literature. As pointed out by the survey in [70], controllers for paths have the following properties: (i) are easier to design due to a decoupling of space and time [32]; (ii) present a smoother convergence pattern [12]; (iii) have a smaller transient error and a stronger robustness [77]; and (iv) have less control effort with control signals that are less likely to saturate [19]. These advantages were already observed in practice. In [82], the authors present comparisons of trajectory and path controllers and show that path controllers perform better in practical situations in which measurement noise is present [82]. Figure 2.1 presents a comparison of the response of a path (left) and a trajectory (right) controllers under the presence of noise. The curves in magenta represent the position of the robots and it is clear that the path controller is less sensitive to measurement noises. To obtain the trajectory tracking controller illustrated on the right, the authors simply replaced the path parameter used in their methodology by the time t.

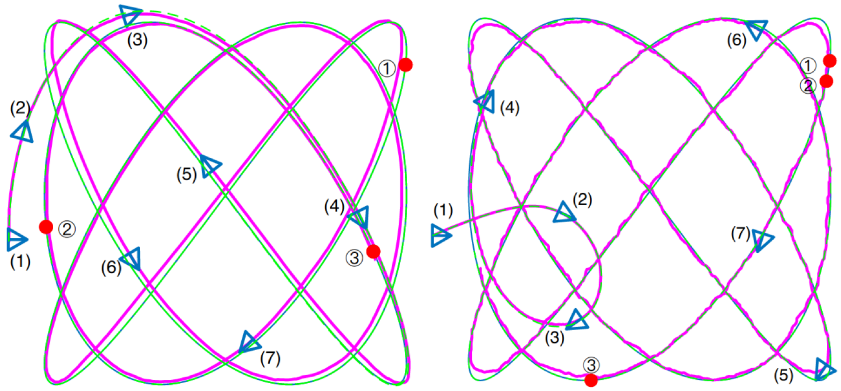

Figure 2.1: Comparison between the robustness against measurement noise of a path controller (left) and a trajectory tracking controller (right). Source: Image extracted from [82].

The work in [65] also presents a comparison between path and trajectory controllers. Inspired by the comparison in [65] and in order to illustrate some of the path following advantages described before, Figure 2.2 presents the result of some simulations in which three common practical failures were induced. The first one is when the initial position of the robot is far from the reference for tracking. The second is when the robot stops for some time interval. This problem can be illustrated by a wheeled robot slipping. The third is when the controllers of the actuators are saturated. Figures 2.2-a shows a response of a trajectory tracking for the first two possible problems. They correspond to undesirable transient behaviors when the initial reference is far from the robot or when

the robot returns to move after a motion failure. Figure 2.2-b shows the third issue, also for trajectory tracking, when there is a saturation of the input velocity. When the robot's maximum velocity is smaller than the velocity of the reference the vehicle tends to "cut corners", and consequently does not follow the desired curve exactly. Depending on the task the robot is supposed to perform, this is an undesired behavior. Figure 2.2-c shows how the path controller passes through these failures without any problem in the curve following. The simple integrator model was considered for the simulation. The norm of the velocity reference for the trajectory and for the path controller is 1. When the saturation was simulated, we considered a maximum velocity of 0.9.

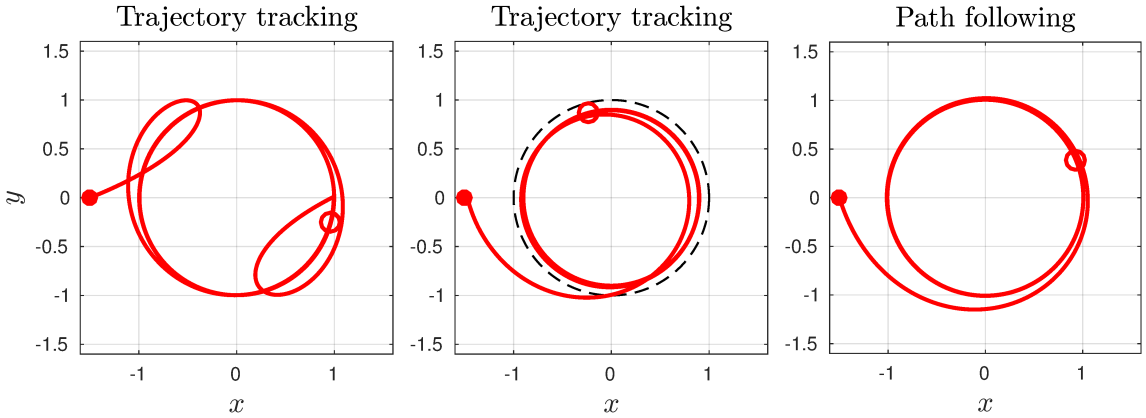

Figure 2.2: Example of three situations in which a traditional trajectory controller behaves undesirably and a path controller is not affected. The three failures are: initial position far from reference (a); motion failure during some interval (a); and controller saturation (b). In (c) the behavior of the path controller, which is not affected by these problems.

Evidently, all problems of trajectory tracking depicted in Figure 2.2 can be avoided. One option is to consider a supervisor that will detect these situations and take an action to treat them. It is also important to emphasize that in some situations a path following approach may not be applicable [71]. Note that, when using path following, given a specific future time, it may not be possible to determine where in the curve the robot will be. Even if the system is working properly at this time, the robot may be at any point of the curve. This behavior is a drawback when some sort of synchronization of tasks is necessary. For instance, in a production line, multiple manipulators may be required to be synchronized in order to optimize the production and a trajectory tracking may be a better choice.

The fact that a path controller is less aggressive than a trajectory tracking was formally studied in [2]. The authors show that following a path is a less restrictive task than following a trajectory, in the sense that it does not have its performance limited by the energy of the control effort. They demonstrate that, in trajectory tracking, the smallest achievable $\mathcal{L}_2$ norm of the tracking error is limited by the least amount of control energy required to stabilize the zero dynamics of the error system. This limitation is not present for the task of following a path, even when a velocity profile is considered, as in [32, 19]. This fact is an additional motivation for the study of vector field based strategies.

2.1.2 Solutions for the path following problem

In the literature, path controllers can be categorized into two groups, Separated Guidance and Control (SGC) and Integrated Guidance and Control (IGC) [70]. In the SGC approach, a high-level guidance is designed and an inner control loop is considered to impose the guidance to the robot [17]. In the IGC, the robot model is taken into account during the design of the guidance law [13] or the guidance law is considered in the inner loop control layers. In general, SGC are simpler and easier to design, however, many SGC strategies lack convergence proofs.

One of the strategies to obtain a controller for paths is to adapt an existing controller for trajectories. *Backstepping* is a widely used method for trajectory tracking. In [13], the authors present a strategy to obtain path controller from a Backstepping approach. First, Backstepping is used to obtain controllers for the thrust force and torques of a quadcopter. The original law may be undefined in some cases. To solve this issue, the degree of freedom provided by the path parameter is explored and a timing law for this parameter is designed. In a simple way, instead of using the physical time to compute the trajectory, there is an additional law that computes this "artificial time", which is the path parameter. The final result is an IGC for paths. Similar strategies are also proposed in [12] and [72]. Another control strategy that has also been used is Feedback Linearization [69]. In this strategy, a linear model is obtained by inverting the original nonlinear model, then a linear controller can be deployed. In Section 2.2.2, we will specify how this strategy can be used in an inner control loop of the non-holonomic robot described in Section 1.2.3.

Lyapunov Theory has also been used to obtain controllers to solve the path following problem and obtain convergence proofs. The characteristics of such methods are very diverse, and they are usually constrained to the three dimensional space [19, 52]. As an example, in [19], Lyapunov theory is used to obtain convergence proofs for a path following approach designed to guide a quadcopter that has the thrust force and the angular rates as input controls. In [51], the path following problem is tackled with the so-called maneuvering approach. As stated in [76], this strategy consists of two stages: (i) a geometric task that aims to guide the robot towards the paths; and (ii) a dynamic task that aims to assign some velocity or acceleration to the robot. The strategy is similar to defining a timing law for the path parameter. To solve both stages of the strategy, the work in [51] needed a velocity measurement of the vehicle. A velocity observer is implemented with this purpose. Lyapunov theory is also recurrent in the study of vector fields as path controllers, see [41]. A more extensive discussion on these approaches is made in Section 2.1.3.

A geometric approach is also considered to solve the path following problem. Ge-

ometric techniques are usually simple high-level strategies that do not consider a detailed robot model, which are used in an SGC approach. They are characterized by the use of a geometric entity to guide the robot to the path, for instance, straight lines that connect the vehicle and the path. A simple geometric approach is the Pure Pursuit [57], which guides the robot towards a straight line to the path. Such methods are differentiated by the strategy used to define the point on the path to be pursued. For instance, in the Line-Of-Sight (LOS) methods this point is defined as the closest one in the path [3]. The Virtual Target Point (VTP) is another example of a geometric technique [53]. The idea is similar to a trajectory tracking control. However, the target point is not defined by time, but by a propagation along the path of the current robot's position. This method is also referred as Carrot-Chasing. The Non-Linear Guidance Law (NLGL) is also another technique based on the virtual target point idea. The VTP is defined by the intersection of a circumference centered in the robot and the path. Another work that may be classified as a geometric method is [17], in which the Frenet-Serret frames associated with the path are used. They consider the closest point map projection to define the point on the path from which the Frenet-Serret frame will be computed. However, these works are limited to three dimensions [10], and convergence proofs are only available for constant-curvature planar paths.

The works in [78, 72] use a Nonlinear Model Predictive Control (NMPC) to solve the path following problem. This type of strategy consists in the computation of control inputs that minimize a given cost functional in a future time horizon. The controller is able to make the robot deviate from obstacles and simulations considering a unicycle robot are presented to illustrate the approach. Another work that tackles the problem with Model Predictive Control (MPC) is [55], in which spline paths in three dimensions are considered in an SGC structure.

In the survey presented in [70], the authors show that path controllers are mostly used in two forms: (i) adaptation of trajectory tracking strategies, such as the ones based on Feedback Linearization and Backstepping; and (ii) geometric strategies such as LOS and VPT. The former have formal convergence proofs and the latter are less formal. However, the authors conclude that the less formal strategies perform better than the trajectory tracking adaptations for path following. The vector field approach presented in this work, better discussed in Section 2.1.5, has formal convergence proofs, is directly developed for path following, and has exhibited good performance in real applications.

2.1.3 Vector field construction methodologies

In the last decades, several methodologies to construct artificial vector fields were proposed. They usually consider the simple integrator model system. Thus, in the scope of robot navigation, the interpretation of a vector field is a velocity to be imposed to the vehicle. In general, a lower-level controller is responsible to impose the velocity to the robot, categorizing an SGC approach. In this section, we present a review of such methodologies.

In [41], vector fields that converge to circular loiters are developed, and convergence proofs are presented with Lyapunov Theory. Fields that converge to different curve shapes are obtained through diffeomorphism. The methodology presented in [41] considers only planar curves, for instance, fields in 2D or fields in 3D converging to curves with a fixed height. In [26], the authors extend the methodology considering a team of UAVs and moving curves. In [16], a method is developed to make a group of robots converge to and circulate a given beacon, which is possibly moving. However, no specific curve is defined to be circulated in this case.

The authors of [44] propose two static 3D vector field strategies and use them to guide a UAV. The proposed field is based on the tangent vector associated with a given point in the curve and a distance measurement. Convergence proofs are also presented.

In [29], vector fields are built based on optimization techniques. In that case, the user can include constraints such as speed limits and obstacle avoidance into the vector field as constraints of the optimization problem. The technique can be used to drive UAVs towards the curve and also to circulate it. However, it requires the user to provide a function that codifies the target curve as a zero-level set. No constructive methodology for generating these curves is provided. The vector field was tested on a humanoid robot. The technique presented in [46] has a similar philosophy, computing the vector field by solving an optimization problem. In that case, the focus is a surgery application.

In [81], the authors develop a method to construct an artificial vector field from a parametric curve. The authors already pointed out the simplicity of the use of parametric representations over the implicit representations in [27]. In [81], contraction analysis theory is used to provide convergence proofs for time-invariant curves in generic manifolds. The method proposed in this work gives equivalent results when the manifold is $\mathbb{R}^n$, and the used metrics is the Euclidean distance. Besides, we consider time-varying curves and disturbances in the model.

Vector fields are constructed for star-shaped curves in [24] and [25]. The method is limited to curves embedded in three dimensions. A radial Fourier basis set is used to represent the curves and convergence proofs are presented. Curves represented by a sequence of points are also considered, however, the methodology consists in an analyti-

cal approximation. In consequence, the convergence properties of the resulting field are dependent on the behavior of these functions, which can be undesirable.

One of the most used methodologies to construct artificial vector fields available in the literature has been proposed in [27]. Its features include: (i) fields for n-dimensions; (ii) consideration of time-varying curves; and (iii) formal convergence proofs. As we anticipated in Section 1.1.1, this strategy considers a curve represented by the intersection of $n-1$ zero-level sets given by $\alpha_i = 0,\ i = 1, ..., n-1$, in which $\alpha_i : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}$. Figure 2.3 presents two examples of intersection of zero-level sets. Both examples illustrate the space $\mathbb{R}^3$, in which the zero-level sets are two dimensional surfaces, represented by $\alpha_1 = 0$ and $\alpha_2 = 0$. For some particular cases, such as the ones depicted in Figure 2.3, these functions can be easily obtained, however, for more generic cases, this may be a difficult task.

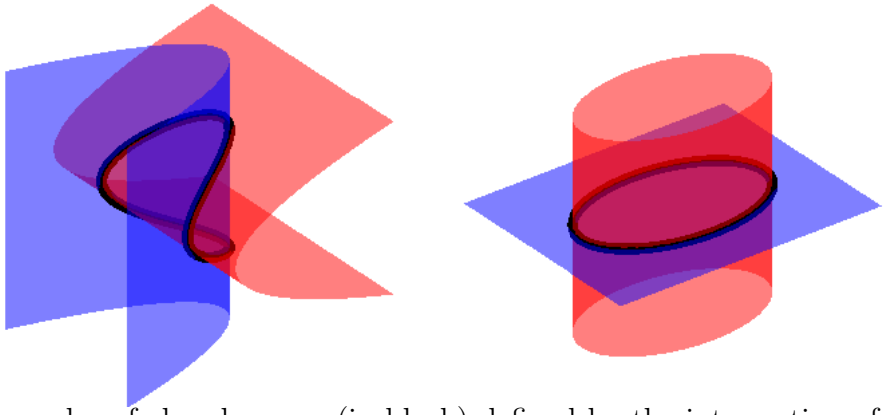

Figure 2.3: Examples of closed curves (in black) defined by the intersection of zero-level sets (blue and red surfaces).

The recent work in [83] presents a vector field methodology for n dimensions and static curves represented by a parametric equation. The methodology consists of an extension of the n-dimensional space with another dimension, which is correspondent to the parameter of the curve's representation. Then, the authors based on [27] to define zero-level sets in this extended space of $n+1$ dimensions. The curve in $\mathbb{R}^n$ is defined by a vector of parametric equations $\mathbf{r}(s) = [r_1(s),\ ...,\ r_n(s)]^T$. For the original problem in n dimensions, n functions $\alpha_i : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ are defined as $\alpha_i(x_i, s) = x_i - r_i(s),\ i = 1, ..., n$. An important feature reached by this approach is that the resulting vector field is singularity-free, meaning that there are no points in the space in which the field is not properly defined or is the null vector. In addition, the methodology can be applied to curves with self-intersections, since in the extended dimension the curve does not intersect itself. Figure 2.4 illustrates the idea of the work in [83] for a two dimensional space. On the left, we show the original curve in dashed black and the curve in the extended dimension in solid black. On the right, we show the zero-level surfaces of the functions $\alpha_1 = x_1 - r_1(s)$ (red) and $\alpha_2 = x_2 - r_2(s)$ (blue). The level set in red extends to $-\infty$ and $+\infty$ in the x_2 direction, while the set in blue extends analogously in the x_1 direction. The sets were limited on

the figure only for a better visualization. The obtainment of the two level sets may be visualized as the area swept by the green cross while it moves along the extended curve.

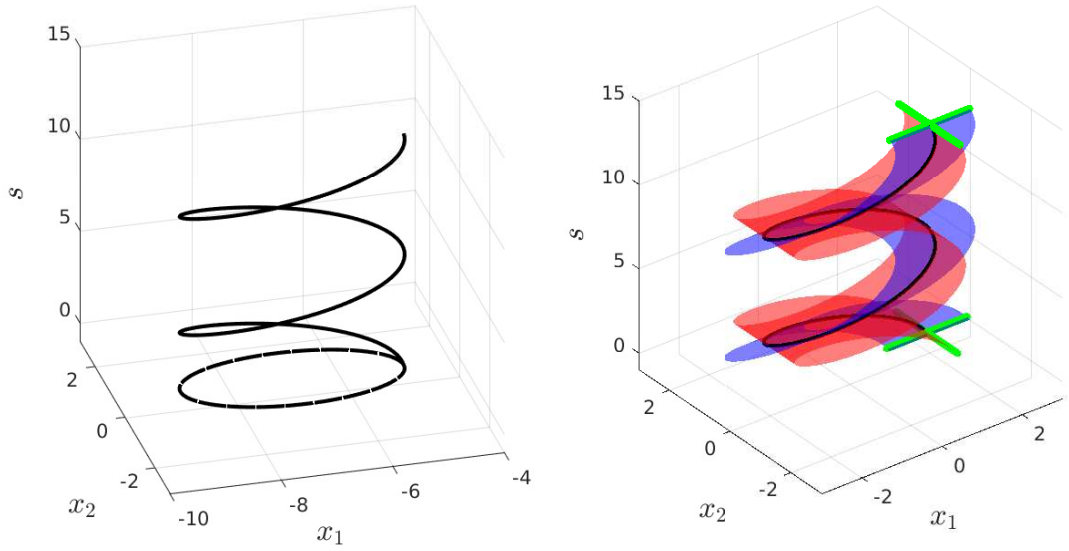

Figure 2.4: Expansion of the curve in two dimensions to $\mathbb{R}^3$ on the left. Zero-level sets in the extended dimension on the right.

2.1.4 Obstacle avoidance

A recurrent issue in the robot navigation problem is the presence of obstacles. While some strategies assume a workspace free of obstacles [27, 83], others incorporate the ability to deviate from these structures.

In general, the methodologies that tackle the presence of obstacles in the robot's workspace consider approaches that can be classified either as reactive or deliberative. A reactive approach considers a navigation free of obstacles in a first moment and takes an action to avoid a collision when an obstacle appears. It is, in general, detected by a sensor. A deliberative approach considers some path planning that computes a path free of collisions. This planning usually requires a knowledge of the map and the geometry of the obstacles. An advantage of a reactive approach is that it is usually simpler, can be easily applied to unknown environments, and has a low computational cost. However, it may lead the robot to undesirable conditions and generate non smooth paths. Deliberative approaches may solve these issues but, usually, under a higher computational cost.

A simple and widely known reactive strategy is the repulsive field action [38]. It consists on the addition of a reference velocity component that points in the opposite direction of a close detected obstacle. In the case this repulsive action points in the opposite direction of the original reference, and cancels it, the robot is locked. Since this

methodologies are originated from a scalar potential field, we usually say that the robot is stuck in a local minimum of this field.

The Bug algorithms [18] are also good examples of navigation algorithms that incorporate the ability to deviate from obstacles in two dimensions. The navigation method has the primary objective of taking the robot to a target point. The deviation strategy consists in the circulation of the detected obstacles. The adopted approach is then reactive. In fact, the algorithms switch between two behaviors. The first is to follow the so called m-line, which connects the start and goal points. The second is the obstacle contouring, referred also as boundary following. Bug 1 and Bug 2 algorithms [45] assume a contact sensor that allows the robot to detect the obstacle and follow its boundary. The Tangent Bug [33] uses a range sensor that allows the robot to choose shorter distances when moving towards the goal and deviating from obstacles. The algorithms are simple to implement, have a low computational cost and have formal proofs that the goal point will be reached when it is possible (they are complete algorithms).

The Bug algorithms have the primary objective of reaching a single goal point. Other obstacle avoidance strategies have been studied when the primary objective of the navigation is to follow a path. In [31] the authors consider a deliberative approach in which a vector field together with a Rapidly-exploring Random Tree (RRT) make a fixed-wing UAV converge to a curve while deviates from obstacles. A RRT [35] is a probabilistic planner based on randomly sampled points and a tree structure. The work in [31] uses this planner to contour the obstacles that may be in the robot's way when it is following the vector field. In fact, the authors use the Vector Field Rapidly-exploring Random Tree (VFRRT) [39] to grow the tree that defines a path that deviates from the obstacles. This algorithm uses the original vector to favor the growth direction of the tree. The authors in [31] also incorporate a Dubins airplane model [48] to better represent the dynamics of the UAV.

Adopting a similar idea of [31], in [58], the authors use the RRT*, cost optimum version of RRT, together with a vector field approach to make a semi-autonomous robot execute user defined movements. The commands defined by the human operator are used to define a simple vector field. The RRT* is grown to maximize the alignment of the path with the field's trajectories, or integral lines. When obstacles are in the way, the planner generates a smooth path that attempts to follow the vector field as closely as the obstacle sets allow, *i.e.*, the path is collision free. Although being a deliberative approach, the methodology presented in [58] does not require a global localization, but only a local map and the detection of obstacles.

In [60], the authors consider a reactive approach that makes a fixed-wing UAV deviate from obstacles when it is following a vector field that converges to a path. The strategy is close to the one in the Bug algorithms in the sense that the obstacles are contoured when they are detected in the robot's way. When the obstacles are detected,

with a range sensor for instance, the algorithm switches to a secondary vector field that contours the obstacle until the direction of the original field is safe for navigation. Although the UAV is allowed to move in three dimensions, the obstacles are only circulated horizontally. In fact, the authors assume that the obstacles are vertical cylinders.

2.1.5 Path following approach in this thesis

The approach used in this thesis to tackle the path following problem is based on the design of a novel artificial vector field, presented in Chapter 3. Although we later focus on the control of a quadcopter, this field can be used in the guidance of several robotic platforms. Its main feature is the use of the Euclidean distance function, which provides interesting and useful properties. In the following, we emphasize these advantages over the already discussed methodologies.

The vector field presented here may be considered as an improvement of the work in [27]. Although they both, basically, consist of three components, associated with convergence, circulation, and time-feedforward movement, the current approach has the following advantages: simplicity on the representation of the curves; homogeneous convergence pattern, which comes from the use of the Euclidean Distance; possibility to generate constant norm time-varying vector fields; and absence of undesirable equilibrium points.

The Euclidean distance field can be constructed from a parametric representation of the curve, a feature already present in the literature [83, 81]. This is an important advantage over the implicit definition considered in [27], which may complicate the obtainment of the field for more complex curve shapes. Similar to [44] and [17], we use the vector tangent to the curve, easily obtained from the parametric representation, as the traversal component of the field. Although our proposed field can be constructed upon a parametric representation, it is entirely defined given the unidimensional set that defines the curve. In other words, *the approach is invariant and independent of the parametrization*. This enables an easy computation of the field given a curve represented by a sequence of points. If a large and smooth sequence of points is available, the method can be easily applied numerically. If only a few way-points are available, the proposed strategy can be applied after a simple polynomial interpolation [4, 68].

The feature that characterizes the methodology of this work is that it uses the n-dimensional Euclidean distance function to encode the path following error, instead of other analytic Lyapunov functions [27]. In fact, the distance function may be seen as a Lyapunov function for the problem. This idea is similar to the closest point map projection considered in [17]. The use of this function originates a homogeneous convergence pattern

that depends only on the distance from the curve. That means that the field's convergence action has an equal intensity along the whole curve. When this homogeneity is not present, the calibration of the field, *i.e.* the definition of the convergence gains, may be more complicated. For instance, the field behavior may be satisfactory around some parts of the curve and undesirable, too aggressive or too smooth, around others.

As all the other methodologies to construct vector fields, our method assumes an integrator model. Then, the vector field consists in a velocity reference for the robot. Our theory presents a disturbance analysis that evaluates the maximum error in the path following given a limited error on the execution of the commanded velocity. As in [64], an ultimate bound set is achieved when disturbances are present. In practice, these disturbances may incorporate several effects, such as wind, and imperfections of lower-level controllers. One interesting fact about this result is that the boundary of the invariant set is defined by a level set of the Euclidean distance to the curve. It is more intuitive than the invariant set that can be computed by using the zero-level sets of [27], whose size and shape will depend on the analytic properties of the functions. Besides, the set is isotropic, meaning that the bound of the invariant set has a constant distance to the curve.

An important and novel property of the vector field presented here is that it has a constant norm even for time-dependent cases, a useful feature for practical applications. Of course, this requires the assumption that is possible to match the curve's speed with the robot's speed. The normalization of the field is a common practice in the literature [27, 81, 83], since it is often desirable to make the robot move at a constant speed. The normalization problem for time-varying fields is not as trivial as the normalization of static ones. This is due to the necessity of maintenance of the computed feedforward component that accounts for the movements of the curve. For example, in [28], a constant norm vector field is obtained by neglecting the time feedforward component. Although stability is obtained, asymptotic stability is lost. In our methodology, the normalization of time-dependent fields is achieved without negative effects on the asymptotic convergence.

The distance function between a point and a curve is defined globally. However, the closest point on the curve may not be uniquely defined at some particular points in $\mathbb{R}^n$. Take the center of a circle as an example, the distance function is the circle radius, but all points in the circle have the same distance to the center. This fact makes our vector field be non-uniquely defined in a set, proved to have measure zero in $\mathbb{R}^n$. In this sense, the work in [83] has an advantage over our approach, since they propose a singularity-free vector field, obtained with the consideration of an extra dimension. However, they do not consider time-varying curves nor disturbances on the velocity commands.

In fact, we can not say that the Euclidean distance vector field is singularity free as the one in [83]. In this scope, we provide a discussion section that argues that these singular points of the vector field do not offer a problem in practice. Besides, as an attempt

to weaken this issue, we also propose a slightly modification in the original vector field that deals with these singular points. The approach, is inspired on [83], considers an additional state variable that represents the parameter of the curve that is being followed. A better mathematical formulation of this method will be left for a future research, as discussed in Chapter 6

Regarding the obstacle avoidance strategy, this thesis presents a methodology that allows the robot autonomously deviate from obstacles. We consider a reactive approach that makes the robot circulate the obstacle until the direction indicated by the original vector field is safe for navigation. The approach may be considered as an extension of the one presented in [60], where the obstacles are assumed to be cylinders. Here, there is no assumption on the shape of the obstacles. Moreover, the proposed extended vector field may deviate in all possible directions, not only on the horizontal. First, we show how the Euclidean distance vector field can be easily applied to contour an obstacle. Basically, we show how the closest point of an imaginary path around the obstacle can be inferred from the closest point of the obstacle. Then we use this contouring vector field to define a composite field that incorporates the ability to guide the robot towards the curve while deviating from the obstacles.

The proposed Euclidean distance vector field is used in a SGC structure to control a ground robot in a real world experiment and a 6 DOF manipulator in a simulation. Besides, simulations and real robot experiments with quadcopters are also performed in an IGC structure, see the next Section. Other experiments and simulations show the additional features of obstacle deviation and treatment of singularities on the original field.

2.2 Control

In this Section, we review some related literature on controllers destined to quadcopters. Then, we present some works that use artificial vector fields to control specific robots to follow a given vector field. Finally, we show how the controller presented in this work fits in the current state-of-the-art.

2.2.1 Quadcopter controllers

The control of quadcopters has been extensively studied in the last years. The use of trajectory tracking for the control of quadcopters is much more present in the literature than the use of path controllers. For instance, in [84], the authors based their trajectory tracking control on Feedback Linearization. The vehicle is modeled with the Newton-Euler approach and a cascade connection is used to control the translation and rotational dynamics.

Backstepping is also a widely used technique to solve the trajectory tracking problem for quadcopters. In [14], the authors consider a Backstepping approach that is able to reject constant force disturbances. Two modes of operations are considered, each one considers a different actuation of the rotation dynamics. The first mode assumes the torques as inputs, while the second assumes the angular velocities. One interesting advantage of [14] is that actuation laws depend only on bounded functions of the position, which prevents very aggressive flights when the initial position error is elevate. The authors present real robot experiments that use a VICON system for localization[1].

Recent works have also tackled the quadcopter control with MPC. In [54], an MPC strategy is used to make a quadcopter reach a given point in minimum time while deviating from previously unknown obstacles. In [59], an NMPC is developed to make a quadcopter track a planned trajectory, which provides references for position, velocity, yaw angle and its derivative. The model assumes the total thrust and the torques as control inputs. The controller is also able to deviate the vehicle from obstacles whose positions and sizes are provided. One advantage of the MPC based approaches is that they are able to explicitly consider the saturation of the actuators.

Trajectory tracking controllers have also been designed by using the differential flatness property. A given system is said to be differentially flat if there exists a set of outputs, the flat outputs, from which all states can be derived. In other words, the evolution of the states up to a system can be obtained from the flat outputs and their derivatives of a limited order. In [50], the authors show that the vector that contains the 3D positions and the yaw angle of a quadcopter is a flat output for the system. The designed controller has a Proportional Derivative (PD) action and needs a linearization over the hover point. A similar controller is presented in [43], in which a nonlinear tracking controller is developed on the special Euclidean group SE(3). In [23], the authors show that the quadcopter model with the inclusion of drag effects is also a flat system. They design a controller and show that the considerations of drag improve the precision of the trajectory tracking. In [87], the authors use the differential flatness property to make a

[1] https://www.vicon.com/

quadcopter follow a vector field. Their approach will be better discussed in Section 2.2.2.

Recently, the trajectory tracking problem has also been tackled with optimal control theory. In [15], a controller in the weighted Sobolev Space is formulated via dynamic programming. The proposed theoretical controllers are synthesized for a quadcopter. Simulations demonstrate the efficiency of the framework. Another different approach to the quadcopter control is presented in [11], where the authors prioritize the pitch and roll angle control over the heading. With this strategy, it is possible to prioritize the tracking of the position over the yaw. Comparison flights with a physical vehicle illustrate the achieved improvements.

2.2.2 Vector fields for robot control

When vector fields are used to control a robot, the system may fall into the categories IGC or SGC. In the IGC approach, the design of the controllers is tied to the vector field itself. In the SGC approach, there is an independent inner loop controller that receives the vector field velocity reference and tries to make the robot follow this command. In this latter case, an extensive variety of controllers are used.

Feedback Linearization is a widely used technique to impose the behavior of a vector field to robots with the "knife-edge" nonholonomic constraint, such as the unicycle and the differential drive robot [27, 65]. The SGC method computes a linear velocity, v_{rob}, and an angular velocity, ω_{rob}, such that a reference $F = [F_x\ F_y]^T$ is applied to the robot [75]. The linearization is obtained with inverse dynamics of a point at a distance d in the forward direction of the robot, the blue point in Figure 2.5. The Feedback linearization

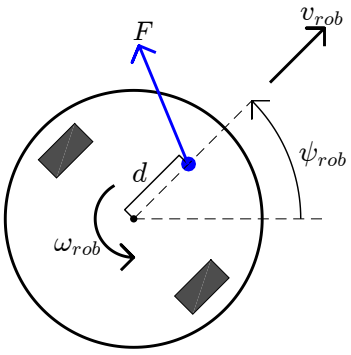

Figure 2.5: Illustration of a differential drive robot. The dynamic of the blue point is inverted in the Feedback Linearization approach in order to impose the velocity F to the robot.

gives a simple law to impose the velocity of a vector field to the robot. The law is given

by

$$\begin{bmatrix} v_{rob} \\ \omega_{rob} \end{bmatrix} = \begin{bmatrix} \cos(\psi_{rob}) & \sin(\psi_{rob}) \\ -\frac{\sin(\psi_{rob})}{d} & \frac{\cos(\psi_{rob})}{d} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}, \tag{2.1}$$

in which ψ_{rob} is the robot's yaw angle. One should have in mind that, in practice, additional inner loop controllers may be necessary. In the case of a differential drive robot, these loops compute the velocity of the wheels, in the case of a fixed-wing UAV, they compute the angles associated with the airplane aerodynamic control surfaces (e.g. ailerons, rudder, and elevators) [56].

The SGC approach is also used to control quadcopters. These vehicles are usually equipped with low-level PID controllers able to make the vehicle track different types of reference. Some examples of low-cost commercial hardware able to perform these controls are the Pixhawk[2] and DJI Naza[3]. Even open source controllers, such as Ardupilot[4] and PX4[5], are able to operate in the so-called Position mode. In this mode, the vehicle responds to velocity commands. For small velocities, absence of significant atmospheric disturbances, and tasks that do not require a high precision, the closed-loop system may be considered as a simple integrator, which is able to follow the velocity given by the vector field. In [28] the authors consider a similar structure to make a quadcopter follow the vector field presented in [27].

When a given vehicle can not be modeled as a single integrator, the vector field equations cannot be directly applied to control the system. For instance, in [27] the authors consider a second-order integrator model and derive additional control laws that impose the dynamics of the vector field to the position states of the system. The structure of the field was taken into account during the design of the controller for the second-order model, which characterizes an IGC structure. In general, these laws depend on the field derivatives, *i.e.* the Jacobian matrix. Each different robotic platform will require a specific controller design.

Vector fields are widely used to control fixed-wing UAVs [56, 64]. In these works, the controllers designed for the airplanes take into account the vector field. In [64], the structure of the field is incorporated in the controller design by using the field's derivatives. These approaches are then characterized as IGC structures. In fact, they assume a simplified nonholonomic model of the vehicle, which is achieved by lower-level PID controllers. From this point of view, one may classify the approach as SGC. Moreover, they consider that the models are uncertain. This is of great interest because it is important to evaluate the application of the method in real scenarios, in which uncertainties are always present. The authors show that in the presence of bounded additive disturbances in the

[2] https://pixhawk.org/
[3] https://www.dji.com/naza-m-v2
[4] https://ardupilot.org/
[5] https://px4.io/

model, the robot converges to an invariant set around the desired limit cycle of the vector field. In [34] the authors use the technique presented in [64] to control a differential drive robot.

Artificial vector fields have already been used to control quadcopters represented by their complete model, *i.e.* not assumed to be well represented by a simple integrator. In [87], a model with 6 degrees of freedom and 12 states is assumed and has the velocity of the propellers as control inputs. The authors use the differential flatness property of the vehicle to design control laws that make the robot behave according to a given vector field. The considered flat outputs are the positions and the yaw angle. Different types of vector fields are also considered in [87], not only the ones that admit a limit cycle. The methodology can then be used to stabilize the system in a given fixed point for example. A limitation of that work is that it does not consider fields with time dependence.

2.2.3 Control approach in this thesis

The control method proposed in Chapter 4 is a solution to the quadcopter path following problem based on artificial vector fields. We assume a vector field proposed by the method presented in Chapter 3. In the following, we compare the properties of our method to the references discussed.

In the literature review presented in [70], the authors classify vector field methodologies for path control as SGC. In fact, the majority of the works that use vector fields to guide these types of vehicles consider independent lower-level controllers to make the closed-loop system behave as the simple integrator. The theory that we present in this text has an IGC structure, which shows an improvement over several existing methodologies. To the best of our knowledge, the work in [87] is the only vector field based methodology to control quadcopters that assumes an IGC structure. However, the authors do not consider time-varying fields nor any type of disturbance. These features are present in the current work.

The controller presented here may be viewed as an extension of the fixed-wing UAV controller presented in [64] to quadcopters. While the fixed-wing UAV has a nonholonomic constraint of first order, the quadcopter has an equivalent constraint but of second order. In fact, the thrust force can only act in the direction of the z body axis. To deal with that, the attitude of the robot is controlled to keep its z body axis in the necessary direction. The z axis of the quadrotor body is aligned with a reference acceleration given by the vector field. This is an analogous idea to the alignment of the UAV heading with the velocity field presented in [64].

In our work, we use the vector field to design control laws that make a double integrator model converge to an integral line of the vector field [27] and, consequently, converge to the curve. Finally, controllers are designed to impose the dynamics of the controlled double integrator to a quadcopter model. The consideration of the field derivatives into these controllers characterizes an IGC structure. This last attitude controller is similar to the one presented in [50]. However, we do not consider any small angle assumption, as [50] considered to compute the angular errors.

In fact, we propose a cascade-like control system, [9, 84], which has the vector field in the external loop. Different from other works, such as [85], we prove that this cascade system guarantees convergence in scenarios of time-varying curves and disturbed control inputs. In order to prove asymptotic convergence, the input-to-state stability (ISS) of the systems is used together with Lyapunov Theory. This stability definition has already been studied when guiding vector fields are used to control a fixed-wing UAV [86]. In the scope of a quadcopter, this ISS approach contrasts with some works in the literature, which use different tools to design the controller for the system, such as differential flatness [87, 50], optimal control theory [15], and MPC [54, 59].

Some works that consider the control of quadcopters [43] assume that the control inputs of the model are the velocity of the propellers. In this work, the thrust and the angular rates are considered as the control inputs [14]. This configuration is called the Acro mode of operation. It is used in several drone races and is available in many commercial flight controllers, such as the Ardupilot and PX4, discussed in Section 2.2.2. We assume the existence of such low-level controllers that will assign the signals for the propellers given the commands of thrust and angular body speed. This Acro mode has a much simpler, and accurate, low-level control layer than the Position mode, considered by the SGC approaches. Consequently, the mathematical model of a quadcopter in the Acro mode is more trustworthy than the integrator model for the vehicle in the Position mode. This fact, together with the controller we present, will enable successful maneuvers in higher speeds. In Section 5.2.5, results in a realistic simulator show that the cascade connection approach presented here is able to control a quadcopter at high speeds.

To deal with imperfections of the lower-level controllers of the Acro mode, our theory incorporates a formal disturbance analysis. We prove that norm-bounded disturbances in the control inputs cause limited deviation from the curve. The works in [56] and [64], which focus on a fixed-wing UAV, also present a disturbance analysis when uncertainties in the model are present. The disturbance analysis presented here, based on Lyapunov theory, is similar to the one in [64]. Given norm-bounded disturbances on the control inputs, thrust and angular velocities, we establish an ultimate bound for the position error.

The proposed controller is originally designed for the Euclidean distance vector field strategy presented in Chapter 3. Thus, the focus of our analysis is on the following

of a predefined path, possibly time-varying. Nonetheless, it is important to emphasize that the quadcopter's control laws that will be developed here do not depend on this specific vector field, similar to [87]. In fact, our strategy can be used with any vector field, as long as this other field structure preserves the ISS, which will be explicitly proved only for the Euclidean distance field proposed in the next Chapter. Our controller can also be used to impose the behavior of other recent vector fields [81, 83] to the quadcopter. The advantage of using the Euclidean distance based vector field is that the boundary of the invariant set is defined by a level set of the Euclidean distance to the curve, see Section 2.1.5.

In order to validate our theory, we present numerical simulations where the convergence can be observed, as in [84, 15]. We show indoor experiments with the ESPcopter and a OptiTrack motion capture system to validate the theory and illustrate the controller ability to deal with external disturbances. More importantly, we present outdoor experiments with a physical drone to demonstrate the robustness of the system in a real world scenario with an onboard controller [36] and a GPS based localization system. These experiments differ from the ones presented in [14, 11, 87, 50], which are performed indoors in a structured environment that counts with precise localization systems, such as the ones based on motion capture technologies. In general, outdoor experiments are less likely to be successful given the difficulties associated with localization. The presented results validate the robustness of the vector field controller under noisy pose and velocity estimations.

Chapter 3

Euclidean distance vector field

In this Chapter, the novel artificial vector field methodology for robot navigation is presented. First, in Section 3.1 the problem is formally stated. In Section 3.2 we present the vector field methodology and an analysis regarding the robustness against input noise. A brief discussion regarding the points of the domain that have more than one closest point to the curve is made in Section 3.3. A more detailed analysis is left to Appendix A. Finally, in Section 3.4, we show how the ability to deviate from obstacles can be incorporated to the theory.

Some theory presented in this Chapter, in Sections 3.1 and 3.2 specifically, were published in the IEEE Transaction on Robotics (TRO) journal under the title of "*Constructive time-varying vector fields for robot navigation*" [67]. Results close to the ones presented in Section 3.4, associated to the obstacle deviation approach, were published in the IEEE International Conference on Robotics and Automation (ICRA) under the title of "*Collision-free vector field guidance and MPC for a fixed-wing UAV*" [60].

3.1 Vector field problem setup

Henceforth, we assume that all the vectors are column vectors. All vector quantities are denoted by a bold symbol, as $\mathbf{x}$. The symbol $\|\cdot\|$ denotes the Euclidean norm of a n-dimensional vector. If Q is a matrix (or vector), Q^{T} denotes its transpose.

Consider a time-varying curve $\mathcal{C}(\mathrm{t}) \subset \mathbb{R}^{\mathrm{n}}$, $\mathrm{n} > 1$. Let $\mathbf{r}(\mathrm{s}, \mathrm{t}) : \mathbb{R}\times\mathbb{R}^{+} \to \mathbb{R}^{\mathrm{n}}$ be a parametrization for the curve $\mathcal{C}(\mathrm{t})$, in which s is the arc-length[1] parameter and t is time. Let $\mathbf{x} \in \mathbb{R}^{\mathrm{n}}$ be a point. There are many possibilities for the curve $\mathcal{C}(\mathrm{t})$. It could be, for instance, a finite straight line, or two disjoint circles. In order to clarify which kind of curves we will consider in this work, we will use the following assumption.

[1]The use of the arc-length parameter is only for convenience, since it will simplify our mathematical analysis. The methodology does not require this special representation to be implemented.

Assumption 1. *We assume that the curve* $\mathcal{C}(t)$ *is such that, for all fixed* t*, it is either an unbounded curve homeomorphic to an infinite straight line ("open curve") or bounded and homeomorphic to a circle ("closed curve").*

Remark 1. *Later in Section 3.3.3 we present an extension of the vector field that will weaken Assumption 1 by incorporating curves with self-intersection.*

The objective is to compute a time-varying vector field $\mathbf{\Phi}(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$ such that the trajectories of the system $\dot{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x}, t)$ converge to and follow the target curve $\mathcal{C}(t)$. The computation of $\mathbf{\Phi}$ will be made with basis on the parametric representation of the curve and on the distance to the curve. With this in mind, we then define important functions for our method.

Definition 1. *The parameter of the closest point on the curve* $s^*(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ *is given by*

$$s^*(\mathbf{x}, t) = \arg\min_s \|\mathbf{x} - \mathbf{r}(s, t)\|^2. \tag{3.1}$$

Definition 2. *The* closest point map*, or closest point projection,* $\mathbf{x}^*(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$ *is*

$$\mathbf{x}^*(\mathbf{x}, t) = \mathbf{r}(s^*(\mathbf{x}, t), t). \tag{3.2}$$

Definition 3. *The* distance vector $\mathbf{D}(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$ *and the* scalar distance $D(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^+$ *are obtained as*

$$\mathbf{D}(\mathbf{x}, t) = \mathbf{x} - \mathbf{x}^*(\mathbf{x}, t), \tag{3.3}$$

$$D(\mathbf{x}, t) = \|\mathbf{D}(\mathbf{x}, t)\|. \tag{3.4}$$

Remark 2. *We will omit the dependence on the variable* t *in the functions (3.1) - (3.4) when the curve is not time-varying.*

The value s^* is the parameter equivalent to the point on $\mathcal{C}(t)$ that is the closest to $\mathbf{x}$. The vector $\mathbf{D}(\mathbf{x}, t)$ goes from the closest point on $\mathcal{C}(t)$ to the position $\mathbf{x}$, thus its norm is the scalar distance function $D(\mathbf{x}, t)$. Note that, despite the utilization of a parametric representation of the curve, the functions $\mathbf{D}$ and D are independent of the parametrization used. In fact, function $\mathbf{x}^*(\mathbf{x}, t)$ is completely defined by purely geometric characteristics of $\mathcal{C}(t)$. The only dependence the field $\mathbf{\Phi}(\mathbf{x}, t)$ will have on the parametrization $\mathbf{r}(s, t)$ is on the direction in which the curve will be followed.

It is important to observe that, for some points (like the center of a circle), function s^* is multi-valued: there may be more than one point in the curve that has the smallest distance from $\mathbf{x}$ to the curve. In those points, our proposed vector field would be multiply defined. Regarding this aspect, the following definition is required.

Definition 4. *Let* $\mathcal{U} \subseteq \mathbb{R}^n \times \mathbb{R}^+$ *be the set of points* $(\mathbf{x}, t)$ *such that* $s^*(\mathbf{x}, t)$ *is a singleton*[2].

Therefore, for $(\mathbf{x}, t) \notin \mathcal{U}$ the expression $\mathbf{D}(\mathbf{x}, t)$ is also multi-valued. Henceforth, we will assume that the domain of the functions $\mathbf{\Phi}$, s^*, $\mathbf{x}^*$, and $\mathbf{D}$ is $\mathcal{U}$, and thus they all will be single-valued. The vector field will not be defined for points outside of $\mathcal{U}$. We will soon establish that there are few such points, *i.e.* for any fixed t the set of points $\mathbf{x}$ such that $(\mathbf{x}, t) \notin \mathcal{U}$ has measure zero in $\mathbb{R}^n$. It will be also mathematically shown that, at least for the time-invariant case, these points are unapproachable if we choose the parameters of the vector field accordingly (Appendix A). In the time-varying case or in the presence of disturbances we can simply choose one of the possible vectors. For instance, the one which is closest to the last velocity vector sent to the robot.

We also need other definitions regarding a traversal term, which will be mostly responsible for the curve following.

Definition 5. *The* curve's tangent vector $\mathbf{T}(s, t) : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}^n$ *is given by*

$$\mathbf{T}(s, t) = \frac{\partial \mathbf{r}(s, t)}{\partial s}. \tag{3.5}$$

Definition 6. *The* closest point's tangent $\mathbf{T}^*(\mathbf{x}, t) : \mathcal{U} \to \mathbb{R}^n$ *is defined as*

$$\mathbf{T}^*(\mathbf{x}, t) = \mathbf{T}(s^*(\mathbf{x}, t), t). \tag{3.6}$$

Remark 3. *We will omit the dependence on the variable* t *in* $\mathbf{T}$ *and* $\mathbf{T}^*$ *when the curve is not time-varying.*

Note that, since s is the arc length, it holds that $\|\mathbf{T}(s, t)\| = 1$. In practical cases, if one uses a parametrization that is not the arc-length, the tangent vector must be normalized. In fact, the definition of the normalized tangent vector does not depend on the parametrization either, except for a factor of –1 that may invert the direction in which the curve is traversed. Geometrically, this vector can be seen as the unit vector that spans the tangent space of the set $\mathcal{C}(t)$ at time t. Now, we also present some auxiliary definitions regarding the curve's normal vector and curvature.

Definition 7. *The* normal vector $\mathbf{N}(s, t) : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}^n$ *is given by*

$$\mathbf{N}(s, t) = \frac{\partial^2 \mathbf{r}(s, t)}{\partial s^2} \left\| \frac{\partial^2 \mathbf{r}(s, t)}{\partial s^2} \right\|^{-1}. \tag{3.7}$$

Definition 8. *The* curvature $\kappa(s, t) : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}^+$ *is given by*

$$\kappa(s, t) = \frac{\partial \mathbf{T}}{\partial s}(s, t)^{\mathrm{T}} \mathbf{N}(s, t). \tag{3.8}$$

These definitions come from the Frenet-Serret formulas [8]. In order to use them, a weak assumption is necessary.

[2] Set with only one element.

Assumption 2. *The function* $\mathbf{r}(\mathrm{s},\mathrm{t})$ *is* C^2 *on* s *for all* t. *Furthermore,* $\mathbf{r}(\mathrm{s},\mathrm{t})$ *is also differentiable with respect to* t*, for all fixed* s.

We may also require that the vector field has a constant norm given by $\mathrm{v_r}$, *i.e.*, $\|\mathbf{\Phi}(\mathbf{x},\mathrm{t})\| = \mathrm{v_r}\ \forall\ (\mathbf{x},\mathrm{t}) \in \mathcal{U}$. In order to make this requirement possible, we must assume that the curve $\mathcal{C}(\mathrm{t})$ is slow-varying on time when compared with the robot speed $\mathrm{v_r}$. To understand this necessity, consider a rigid curve that moves sideways with a speed greater than $\mathrm{v_r}$. In such case, if the robot speed is $\mathrm{v_r}$, it will not be able to follow the path. With the purpose to solve this issue, we will need another definition.

Definition 9. *The* null space of $\mathbf{T}^*$ projection operator*,* $\Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t}) : \mathcal{U} \to \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$*, is defined as the matrix*

$$\Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t}) = \mathrm{I}_{\mathrm{n}\times\mathrm{n}} - \mathbf{T}^*(\mathbf{x},\mathrm{t})\mathbf{T}^*(\mathbf{x},\mathrm{t})^{\mathrm{T}}, \tag{3.9}$$

in which $\mathrm{I}_{\mathrm{n}\times\mathrm{n}}$ *is the identity matrix of order* n.

This null space projection matrix is such that $\Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t})\mathbf{h}$ is in the (right) null space of $\mathbf{T}^*(\mathbf{x},\mathrm{t})^{\mathrm{T}}$ for all $\mathbf{h}$. In other words, $\Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t})\mathbf{h}$ removes the component in the direction of $\mathbf{T}^*(\mathbf{x},\mathrm{t})$ of the vector $\mathbf{h}$. With this definition, we can formally assume the following.

Assumption 3. *For any point* $\mathbf{x} \in \mathbb{R}^{\mathrm{n}}$ *at any time* $\mathrm{t} \in \mathbb{R}^{+}$*, such that* $(\mathbf{x},\mathrm{t}) \in \mathcal{U}$*, the partial derivative of the distance vector* $\mathbf{D}(\mathbf{x},\mathrm{t})$*, defined in (3.3), with respect to time is finite. We will impose that when we remove the traversal component from this vector the resulting vector has a norm smaller than* $\mathrm{v_r}$*. Formally*

$$\max_{(\mathbf{x},\mathrm{t})\in\mathcal{U}} \left\| \Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t}) \frac{\partial \mathbf{D}}{\partial \mathrm{t}}(\mathbf{x},\mathrm{t}) \right\| \equiv \mathrm{v_m} < \mathrm{v_r}, \tag{3.10}$$

in which $\mathrm{v_m}$ *is called the maximum "local speed" of* $\mathcal{C}(\mathrm{t})$.

An intuitive interpretation of the term inside the norm operator in equation (3.10) is that $\frac{\partial \mathbf{D}}{\partial \mathrm{t}}$ is the necessary velocity of a point $\mathbf{x}(\mathrm{t}) \in \mathcal{C}(\mathrm{t})$ to continue in the curve at time $\mathrm{t}+\Delta\mathrm{t}$, *i.e.* $\mathbf{x}(\mathrm{t}+\Delta\mathrm{t}) \in \mathcal{C}(\mathrm{t}+\Delta\mathrm{t})$. However, any component of this velocity on the tangent direction of the curve just causes an internal movement, as if $\mathbf{x}(\mathrm{t})$ is traversing the curve. For instance, if the time-varying curve is a circle moving in space, $\frac{\partial \mathbf{D}}{\partial \mathrm{t}}$ makes the point move with the curve from $\mathcal{C}(\mathrm{t})$ to $\mathcal{C}(\mathrm{t}+\Delta\mathrm{t})$, but the tangent component of this velocity only makes the point rotate inside $\mathcal{C}(\mathrm{t}+\Delta\mathrm{t})$. Therefore, we can remove this component to make the assumption less strict.

Note that Assumption 3 is weak, since it is reasonable that a robot with speed $\mathrm{v_r}$ cannot track a curve that moves with a speed greater than $\mathrm{v_r}$. In real scenarios, it is also important to emphasize that the curve $\mathcal{C}(\mathrm{t})$ must be feasible to be traversed by the robot at the desired speed $\mathrm{v_r}$. For instance, curves with high curvatures may not be tracked with high velocities as real robots might be subject to mechanical constraints, such as maximum accelerations and angular velocities.

Essentially, the problem addressed in this chapter is:

Problem 1. *Given a time-varying curve* $\mathcal{C}(\mathrm{t})$ *embedded in an* n*-dimensional Euclidean space, under Assumption 3, defined in the parametric form* $\mathbf{r}(\mathrm{s},\mathrm{t})$*, under Assumption 2, find a vector field* $\boldsymbol{\Phi}(\mathbf{x},\mathrm{t}) : \mathcal{U} \mapsto \mathbb{R}^{\mathrm{n}}$ *such that* $\|\boldsymbol{\Phi}\| = \mathrm{v_r}$*, and the trajectories of the system* $\dot{\mathbf{x}} = \boldsymbol{\Phi}(\mathbf{x},\mathrm{t})$ *converge to and traverse (follow, circulate)* $\mathcal{C}(\mathrm{t})$*.*

3.2 Vector field formulation

In this section, we will state the solution to Problem 1 defined in the previous section, announce its properties and then proceed to establish them formally. Before stating our solution, two definitions are necessary.

Definition 10. *A function* $\mathrm{G} : \mathbb{R}^+ \to [0,1)$ *is said to be of* G*-type if it is Lipschitz continuous, strictly increasing and* $\mathrm{G}(0) = 0$*, thus* $\mathrm{G}(\mathrm{D}) > 0$ *for* $\mathrm{D} > 0$*.*

Some simple examples of G-type functions include $\mathrm{G}(\mathrm{D}) = (2/\pi)\,\mathrm{atan}(\mathrm{k_f D})$, for $\mathrm{k_f} > 0$ and $\mathrm{G}(\mathrm{D}) = \mathrm{D}/\sqrt{\mathrm{D}^2 + \mathrm{k_f}}$ also for $\mathrm{k_f} > 0$.

Definition 11. *Let* G *be a* G*-type function, then* $\mathrm{H}(\mathrm{D}) = \sqrt{1 - \mathrm{G}(\mathrm{D})^2}$*. Henceforth we will also use the definitions*

$$\hat{\mathrm{G}}(\mathbf{x},\mathrm{t}) \equiv \mathrm{G}(\mathrm{D}(\mathbf{x},\mathrm{t})), \qquad \hat{\mathrm{H}}(\mathbf{x},\mathrm{t}) \equiv \mathrm{H}(\mathrm{D}(\mathbf{x},\mathrm{t})). \tag{3.11}$$

Remark 4. *We will omit the dependence on the variable* t *in* $\hat{\mathrm{G}}$ *and* $\hat{\mathrm{H}}$ *when the curve is not time-varying.*

Note that $\mathrm{G},\ \mathrm{H} : \mathbb{R}^+ \to \mathbb{R}^+$, while $\hat{\mathrm{G}},\ \hat{\mathrm{H}} : \mathcal{U} \to \mathbb{R}^+$.

Given these definitions, a solution to Problem 1 is the vector field $\boldsymbol{\Phi}(\mathbf{x},\mathrm{t})$ defined by the following components:

$$\boldsymbol{\Phi}(\mathbf{x},\mathrm{t}) = \underbrace{-\eta(\mathbf{x},\mathrm{t})\hat{\mathrm{G}}(\mathbf{x},\mathrm{t})\frac{\mathbf{D}(\mathbf{x},\mathrm{t})}{\mathrm{D}(\mathbf{x},\mathrm{t})}}_{\text{convergence}} + \underbrace{\eta(\mathbf{x},\mathrm{t})\hat{\mathrm{H}}(\mathbf{x},\mathrm{t})\mathbf{T}^*(\mathbf{x},\mathrm{t})}_{\text{traversal}} \underbrace{-\Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t})\frac{\partial \mathbf{D}}{\partial \mathrm{t}}(\mathbf{x},\mathrm{t})}_{\text{feedforward}}, \tag{3.12}$$

in which $\eta(\mathbf{x},\mathrm{t}) : \mathcal{U} \to \mathbb{R}$ is a positive scalar chosen so that $\boldsymbol{\Phi}$ has a norm equal to $\mathrm{v_r}$. This scalar can be found by obtaining the greatest root of the second-order polynomial for η: $\|\boldsymbol{\Phi}[\eta]\|^2 = \mathrm{v_r^2}$.

Note that, since $\mathbf{D}$ and $\mathbf{T}$ are invariant to the parametrization, *i.e.* they depend only on the curve's geometry, the vector field is also purely geometric. The curve's parametrization function is only a convenient tool to enable the computation of $\boldsymbol{\Phi}$.

Likewise [27], the vector field has three components, each one responsible for a different behavior. Note, however, that the computation of the terms is much more straightforward in comparison to previous works that consider implicit definitions for the target curve, requiring no knowledge of zero level sets.

This vector field has the following features, that will be formally shown in the following subsections:

- Its integral curves converge asymptotically to $\mathcal{C}(t)$ for each initial condition $(\mathbf{x}_0, 0) \in \mathcal{U}$. There are no other equilibrium points, stable or unstable (Proposition 2).
- Under Assumption 3, η can always be chosen so that the time-varying vector field has a constant norm: $\|\mathbf{\Phi}(\mathbf{x}, t)\| = v_r$ (Lemma 8).
- Points outside of $\mathcal{U}$ form a set of measure zero (Proposition 1). Furthermore, for the normalized time-invariant case, one can choose the parameter $\hat{G}$ such that no points outside of $\mathcal{U}$ are approachable (Proposition 6, Appendix A). Nevertheless, even if these points are reached, which can happen in the case of time-varying curves or if there are disturbances on the system, one can simply choose one of the possible values of $\mathbf{\Phi}(\mathbf{x}, t)$ and continue moving along the field (Section 3.3.1). For instance, a natural choice is the closest to the last vector that has been sent to the robot to follow.
- It is robust to norm-bounded additive disturbances in the assumed simple integrator model. In this case, the maximum deviation from the curve is finite and can be computed from the bound of the disturbance (Proposition 3).
- Its convergence behavior is isotropic on space. It means that the weight of the convergent component does not depend on the direction of the vector $\mathbf{D}$, but only on the distance D (also Proposition 3).

3.2.1 Distance function properties

The Euclidean distance function defined in (3.4) has interesting and useful properties. In this section, they will be presented. We begin by showing that the set $\mathcal{U}$ is "small". For this purpose, we begin with a Lemma.

Lemma 1. *Consider a point* $\mathbf{x} \in \mathbb{R}^n$*, let* $\mathbf{x}^*$ *be one of the possible closest points on the curve* $\mathcal{C}$ *to the point* $\mathbf{x}$*. Let* $0 < \alpha < 1$*. Then the closest point to the curve of the point*

$\mathbf{x}_\alpha = (1-\alpha)\mathbf{x}^* + \alpha\mathbf{x}$ *is also* $\mathbf{x}^*$. *Furthermore, this point is unique: there is no other point on the curve that has the same distance to* $\mathbf{x}_\alpha$.

Proof. Suppose there is another point on the curve, $\mathbf{x}^*_{\text{other}} \neq \mathbf{x}^*$ that is closer to $\mathbf{x}_\alpha$ (that is, $\mathbf{x}^*$ is not the closest point at the curve to $\mathbf{x}_\alpha$) *or* has the same distance to $\mathbf{x}_\alpha$ as $\mathbf{x}^*$ (that is, $\mathbf{x}^*$ is not unique). In the following, we show that this induces a contradiction and therefore it is false.

Suppose it is true. Let $\mathbf{u} = \mathbf{x}^* - \mathbf{x}_\alpha$ and $\mathbf{v} = \mathbf{x}^*_{\text{other}} - \mathbf{x}^*$. Then, $\|\mathbf{u}+\mathbf{v}\| \leq \|\mathbf{u}\|$, since $\mathbf{x}^*_{\text{other}}$ is closer, or at the same distance, to $\mathbf{x}_\alpha$ than $\mathbf{x}^*$ is. This implies after squaring that (i) $2\mathbf{u}^{\mathrm{T}}\mathbf{v} + \|\mathbf{v}\|^2 \leq 0$, and also that (ii) $\mathbf{u}^{\mathrm{T}}\mathbf{v} < 0$, because $\mathbf{v} \neq \mathbf{0}$ since $\mathbf{x}^*_{\text{other}} \neq \mathbf{x}^*$.

Since $0 < \alpha < 1$, we have from (i) and (ii) that $\|\mathbf{v}\|^2 + 2\mathbf{u}^{\mathrm{T}}\mathbf{v}/\alpha < 0$. Summing $\|\mathbf{u}\|^2/\alpha^2$ in both sides it is possible to conclude that (iii) $\|\mathbf{u}/\alpha + \mathbf{v}\| < \|\mathbf{u}\|/\alpha$.

Finally, note that $(\mathbf{x}^* - \mathbf{x}) = \mathbf{u}/\alpha$. Furthermore, that $(\mathbf{x}^*_{\text{other}} - \mathbf{x}) = \mathbf{u}/\alpha + \mathbf{v}$. Therefore, (iii) implies that $\|\mathbf{x}^*_{\text{other}} - \mathbf{x}\| < \|\mathbf{x}^* - \mathbf{x}\|$, which is contradictory with the fact that $\mathbf{x}^*$ is the closest point at the curve to $\mathbf{x}$. □

Now, it is possible to prove that the set is indeed "small".

Proposition 1. *For any curve in* $\mathbb{R}^{\mathrm{n}}$*, the set of points that have more than one closest point on the curve has measure zero in* $\mathbb{R}^{\mathrm{n}}$.

Proof. The proof comes from contradiction. Suppose there is a set $\mathcal{V}$ of points not in $\mathcal{U}$ that has a non-zero measure in $\mathbb{R}^{\mathrm{n}}$. Therefore, it is possible to consider a ball $\mathcal{B} \subseteq \mathcal{V}$ with nonzero volume. Let $\mathbf{x}$ be the center of such a ball, which is also a point with more than one closest point on the curve. Let $\mathbf{x}^*$ be one of such points.

Choose an $0 < \alpha < 1$ such that the point $\mathbf{x}_\alpha = (1-\alpha)\mathbf{x}^* + \alpha\mathbf{x}$ lies in $\mathcal{B}$. This is always possible because it has a non-zero measure. Since it lies in $\mathcal{B}$, $\mathbf{x}_\alpha$ must also have more than one point closest to the curve. However, Lemma 1 forbids it. This contradiction concludes the proof. □

Even if this set is small, there is the possibility that the trajectory indeed passes through it, thus, in such state, the field $\mathbf{\Phi}(\mathbf{x}, \mathrm{t})$ will be multiply defined if we consider the domain $\mathbb{R}^{\mathrm{n}} \times \mathbb{R}^+$. Further in this work, we establish two facts: (i) convergence is still achieved if one of the possible values of the field is chosen (Section 3.3.1); and (ii) there are sufficient conditions, for the time-invariant case, that guarantee that the trajectory never reaches these singular points (Appendix A). Also, in Section 3.3.3, we propose a novel strategy to deal with these issues by saving curve's parameter as a state variable.

Now we derive other properties of the distance function, and its associated functions, which will be important to prove the convergence of the vector field.

Lemma 2. *If* $\mathrm{s}^*(\mathbf{x}, \mathrm{t})$ *is defined by (3.1), the following identity holds*

$$\mathbf{T}^*(\mathbf{x}, \mathrm{t})^{\mathrm{T}}\mathbf{D}(\mathbf{x}, \mathrm{t}) = \mathbf{T}(\mathrm{s}^*(\mathbf{x}, \mathrm{t}), \mathrm{t})^{\mathrm{T}}\mathbf{D}(\mathbf{x}, \mathrm{t}) = 0. \tag{3.13}$$

Proof. First, note that, given Assumption 1, the curve $\mathcal{C}(\mathrm{t})$ has no borders. Thus, we can apply the first-order optimality condition to the optimization problem in (3.1) to conclude that

$$\frac{\partial \mathbf{r}}{\partial \mathrm{s}}(\mathrm{s}^*(\mathbf{x},\mathrm{t}),\mathrm{t})^{\mathrm{T}}\Big(\mathbf{x}-\mathbf{r}\big(\mathrm{s}^*(\mathbf{x},\mathrm{t}),\mathrm{t}\big)\Big)=0. \tag{3.14}$$

The first factor of the dot product in (3.14) is the tangent vector $\mathbf{T}$ defined in (3.5). The second factor in (3.14) is the distance vector $\mathbf{D}(\mathbf{x},\mathrm{t})$. Using these facts into (3.14), (3.13) is established. □

Lemma 3. *It holds that*

$$\frac{\partial \mathrm{s}^*}{\partial \mathbf{x}}(\mathbf{x},\mathrm{t})=\frac{\mathbf{T}^*(\mathbf{x},\mathrm{t})}{\Omega(\mathbf{x},\mathrm{t})}, \tag{3.15}$$

in which $\Omega(\mathbf{x},\mathrm{t}):\mathcal{U}\to\mathbb{R}$ *is given by*

$$\Omega(\mathbf{x},\mathrm{t})=1-\kappa(\mathrm{s}^*(\mathbf{x},\mathrm{t}),\mathrm{t})\mathbf{N}(\mathrm{s}^*(\mathbf{x},\mathrm{t}),\mathrm{t})^{\mathrm{T}}\mathbf{D}(\mathbf{x},\mathrm{t}). \tag{3.16}$$

Proof. Take the optimality condition in equation (3.14), differentiate both sides in $\mathbf{x}$ and solve for $\frac{\partial \mathrm{s}^*}{\partial \mathbf{x}}$. In that case, one needs to remember that since s is the arc-length, $\|\frac{\partial \mathbf{r}}{\partial \mathrm{s}}\|=1$ and $\frac{\partial^2 \mathbf{r}}{\partial \mathrm{s}^2}=\kappa(\mathrm{s},\mathrm{t})\mathbf{N}(\mathrm{s},\mathrm{t})$ due to Frenet-Serret formulas. □

Lemma 4. *The vector* $\mathbf{D}(\mathbf{x},\mathrm{t})$ *is in the (right) null space of the matrix* $\frac{\partial \mathbf{x}^*}{\partial \mathbf{x}}(\mathbf{x},\mathrm{t})$. *Mathematically*

$$\frac{\partial \mathbf{x}^*}{\partial \mathbf{x}}(\mathbf{x},\mathrm{t})^{\mathrm{T}}\mathbf{D}(\mathbf{x},\mathrm{t})=\mathbf{0}. \tag{3.17}$$

Proof. Consider the fact that $\mathbf{x}^*(\mathbf{x},\mathrm{t})=\mathbf{r}(\mathbf{s}^*(\mathbf{x},\mathrm{t}),\mathrm{t})$. Differentiate both sides in $\mathbf{x}$, use the chain-rule, the fact that $\frac{\partial \mathbf{r}}{\partial \mathrm{s}}=\mathbf{T}(\mathrm{s},\mathrm{t})$ and Lemma 3 for $\frac{\partial \mathrm{s}^*}{\partial \mathbf{x}}$ to conclude that

$$\frac{\partial \mathbf{x}^*}{\partial \mathbf{x}}(\mathbf{x},\mathrm{t})=\frac{\mathbf{T}^*(\mathbf{x},\mathrm{t})\mathbf{T}^*(\mathbf{x},\mathrm{t})^{\mathrm{T}}}{\Omega(\mathbf{x},\mathrm{t})}. \tag{3.18}$$

Multiplying both sides by $\mathbf{D}(\mathbf{x},\mathrm{t})$ and using Lemma 2, we obtain (3.17). □

Lemma 5. *For each* $(\mathbf{x},\mathrm{t})\in\mathcal{U}$ *with* $\mathbf{x}\notin\mathcal{C}(\mathrm{t})$, $\nabla\mathrm{D}(\mathbf{x},\mathrm{t})$ *is a unit norm vector and is given by* $\nabla\mathrm{D}(\mathbf{x},\mathrm{t})=\mathbf{D}(\mathbf{x},\mathrm{t})/\mathrm{D}(\mathbf{x},\mathrm{t})$.

Proof. Consider $\mathbf{D}=\mathbf{x}-\mathbf{x}^*(\mathbf{x},\mathrm{t})$ and $\mathrm{D}=\sqrt{\mathbf{D}^{\mathrm{T}}\mathbf{D}}$. It is possible to write

$$\nabla\mathrm{D}=\left(\mathrm{I}_{\mathrm{n}\times\mathrm{n}}-\frac{\partial \mathbf{x}^*}{\partial \mathbf{x}}(\mathbf{x},\mathrm{t})\right)^{\mathrm{T}}\frac{\mathbf{D}}{\mathrm{D}}=\frac{\mathbf{D}}{\mathrm{D}}, \tag{3.19}$$

in which $\mathrm{I}_{\mathrm{n}\times\mathrm{n}}$ is the identity matrix of order n and the second equivalence comes from Lemma 4. Since $(\mathbf{x},\mathrm{t})\in\mathcal{U}$, the point $\mathbf{x}^*$ and vector $\mathbf{D}$ are unique. Furthermore, since $\mathbf{x}\notin\mathcal{C}(\mathrm{t})\implies\mathrm{D}\neq 0$ the vector $\nabla\mathrm{D}$ is well defined. □

At last, we will show another Lemma that will be useful to prove convergence for the proposed field:

Lemma 6. *Let* $\mathrm{P} : \mathbb{R}^+ \to \mathbb{R}^+$ *be defined by* $\mathrm{P} = \frac{1}{2}\mathrm{D}^2$. *For all* $(\mathbf{x}, \mathrm{t}) \in \mathcal{U}$ *the following identity holds*

$$\frac{\partial \mathrm{P}}{\partial \mathrm{t}} = \nabla \mathrm{P}^{\mathrm{T}} \Pi_{\mathbf{T}} \frac{\partial \mathbf{D}}{\partial \mathrm{t}}. \tag{3.20}$$

Proof. First, note that with the chain rule we obtain $\nabla \mathrm{P} = \mathrm{D}\nabla \mathrm{D}$. Using Lemma 5 and noting that $\nabla \mathrm{P}$ is also defined when $\mathrm{D} = 0$, we conclude that:

$$\nabla \mathrm{P}(\mathbf{x}, \mathrm{t}) = \mathbf{D}(\mathbf{x}, \mathrm{t}). \tag{3.21}$$

Now, note that given the definition of $\Pi_{\mathbf{T}}$ in (3.9) and that $\nabla \mathrm{P} \parallel \mathbf{D} \perp \mathbf{T}$, we have that (3.20) reduces to:

$$\frac{\partial \mathrm{P}}{\partial \mathrm{t}} = \nabla \mathrm{P}^{\mathrm{T}} \frac{\partial \mathbf{D}}{\partial \mathrm{t}}. \tag{3.22}$$

Now, taking the partial derivative of the identity $\mathrm{P} = \frac{1}{2}\mathbf{D}^{\mathrm{T}}\mathbf{D}$ with respect to time

$$\frac{\partial \mathrm{P}}{\partial \mathrm{t}} = \frac{1}{2}\frac{\partial \mathbf{D}}{\partial \mathrm{t}}^{\mathrm{T}} \mathbf{D} + \frac{1}{2}\frac{\partial \mathbf{D}}{\partial \mathrm{t}}^{\mathrm{T}} \mathbf{D} = \mathbf{D}^{\mathrm{T}} \frac{\partial \mathbf{D}}{\partial \mathrm{t}}. \tag{3.23}$$

Using (3.21) in (3.23), equation (3.22), which is equivalent to (3.20), is established. □

3.2.2 Normalized vector field methodology

Before presenting the definition of the field $\mathbf{\Phi}$, we will discuss its three individual components: the *convergence* term, the *traversal* term and the *feedforward* term.

First, a field that converges to the curve $\mathcal{C}(\mathrm{t})$ is sought. Therefore, the *convergent component* $\mathbf{\Phi}_{\mathrm{G}}(\mathbf{x}, \mathrm{t}) : \mathcal{U} \to \mathbb{R}^{\mathrm{n}}$ is defined as

$$\mathbf{\Phi}_{\mathrm{G}}(\mathbf{x}, \mathrm{t}) = -\hat{\mathrm{G}}(\mathbf{x}, \mathrm{t}) \frac{\nabla \mathrm{P}(\mathbf{x}, \mathrm{t})}{\|\nabla \mathrm{P}(\mathbf{x}, \mathrm{t})\|} = -\hat{\mathrm{G}}(\mathbf{x}, \mathrm{t}) \nabla \mathrm{D}(\mathbf{x}, \mathrm{t}), \tag{3.24}$$

in which $\hat{\mathrm{G}}(\mathbf{x}, \mathrm{t}) \equiv \mathrm{G}(\mathrm{D}(\mathbf{x}, \mathrm{t}))$, as in Definition 11, and the second equality is obtained from equations in (3.19) and (3.21). When $\mathrm{D} = 0$ the vector $\nabla \mathrm{D}$ is not defined. However, $\lim_{\mathbf{x} \to \mathcal{C}(\mathrm{t})} \hat{\mathrm{G}}(\mathbf{x}, \mathrm{t}) \nabla \mathrm{D}(\mathbf{x}, \mathrm{t}) = \mathbf{0}$, since $\mathrm{G}(0) = 0$ and $\lim_{\mathbf{x} \to \mathcal{C}(\mathrm{t})} \|\nabla \mathrm{D}(\mathbf{x}, \mathrm{t})\| = 1$. The intuition behind this term is that it always points to the curve, more precisely to the closest point at the curve. Therefore, it guides the point $\mathbf{x}$ towards $\mathcal{C}(\mathrm{t})$.

Now, a field that provides the tracking of the curve is sought. Therefore, the *traversal component* $\mathbf{\Phi}_{\mathrm{H}}(\mathbf{x}, \mathrm{t}) : \mathcal{U} \to \mathbb{R}^{\mathrm{n}}$ is defined as

$$\mathbf{\Phi}_{\mathrm{H}}(\mathbf{x}, \mathrm{t}) = \hat{\mathrm{H}}(\mathbf{x}, \mathrm{t}) \mathbf{T}^*(\mathbf{x}, \mathrm{t}), \tag{3.25}$$

in which $\mathbf{T}^*(\mathbf{x}, \mathrm{t})$ is the tangent vector computed according to (3.6) and $\hat{\mathrm{H}}$ is defined as in Definition 11. The intuition behind this component is that, when we are in $\mathcal{C}(\mathrm{t})$ at a

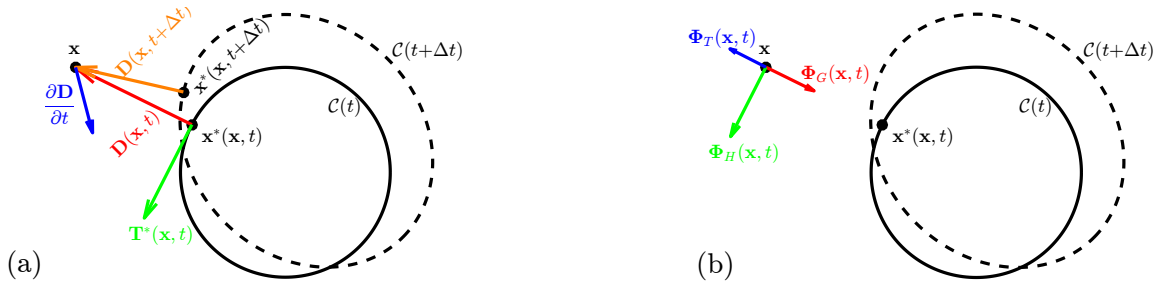

Figure 3.1: Example showing the components of $\mathbf{\Phi}$. In (a) we show some elements necessary to compute the components. In (b) we show the three components.

point $\mathbf{x}$, this component is always tangent to the curve. Therefore, it induces $\mathbf{x}$ to follow the curve.

The *time feedforward component* $\mathbf{\Phi}_{\mathrm{T}}(\mathbf{x}, \mathrm{t}) : \mathcal{U} \to \mathbb{R}^{\mathrm{n}}$, important in time-dependent scenarios, is defined as

$$\mathbf{\Phi}_{\mathrm{T}}(\mathbf{x}, \mathrm{t}) = -\Pi_{\mathbf{T}}(\mathbf{x}, \mathrm{t}) \frac{\partial \mathbf{D}}{\partial \mathrm{t}}(\mathbf{x}, \mathrm{t}). \tag{3.26}$$

It is not as easy to have an intuition for this term, but we will see mathematically that it is necessary to ensure convergence. Furthermore, it is clearly related to the time-varying properties of the curve, since for static curves it vanishes.

Figure 3.1 illustrates these three components. Except for the norm requirement, $\|\mathbf{\Phi}\| = \mathrm{v_r}$, we could sum up all these components and obtain a field that solves Problem 1. If it is desirable to have a constant speed, $\mathrm{v_r}$, for time-invariant vector fields, it is usual to normalize the vector field such that a vector $\mathbf{\Phi}$ has the same norm for all $\mathbf{x}$. However, time-varying fields can not be as easily normalized without compromising convergence. This happens because if we multiply the component $\mathbf{\Phi}_{\mathrm{T}}$ by a scaling factor, the vector field will not properly compensate for the movement of the curve, thus, the convergence will be affected. To solve this issue, we propose a different normalization method. It will scale only part of the vector field to obtain a constant norm of the composite vector. The component associated with the changes on time of the curve will remain unchanged.

Let us divide the three components into two groups. The first one, $\mathbf{\Phi}_{\mathrm{S}}$, composed of $\mathbf{\Phi}_{\mathrm{G}}$ and $\mathbf{\Phi}_{\mathrm{H}}$, is not dependent on partial derivatives with respect to time, thus, it is the *scalable component*. The second one, composed only of $\mathbf{\Phi}_{\mathrm{T}}$, is dependent on the partial derivatives with respect to time, thus it is called the *not scalable component*.

We propose the following field:

$$\mathbf{\Phi} = \eta \mathbf{\Phi}_{\mathrm{S}} + \mathbf{\Phi}_{\mathrm{T}}, \tag{3.27}$$

in which η is a scaling factor that will be determined soon.

A Lemma is necessary:

Lemma 7. *For all* $(\mathbf{x}, \mathrm{t}) \in \mathcal{U}$*, the vector* $\mathbf{\Phi}_{\mathrm{S}} = \mathbf{\Phi}_{\mathrm{G}} + \mathbf{\Phi}_{\mathrm{H}}$ *has a unit norm.*

Proof. From Lemma 2, $\boldsymbol{\Phi}_{\mathrm{G}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{H}} = -\hat{\mathrm{G}}\hat{\mathrm{H}}\mathbf{D}^{\mathrm{T}}\mathbf{T}^{*}/\mathrm{D} = 0$. Thus $\|\boldsymbol{\Phi}_{\mathrm{G}}+\boldsymbol{\Phi}_{\mathrm{H}}\|^2 = \|\boldsymbol{\Phi}_{\mathrm{G}}\|^2+\|\boldsymbol{\Phi}_{\mathrm{H}}\|^2$. Now, since $\|\mathbf{T}^{*}\| = 1$ and $\|\nabla\mathrm{D}\| = 1$, we have that $\|\boldsymbol{\Phi}_{\mathrm{G}} + \boldsymbol{\Phi}_{\mathrm{H}}\|^2 = \hat{\mathrm{G}}^2 + \hat{\mathrm{H}}^2$. From Definition 11, we have that $\hat{\mathrm{G}}^2 + \hat{\mathrm{H}}^2 = 1$. □

Now, we can see, using the fact $\|\boldsymbol{\Phi}_{\mathrm{S}}\| = 1$ (Lemma 7), that using (3.27) the condition $\|\boldsymbol{\Phi}\|^2 = \mathrm{v_r^2}$ reduces to

$$\eta^2 + \left(2\boldsymbol{\Phi}_{\mathrm{S}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{T}}\right)\eta + \left(\|\boldsymbol{\Phi}_{\mathrm{T}}\|^2 - \mathrm{v_r^2}\right) = 0. \tag{3.28}$$

As will be clear later, in order to ensure $\eta > 0$, the following root of the second-order polynomial in (3.28) should be considered

$$\eta = -\boldsymbol{\Phi}_{\mathrm{S}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{T}} + \sqrt{\left(\boldsymbol{\Phi}_{\mathrm{S}}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{T}}\right)^2 + \mathrm{v_r^2} - \|\boldsymbol{\Phi}_{\mathrm{T}}\|^2}. \tag{3.29}$$

Note that $\|\boldsymbol{\Phi}_{\mathrm{T}}\| < \mathrm{v_r}$, due to Assumption 3, ensures that η is real.

The normalized vector field $\boldsymbol{\Phi}$ can then be written as

$$\boldsymbol{\Phi} = -\eta\mathrm{G}\nabla\mathrm{D} + \eta\mathrm{H}\mathbf{T}^{*} - \Pi_{\mathbf{T}}\frac{\partial\mathbf{D}}{\partial\mathrm{t}}. \tag{3.30}$$

Asymptotic convergence of the integral curves to the curve $\mathcal{C}(\mathrm{t})$ for this normalized field will be established, but first, it is necessary to show two lemmas.

Lemma 8. *Let*

$$\mathrm{v_m} = \max_{(\mathbf{x},\mathrm{t})\in\mathcal{U}} \left\| \Pi_{\mathbf{T}}(\mathbf{x},\mathrm{t})\frac{\partial\mathbf{D}}{\partial\mathrm{t}}(\mathbf{x},\mathrm{t}) \right\|. \tag{3.31}$$

Given Assumption 3, the value of η in (3.29) is such that

$$0 < \mathrm{v_r} - \mathrm{v_m} \leq \eta \leq \mathrm{v_r} + \mathrm{v_m}. \tag{3.32}$$

Proof. From the fact that $\|\boldsymbol{\Phi}_{\mathrm{T}}\| \leq \mathrm{v_r}$, it is clear that $\eta \geq 0$. In addition, from $\|\eta\boldsymbol{\Phi}_{\mathrm{S}} + \boldsymbol{\Phi}_{\mathrm{T}}\| = \mathrm{v_r}$, $\|\boldsymbol{\Phi}_{\mathrm{S}}\| = 1$ and $\|\boldsymbol{\Phi}_{\mathrm{T}}\| \leq \mathrm{v_m}$ (given Assumption 3 and equation (3.26)), it can be inferred, by the triangle inequality $\|\mathbf{a}+\mathbf{b}\| \leq \|\mathbf{a}\| + \|\mathbf{b}\|$, that

$$\mathrm{v_r} = \|\eta\boldsymbol{\Phi}_{\mathrm{S}} + \boldsymbol{\Phi}_{\mathrm{T}}\| \leq \eta\|\boldsymbol{\Phi}_{\mathrm{S}}\| + \|\boldsymbol{\Phi}_{\mathrm{T}}\| \leq \eta + \mathrm{v_m}, \tag{3.33}$$

from which $\eta \geq \mathrm{v_r} - \mathrm{v_m}$ is obtained. Furthermore, using the same information and from the reverse triangle inequality, $\|\mathbf{a}+\mathbf{b}\| \geq \|\mathbf{a}\| - \|\mathbf{b}\|$, we have

$$\mathrm{v_r} = \|\eta\boldsymbol{\Phi}_{\mathrm{S}} + \boldsymbol{\Phi}_{\mathrm{T}}\| \geq \eta\|\boldsymbol{\Phi}_{\mathrm{S}}\| - \|\boldsymbol{\Phi}_{\mathrm{T}}\| \geq \eta - \mathrm{v_m}, \tag{3.34}$$

from which $\eta \leq \mathrm{v_r} + \mathrm{v_m}$ is obtained. □

Lemma 9. *Let $\mathrm{z} : \mathbb{R} \mapsto \mathbb{R}$, a constant $\gamma > 0$ and a G-type function G, such that the differential equation $\dot{\mathrm{z}}(\mathrm{t}) = -\mathrm{G}(\mathrm{z}(\mathrm{t}))$ holds for any initial condition $\mathrm{z}(0) = \mathrm{z}_0 \geq 0$. Then,* $\lim_{\mathrm{t}\to\infty} \mathrm{z}(\mathrm{t}) = 0$.

Proof. Use the Lyapunov function $V(z) = \frac{z^2}{2}$. We then have that $\dot{V} = -\gamma z G(z)$, that, due to the properties of G in Definition 10, is nonpositive and 0 if and only if $z = 0$. And the proof is complete. □

And then, an assumption is necessary.

Assumption 4. *Given an initial condition* $(\mathbf{x}(t_0), t_0) \in \mathcal{U}$*, the trajectories of the system* $\dot{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x}, \mathbf{t})$ *are such that* $(\mathbf{x}(t), t) \in \mathcal{U}\ \forall\ t > t_0$.

Assumption 4 is only necessary to formalize our next proposition by excluding the points $\mathbf{x}$ such that $\mathbf{x}^*(\mathbf{x}, t)$ is not uniquely defined. Later, in Section 3.3, we derive sufficient conditions to state that Assumption 4 holds for the time-invariant case. We are now able to state the main result of this subsection.

Proposition 2. *Given Assumption 4, the trajectories of the system* $\dot{\mathbf{x}}(t) = \mathbf{\Phi}(\mathbf{x}, t)$ *converge to* $\mathcal{C}(t)$.

Proof. Consider the Lyapunov candidate function $P = \frac{1}{2}D^2$. Given $\dot{\mathbf{x}} = \mathbf{\Phi}$, the time derivative of the Lyapunov candidate function P can be written as $\dot{P} = \nabla P^T \mathbf{\Phi} + \partial P/\partial t$. Thus, using (3.30), we have

$$\dot{P} = -\eta G \nabla P^T \nabla D + \eta H \nabla P^T \mathbf{T}^* - \nabla P^T \Pi_{\mathbf{T}} \frac{\partial \mathbf{D}}{\partial t} + \frac{\partial P}{\partial t}. \tag{3.35}$$

From equation (3.21) and Lemma 2, it holds that $\mathbf{D}^T \mathbf{T}^* = \nabla P^T \mathbf{T}^* = 0$. Thus, the second term in (3.35) is null. From Lemma 6 it is clear that the last two terms in (3.35) cancel each other. Noting that $\nabla P^T \nabla D = D \nabla D^T \nabla D = D$, (3.35) becomes

$$\dot{P} = -\eta G D. \tag{3.36}$$

Now, since $P = \frac{1}{2}D^2$, $\dot{P} = D\dot{D}$. Furthermore, using Lemma 8 and denoting $\eta > \gamma = v_r - v_m > 0$, we have that

$$\dot{D} \leq -\gamma G(D). \tag{3.37}$$

Now, we use the *strong comparison lemma* for ordinary differential equations (see [49]). It states that if we have two functions $z(t)$ and $D(t)$ such that $\dot{z} = -\gamma G(z)$ and $\dot{D} \leq -\gamma G(D)$, $D(0) < z(0)$ and G being Lipschitz continuous, then $D(t) < z(t)$ for all (finite) t. Consider any $z(0) > D(0)$, applying Lemma 9, we conclude that $z(t) \to 0$, and since $D(t) \geq 0$ and $D(t) < z(t)$, this forces $D(t) \to 0$ as well. □

Note that, in equation (3.35), the last two terms would not cancel each other if we had applied the standard normalization in the time varying-field. This happens because the term involving ∇P would be scaled. Consequently, we would not be able to ensure $\dot{P} < 0$. Note also that, with Proposition 2, we have that, as $t \to \infty$, the convergence term of (3.30) vanishes as the robot converges to the curve. Since $\eta > 0$, from Lemma 8, and

the feedforward component has no projection onto the tangential direction, the traversal term guarantees that the curve is being traversed.

In conclusion of this section, equation (3.30) represents the solution to Problem 1.

3.2.3 Robustness analysis

Consider a robot model given by $\dot{\mathbf{x}} = \mathbf{u} + \delta_v$, in which $\mathbf{u} \in \mathbb{R}^n$ is the input signal and $\delta_v \in \mathbb{R}^n$ is a bounded disturbance with limited norm, *i.e.* $\|\delta_v\| < \Delta_v \ \forall \ t \geq 0$. In this subsection, we present some robustness results based on ultimate boundedness arguments. In order to establish an ultimate bound for the function D in this perturbed system, consider the following Assumption.

Assumption 5. *The bound* Δ_v *is such that* $\Delta_v < v_r - v_m$.

Now, in the following proposition, we establish the ultimate bound result.

Proposition 3. *Given Assumptions 3, 4 and 5, the trajectories of the system* $\dot{\mathbf{x}}(t) = \mathbf{u} + \delta_v$, *with* $\|\delta_v\| < \Delta_v$, *under the control law* $\mathbf{u} = \boldsymbol{\Phi}(\mathbf{x}, t)$, *converge to the following positive invariant set*

$$\mathcal{I}(t) = \left\{ \mathbf{x} \in \mathbb{R}^n \mid D(\mathbf{x}, t) < G^{-1}\left(\frac{\Delta_v}{v_r - v_m}\right) \right\}. \tag{3.38}$$

Proof. First, note that Assumption 4 ensures $\boldsymbol{\Phi}(\mathbf{x}, t)$ is well defined. Assuming the perturbed system, the time derivative of the Lyapunov function P becomes

$$\dot{P} = \nabla P^T \dot{\mathbf{x}} + \frac{\partial P}{\partial t} = \nabla P^T \boldsymbol{\Phi} + \nabla P^T \delta_v + \frac{\partial P}{\partial t}. \tag{3.39}$$

Following the steps of Proposition 2, we have that $\dot{P} = -\eta G D + \nabla P^T \delta_v$. Let $M = \max \|\delta_v\|$, then, using the fact $\dot{P} = D\dot{D}$, we have

$$\dot{D} \leq -\eta G + M < -\eta G + \Delta_v, \tag{3.40}$$

in which the last inequality is true given that $M < \Delta_v$.

Remember that the function $G \equiv G(D)$ is continuous and strictly increasing, thus, invertible and $G(a) > b$ if and only if $a > G^{-1}(b)$. Thus, given (3.38), for every $\mathbf{x} \notin \mathcal{I}(t)$ we have

$$G \geq \frac{\Delta_v}{v_r - v_m} \geq \frac{\Delta_v}{\eta}, \qquad \forall \ \mathbf{x}(t) \notin \mathcal{I}(t), \tag{3.41}$$

in which the second inequality comes from Lemma 8. Using (3.41) in (3.40), we conclude

$$\dot{D} \leq -\Delta_v + M < 0, \qquad \forall \ \mathbf{x}(t) \notin \mathcal{I}(t). \tag{3.42}$$

Given the result in (3.42), an initial instant t_0 and an initial distance $D_0 = D(t_0)$, we can compute a conservative upper bound time $T(D_0, t_0)$ that ensures $\mathbf{x} \in \mathcal{I}(t)$ for $t > T(D_0, t_0)$. It is defined by:

$$T(D_0, t_0) = t_0 + \frac{D_0}{\Delta_v - \max \|\delta_v\|}. \tag{3.43}$$

Thus:

$$t > T(D_0, t_0) \implies \mathbf{x} \in \mathcal{I}(t). \tag{3.44}$$

Result in (3.44) is the statement of the ultimate boundedness of the perturbed system. □

Note that, according to Proposition 3, the volume of the set $\mathcal{I}$ increases as Δ_v increases, as expected. It is also interesting to observe that the faster is the movement of the curve, represented by the value of v_m, the greater is the volume of $\mathcal{I}$. If $\Delta_v = 0$, it follows that $\mathcal{I}(t)$ collapses to the curve $\mathcal{C}(t)$.

The result in Proposition 3 clarifies one more geometric interpretation of our result. In the presence of uncertainties, the distance that defines the system's ultimate bound is dependent on the function $G(D)$ and on the maximum velocity error Δ_v. The faster the function G approaches 1 as D increases, the smaller is the set $\mathcal{I}$. It also shows the isotropic property of the field, meaning that deviations from the curve are equally compensated by the convergent component in all directions perpendicular to the tangent vector.

In conclusion, in the presence of limited disturbances, the state $\mathbf{x}$ converges to the set $\mathcal{I}(t)$, which corresponds to a region around the curve $\mathcal{C}(t)$. Given the isotropic property of the field, this region is a circular tube around the curve and its cross-section radius is given by the bound in (3.38). If the system $\dot{\mathbf{x}} = \boldsymbol{\Phi}(\mathbf{x}, t)$, is assumed as a nominal system and δ_v is assumed as an input of that system, Proposition 3 implies that $\dot{\mathbf{x}} = \boldsymbol{\Phi}(\mathbf{x}, t) + \delta_v$ is input-to-state stable[3], see [37]. In other words, a norm-bounded perturbation on the system causes a norm-bounded perturbation on the ultimate value of D. Now consider a perturbation $\delta_v(t)$ that vanishes over time. Since the perturbed system is ISS, we have that $\lim_{t\to\infty} \delta_v(t) = 0 \implies \mathbf{x}(t) \to \mathcal{C}(t)$ as $t \to \infty$, see Chapter 4 of [37]. In the next Chapter, this property is used to show that the vector field $\boldsymbol{\Phi}(\mathbf{x}, t)$ can be used to control a quadcopter.

3.3 Singular points considerations

By using Assumption 4, we avoid a definition problem of $\boldsymbol{\Phi}(\mathbf{x}, t)$ when the point $\mathbf{x}^*$, the closest on the curve, is not uniquely defined. In this Section, we first provide some

[3] For $\delta_v < \Delta_v < v_r - v_m$, as in Assumption 5.

discussion on how these *singular points* are not a problem in practice. Then we show how a constraint on the system's initial condition can ensure the absence of singularities in the robot's trajectory. Finally, we present a novel strategy, inspired in [83], to tackle the issue of these singularity points by using a memory variable.

3.3.1 Repulsiveness of singular points

It is important to emphasize that $\dot{\mathrm{D}}$, in equation (3.37), is negative definite and has a bound that depends only on D. Thus, even if the trajectory $\mathbf{x}(\mathrm{t})$ passes through a point outside $\mathcal{U}$ (those that form a set of measure zero), we can simply choose one vector $\mathbf{\Phi}$ computed according to one of the possible minimizers $\mathbf{x}^*$. If we adopt this strategy, independently of the vector $\mathbf{\Phi}$ that we choose, the distance D will keep decreasing. Thus, there is no way the trajectory can get "stuck" on any region other than the curve $\mathcal{C}(\mathrm{t})$, *i.e.* outside the target curve there are no equilibrium points.

Assuming the incorporation of any criteria that chooses among one of the possible values of $\mathrm{s}^*(\mathbf{x}, \mathrm{t})$, when it is not unique, the vector field $\mathbf{\Phi}(\mathbf{x}, \mathrm{t})$ can be defined in $\mathbb{R}^{\mathrm{n}} \times \mathbb{R}^{+}$, instead of only in $\mathcal{U}$. However, it is important to emphasize that, depending on the choice we make for one of the possible values of s^*, the vector field $\mathbf{\Phi}$ could be switching, although this is not observed in practice. In fact, this vector field will be discontinuous in the points outside $\mathcal{U}$. The importance of this discussion is to show that the trajectory will not be stuck outside the set $\mathcal{U}$.

In Appendix A we derive, for the time-invariant case, an analysis of the *approachability* of the points outside $\mathcal{U}$. There, we show a sufficient condition to ensure that, if $(\mathbf{x}, 0) \in \mathcal{U}$, no point outside $\mathcal{U}$ will be reached. In other words, for any initial condition in $\mathcal{U}$, Assumption 4 holds. From the result obtained in equation (A.15), a sufficient, and conservative, condition to ensure that no point outside $\mathcal{U}$ will be reached is $\hat{\mathrm{G}}(\mathbf{x}) = 1 \; \forall \; \mathbf{x} \notin \mathcal{U}$. We can use this result to establish some interesting G functions.

Let $\mathrm{E}(\mathbf{x})$ be the distance between $\mathbf{x}$ and the complement of $\mathcal{U}$. If we allow a $\bar{\mathrm{G}}$ to be a function of $\mathrm{D}(\mathbf{x})$ and $\mathrm{E}(\mathbf{x})$, we can obtain a simple result that ensures the non-approachability of points outside $\mathcal{U}$. Consider the following function:

$$\bar{\mathrm{G}}(\mathrm{D}, \mathrm{E}) = \frac{(\mathrm{e}^{\mathrm{D}(\mathbf{x})} - 1)\mathrm{e}^{\mathrm{E}(\mathbf{x})}}{\mathrm{e}^{\mathrm{D}(\mathbf{x})}\mathrm{e}^{\mathrm{E}(\mathbf{x})} - 1}. \tag{3.45}$$

The function $\bar{\mathrm{G}}$ defined in (3.45) is: (i) positive definite; (ii) null if and only if $\mathrm{D} = 0$; and (iii) one if $\mathrm{E} = 0$. Thus, it has the necessary properties of a G-type like function and ensures that points outside $\mathcal{U}$ will not be approachable. It is important to emphasize that

for $\mathrm{D} = \mathrm{E} = 0$ we have that $\bar{\mathrm{G}}(\mathrm{D}, \mathrm{E})$ is not defined. However, this only happens on curves with self-intersections, which are out of scope by now.

There is also another way we can define the function $\mathrm{G}(\mathrm{D})$ in order to obtain a field that does not lead the trajectory $\mathbf{x}(\mathrm{t})$ to leave the set $\mathcal{U}$. This has the advantage that computing the distance E is not necessary. It is based on the notion that the convergent component alone never makes a point $\mathbf{x} \notin \mathcal{U}$ be approachable. Let δ_0 be the smallest distance $\mathrm{D}(\mathbf{x}_\mathrm{P})$ of any point $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$. For the time invariant case, we can define:

$$\delta_0 = \min_{\mathbf{x}_\mathrm{P} \notin \mathcal{U}} \mathrm{D}(\mathbf{x}_\mathrm{P}). \tag{3.46}$$

If $\mathrm{G}(\mathrm{D}) = 1$ for every $\mathrm{D} \geq \delta_0$, thus for every point $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$, we have that these points can not be approachable. A simple function G that ensures that is given by:

$$\mathrm{G}(\mathrm{D}) = \begin{cases} -\dfrac{\mathrm{D}^2}{\delta_0^2} + \dfrac{2\mathrm{D}}{\delta_0} & \text{if} \quad \mathrm{D} \leq \delta_0, \\ 1 & \text{if} \quad \mathrm{D} > \delta_0. \end{cases} \tag{3.47}$$

Note that $\mathrm{G}(0) = 0$ and $\mathrm{D} \geq \delta_0 \implies \mathrm{G}(\mathrm{D}) = 1$. Moreover, $\mathrm{G}(\mathrm{D})$ is Lipschitz and strictly increasing in the interval $[0,\ \delta_0]$. If we assume a co-domain in the interval $[0,\ 1)$ we can consider the inverse function G^{-1} used in the computation of the invariant set $\mathcal{I}(\mathrm{t})$ in Proposition 3.

On the left of Figure 3.2, we exemplify a vector field computed using the function $\bar{\mathrm{G}}(\mathrm{D}, \mathrm{E})$ defined in (3.45). On the right of the Figure, we exemplify a field that uses the function $\mathrm{G}(\mathrm{D})$ in (3.47). The curve $\mathcal{C}$ is an ellipse with semiaxis $\mathrm{a} = 2.0$ and $\mathrm{b} = 1.2$. In this example, it can be shown that $\delta_0 = \mathrm{b}^2/\mathrm{a} = 0.72$. The green line corresponds to the set of points $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$. The initial condition of each trajectory was in $\mathcal{U}$ but very close to the green line. Nevertheless, no trajectory left $\mathcal{U}$. In other words, no trajectory reached the green line. Note that the function $\mathrm{G}(\mathrm{D})$ in (3.47) is more aggressive, and more conservative, in the sense that it is 1 for every point $\mathbf{x}$ such that $\mathrm{D}(\mathbf{x}) \geq \delta_0$.

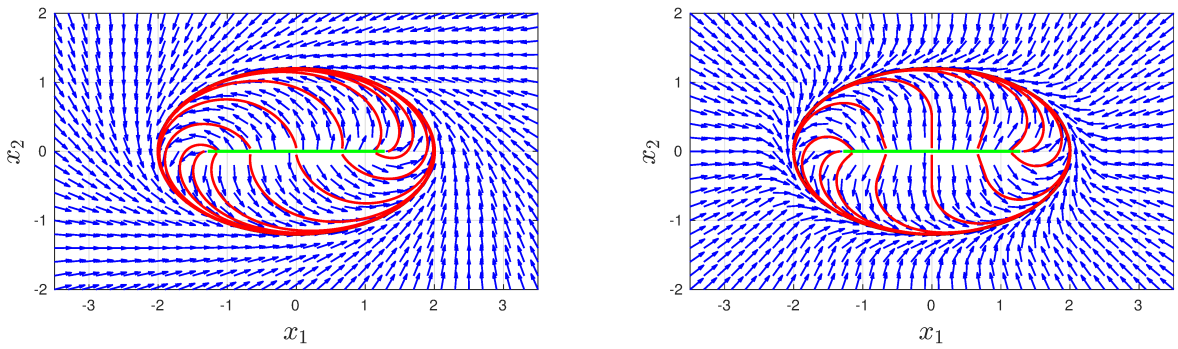

Figure 3.2: On the left, an example of a vector field computed with the function $\bar{\mathrm{G}}(\mathrm{D}, \mathrm{E})$ in (3.45). On the right, an example of a vector field computed with the function $\mathrm{G}(\mathrm{D})$ in (3.47). No trajectory reaches the green line, which is the set of points out of $\mathcal{U}$.

3.3.2 Initial condition limitation

A possibility to ensure that Assumption 4 holds is to consider a constraint on the initial condition of the system $\dot{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x}, \mathrm{t})$. Let δ_0 be the smallest distance $\mathrm{D}(\mathbf{x}_\mathrm{P}, \mathrm{t})$ to the curve of any point $(\mathbf{x}_\mathrm{P}, \mathrm{t}) \notin \mathcal{U}$ for all $\mathrm{t} > 0$. Now, considering the time varying case, $\delta_0(\mathrm{t})$ is redefined as:

$$\delta_0 = \min_{(\mathbf{x}_\mathrm{P}, \mathrm{t}) \notin \mathcal{U}} \mathrm{D}(\mathbf{x}_\mathrm{P}, \mathrm{t}). \tag{3.48}$$

Note that the scalar distance $\mathrm{D}(\mathbf{x}, \mathrm{t})$ is well defined as $\mathrm{D}(\mathbf{x}, \mathrm{t}) = \min_{\mathrm{s}} \|\mathbf{x} - \mathbf{r}(\mathrm{s}, \mathrm{t})\|$. Consider the set $\mathcal{X}_0$ given by:

$$\mathcal{X}_0 = \{\mathbf{x} \in \mathbb{R}^\mathrm{n} \mid \mathrm{D}(\mathbf{x}, 0) < \delta_0\}. \tag{3.49}$$

Consider the system $\dot{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x}, \mathrm{t})$ with initial condition $\mathbf{x}(0) = \mathbf{x}_0$. Assumption 4 can also be ensured if $\mathbf{x}_0 \in \mathcal{X}_0$. This is true because $\mathrm{D}(\mathbf{x}(0), 0) < \delta_0$ and since $\dot{\mathrm{D}} \leq 0$ we have $\mathrm{D}(\mathbf{x}(\mathrm{t}), \mathrm{t}) \leq \mathrm{D}(\mathbf{x}(0), 0) < \delta_0 \ \forall \ \mathrm{t} \geq 0$, which ensures that no point outside $\mathcal{U}$ will be reached.

In addition, note that if Assumptions 1 and 2 hold, we have that $\delta_0 > 0$. The problem with this approach is that the computation of δ_0 depends on the computation of the set $\mathcal{U}$, which is not trivial in the general case. In Section 3.3.3 we show how an equivalent initial condition can be achieved by considering a slightly different vector field.

3.3.3 Vector field in augmented space

In this Section, we propose a strategy to solve the issue when the point $\mathbf{x}^*$, or the parameter s^* equivalently, is not uniquely defined. The idea consists on the consideration of a virtual variable, let us call it $\mathrm{s_v}$, in place of s^* to compute the vector field. A propagation law for this variable will attempt to make it converge to s^* (or at least an equivalent local optimum). This approach is inspired on the one proposed in [83].

From now on, we assume the time invariant case. Thus, the augmented vector field will have only a convergent and a tangential component. As we show, the resulting vector field has many similarities with the one presented in Section 3.2. However, we do not present formal proofs for this novel strategy as we do in that Section.

3.3.3.1 Initial definitions

In this modified strategy, the point on the curve used to compute the field is not the one given by the parameter s^*, achieved via an optimization problem. We now consider a parameter s_v as a virtual variable. This idea may seem dissonant to the original idea of considering the closest point on the curve $\mathcal{C}$. However, we will attempt to make $s_v \to s^*$. For this new, and slightly different framework, consider the following definitions

Definition 12. *The variable* $s_v \in \mathbb{R}$ *is a virtual state of the system that defines the point on the curve* $\mathcal{C}$*, let it be* $\mathbf{r}(s_v)$*, that will be used to compute the vector field.*

Definition 13. *The* reference point[4] $\mathbf{x}_v(s_v) : \mathbb{R} \to \mathbb{R}^n$ *is*

$$\mathbf{x}_v(s_v) = \mathbf{r}(s_v). \tag{3.50}$$

Definition 14. *The* virtual distance vector $\mathbf{D}_v(s_v, \mathbf{x}) : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^n$ *and the* virtual scalar distance $D_v(s_v, \mathbf{x}) : \mathbb{R}\times\mathbb{R}^n \to \mathbb{R}^+$ *are obtained as*

$$\mathbf{D}_v(s_v, \mathbf{x}) = \mathbf{x} - \mathbf{x}_v(s_v), \tag{3.51}$$

$$D_v(s_v, \mathbf{x}) = \|\mathbf{D}_v(s_v, \mathbf{x})\|. \tag{3.52}$$

Note that Definitions 12, 13, and 14 are analogous to Definitions 1, 2, and 3. The difference is that the definitions above consider the parameter s_v instead of the closest point parameter s^*. Moreover, the time dependence of the variables is not present, since we are considering static curves.

3.3.3.2 Augmented space vector field proposal

The vector field we propose to tackle the singularities is analogous to the one defined in equation (3.12). The difference is that instead of using $\mathbf{D}$ and $\mathbf{T}^*$, we will use $\mathbf{D}_v$ and $\mathbf{T}(s_v)$. We define the *augmented space vector field* as

$$\mathbf{\Phi}_v(s_v, \mathbf{x}) = -v_r G(D_v)\frac{\mathbf{D}_v}{D_v} + v_r H(D_v)\mathbf{T}(s_v). \tag{3.53}$$

Note the similarity between $\mathbf{\Phi}_v$ in (3.53) and $\mathbf{\Phi}$, in (3.12). When $s_v = \mathbf{s}^*$, vector $\mathbf{D}_v/D_v$ corresponds to the gradient of the distance function $D(\mathbf{x})$. The case when $D_v = 0$ is

[4]This point $\mathbf{x}_v$ plays a role of a reference point on the curve. However, it is important to emphasize that is not the same role as the one of a reference trajectory. This will be formally discussed latter in this Section.

not a problem for the same reason discussed before, since $G(D_v)$ vanishes in this case. The feedforward term, present in (3.12), is out here and $\eta(t) = v_r$, given that we are considering a static curve $\mathcal{C}$.

Remark 5. *Rigorously, we cannot say that* $\mathbf{\Phi}_v$ *in (3.53) is a vector field in the sense that it does map the space* $\mathbb{R}^n$ *into itself. It happens because* $\mathbf{\Phi}_v$ *depends on* $\mathbf{x}$ *but also on* s_v*. In this Section, we overlook this issue. We may consider that* $\mathbf{\Phi}_v$ *is computed via an algorithm for instance. However, we will keep with the name vector field.*

Since s_v is now a state variable, it needs to be assigned an initial value. Since it is a virtual variable, *i.e.* it does not represent a physical state, we have the freedom to assign it with the initial condition we desire. A natural choice for $s_v(0)$ is the parameter of any of the closest points $\mathbf{x}^*$ associated with the initial point $\mathbf{x}(0)$. Thus, we define

$$s_v(0) = \arg\min_s \|\mathbf{x}(0) - \mathbf{r}(s)\|^2. \tag{3.54}$$

Note that, given equation (3.54), the condition $\mathbf{D}_v(s_v(0), \mathbf{x}(0))^T \mathbf{T}(s_v(0)) = 0$ holds. As discussed in Section 3.2, the original relation $\mathbf{D}^T\mathbf{T}^* = 0$ (see Lemma 2) is fundamental to achieve the convergence of the vector field to the curve $\mathcal{C}$. If we manage to have the $\mathbf{D}_v(s_v(t), \mathbf{x}(t))^T \mathbf{T}(s_v(t)) = 0$ for all $t > 0$, the vector field $\mathbf{\Phi}_v$ will converge to the curve as the original field $\mathbf{\Phi}$ in (3.12). We can accomplish that through the definition of the dynamics of s_v.

Consider that we define $\dot{s}_v$ as

$$\dot{s}_v = \frac{\mathbf{T}(s_v)}{\Omega_v}\dot{\mathbf{x}}, \tag{3.55}$$

in which $\Omega_v \equiv \Omega_v(s_v, \mathbf{x})) = 1 - \kappa(s_v)\mathbf{N}(s_v)^T\mathbf{D}_v$. Recalling, $\kappa(s_v)$ is the curve's curvature computed at the parameter s_v and $\mathbf{N}(s_v)$ is the curve's unit normal vector, also computed at s_v. Recalling Lemma 3, we have that $\dot{s}_v$ in (3.55) is the time derivative of the optimum parameter s^* (assuming $s_v = s^*$). Thus, if we start with condition in (3.54), and impose the dynamics in (3.55) to the virtual parameter, we maintain the orthogonality condition $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$.

Note that, with this approach, we are not seeking the global optimum anymore, but only the local one. In fact, the condition $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$ holds for every local optimum of the distance to the curve. The desired behavior is that the field will not be based exactly on the Euclidean distance function. It will be based on a distance to the curve that may be the shortest only locally. Thus, if the current parameter is $s_v = s^*$ and the trajectory passes through a singularity, s_v will no longer correspond to the shortest distance to the curve, but will still ensure the property of interest, which is $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$. Also, the variable s_v will not be subjected to a discontinuity.

Remark 6. *The computation of* Ω_v *in (3.55) requires that* $\mathbf{N}(s_v)$ *is well defined. The case it is not defined corresponds to a straight path or an inflection point of* $\mathcal{C}$*. However, in such cases we have* $\kappa(s_v) = 0$*. Since* $\|\mathbf{N}\| = 1$ *we have that* $\Omega_v = 1$*.*

Ideally, if we impose the dynamics in (3.55) to s_v we can keep the condition $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$. However, this would be a feedfoward only controller. Any uncertainty on the variable $\dot{\mathbf{x}}$ would make $\mathbf{x}_v$ no longer be a local closest point on the curve, and thus, $\mathbf{D}_v^T\mathbf{T}(s_v) \neq 0$. In order to solve this issue, we propose a corrective action to maintain the condition $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$. Thus, we consider the following propagation law for s_v

$$\dot{s}_v = \frac{\mathbf{T}(s_v)}{\Omega_v}\dot{\mathbf{x}} + k_s\mathbf{D}_v^T\mathbf{T}(s_v), \tag{3.56}$$

in which $k_s > 0$ is a constant. If the condition $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$ holds, we have no corrective action, and only the first parcel is able to keep the condition true. In the case this condition does not hold, the term $k_s\mathbf{D}_v^T\mathbf{T}(s_v)$ does not vanish. Figure 3.3 illustrates the control action $k_s\mathbf{D}_v^T\mathbf{T}(s_v)$. In the left, it shows a situation in which $\mathbf{x}_v$ is ahead of $\mathbf{x}^*$. In this case, the dot product $\mathbf{D}_v^T\mathbf{T}(s_v)$ is negative, thus, it will slow down the propagation of s_v. On the right, the opposite situation happens, and the term $k_s\mathbf{D}_v^T\mathbf{T}(s_v)$ will accelerate the propagation of s_v. Over time, one may expect that $s_v \to s^*$ even if $s_v(0) \neq s^*$.

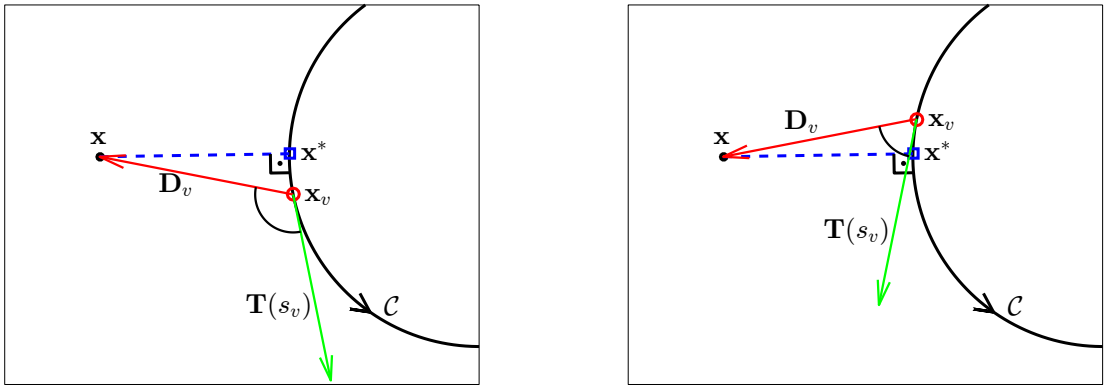

Figure 3.3: Illustration of the control action that tries to make s_v evolve so that $\mathbf{D}_v^T\mathbf{T}(s_v) = 0$. On the left, a situation when s_v is ahead of the local optimum. On the right, the opposite situation, s_v is behind s^*.

Now, we need to comment the situation in which Ω_v, in (3.56), is zero. If $\Omega_v = 0$, we have a problem in the propagation of s_v. Thus, we cannot say that the field $\mathbf{\Phi}_v$ is totally free of singularities. Given the definition $\Omega_v = 1 - \kappa(s_v)\mathbf{N}(s_v)^T\mathbf{D}_v$, we have that the condition for $\Omega_v = 0$ is

$$\mathbf{N}(s_v)^T\mathbf{D}_v = \frac{1}{\kappa(s_v)}. \tag{3.57}$$

Remember that, by definition, $\|\mathbf{N}\| = 1$, which makes $D_v = \|\mathbf{D}_v\| \geq \mathbf{N}(s_v)^T\mathbf{D}_v$. Thus, the condition $\Omega_v = 0$ implies $D_v \geq \kappa(s_v)^{-1}$. Mathematically

$$\Omega_v = 0 \implies \|\mathbf{D}_v\| \geq \frac{1}{\kappa(s_v)}. \tag{3.58}$$

Consider $\kappa_{\max} = \max_{s} \kappa(s)$. If we assume $D_v < \kappa_{\max}^{-1}$ we avoid the possibility of $\Omega_v = 0$, and consequently, a singularity in the computation of $\dot{s}_v$. In fact, this argument corresponds to the contraposition of the sentence in (3.58). Mathematically

$$\|\mathbf{D}_v\| < \frac{1}{\kappa(s_v)} \implies \Omega_v \neq 0. \tag{3.59}$$

Given the sentence in (3.59), we can ensure $\Omega_v \neq 0$ if we impose the following constraint on the initial condition

$$\mathcal{X}_v = \left\{ \mathbf{x} \in \mathbb{R}^n \mid D(\mathbf{x}) < \frac{1}{\kappa_{\max}} \right\}. \tag{3.60}$$

The initial set $\mathcal{X}_v$, in (3.60), has a definition similar to the set $\mathcal{X}_0$, in (3.49), however it is much easier to compute. Since $\kappa(s)$ is a local property, $\kappa_{\max}$ can be computed from the simple evaluation of $\kappa(s)$ in the unidimensional curve $\mathcal{C}$.

As we comment in the beginning of this Section, it is not our intention to prove the stability of the guidance strategy using the augmented vector field $\mathbf{\Phi}_v$, in (3.53), along with the propagation law in $\dot{s}_v$, in (3.56). The methodology presented in this Section has a practical purpose. A better mathematical investigation of this theory is left to a future research. In practice, the use of the augmented field approach proposed in this section has some advantages. We list them below:

- It provides a vector field, $\mathbf{\Phi}_v$, which is well defined when the set s^* is not a singleton.
- The field $\mathbf{\Phi}_v$ enables the consideration of curves with self intersection, which were intractable with the original field $\mathbf{\Phi}$. See Assumption 1.
- The computation of Φ_v does not require an optimization problem to be solved every time step. The cost of propagating the parameter s_v is way computationally cheaper. The optimization problem is solved only once, to compute the initial value of the virtual variable s_v.
- It enables the computation of an initial set, $\mathcal{X}_v$ in (3.60), that is only dependent on the curve's maximum curvature. Starting on this set singularities on the computation of $\mathbf{\Phi}_v$ are prevented.

The point $\mathbf{x}_v(s_v)$ in Definition 13 may look like as a reference trajectory. However, note that it is not defined by time, but on the propagation of the virtual variable s_v. As we discussed in Section 2.1.1 of the related works (Chapter 2), the vector field path controller does not have some issues that may occur in simple trajectory tracking approaches. The propagation of s_v is dependent on the robot's speed $\dot{\mathbf{x}}$. If a temporary mechanical problem makes the robot stop, for instance, the propagation of s_v stops as well, since $\dot{s}_v$ in (3.56) will be zero. We emphasize that the computation of $\dot{s}_v$ requires a speed measurement $\dot{\mathbf{x}}$. This is what enables the propagation to stop when the robot stops. If this measurement

is not available, we can assume that $\dot{\mathbf{x}} = \mathbf{\Phi}_v$, *i.e.*, the robot is following exactly the field reference. In this case, if the robot stops, the propagation of s_v will continue. In practice, the system will converge to a situation similar to the one in the left of Figure 3.3, since the term $k_s \mathbf{D}_v^T \mathbf{T}(s_v)$ will start to act. If we choose k_s high enough, the distance between $\mathbf{x}^*$ and $\mathbf{x}_v$ will be small. Since s_v is a virtual variable, the gain k_s can be set to a high value without any practical issue.

This effect that the "reference point" stays in front of the robot when it stops is also present in the singularity free vector field theory proposed in [83]. However, here we have an independent parameter, k_s, that allows to play with the distance in which the system will stabilize. Equivalently, how far from s^* will the parameter s_v stop.

To end this Section, we emphasize that the theory presented here shall be further analyzed. A better mathematical investigation may be done to evaluate the behavior of the field $\mathbf{\Phi}_v$. For now, it can be used in practical implementations mainly to improve the computational efficiency of the controller and to incorporate the ability to consider curves with self intersections.

3.4 Incorporation of obstacles

In this Section, we show how the vector field designed in Section 3.2 can be augmented to incorporate an obstacle deviation feature. Similar to what was proposed in [60], the approach to deviate from the obstacles will be their circulation. We show how the specific vector field structure presented in Section 3.2 can be easily applied to define a field that circulates an obstacle. As the field that converges to a path is computed from the closest point in this path, the *contouring field* can be inferred from the knowledge of the closest point in the obstacle.

In practice, to apply the vector field designed in this Section we can adopt two approaches. In the first approach, one needs to know the geometry of all obstacles in the workspace. In this case, an algorithm must compute the closest point of the obstacle set. The second option is to have a LiDAR attached to the robot, so that it can detect the obstacle points around it. Then, a simple iteration through these points can provide the closest one.

3.4.1 Obstacle contour

In this Subsection, we first establish some basic definitions and then we proceed to the definition of the vector field that contour an obstacle.

3.4.1.1 Initial definitions

We aim to show how the vector field described in Section 3.2 can be used to make a robot contour an obstacle while keeping a fixed distance from it. The Euclidean vector field is computed from the the closest point on the reference curve. In order to contour an obstacle, its closest point is used with a similar purpose.

Consider the set of collision points $\mathcal{O}(t)$. A point $\mathbf{x}_o \in \mathcal{O}(t)$ if $\mathbf{x}_o$ is a collision point, in other words, a point that belongs to an obstacle. In order to navigate safely, the robot's position $\mathbf{x}$ must be at a minimum distance from any $\mathbf{x}_o \in \mathcal{O}(t)$, $\forall$ t.

In Section 3.2, the vector field was computed from the closest point on the path, called $\mathbf{x}^*$. Now, in order to apply the vector field to make the robot contour an obstacle, we will consider the closest point $\mathbf{x}_o^*$ on the obstacle. Consider then the following definition:

Definition 15. *The closest point* $\mathbf{x}_o^*(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$ *that belongs to an obstacle is given by*

$$\mathbf{x}_o^*(\mathbf{x}, t) = \arg \min_{\mathbf{x}_o \in \mathcal{O}(t)} \|\mathbf{x} - \mathbf{x}_o\|^2. \tag{3.61}$$

We also define the distance quantities between the robot and the obstacle set:

Definition 16. *The* distance vector to the obstacle set $\mathbf{D}_o(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n$ *and the* scalar distance to the obstacle set $D_o(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^+$ *are obtained as*

$$\mathbf{D}_o(\mathbf{x}, t) = \mathbf{x} - \mathbf{x}_o^*(\mathbf{x}, t), \tag{3.62}$$

$$D_o(\mathbf{x}, t) = \|\mathbf{D}_o(\mathbf{x}, t)\|. \tag{3.63}$$

In order to circulate the obstacle, the robot will attempt to keep a distance $\lambda > 0$ from the it. In practice, it is also necessary that λ is greater than the robot's safety radius. Thus, we consider the set of points that are a distance λ from the obstacle set $\mathcal{O}(t)$.

Definition 17. *The equidistant set from the obstacle set is defined as*

$$\mathcal{S}_o(t) = \{\mathbf{x} \in \mathbb{R}^n : D_o(\mathbf{x}, t) = \lambda\}. \tag{3.64}$$

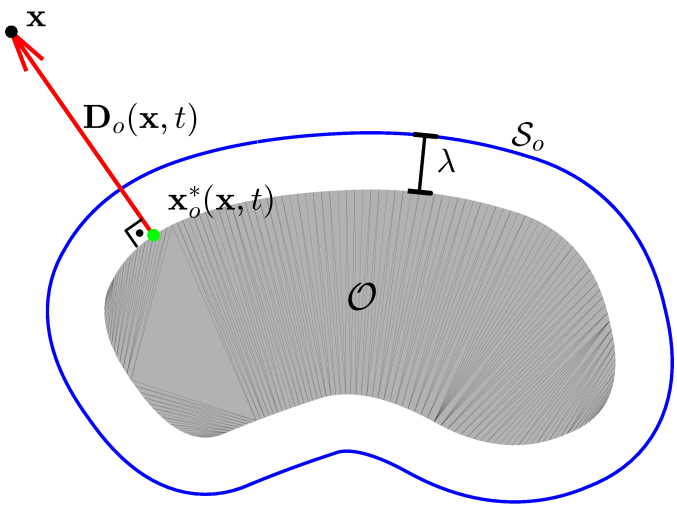

Figure 3.4: Illustration of Definitions 15, 16, and 17. Obstacle set $\mathcal{O}$ in gray, distance vector $\mathbf{D}_o(\mathbf{x}, t)$ in red, and equidistant set $\mathcal{S}_o$ in blue.

Note that, for the case $n = 2$, the set $\mathcal{S}_o$ is a unidimensional set, thus is equivalent to a path. In this case, the obstacle will be circulated in a defined path. For $n > 2$, the set $\mathcal{S}_o$ is a hyper surface around the obstacle. In this second case, the path that the obstacle will be circulated is not defined. The method will only ensure that the robot will keep at a distance λ from the obstacle and moving along it.

Figure 3.4 illustrates Definitions 15, 16, and 17 for the bidimensional case. The grey region represents the set $\mathcal{O}$. The green dot is the point $\mathbf{x}_o^*(\mathbf{x}, t)$, and the red vector is $\mathbf{D}_o(\mathbf{x}, t)$. The set $\mathcal{S}_o$ is depicted by the blue path, which is at a distance λ from the set $\mathcal{O}$.

3.4.1.2 Obstacle contour methodology

Now we will compute a vector field with the structure presented in Section 3.2 that allows the robot to circulate an obstacle. In order to compute this field, we need to compute the distance vector, let it be $\mathbf{D}_\lambda$, to the desired set $\mathcal{S}_o$. This distance can be easily achieved from the vector $\mathbf{D}_o$ as

$$\mathbf{D}_\lambda = \mathbf{D}_o - \lambda \frac{\mathbf{D}_o}{D_o}. \tag{3.65}$$

We can also define $D_\lambda = \|\mathbf{D}_\lambda\|$. Then, a convergent component to the equidistant set $\mathcal{S}_o$, let it be $\mathbf{\Psi}_G(\mathbf{x}, t)$, can be computed as

$$\mathbf{\Psi}_G(\mathbf{x}, t) = -G(D_\lambda)\frac{\mathbf{D}_\lambda}{D_\lambda}. \tag{3.66}$$

Thus, the vector $\mathbf{D}_o$ can easily be used to achieve the convergent component of the field. In order to compute a tangent component, we need to respect the requirement that it is perpendicular to the vector $\mathbf{D}_\lambda \parallel \mathbf{D}_o$.

In the case of fields in two dimensions, $n = 2$, there is only one possible direction for the tangent component. In the case $n > 2$ there is more than one possible direction. We propose an additional vector field to choose among the possible directions. Let $\mathcal{A}(\mathbf{x}, t)$ be this additional auxiliary field. Then the tangent component of the field to circulate the obstacle, let us call it $\mathbf{T}_\lambda$, is computed by removing the component of $\mathcal{A}(\mathbf{x}, t)$ in the direction of $\boldsymbol{\Psi}_G$. Let the unit vector $\nabla D_\lambda = \mathbf{D}_\lambda / D_\lambda$. Then we have

$$\mathbf{T}_\lambda(\mathbf{x}, t) = \frac{\Pi_{\nabla D_\lambda}(\mathbf{x}, t)\mathcal{A}(\mathbf{x}, t)}{\|\Pi_{\nabla D_\lambda}(\mathbf{x}, t)\mathcal{A}(\mathbf{x}, t)\|}, \tag{3.67}$$

in which $\Pi_{\nabla D_\lambda}(\mathbf{x}, t)$ is the projector on the null space of ∇D_λ. Vector $\mathbf{T}_\lambda(\mathbf{x}, t)$ is basically the normalized version of $\mathcal{A}(\mathbf{x}, t)$ without its component in the direction of $\boldsymbol{\Psi}_G(\mathbf{x}, t)$. For now, we assume that $\mathcal{A}(\mathbf{x}, t)$ is such that $\Pi_{\nabla D_\lambda}(\mathbf{x}, t)\mathcal{A}(\mathbf{x}, t)$ never vanishes.

Now, given that $\boldsymbol{\Psi}_G^T \mathbf{T}_\lambda = 0$, the tangent component of the vector field that contours the obstacle is given by

$$\boldsymbol{\Psi}_H(\mathbf{x}, t) = H(D_\lambda)\mathbf{T}_\lambda(\mathbf{x}, t). \tag{3.68}$$

In the case of moving obstacles, the following time feed-forward component can be computed:

$$\boldsymbol{\Psi}_T(\mathbf{x}, t) = \Pi_{\mathbf{T}_\lambda}(\mathbf{x}, t)\frac{\partial \mathbf{D}_\lambda}{\partial t}. \tag{3.69}$$

in which $\Pi_{\mathbf{T}_\lambda}(\mathbf{x}, t)$ is the projector on the null space of $\mathbf{T}_\lambda$.

Finally, the vector field $\boldsymbol{\Psi}(\mathbf{x}, t)$ that contours an obstacle at a fixed distance is defined by

$$\boldsymbol{\Psi}(\mathbf{x}, t) = \eta\boldsymbol{\Psi}_G(\mathbf{x}, t) + \eta\boldsymbol{\Psi}_H(\mathbf{x}, t) + \boldsymbol{\Psi}_T(\mathbf{x}, t), \tag{3.70}$$

where η is computed to achieve $\|\boldsymbol{\Psi}(\mathbf{x}, t)\| = v_r$, as described in Section 3.2.

Note that the vector field defined in (3.70) preserves the structure of the field presented in Section 3.2. The considered convergence component is not directly defined from the closest point on the path, but passes through the closest point on the obstacle. The tangent component preserves its fundamental property, that is the orthogonality with the convergent one. The difference now it that, for the n dimensional case, the field does not converge to a path, but to the surface $\mathcal{S}_o$ around the obstacle and moves along it according to the indication of the auxiliary field $\mathcal{A}$.

Figure 3.5 depicts, for the time independent case and $n = 2$, two fields that contour an obstacle. On the left we show the auxiliary fields $\mathcal{A}(\mathbf{x})$, and on the right we show the associated field $\boldsymbol{\Psi}(\mathbf{x})$ that circulates the gray obstacle. In the top example, the field $\mathcal{A}(\mathbf{x})$ is constant. In this case, the field $\boldsymbol{\Psi}(\mathbf{x})$ circulates the obstacle clockwise in the top and

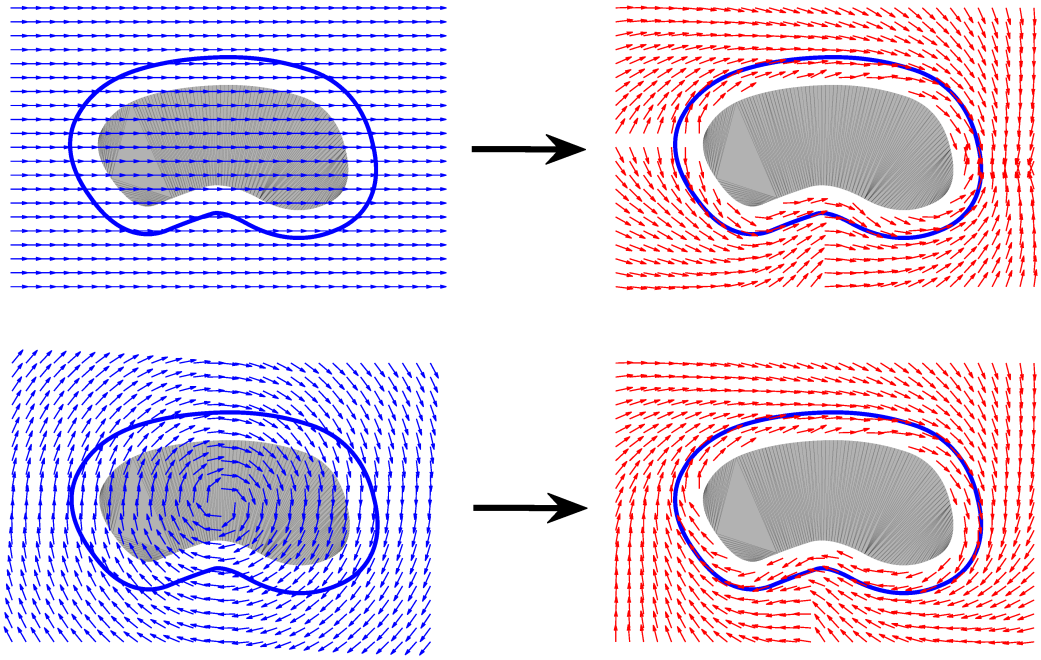

Figure 3.5: Two examples (top and bottom) of fields that contour the gray obstacle. On the left we show the auxiliary field $\mathcal{A}(\mathbf{x})$ that originated the fields $\boldsymbol{\Psi}(\mathbf{x})$ in the right.

counter-clockwise in the bottom. In the bottom example, the field $\mathcal{A}(\mathbf{x})$ is a circulating field, thus, the resulting field $\boldsymbol{\Psi}$ always circulates the obstacle in the same direction.

In these bidimensional cases, the only influence of field $\mathcal{A}$ is in the sense of direction of field $\boldsymbol{\Psi}$, since there is only one direction perpendicular to the convergence component. Note also that, in general, the field $\boldsymbol{\Psi}$ may be prone to singularities, as in the top example of Figure 3.5. As discussed in the Conclusion (Chapter 6), this issue will be left to a future work.

In the following Section, we show how the strategy presented here and the original vector field proposed in Section 3.2 can be put together to provide a safe navigation towards a path.

3.4.2 Collision free navigation

In this Subsection, we show how the strategies presented in Sections 3.2 and 3.4.1 can be used together to implement a collision free navigation method. Basically, the new approach will switch between two behaviors: path following (Section 3.2) and obstacle contouring (Section 3.4.1).

From now on, let us call the field $\boldsymbol{\Phi}(\mathbf{x}, \mathrm{t})$, defined in (3.30), as the *original field*. It is the one that makes the robot converge to and follow the original curve $\mathcal{C}$. The field

$\boldsymbol{\Psi}(\mathbf{x}, \mathrm{t})$, defined in (3.70), will be called the *contouring field*, since it is designed to contour the obstacle.

The idea of the navigation approach is to follow the vector field to the original path $\mathcal{C}$ when there is no eminent collision with any obstacle. When an obstacle is in the way, the navigation approach will be the circulation of the obstacle.

By eminent collision we mean a situation that respects two conditions. First, the direction of the original field points in the direction that makes the distance to the obstacle decrease. Second, the distance between the state $\mathbf{x}$ and the closest obstacle is less than a threshold $\mathrm{D_{in}}$. If these conditions are met, the navigation uses the contouring field and the robot will attempt to contour the obstacle at a distance λ. It is necessary that the value of $\mathrm{D_{in}}$ is such that $\lambda < \mathrm{D_{in}}$.

In order to compute the field $\boldsymbol{\Psi}(\mathbf{x}, \mathrm{t})$ we use the auxiliary field $\mathcal{A}(\mathbf{x}, \mathrm{t})$ as the original field $\boldsymbol{\Phi}(\mathbf{x}, \mathrm{t})$, *i.e.* $\mathcal{A}(\mathbf{x}, \mathrm{t}) = \boldsymbol{\Phi}(\mathbf{x}, \mathrm{t})$. With this strategy, the system will attempt to contour the obstacle as close as possible to the original field that converges to the curve. Note that, by making $\mathcal{A} = \boldsymbol{\Phi}$, we can not ensure that, in equation (3.67), $\Pi_{\nabla \mathrm{D}_\lambda} \boldsymbol{\Phi} \neq \mathbf{0}$. Deeper discussions on this issue are left for a future work.

In order to avoid abrupt transitions between the fields $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$, we also consider an intermediate state when the distance between $\mathbf{x}$ and the set $\mathcal{O}$ is greater than $\mathrm{D_{in}}$ but smaller than a distance $\mathrm{D_{in}^0} > \mathrm{D_{in}}$. In this state, the field will be defined as a combination of the fields. To compute this composite field, consider the parameter θ defined as

$$\theta(\mathbf{x}, \mathrm{t}) \equiv \theta = \frac{\mathrm{D_o}(\mathbf{x}, \mathrm{t}) - \mathrm{D_{in}}}{\mathrm{D_{in}^0} - \mathrm{D_{in}}}. \tag{3.71}$$

Note that $\theta = 1$ when $\mathrm{D_o} = \mathrm{D_{in}^0}$, in other words, when the transition $\boldsymbol{\Phi} \to \boldsymbol{\Psi}$ starts. When the transition is complete $\theta = 0$. In the transition state, when $\mathrm{D_{in}} < \mathrm{D_o} < \mathrm{D_{in}^0}$, we have $\theta \in [0, 1]$

The navigation vector field $\mathbf{F}(\mathbf{x}, \mathrm{t})$ that is able to follow a path while deviating from the obstacles is defined as

$$\mathbf{F}(\mathbf{x}, \mathrm{t}) = \begin{cases} \boldsymbol{\Phi}(\mathbf{x}, \mathrm{t}) & \text{if } \ \mathbf{D_o^T}\boldsymbol{\Phi} \geq 0 \ \text{ or } \ \mathrm{D_o} > \mathrm{D_{in}^0}, \\ \mathrm{v_r} \dfrac{\theta\boldsymbol{\Phi}(\mathbf{x}, \mathrm{t}) + (1-\theta)\boldsymbol{\Psi}(\mathbf{x}, \mathrm{t})}{\|\theta\boldsymbol{\Phi}(\mathbf{x}, \mathrm{t}) + (1-\theta)\boldsymbol{\Psi}(\mathbf{x}, \mathrm{t})\|} & \text{if } \ \mathbf{D_o^T}\boldsymbol{\Phi} < 0, \ \ \mathrm{D_{in}} \leq \mathrm{D_o} \leq \mathrm{D_{in}^0}, \\ \boldsymbol{\Psi}(\mathbf{x}, \mathrm{t}) & \text{if } \ \mathbf{D_o^T}\boldsymbol{\Phi} < 0, \ \ \mathrm{D_o} < \mathrm{D_{in}}, \end{cases} \tag{3.72}$$

in which the field $\mathcal{A}$ used to compute $\boldsymbol{\Psi}$ is defined as $\mathcal{A} = \boldsymbol{\Phi}$. In the transition phase ($\mathrm{D_{in}} \leq \mathrm{D_o} \leq \mathrm{D_{in}^0}$) the composite field is computed through a convex combination of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ followed by a normalization.

Figure 3.6 illustrates the vector field defined in (3.72). In the figure, the dark shapes are the obstacles. The field depicted in black converges to the curve in black while avoiding the obstacles. In the blue region, the original field $\boldsymbol{\Phi}$ is totally active. In the green region, we have the composite field, and in the yellow region the field $\boldsymbol{\Psi}$ is totally

active. In the top right, we see that the example trajectory (in red) deviates from an obstacle that is far away from the target curve. In the bottom, we see that the trajectory also deviates from an obstacle on the original path. In this case, the trajectory is forced to leave the path. This effect may also happen when the obstacle is close to the curve (top left).

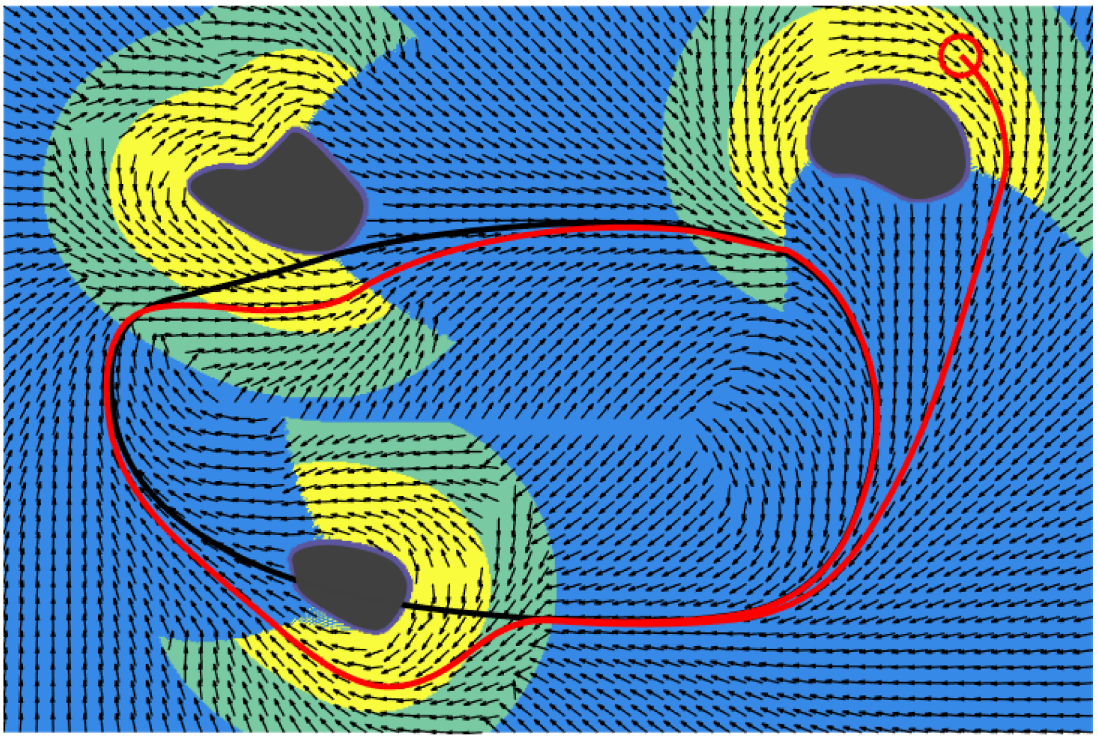

Figure 3.6: Example of the vector field defined in (3.72). The red trajectory attempts to circulate the path in black while deviating from the obstacles in dark gray. The regions in blue, green, and yellow show which of the three conditions in equation (3.72) is active.

Chapter 4

Quadcopter control

In this Chapter, we show how the vector field described in Chapter 3 can be used to control a quadcopter. A controller that takes into account the field's derivatives is proposed and its convergence properties formally analyzed. An analogous approach, applied to a fixed-wing UAV, was presented in [64]. The idea is now extended to a quadcopter model that takes the thrust force and the angular velocities as control inputs. As we will define, the position of the aerial vehicle is represented by $\mathbf{p}$. The objective is to make this state behave as the state $\mathbf{x}$ in the simple integrator of the previous Chapter.

A paper with the results presented in this Chapter was published on the International Conference on Robotics and Automation (ICRA), in 2020, with the title "Robust quadcopter control with artificial vector fields" [63].

4.1 Control problem setup

Consider $\mathbf{p} \in \mathbb{R}^3$ the position of the robot and $\mathbf{v} \in \mathbb{R}^3$ its velocity, represented in the world frame. Assume a world frame with the z axis pointing up and gravity acceleration pointing down with value g. Let $\mathrm{R}_\mathrm{b}^\mathrm{w} \in \mathrm{SO}(3)$ be a rotation matrix representing the attitude of the quadcopter (body) with respect to the world frame. Based on [43], the following vehicle model is assumed

$$\dot{\mathbf{p}} = \mathbf{v}, \tag{4.1a}$$

$$\dot{\mathbf{v}} = \mathrm{R}_\mathrm{b}^\mathrm{w}\hat{\mathbf{z}}\frac{\tau}{\mathrm{m}} - \mathrm{g}\hat{\mathbf{z}} + \frac{\mathrm{f_d}(\mathbf{v})}{\mathrm{m}} + \frac{\delta_\tau}{\mathrm{m}}, \tag{4.1b}$$

$$\dot{\mathrm{R}}_\mathrm{b}^\mathrm{w} = \mathrm{R}_\mathrm{b}^\mathrm{w}\mathrm{S}(\omega + \delta_\omega), \tag{4.1c}$$

in which $\hat{\mathbf{z}}$ is the vector $[0,\ 0,\ 1]^\mathrm{T}$, m is the mass of the vehicle, and $\mathrm{f_d}(\mathbf{v})$ is a known drag force, which will be specified later, expressed in the world frame. The control inputs are the total thrust force τ and the angular velocity vector $\omega \in \mathbb{R}^3$, expressed in the body frame. The matrix $\mathrm{S}(\omega)$ is the skew-symmetric matrix associated with the angular

velocity $\omega = [\omega_x,\ \omega_y,\ \omega_z]^T$. The inputs $\delta_\tau \in \mathbb{R}^3$ and $\delta_\omega \in \mathbb{R}^3$ represent unknown but bounded disturbances on the thrust and on the angular velocity command. The term δ_τ may also account for errors in the model of the drag force. We assume that these uncertainties are norm-bounded, i.e. $\|\delta_\tau\| \leq \Delta_\tau$ and $\|\delta_\omega\| \leq \Delta_\omega$.

The goal is to make the vector $\mathbf{p}$ converge to a time-varying curve $\mathcal{C}(t) \subset \mathbb{R}^3 \times \mathbb{R}^+$, by establishing a control law based on the artificial vector field $\boldsymbol{\Phi}(\mathbf{p}, t) : \mathbb{R}^3 \times \mathbb{R}^+ \rightarrow \mathbb{R}^3$, designed in Chapter 3 (or any other vector field). It is required that the vector field $\boldsymbol{\Phi}$ is twice differentiable, *i.e.*, $\boldsymbol{\Phi} \in C^2$. Considering the methodology presented in Chapter 3, a field with such property can be achieved by assuming a curve $\mathcal{C}(t)$ that is three times differentiable, since one degree of continuity is lost with the computation of the tangent vector. Besides the definition of a control law, it is necessary to analyze the influence of the bounded disturbances δ_τ and δ_ω. In other words, how far the robot will be from the curve when disturbances are present.

The problem addressed in this Chapter can be divided in the following two sub-problems:

Problem 2. *Find control laws for the signals τ and ω such that the position state $\mathbf{p}$ of (4.1), with $\delta_\tau = \mathbf{0}$ and $\delta_\omega = \mathbf{0}$, converges to an integral line of the field $\boldsymbol{\Phi}(\mathbf{p}, t)$ and, furthermore, $\mathbf{p}(t) \rightarrow \mathcal{C}(t)$ as $t \rightarrow \infty$.*

Problem 3. *Given the bounds $\|\delta_\tau\| \leq \Delta_\tau$ and $\|\delta_\omega\| \leq \Delta_\omega$, find a maximum tracking error for the state $\mathbf{p}$. In other words, find an ultimate bound for the robot's position error.*

4.2 Control scheme

In this section, we will develop a three-layer controller to solve Problem 2. The outer loop is the guidance strategy based on the vector field proposed in Chapter 3. The intermediate layer is a controller that computes the necessary acceleration for the vehicle to follow the vector field. The inner loop controls the orientation of the vehicle necessary to impose the computed acceleration. The convergence proofs will rely on the ISS of the closed loop systems. Figure 4.1 depicts the control that will be presented. Although we illustrate the controller as a cascade connection, we emphasize that all modules run in the same frequency. The velocity reference $\boldsymbol{\Phi}$ and the acceleration reference a_r are internal variables of the controller.

In this Chapter, we consider the vector field $\boldsymbol{\Phi}$ previously presented. However, one should have in mind that the quadcopter controller presented here can be used along with the fields $\boldsymbol{\Psi}$, $\boldsymbol{\Phi}_v$ and $\mathbf{F}$, which are variations of $\boldsymbol{\Phi}$.

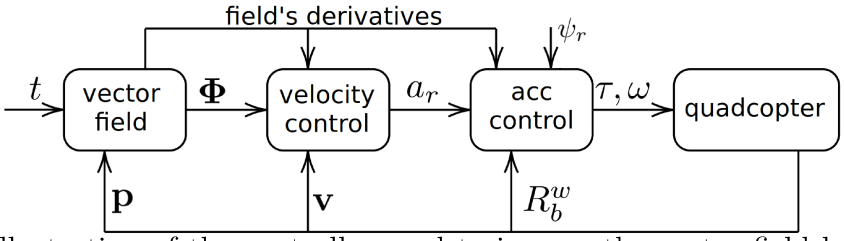

Figure 4.1: Illustration of the controller used to impose the vector field behavior to the quadcopter. Since the field does not provide any reference for heading, a reference ψ_r is provided to the inner controller.

4.2.1 Second-order control

In the previous Chapter, we have shown that the system $\dot{\mathbf{p}} = \mathbf{\Phi}(\mathbf{p}, \mathrm{t}) + \delta_\mathrm{v}$ (or equivalently $\dot{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x}, \mathrm{t}) + \delta_\mathrm{v}$), which considers a simple integrator model, is ISS with respect to additive perturbations. Now, consider the second-order integrator given by

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v}, \\ \dot{\mathbf{v}} &= \mathbf{a}_\mathrm{d}, \end{aligned} \tag{4.2}$$

in which $\mathbf{a}_\mathrm{d} \in \mathbb{R}^3$ is the input signal. The objective now is to establish a control law for $\mathbf{a}_\mathrm{d}$ such that $\mathbf{p}(\mathrm{t}) \to \mathcal{C}(\mathrm{t})$ as $\mathrm{t} \to \infty$. Consider the following control law, [27]

$$\mathbf{a}_\mathrm{d} = \mathrm{J}_{\mathbf{\Phi}}\mathbf{v} + \mathrm{k}_\mathrm{v}(\mathbf{\Phi} - \mathbf{v}) + \frac{\partial \mathbf{\Phi}}{\partial \mathrm{t}}, \tag{4.3}$$

in which $\mathrm{J}_{\mathbf{\Phi}} \equiv \mathrm{J}_{\mathbf{\Phi}}(\mathbf{p}, \mathrm{t})$ is the Jacobian matrix of the field $\mathbf{\Phi}(\mathbf{p}, \mathrm{t})$ with respect to $\mathbf{p}$ and $\mathrm{k}_\mathrm{v} > 0$ is a gain. Now, since $\mathbf{\Phi} \in \mathrm{C}^2$, we have that $\mathbf{a}_\mathrm{d} \in \mathrm{C}^1$.

First, let us prove that the law in (4.3) guarantees $\lim_{\mathrm{t}\to\infty} \mathbf{v} = \mathbf{\Phi}$. Using the Lyapunov candidate function $\mathrm{V}_\mathrm{v} = (1/2)(\mathbf{\Phi} - \mathbf{v})^\mathrm{T}(\mathbf{\Phi} - \mathbf{v})$, it follows that

$$\dot{\mathrm{V}}_\mathrm{v} = (\mathbf{\Phi} - \mathbf{v})^\mathrm{T}(\dot{\mathbf{\Phi}} - \dot{\mathbf{v}}). \tag{4.4}$$

Replacing (4.2) and (4.3) into (4.4) and noting that $\dot{\mathbf{\Phi}} = \mathrm{J}_{\mathbf{\Phi}}\dot{\mathbf{p}} + \partial\mathbf{\Phi}/\partial\mathrm{t} = \mathrm{J}_{\mathbf{\Phi}}\mathbf{v} + \partial\mathbf{\Phi}/\partial\mathrm{t}$ we obtain

$$\dot{\mathrm{V}}_\mathrm{v} = (\mathbf{\Phi} - \mathbf{v})^\mathrm{T}(-\mathrm{k}_\mathrm{v}(\mathbf{\Phi} - \mathbf{v})) = -\mathrm{k}_\mathrm{v}\|\mathbf{\Phi} - \mathbf{v}\|^2 \leq 0, \tag{4.5}$$

with $\dot{\mathrm{V}}_\mathrm{v} = 0 \iff \mathbf{v} = \mathbf{\Phi}$. Thus, it follows that $\lim_{\mathrm{t}\to\infty} \|\mathbf{\Phi} - \mathbf{v}\| = 0$.

The system in (4.2), under the control law in (4.3), can be put in the form used in Lemma 4.7 of [37]. Let $\delta_\mathrm{v} = \mathbf{v} - \mathbf{\Phi}$. Then $\dot{\delta}_\mathrm{v} = \dot{\mathbf{v}} - \dot{\mathbf{\Phi}} = \mathbf{a}_\mathrm{d} - \mathrm{J}_{\mathbf{\Phi}}\mathbf{v} - \partial\mathbf{\Phi}/\partial\mathrm{t} = \mathrm{k}_\mathrm{v}(\mathbf{\Phi} - \mathbf{v}) = -\mathrm{k}_\mathrm{v}\delta_\mathrm{v}$. Let also $\dot{\mathbf{p}} = \mathbf{v} = \mathbf{\Phi} + \delta_\mathrm{v}$. In Section 3.2.3 we have shown that the system $\dot{\mathbf{p}} = \mathbf{\Phi} + \delta_\mathrm{v}$ is input-to-state stable. Now, we have shown that $\lim_{\mathrm{t}\to\infty} \delta_\mathrm{v} = 0$, since $\dot{\delta}_\mathrm{v} = -\mathrm{k}_\mathrm{v}\delta_\mathrm{v}$. Thus, Lemma 4.7 of [37] can be used to prove that $\mathbf{p}$, of the system (4.2), converges to $\mathcal{C}(\mathrm{t})$.

Similar to what was made in the previous Chapter, let us now insert a disturbance in equation (4.2), which will become $\dot{\mathbf{v}} = \mathbf{a}_\mathrm{d} + \delta_\mathrm{a}$. The disturbance δ_a is norm-bounded, *i.e.* $\|\delta_\mathrm{a}\| \leq \Delta_\mathrm{a}$.

Lemma 10. *Given a norm-bounded disturbance δ_a, $\|\delta_\mathrm{a}\| \leq \Delta_\mathrm{a}$, the error of the velocity* $\mathbf{v}$ *converges to the set*

$$\mathcal{I}_\mathrm{v}(\mathbf{p}, \mathrm{t}) = \left\{ \mathbf{v} \in \mathbb{R}^3 \mid \|\boldsymbol{\Phi}(\mathbf{p}) - \mathbf{v}\| < \frac{\Delta_\mathrm{a}}{\mathrm{k_v}} \right\}. \tag{4.6}$$

Proof. Assuming the perturbed system, the time derivative of the Lyapunov function $\mathrm{V_v}$ becomes

$$\dot{\mathrm{V}}_\mathrm{v} = -\mathrm{k_v}\|\boldsymbol{\Phi} - \mathbf{v}\|^2 + (\boldsymbol{\Phi} - \mathbf{v})^\mathrm{T}\delta_\mathrm{a}. \tag{4.7}$$

Using the bound $\|\delta_\mathrm{a}\| \leq \Delta_\mathrm{a}$, we have the following:

$$\dot{\mathrm{V}}_\mathrm{v} \leq -\mathrm{k_v}\|\boldsymbol{\Phi} - \mathbf{v}\|^2 + \|\boldsymbol{\Phi} - \mathbf{v}\|\Delta_\mathrm{a}. \tag{4.8}$$

From (4.8), we see that $\|\boldsymbol{\Phi} - \mathbf{v}\| > \Delta_\mathrm{a}/\mathrm{k_v} \implies \dot{\mathrm{V}}_\mathrm{v} < 0$. Note also that $\|\boldsymbol{\Phi} - \mathbf{v}\| = \Delta_\mathrm{a}/\mathrm{k_v}$ represents a closed level set of $\mathrm{V_v}$, since $\mathrm{V_v}$ is radially unbounded in $\mathbf{v}$ for a fixed $\mathbf{p}$. □

Lemma 11. *Given a norm-bounded disturbance δ_a, $\|\delta_\mathrm{a}\| \leq \Delta_\mathrm{a}$, the error of the position* $\mathbf{p}$*, from (4.2), converges to the invariant set (3.38) with $\Delta_\mathrm{v} = \Delta_\mathrm{a}/\mathrm{k_v}$.*

Proof. From Lemma 10, there exists an instant $\mathrm{t_a} > 0$ from which the error of velocity will remain below $\Delta_\mathrm{a}/\mathrm{k_v}$ for all $\mathrm{t} > \mathrm{t_a}$. From that instant, an upper bound of $\delta_\mathrm{v} = \mathbf{v} - \boldsymbol{\Phi}$ is $\Delta_\mathrm{v} = \Delta_\mathrm{a}/\mathrm{k_v}$, which can then be used with Proposition 3 when $\mathrm{t} > \mathrm{t_a}$. □

In conclusion, the system in (4.2) is also input-to-state stable when the input is an additional term δ_a in the dynamics of $\mathbf{v}$. Given a bounded $\|\delta_\mathrm{a}\| < \Delta_\mathrm{a}$, the bounded disturbances in $\mathbf{v}$ and $\mathbf{p}$ are given by Lemma 10 and Lemma 11, respectively.

In this section, the controller in (4.3) was used to impose the dynamics $\dot{\mathbf{p}} = \boldsymbol{\Phi}(\mathbf{p}, \mathrm{t})$ to a system described by the double integrator model in (4.2). In the coming section, the same idea will be used to impose the dynamics of the second-order integrator to the quadcopter model in (4.1).

4.2.2 Quadcopter control

Finally, the complete system in (4.1) can be considered. Now, the objective is to compute a thrust τ and an angular velocity ω so that the dynamics of $\mathbf{p}$, in (4.1), behaves

according to $\mathbf{\Phi}(\mathbf{p}, t)$. This will be possible by the imposition of the dynamics in (4.2) to the model in (4.1). For now, consider that $\delta_\tau = 0$, and $\delta_\omega = 0$.

Consider $\hat{\mathbf{z}}_b = R_b^w \hat{\mathbf{z}}$, as the z body axis of the vehicle written with respect to the inertial frame, *i.e.* the direction where the thrust acts on. From (4.1b) we have that

$$\hat{\mathbf{z}}_b \tau = m\dot{\mathbf{v}} + mg\hat{\mathbf{z}} - f_d. \tag{4.9}$$

In fact, the quadcopter is an underactuated vehicle [85], and equation (4.9) represents a nonholonomic constraint of second order. Basically, it shows that the total thrust only acts in one direction, specified by the vehicle's attitude. Consider now a desired acceleration $\mathbf{a}_d$, defined in (4.3), which we want to impose to the system in (4.1). It is clear that the dynamics of the double integrator ($\dot{\mathbf{v}} = \mathbf{a}_d$) in (4.2) can be instantaneously imposed to (4.1) if and only if $\mathbf{a}_d$ is such that $\hat{\mathbf{z}}_b \tau \parallel m\mathbf{a}_d + mg\hat{\mathbf{z}} - f_d$, where $\parallel$ indicates that one vector is parallel to the other.

The idea to solve this problem is to consider an inner control loop in order to align the vector $\hat{\mathbf{z}}_b$ with the vector $\mathbf{a}_r = \mathbf{a}_d + g\hat{\mathbf{z}} - f_d/m$. Thus, if $\hat{\mathbf{z}}_b \parallel \mathbf{a}_r$ the desired acceleration $\mathbf{a}_d$ can be imposed by selecting $\tau = m\|\mathbf{a}_r\|$.

In order to align $\hat{\mathbf{z}}_b$ with $\mathbf{a}_r$, consider a reference rotation matrix $R_r^w \equiv R_r^w(t)$. Then, the input ω will be used in the input of the inner loop to allow the state matrix R_b^w, the vehicle's attitude, to track the reference R_r^w. Formally, the objective is to make $R_b^w(t) \to R_r^w(t)$ as $t \to \infty$.

In order to respect constraint (4.9), with $\dot{\mathbf{v}} = \mathbf{a}_d$, the third column of R_r^w must be given by $\hat{z}_r = \mathbf{a}_r/\|\mathbf{a}_r\|$. Now, since the final objective is to make the drone converge to the curve $\mathcal{C}(t)$, its heading angle can be freely chosen. Thus, given a continuous reference of heading ψ_r, the vector $\hat{\mathbf{x}}_r$, first column of R_r^w, can be defined from a vector w_ψ that points in the direction with heading angle ψ_r. Let $w_\psi = [\cos(\psi_r)\ \sin(\psi_r)\ 0]^T$. The vector $\hat{\mathbf{x}}_r$ must be orthogonal to $\hat{\mathbf{z}}_b$ and have a unit norm, thus, we define

$$\hat{\mathbf{x}}_r = \frac{w_\psi - w_\psi^T \hat{\mathbf{z}}_b \hat{\mathbf{z}}_b}{\|w_\psi - w_\psi^T \hat{\mathbf{z}}_b \hat{\mathbf{z}}_b\|}. \tag{4.10}$$

In (4.10) the vector $\hat{\mathbf{x}}_r$ is computed by removing the component of w_ψ in the direction of $\hat{\mathbf{z}}_r$ and normalizing the result. Given the definitions of $\hat{\mathbf{z}}_r$ and $\hat{\mathbf{x}}_r$, the vector $\hat{\mathbf{y}}_r$ is computed according to $\hat{\mathbf{y}}_r = \hat{\mathbf{z}}_r \times \hat{\mathbf{x}}_r$. Finally, $R_r^w = [\hat{\mathbf{x}}_r\ \hat{\mathbf{y}}_r\ \hat{\mathbf{z}}_r]$. Note that $\mathbf{a}_r \in C^1 \implies \hat{\mathbf{z}}_r \in C^1$, and since ψ_r is continuous, $\hat{\mathbf{x}}_r \in C^1$. Given the definition of $\hat{\mathbf{y}}_r$, it belongs to class C^1 and then $R_r^w \in C^1$.

In the case that $R_b^w = R_r^w$, the control law for τ should be $\tau = m\|\mathbf{a}_r\|$. In this way, the double integrator dynamics is imposed with $\dot{\mathbf{v}} = \mathbf{a}_d$. However, when $R_b^w \neq R_r^w$, the law $\tau = m\|\mathbf{a}_r\|$ is not required. In this case, neither this simple law nor any other will be able to impose the dynamics of the double integrator instantaneously. Here, the law for τ will be chosen so that the norm of the difference between the reference acceleration, $\mathbf{a}_r$,

and the applied one, $\hat{\mathbf{z}}_\mathrm{b}\tau/\mathrm{m}$, is minimized. Thus, before the definition of a control law for ω, the control law for τ is defined as

$$\tau = \arg\min_{\tau'} \left\| \mathbf{a}_\mathrm{r} - \frac{\tau'}{\mathrm{m}}\hat{\mathbf{z}}_\mathrm{b} \right\|^2 = \mathrm{m}(\hat{\mathbf{z}}_\mathrm{b}^\mathrm{T}\mathbf{a}_\mathrm{r}). \tag{4.11}$$

Note that if $\hat{\mathbf{z}}_\mathrm{b} = \hat{\mathbf{z}}_\mathrm{r}$ we have $\tau = \mathrm{m}\|\mathbf{a}_\mathrm{r}\|$.

Let us now define the control law for ω. In order to do that, consider the error matrix $\mathrm{R_e}$, which represents the misalignment between the reference frame $\mathrm{R}_\mathrm{r}^\mathrm{w}$ and the body frame $\mathrm{R}_\mathrm{b}^\mathrm{w}$. Thus, we have $\mathrm{R_e} = (\mathrm{R}_\mathrm{b}^\mathrm{w})^\mathrm{T}\mathrm{R}_\mathrm{r}^\mathrm{w}$. The control law for ω must make $\lim_{\mathrm{t}\to\infty} \mathrm{R_e} = \mathrm{I}$. The time derivative of $\mathrm{R_e}$ is:

$$\dot{\mathrm{R}}_\mathrm{e} = (\dot{\mathrm{R}}_\mathrm{b}^\mathrm{w})^\mathrm{T}\mathrm{R}_\mathrm{r}^\mathrm{w} + (\mathrm{R}_\mathrm{b}^\mathrm{w})^\mathrm{T}\dot{\mathrm{R}}_\mathrm{r}^\mathrm{w}. \tag{4.12}$$

Equation (4.12) can be written as

$$\mathrm{S}(\omega_\mathrm{e})\mathrm{R_e} = (\mathrm{S}(\mathrm{R}_\mathrm{b}^\mathrm{w}\omega)\mathrm{R}_\mathrm{b}^\mathrm{w})^\mathrm{T}\mathrm{R}_\mathrm{r}^\mathrm{w} + (\mathrm{R}_\mathrm{b}^\mathrm{w})^\mathrm{T}\mathrm{S}(\mathrm{R}_\mathrm{b}^\mathrm{w}\omega_\mathrm{r})\mathrm{R}_\mathrm{r}^\mathrm{w}, \tag{4.13}$$

in which $\omega_\mathrm{r} \in \mathbb{R}^3$ is the rotation velocity of the reference rotation matrix $\mathrm{R}_\mathrm{r}^\mathrm{w}$, written with respect to the body frame. The velocity $\omega_\mathrm{e} \in \mathbb{R}^3$ is the rotation of the error matrix $\mathrm{R_e}$, also written in the body frame. In (4.13), note that $\mathrm{R}_\mathrm{b}^\mathrm{w}\mathrm{S}(\omega) = \mathrm{S}(\mathrm{R}_\mathrm{b}^\mathrm{w}\omega)\mathrm{R}_\mathrm{b}^\mathrm{w}$.

Now, let vectors $\mathbf{r}, \mathbf{s} \in \mathbb{R}^3$ and a matrix $\mathrm{R} \in \mathrm{SO}(3)$. Given the facts that $\mathrm{S}(\mathrm{R}\mathbf{r}) = \mathrm{R}\mathrm{S}(\mathbf{r})\mathrm{R}^\mathrm{T}$, $\mathrm{S}(\mathbf{r})^\mathrm{T} = -\mathrm{S}(\mathbf{r}) = \mathrm{S}(-\mathbf{r})$, $\mathrm{S}(\mathbf{r}) + \mathrm{S}(\mathbf{s}) = \mathrm{S}(\mathbf{r}+\mathbf{s})$, and $\mathrm{R_e} = (\mathrm{R}_\mathrm{b}^\mathrm{w})^\mathrm{T}\mathrm{R}_\mathrm{r}^\mathrm{w}$, after applying some algebraic manipulation in (4.13), we obtain the relation $\mathrm{S}(\omega_\mathrm{e})\mathrm{R_e} = \mathrm{S}(-\omega + \omega_\mathrm{r})\mathrm{R_e}$, thus:

$$\omega_\mathrm{e} = \omega_\mathrm{r} - \omega. \tag{4.14}$$

Now, it is necessary to compute ω_r. We know that $\dot{\mathrm{R}}_\mathrm{r}^\mathrm{w} = \mathrm{S}(\mathrm{R}_\mathrm{b}^\mathrm{w}\omega_\mathrm{r})\mathrm{R}_\mathrm{r}^\mathrm{w}$. Using the fact $\mathrm{S}(\mathrm{R}\mathbf{r}) = \mathrm{R}\mathrm{S}(\mathbf{r})\mathrm{R}^\mathrm{T}$ we obtain:

$$\mathrm{S}(\omega_\mathrm{r}) = (\mathrm{R}_\mathrm{b}^\mathrm{w})^\mathrm{T}\dot{\mathrm{R}}_\mathrm{r}^\mathrm{w}(\mathrm{R_e})^\mathrm{T}. \tag{4.15}$$

Thus, if we compute the matrix $\dot{\mathrm{R}}_\mathrm{r}^\mathrm{w}$ numerically, ω_r can be obtained from equation (4.15). Note that, since $\mathrm{R}_\mathrm{r}^\mathrm{w} \in \mathrm{C}^1$, it is differentiable and therefore we can obtain ω_r.

Now, let us project a law for ω_e so that $\lim_{\mathrm{t}\to\infty} \mathrm{R_e} = \mathrm{I}$. As stated by the Euler's rotation theorem, any orientation displacement can be expressed as a single rotation of an angle β around a given axis, indicated by a unit vector $\hat{\mathbf{n}}$. Thus, consider the following map:

$$\mathrm{R_e} \mapsto (\hat{\mathbf{n}}, \beta). \tag{4.16}$$

The following lemma establishes the influence of an angular velocity on the angle parameter β.

Lemma 12. *Consider the map* $\mathrm{R} \mapsto (\hat{\mathbf{n}}, \beta)$. *If an angular velocity* $\mathbf{r} \in \mathbb{R}^3$ *is applied to a frame represented by the matrix* R *(such that* $\dot{\mathrm{R}} = \mathrm{S}(\mathbf{r})\mathrm{R}$*), the following relation holds:*

$$\dot{\beta} = \hat{\mathbf{n}}^\mathrm{T}\mathbf{r}. \tag{4.17}$$

Proof. Consider a quaternion [22] $q = [q_w,\ q_x,\ q_y,\ q_z]$ equivalent to the rotation matrix R. Given the representation $(\hat{\mathbf{n}}, \beta)$, the element q_w of the quaternion can be written as $q_w = \cos(\beta/2)$. Taking the time derivative of q_w we obtain (i) $\dot{q}_w = -(1/2)\sin(\beta/2)\,\dot{\beta}$. Now, if an angular velocity $\mathbf{r}$ is applied to the quaternion q, we know, by the rule of quaternion derivative, that the time derivative of q_w is given by (ii) $\dot{q}_w = -(1/2)\sin(\beta/2)\,\hat{\mathbf{n}}^T\mathbf{r}$. Comparing (i) and (ii) we obtain (4.17). □

We need that $\lim_{t\to\infty} R_e = I$, which occurs if and only if $\beta \to 0$. Consider $\omega_e = -k_\beta \sin(\beta)\hat{\mathbf{n}}$, with $k_\beta > 0$. From (4.14), the control law for ω will then be defined as

$$\omega = \omega_r + k_\beta \sin(\beta)\hat{\mathbf{n}}. \tag{4.18}$$

Intuitively, the first term in (4.18) is a feedforward that compensates for changes on the reference rotation matrix R_r^w. The second term acts to decrease the angle error β. Seeking a formal proof, we establish the following Lemma.

Lemma 13. *Given the control law in (4.18),* $\lim_{t\to\infty} R_e = I$.

Proof. Consider the Lyapunov candidate function $V_\beta = 1 - \cos(\beta)$, as in [64]. Note that $V_\beta(\beta) \geq 0$ and $V_\beta(0) = 0$. The time derivative of V_β is

$$\dot{V}_\beta = \sin(\beta)\dot{\beta}. \tag{4.19}$$

Using the result in Lemma 12 and equation (4.14), we have that

$$\dot{V}_\beta = \sin(\beta)\hat{\mathbf{n}}^T\omega_e = \sin(\beta)\hat{\mathbf{n}}^T(\omega_r - \omega). \tag{4.20}$$

Replacing the control law in (4.18) into (4.20), we obtain

$$\dot{V}_\beta = \sin(\beta)\hat{\mathbf{n}}^T(-k_\beta \sin(\beta)\hat{\mathbf{n}}) = -k_\beta \sin(\beta)^2 \leq 0. \tag{4.21}$$

We have that $-\pi < \beta < \pi \implies \dot{V}_\beta \leq 0$ with $\dot{V}_\beta = 0 \iff \beta = 0$. If $\beta \sim \pi$ we also have $\dot{V}_\beta = 0$. However, this corresponds to an unstable equilibrium, since $\beta \sim \pi$ is the maximizer of $V_\beta(\beta)$ and $\dot{V}_\beta(\beta + \epsilon) < 0$ for any small ϵ. Thus, we can conclude that $\lim_{t\to\infty} \beta = 0$, which corresponds to $R_e \to I$. □

Note that the law in (4.18) is invariant to shifts of 2π in β. The price we pay is the existence of a fixed point in $\beta \sim \pi$. In practice, this is not a big problem, since this equilibrium is unstable, [64]. Note also that when $\beta \sim 0$ or $\beta \sim \pi$ the vector $\hat{\mathbf{n}}$ is not defined. This is not a problem because in both cases we have $\sin(0) = \sin(\pi) = 0$, thus the control law is well defined and, in this case, given by $\omega = \omega_r$.

Proposition 4. *The state* $\mathbf{p}$ *of the system in (4.1), under the control law given by (4.11) and (4.18), converges asymptotically to the curve* $\mathcal{C}(t)$.

Proof. From Lemma 13 we know that the control law in (4.18) with ω_r from (4.15) ensures asymptotic convergence of R_b^w to R_r^w. Also, according to (4.11), as $R_b^w \to R_r^w$ we have that $m\mathbf{a}_r \to \tau\hat{\mathbf{z}}_b$ and consequently $\dot{\mathbf{v}} \to \mathbf{a}_d$. Meaning that, given $\delta_a = \dot{\mathbf{v}} - \mathbf{a}_d$, there exists $t_a > 0$ such that $\|\delta_a\| < \epsilon \ \forall \ t > t_a$, with $\epsilon > 0$ as small as we want, *i.e* the dynamics of the controlled double integrator in (4.2) will be imposed to the quadcopter. When the double integrator has the input $\mathbf{a}_d$, defined in (4.3), we know, from Lemma 10 that its state $\mathbf{p}$ converges to $\mathcal{C}$. A direct consequence of Lemma 11 is that the double integrator is input-to-state stable when the amount δ_a is considered as an input. If $\|\delta_a\| \to 0$, Lemma 4.7 of [37] ensures that the state $\mathbf{p}$ of the vehicle also goes to $\mathcal{C}(t)$. □

Proposition 4 finally states that the control laws in (4.11) and (4.18) are the solution to Problem 2.

4.3 Disturbance on control inputs

Finally, we now consider the influence of the disturbances δ_τ and δ_ω on the controlled system. Note that in the previous sections we considered some sort of "virtual disturbances" on velocity, δ_v, and on acceleration, δ_a. However, these disturbances were only considered with the purpose of proving the convergence of the undisturbed system by showing that each individual layer is ISS. In this section, we use those results to prove that disturbances on the control inputs originate bounded disturbances on the position of the robot.

Consider now the following lemma, which shows that a bounded disturbance on the angular velocity causes a bounded disturbance on the attitude error.

Lemma 14. *If the disturbance δ_ω is considered, with $\|\delta_\omega\| \leq \Delta_\omega$, the angle β associated with the attitude error will converge to the invariant set*

$$\mathcal{I}_\beta = \left\{ \beta \in \mathbb{R} \mid \beta < \Delta_\beta(\Delta_\omega) \equiv \operatorname{asin}\left(\frac{\Delta_\omega}{k_\beta}\right) \right\}, \tag{4.22}$$

in which $\Delta_\beta \equiv \Delta_\beta(\Delta_\omega)$ is the ultimate bound for β.

Proof. Consider the Lyapunov function $V_\beta = 1 - \cos(\beta)$, as in Lemma 13. If we assume $\delta_\omega \neq 0$, $\dot{V}_\beta$ in (4.21) becomes

$$\dot{V}_\beta = -k_\beta \sin(\beta)^2 + \sin(\beta)\hat{\mathbf{n}}^T\delta_\omega. \tag{4.23}$$

Given that $\|\hat{\mathbf{n}}\| = 1$ and $\|\delta_\omega\| < \Delta_\omega$ we have that $\|\hat{\mathbf{n}}^{\mathrm{T}}\delta_\omega\| \leq \Delta_\omega$. Thus

$$\beta > \operatorname{asin}\left(\frac{\Delta_\omega}{\mathrm{k}_\beta}\right) \implies \dot{\mathrm{V}}_\beta < 0. \tag{4.24}$$

Since $\dot{\mathrm{V}}_\beta < 0 \;\forall\; \beta \notin \mathcal{I}_\beta$, the set $\mathcal{I}_\beta$ is positively invariant. □

Note that, given the result in (4.24) the gain k_β must be such that $\mathrm{k}_\beta > \Delta_\omega$, as in [64].

Let us now analyze the influence that the attitude error Δ_β and the error δ_τ have on the vehicle's acceleration. Comparing the right side of (4.1b) in the ideal case $\mathrm{R}_\mathrm{b}^\mathrm{w} = \mathrm{R}_\mathrm{r}^\mathrm{w}$, $\delta_\tau = 0$ with the disturbed case $\mathrm{R}_\mathrm{b}^\mathrm{w} \neq \mathrm{R}_\mathrm{r}^\mathrm{w}$, $\delta_\tau \neq 0$, the error on the vehicle's acceleration can be written as

$$\delta_\mathrm{a} = \left\| \frac{\tau}{\mathrm{m}}\hat{\mathbf{z}}_\mathrm{r} - \frac{\tau}{\mathrm{m}}\hat{\mathbf{z}}_\mathrm{b} - \frac{\delta_\tau}{\mathrm{m}} \right\| \leq \frac{\tau}{\mathrm{m}}\|\hat{\mathbf{z}}_\mathrm{r} - \hat{\mathbf{z}}_\mathrm{b}\| + \left\|\frac{\delta_\tau}{\mathrm{m}}\right\| \leq \|\mathbf{a}_\mathrm{r}\|\,\|\hat{\mathbf{z}}_\mathrm{r} - \hat{\mathbf{z}}_\mathrm{b}\| + \frac{\Delta_\tau}{\mathrm{m}}, \tag{4.25}$$

in which the inequality is obtained by (i) triangular inequality and (ii) making $\tau = \mathrm{m}\|\mathbf{a}_\mathrm{r}\|$. It holds because, in fact, τ is computed accordingly to (4.11), and it is the value that minimizes the norm of $\hat{\mathbf{z}}_\mathrm{r} - \hat{\mathbf{z}}_\mathrm{b}$.

Since the vectors $\hat{\mathbf{z}}_\mathrm{r}$ and $\hat{\mathbf{z}}_\mathrm{b}$ have a unit norm, the law of cosines implies that $\|\hat{\mathbf{z}}_\mathrm{r} - \hat{\mathbf{z}}_\mathrm{b}\| = \sqrt{2\left(1 - \cos(\Delta_\beta)\right)}$, which is the length of the chord with angle Δ_β in a circle with unity radius. Using the fact that the arc with angle Δ_β is no shorter than the chord with the same angle, it is true that $\|\hat{\mathbf{z}}_\mathrm{r} - \hat{\mathbf{z}}_\mathrm{b}\| \leq \Delta_\beta$. Using this result in (4.25) and considering that the thrust force is limited by a maximum τ_max, *i.e.* $\|\mathrm{a}_\mathrm{r}\| \leq \tau_\mathrm{max}/\mathrm{m}$, an upper bound for the acceleration error is

$$\Delta_\mathrm{a} = \frac{\tau_\mathrm{max}}{\mathrm{m}}\Delta_\beta + \frac{\Delta_\tau}{\mathrm{m}}. \tag{4.26}$$

The first term in (4.26) is an acceleration error due to an attitude error. The second one is due to the thrust error directly. Now, using the value of Δ_β from (4.22) we obtain

$$\Delta_\mathrm{a} = \frac{\tau_\mathrm{max}}{\mathrm{m}}\operatorname{asin}\left(\frac{\Delta_\omega}{\mathrm{k}_\beta}\right) + \frac{\Delta_\tau}{\mathrm{m}}. \tag{4.27}$$

In order to state our final result, let us define functions to compute the ultimate bounds defined until now. Let $\gamma_1(\Delta_\mathrm{v})$, $\gamma_2(\Delta_\mathrm{a})$ and $\gamma_3(\Delta_\tau, \Delta_\omega)$ be the functions that return the ultimate bounds defined in (3.38), (4.6) and (4.27), respectively. They are defined as:

$$\begin{aligned} \gamma_1(\Delta_\mathrm{v}) &= \mathrm{G}^{-1}\left(\frac{\Delta_\mathrm{v}}{\mathrm{v_r} - \mathrm{v_m}}\right), \\ \gamma_2(\Delta_\mathrm{a}) &= \frac{\Delta_\mathrm{a}}{\mathrm{k_v}}, \\ \gamma_3(\Delta_\tau, \Delta_\omega) &= \frac{\tau_\mathrm{max}}{\mathrm{m}}\operatorname{asin}\left(\frac{\Delta_\omega}{\mathrm{k}_\beta}\right) + \frac{\Delta_\tau}{\mathrm{m}}. \end{aligned} \tag{4.28}$$

Proposition 5. *If the uncertainties δ_τ and δ_ω are present, with $\|\delta_\tau\| \leq \Delta_\tau$, $\|\delta_\omega\| \leq \Delta_\omega$, the position state of the quadcopter will converge to the invariant set given by:*

$$\mathcal{I}_{\mathrm{D}}(\mathrm{t}) = \left\{ \mathbf{p} \in \mathbb{R}^3 \mid \mathrm{D}(\mathbf{p}, \mathrm{t}) < \gamma_1 \circ \gamma_2 \circ \gamma_3(\Delta_\tau, \Delta_\omega) \right\}. \tag{4.29}$$

Proof. Using the function $\gamma_3(\Delta_\tau, \Delta_\omega)$, we obtain an upper bound for the quadcopter's acceleration error given the control disturbances. This value is then used in function $\gamma_2(\Delta_\mathrm{a})$, according to Lemma 10, to obtain an upper bound to the vehicle's velocity error. Finally, the velocity error Δ_v is used in function $\gamma_1(\Delta_\mathrm{v})$, according to Proposition 3, to obtain an upper bound for the function D, which is a measurement of the vehicle's distance to the curve. □

Proposition 5 presents a solution to Problem 3.

Chapter 5

Results

The first content of this chapter is a brief description of how the methodologies proposed in Chapters 3 and 4 can be implemented. Next, we present some simulations and experiments that validate the theory. The experiments were performed with the robots described in Section 1.2.

5.1 Implementation methods

In order to apply the developed vector field methodology to a given parametric representation $\mathbf{r}(\mathrm{s},\mathrm{t})$ of a curve $\mathcal{C}(\mathrm{t})$, the optimization problem in (3.1) must be solved, *i.e.* $\mathrm{s}^*(\mathbf{x},\mathrm{t})$ must be found. A two-stage strategy is proposed for solving this problem. In the first stage, M equally-spaced samples are computed in the domain of s, *i.e.*, $\mathrm{s_k}$, $\mathrm{k} = 1,2,...,\mathrm{M}$ such that $\mathrm{s_{k+1}} = \mathrm{s_k} + \Delta\mathrm{s}$. By iterating over $\mathrm{s_k}$, the k^* that minimizes $\|\mathbf{x} - \mathbf{r}(\mathrm{s_{k^*}},\mathrm{t})\|$ is selected. In the second stage, a golden search method is used, which has a logarithmic complexity [61]. It assumes the domain $\mathrm{s_{k^*}} - \Delta\mathrm{s} \leq \mathrm{s} \leq \mathrm{s_{k^*}} + \Delta\mathrm{s}$ and finally obtains $\mathrm{s}^*(\mathbf{x},\mathrm{t})$. The first stage solves the problem of possible local minima, finding the region where the global minimum of (3.1) is. The second stage refines the solution of the first stage assuming a limited domain of s, in which the problem in (3.1) is assumed to be convex. This two-stage method gives a low computation cost to obtain the distance function with sufficient precision.

The tangent vector $\mathbf{T}(\mathrm{s}^*,\mathrm{t})$ is easily computed with equation (3.5). An option is to compute it numerically as $\mathbf{T}(\mathrm{s}^*,\mathrm{t}) \approx \left[\mathbf{r}(\mathrm{s}^*+\delta\mathrm{s},\mathrm{t}) - \mathbf{r}(\mathrm{s}^*,\mathrm{t})\right]/\delta\mathrm{s}$, for a sufficiently small $\delta\mathrm{s} > 0$. If s is not the arc length (most common case), we still need to normalize $\mathbf{T}(\mathrm{s}^*,\mathrm{t})$.

The computation of the time feedforward term requires the computation of $\frac{\partial \mathbf{D}}{\partial \mathrm{t}}$. In order to compute it numerically, the optimization problem in (3.1) is solved once more, now considering the curve at a time $\mathrm{t}+\delta\mathrm{t}$. Thus, given a small positive $\delta\mathrm{t}$ we can compute $\frac{\partial \mathbf{D}}{\partial \mathrm{t}} = -\frac{\partial \mathbf{r}}{\partial \mathrm{t}} \approx \left[\mathbf{r}(\mathrm{s}^*(\mathbf{x},\mathrm{t}),\mathrm{t}) - \mathbf{r}(\mathrm{s}^*(\mathbf{x},\mathrm{t}+\delta\mathrm{t}),\mathrm{t}+\delta\mathrm{t})\right]/\delta\mathrm{t}$.

If the curve $\mathcal{C}(\mathrm{t})$ is represented by a sequence of points, the necessary vectors can

also be computed. Basically, we can use only the first stage described before, since the second stage requires the function $\mathbf{r}(s, t)$. The derivatives associated with the computation of $\mathbf{T}$ and $\frac{\partial \mathbf{D}}{\partial t}$ can be estimated from the closest point on $\mathcal{C}(t)$ and its neighbors. If the sequence of points is too sparse, a simple interpolation can be used to obtain a function $\mathbf{r}(s, t)$, for instance, the one considered in [4].

In order to implement the quadcopter control method described in Chapter 4, besides the computation of the vector field, we need to compute the control laws for τ and ω. Assuming a function that returns the vector field, the Jacobian matrix $J_{\mathbf{\Phi}}$, the compensation $\frac{\partial \mathbf{\Phi}}{\partial t}$, and the matrix $\dot{R}_r^w$ can all be computed numerically. If the quadcopter's path is represented by a sequence of points, it is fundamental to obtain an analytic representation for the curve, with polynomial interpolation for example. This is due to the higher-order of the derivatives computed numerically. In our quadcopter examples, we only considered curves defined by parametric equations.

In Section 3.3.3 we commented that the field $\mathbf{\Phi}_v$ is based on a local minimum of the distance to the curve. The parameter s_v corresponds to a local optimum. When the implementation of the vector field departs from a curve represented by a sequence of points, there is a simple strategy to implement this feature. Instead of looking for the closest point in the entire curve (whole set of points), this search is performed only in the vicinity of the current closest point. Considering a cyclic parametrization, this optimization can be represented mathematically as

$$s_v[k+1] = \arg \min_{s_v[k]-\Delta s \leq s' \leq s_v[k]+\Delta s} \|\mathbf{x} - \mathbf{r}(s')\|^2, \tag{5.1}$$

for a $\Delta s > 0$. Again, we emphasize that this methodology is not formal. A better mathematical formulation of this idea is a topic for future research.

5.2 Quadcopter results

In the following, we present simulations and real robot experiments that validate the theory presented in Chapters 3 and 4.

5.2.1 Quadcopter Matlab simulation

In order to exemplify how the vector field can be used to control a quadcopter, we present an example of a vehicle following a vector field by using the control proposed in Chapter 4. In this simulation, the model in (4.1) was simulated. We considered $m = 0.5$ kg, $g = 9.81$ m/s^2, and $f_d(\mathbf{v}) = -k_d\mathbf{v}$, with $k_d = 0.05$ Ns/m. In order to observe the asymptotic convergence in a theoretical scenario, no disturbance was included, thus, $\delta_\tau = \mathbf{0}$ and $\delta_\omega = \mathbf{0}$. We considered $G = (2/\pi)\,\mathrm{atan}(k_f D)$ and the parameters of the controller were set as: $v_r = 4$ m/s, $k_f = 1$ m^{-1}, $k_v = 2$ s^{-1} and $k_\beta = 5$ rad/s. The considered curve has a sort of "knot", and it would be difficult to represent it with functions α_1, and α_2, as required by [27]. For $0 \leq s \leq 2\pi$, the parametric equation that describes this curve is

$$\mathbf{r}(s,t) = \mathbf{r}(s) = \begin{bmatrix} \sin(s) + 2\sin(2s) \\ \cos(s) - 2\cos(2s) \\ -\sin(3s) \end{bmatrix}. \tag{5.2}$$

Figure 5.1 presents the trajectory performed by the quadcopter in four instants of the simulation.

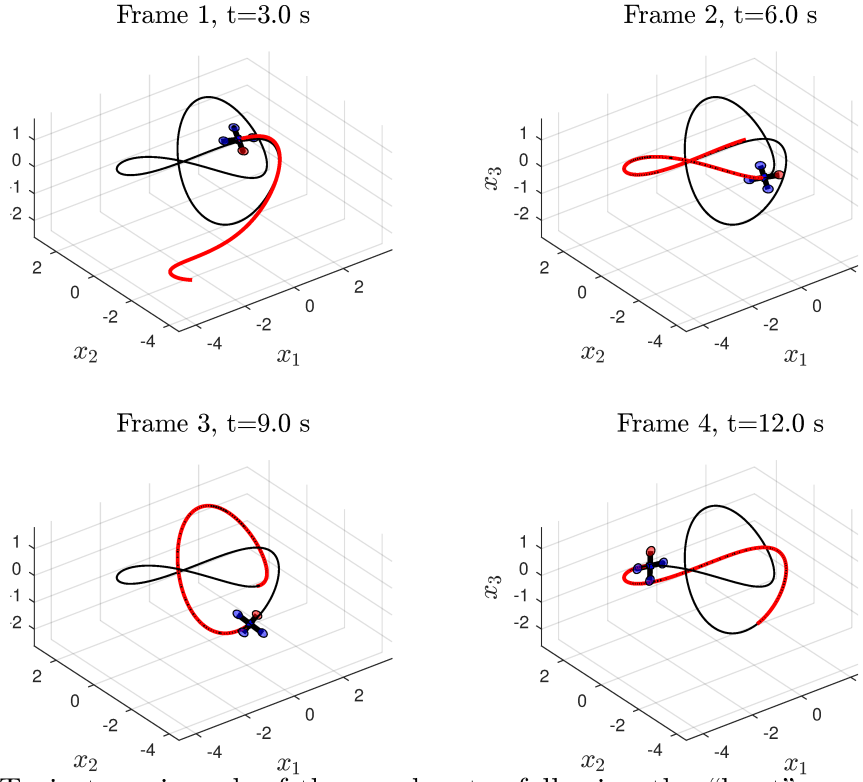

Figure 5.1: Trajectory, in red, of the quadcopter following the "knot" curve in four different instants. No uncertainty was added to the simulation.

.

In Figure 5.2 the error functions are presented. The distance function D is in blue, the velocity error $\|\mathbf{\Phi} - \mathbf{v}\|$ is in green and the angle error β in red. The error with the

fastest dynamics is the angle β. After β goes to 0 the velocity $\mathbf{v}$ starts to follow $\mathbf{\Phi}$. Only then, we observe that D goes to 0. In the bottom, we observe the control signals for thrust (black), and angular velocities (red, green, and blue). Note that the distance function increases in the beginning. This is due to the initial condition of the higher-order system, and after this transient, D goes to 0.

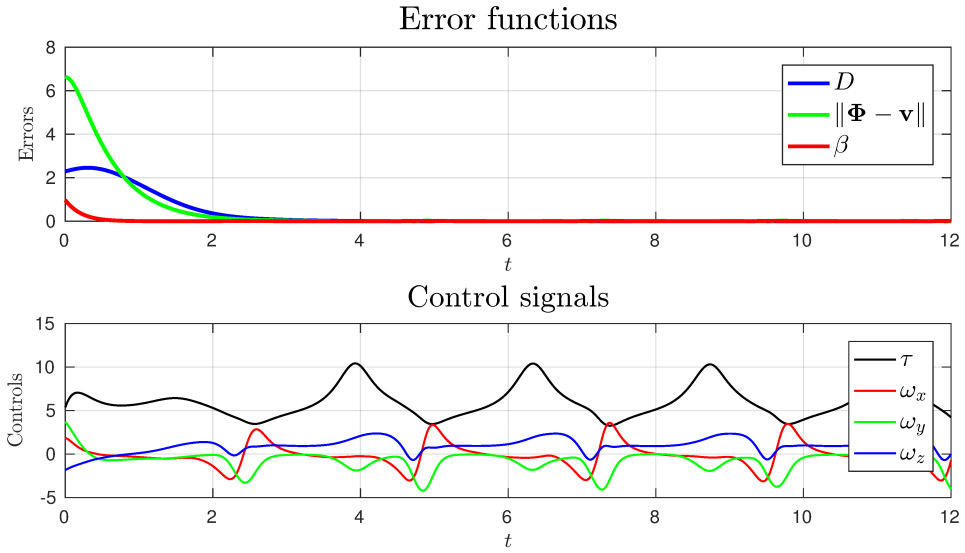

Figure 5.2: In the top are the error functions associated with the trajectory in Figure 5.1. Units for D, $\|\mathbf{\Phi} - \mathbf{v}\|$ and β in m, m/s and rad, respectively. In the bottom are the computed control signals. Units in N for τ and in rad/s for ω_{x}, ω_{y} and ω_{z}.

A video with Matlab animations of simulations of drone following paths is available at https://youtu.be/neDOD-CeBBU.

5.2.2 Obstacle deviation simulation

In order to exemplify the theory presented in Section 3.4, along with the controller in Chapter 4, we used the library vectorfield_stack presented in Section 1.5 to perform a simulation in which a drone follows a curve while deviating from spherical obstacles. The considered curve is described by the following parametric equation

$$\mathbf{r}(\mathrm{s},\mathrm{t}) \equiv \mathbf{r}(\mathrm{s}) = \begin{bmatrix} 2\cos(\mathrm{s}) \\ 2\sin(\mathrm{s}) \\ 0.2\cos(6\mathrm{s}) \end{bmatrix}, \tag{5.3}$$

in which $0 \leq \mathrm{s} \leq 2\pi$. In the simulation, we considered a drone with a mass $\mathrm{m} = 1$ kg. The controller gains were empirically set as: $\mathrm{v_r} = 0.5$ m/s, $\mathrm{k_f} = 4.0$ m^{-1}, $\mathrm{k_v} = 4.0$ s^{-1} and $\mathrm{k}_\beta = 8.0$ rad/s. The parameters associated to the obstacle avoidance are: $\lambda = 0.4$ m, $\mathrm{D_{in}} = 0.55$ m and $\mathrm{D_{in}^0} = 0.7$ m.

Figure 5.3 shows the curve $\mathcal{C}$ in green and the obstacles in red. The drone's trajectory is shown in gray. After the simulation starts, the drone deviates from the first obstacle. After the drone has already approached the curve, the trajectory deviates from two obstacles. In both cases, the deviated trajectory goes in the direction that is closer to the original vector field. One of the obstacles is contoured from below, while the other from above. The white dot is the closest point in the obstacle set $\mathcal{O}$, which is used to compute the contouring field. Evidently, when the robot is already at the curve and starts to contour an obstacle, the distance to the curve increases. However after the obstacle is circulated, the distance to the curve decreases again.

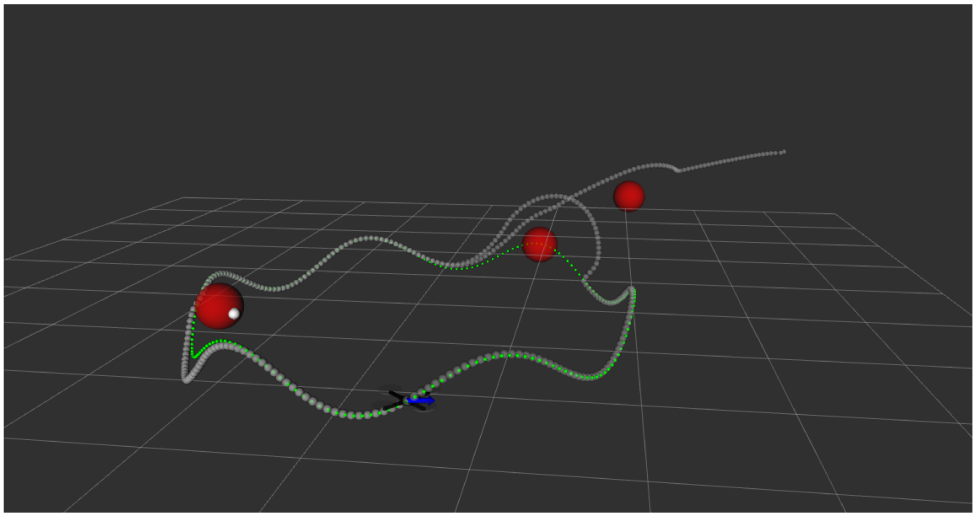

Figure 5.3: Simulation result of a drone following a vector field while deviating from obstacles.

Figure 5.4 shows the distance (to the curve $\mathcal{C}$) function D in blue and the distance (to the obstacle set $\mathcal{O}$) function $\mathrm{D_o}$ in red. Initially, D is approximately 3.8 m, and rapidly decreases to almost 0. When $\mathrm{D_o}$ becomes smaller than $\mathrm{D}_{\mathrm{in}}^{0}$ (yellow line), and the quadcopter starts to contour the obstacle, the distance D increases. However, as soon as the robot returns to follow the original vector field, the distance D decreases again. When the obstacle is being contoured, we observe that $\mathrm{D_o}$ remains approximately constant at the value of $\mathrm{D_o} = \lambda$, the green line. Note also that $\mathrm{D_o}$ approaches λ from above, thus, if λ is set properly, the drone will never get sufficiently close to the obstacle so that a collision occurs.

A video with several simulations using the library vectorfield_stack is available at https://youtu.be/9gtiL80KUsc.

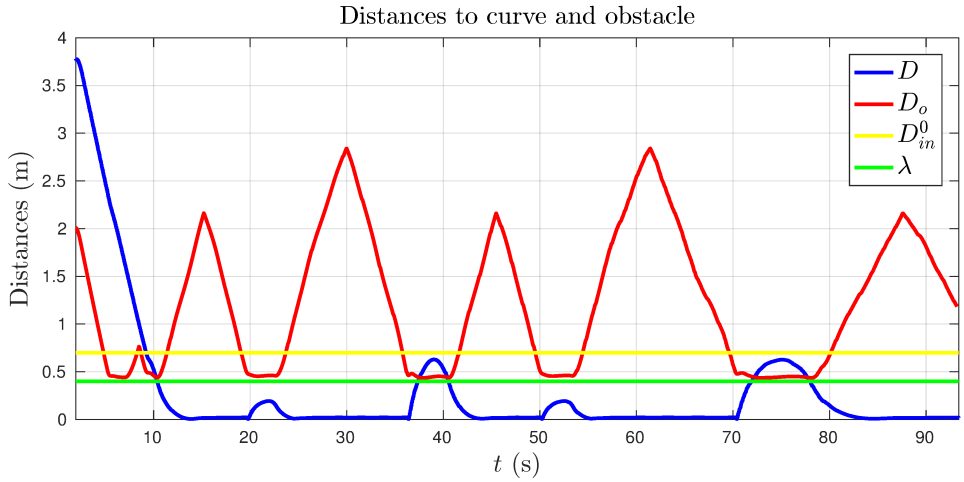

Figure 5.4: Distance function in the drone simulation using the vectorfield_stack library. The distance to the curve (D) is in blue and the distance to the obstacle set (D_o) is in red. The yellow and green lines represent D^0_{in} and λ, respectively.

5.2.3 DJI experiments

In order to exemplify the control strategy, presented in Chapter 4, in a real scenario, two experiments with the quadcopter described in Section 1.2.1 were performed. The controller was implemented in C++ and run under the ROS environment. In all results, the function G(D) was chosen as to $G = (2/\pi)\,\mathrm{atan}(k_f D)$. In these experiments, the mass of the DJI drone was $m = 2.88$ kg. Given the choice of the function G(D), it is possible to use the result of Proposition 3 to conclude the following ultimate bound for the distance to the target curve

$$\mathcal{I}(t) = \left\{ \mathbf{x} \in \mathbb{R}^n \mid D(\mathbf{x}, t) \leq \frac{1}{k_f} \tan\left(\frac{\pi}{2} \frac{\Delta_v}{v_r - v_m} \right) \right\}. \tag{5.4}$$

The DJI drone does not have the Acro rate mode of operation. Thus we used the Stabilized mode of operation. This mode receives as inputs the total thrust, the roll and pitch angles, and the yaw velocity. The controls τ and ω_z, computed in Chapter 4, could be assigned to the drone. The controls for the pitch and roll angles were extracted from the matrix R^w_r. In consequence, in the DJI experiments the controller gain k_β takes effect only on the yaw dynamics.

In the actual robot experiments, the parametric equation of the curve was used, thus, we employed the two-stage algorithm described in Section 5.1. The two experiments are presented in the following. A video with the recording of these experiments is available at https://youtu.be/VkexVe31HD4.

5.2.3.1 Eight curve

An static 8 like curve was considered to be followed by the quadcopter. The parametric equation that describes this curve is

$$\mathbf{r}(s,t) \equiv \mathbf{r}(s) = \begin{bmatrix} c_1 \cos(s) \\ c_2 \sin(2s) \\ h_0 + c_3 \sin(s) \end{bmatrix}, \tag{5.5}$$

in which $0 \leq s \leq 2\pi$, $c_1 = 6$ m, $c_2 = 3$ m, $c_3 = 1$ m and $h_0 = 5$ m. The robot velocity was set to $v_r = 1.2$ m/s. The gain of the function G(D) was set to $k_f = 2.0$ m^{-1}, while $k_v = 1.0$ s^{-1}.

Figure 5.5 shows in red the trajectory executed by the robot during 95 seconds. The initial condition was $\mathbf{x}(0) = [-8.2,\ -16.1,\ 4.7]^T$, in meters. Since the curve is static, $v_m = 0$m/s. Due to practical issues such as measurement errors and robot internal dynamics, $\Delta_v > 0$ and the trajectory of the system does not converge exactly to 0. By analyzing the norm of the difference between the commanded speed $\mathbf{\Phi}$ and the performed one $\mathbf{v}$, it was possible to estimate $\Delta_v = 0.81$m/s. Using equation (5.4), it follows that $\mathcal{I} = \{\mathbf{x} \in \mathbb{R}^3 \mid D(\mathbf{x}) \leq 0.89 \text{ m}\}$. The invariant set $\mathcal{I}$ is illustrated in Figure 5.5 by the blue tube around the curve $\mathbf{r}(s)$, in black. The blue tube does not intersect itself, thus, we have a guarantee that, even in the case of uncertainties, when the drone is passing through the "center of the eight" the closest point will not jump to the other part of the curve.

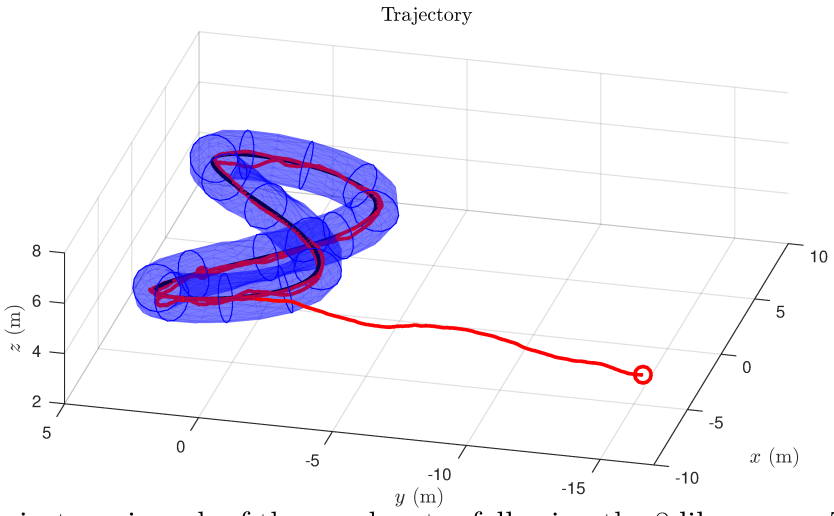

Figure 5.5: Trajectory, in red, of the quadcopter following the 8 like curve. The blue tube represents the positive invariant set $\mathcal{I}$.

In Figure 5.6, the evolution of the distance function D is presented. The ultimate bound for D is represented by the red line. Note that once D is below the red line it does not go back above. In this experiment the error after the convergence was kept below

0.45m, which is in the set $\mathcal{I}$. Figure 5.7 shows the components of the velocity $\mathbf{\Phi}$ in red and the ones executed by the vehicle in blue. Note how the controller was able to track the reference. Note also how noisy are the signals, especially in the z direction. Much of the noise can be justified by uncertainties in the localization system, usually higher in outdoor environments.

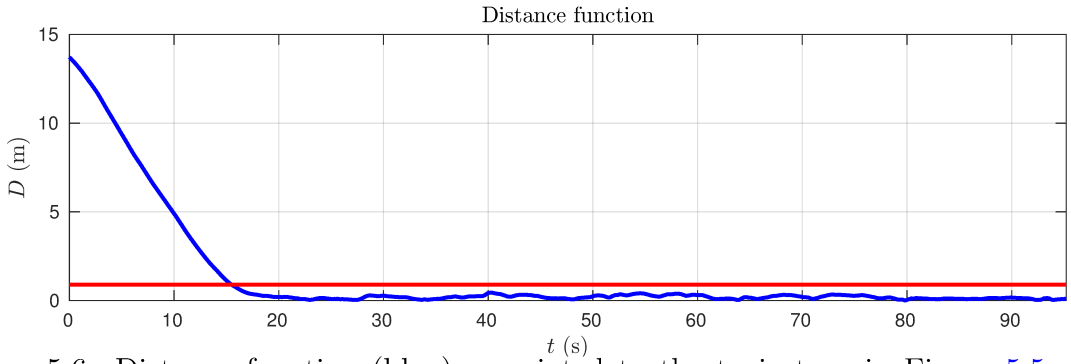

Figure 5.6: Distance function (blue) associated to the trajectory in Figure 5.5 and its ultimate bound (red).

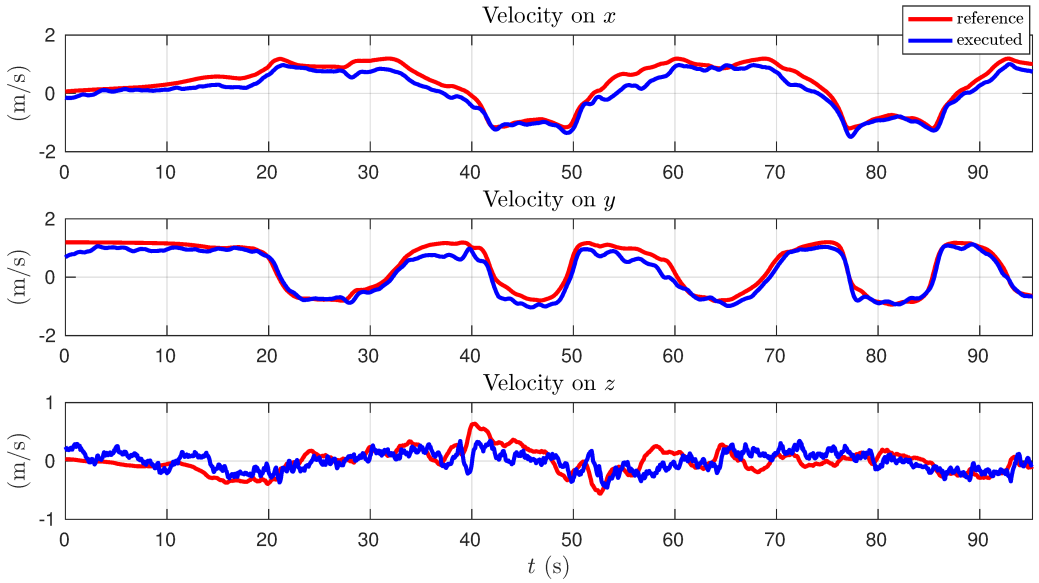

Figure 5.7: Three components of the reference (red) and executed (blue) velocities associated with the trajectory in Figure 5.5.

5.2.3.2 Deforming saddle curve

In order to demonstrate the ability of the strategy to deal with time-varying curves, a saddle-like curve that changes in shape while rotates around the z axis was considered.

The parametric equation that describes this curve is

$$\mathbf{r}(\mathrm{s},\mathrm{t}) = \mathrm{R_z}(\omega_1 \mathrm{t}) \begin{bmatrix} \mathrm{c}_1 \cos(\mathrm{s}) \\ \mathrm{c}_2 \sin(\mathrm{s}) \\ \mathrm{h}_0 + \mathrm{c}_3 \cos(\omega_2 \mathrm{t}) \cos(\mathrm{s})^2 \end{bmatrix}, \tag{5.6}$$

in which $0 \leq \mathrm{s} \leq 2\pi$ and $\mathrm{R_z}(\omega_1 \mathrm{t})$ is a rotation matrix of $\omega_1 \mathrm{t}$ radians around the z axis. The constants are given by $\mathrm{c}_1 = 5$ m, $\mathrm{c}_2 = 5$ m, $\mathrm{c}_3 = 3.5$ m, $\omega_1 = 0.05$ rad/s, $\omega_2 = 0.025$ rad/s and $\mathrm{h}_0 = 7$ m. In addition, $\mathrm{v_r} = 2.1$ m/s and $\mathrm{k_f} = 1.8$ m^{-1}, while $\mathrm{k_v} = 1.0$ s^{-1}.

Figure 5.8 shows four stages of the drone following the moving and deforming saddle-like curve. The plot corresponds to 150 seconds of the experiment. The time-varying curve $\mathcal{C}(\mathrm{t})$, in black, changes between frames, since it assumes a different shape for each instant of the experiment. The past trajectory of the system is shown in red. The initial condition was $\mathbf{x}(0) = [-4.8,\ -4.9,\ 4.3]^{\mathrm{T}}$, in meters. During the experiment, the maximum value of $\|\mathbf{\Phi}_{\mathrm{T}}\|$ was 0.265m/s, thus, in order to analyze the theory in Section 3.2.3, $\mathrm{v_m} = 0.265$ m/s. As in the previous experiment, it is possible to consider $\Delta_{\mathrm{v}} = 1.1$ m/s by analyzing the difference between the commanded speed and the performed one. Again, using equation (5.4), it follows that $\mathcal{I}(\mathrm{t}) = \left\{\mathbf{x} \in \mathbb{R}^3 \mid \mathrm{D}(\mathbf{x},\mathrm{t}) \leq 0.74 \text{ m}\right\}$.

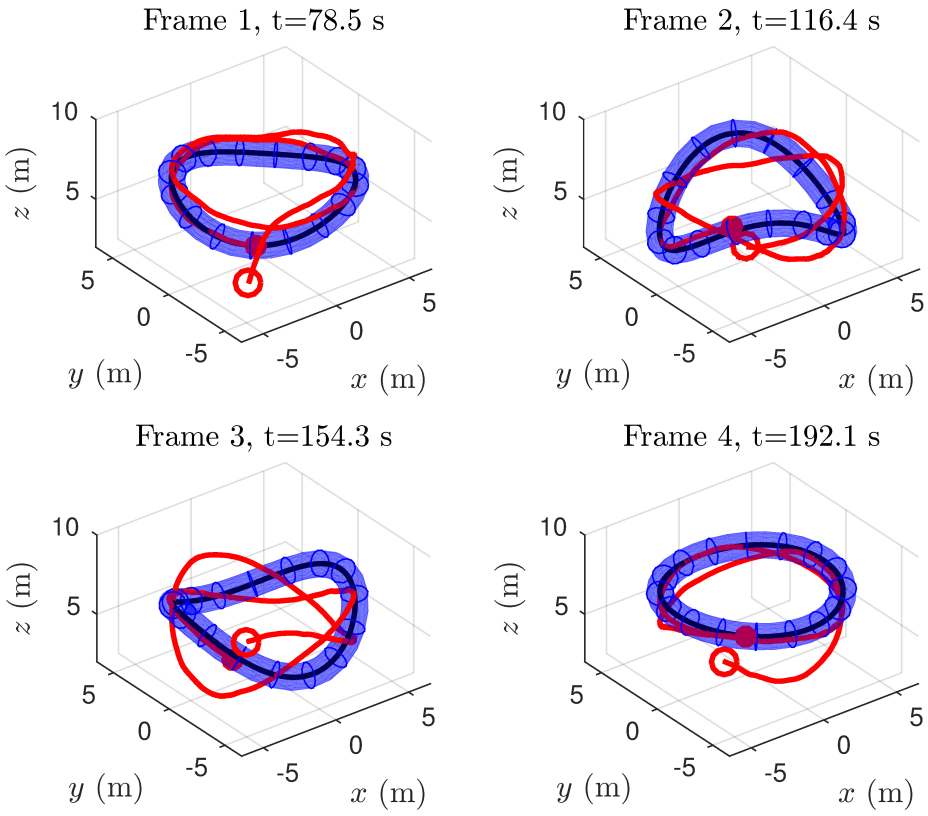

Figure 5.8: Trajectory, in red, of the quadcopter following the saddle-like curve in four different instants. In each frame, the time-varying curve $\mathcal{C}(\mathrm{t})$ has a different shape. The blue tube represents the set $\mathcal{I}(\mathrm{t})$.

In the top of Figure 5.9, the distance function D is presented. The red line represents the ultimate bound for the function D. After convergence, the distance was kept

below 0.58 m. In the bottom, we observe how the norm of the feedforward component, $\mathbf{\Phi}_{\mathrm{T}}$, is bounded by $\mathrm{v_m}$. Figure 5.10 shows how the controller was able to impose the velocity reference $\mathbf{\Phi}$ to the vehicle.

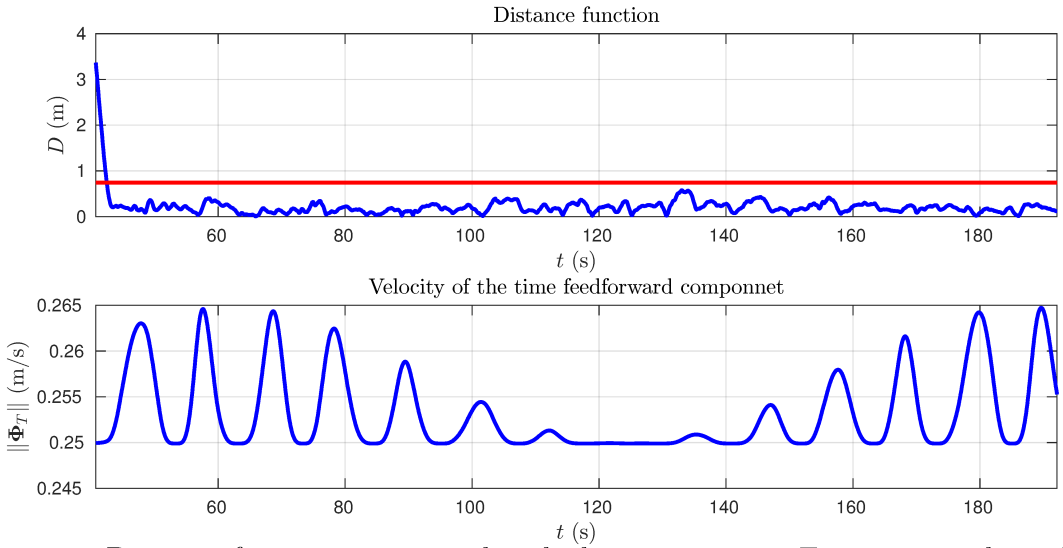

Figure 5.9: Distance function associated with the trajectory in Figure 5.8 and its ultimate bound (top). Norm of the feedforward component (bottom).

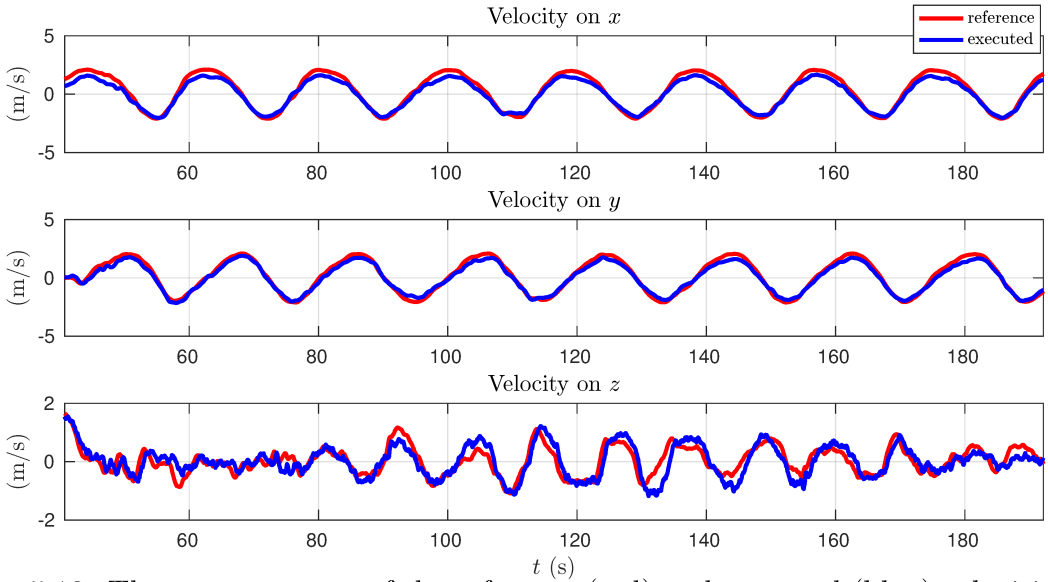

Figure 5.10: Three components of the reference (red) and executed (blue) velocities associated with the trajectory in Figure 5.8.

5.2.4 ESPcopter experiments

The controller designed in Chapter 4 was also deployed to the ESPcopter, described in Section 1.2.2. This time, the experiment was conducted indoors and counted with a

motion capture system. Figure 5.11 illustrates the setup for this experiment. Reflexive

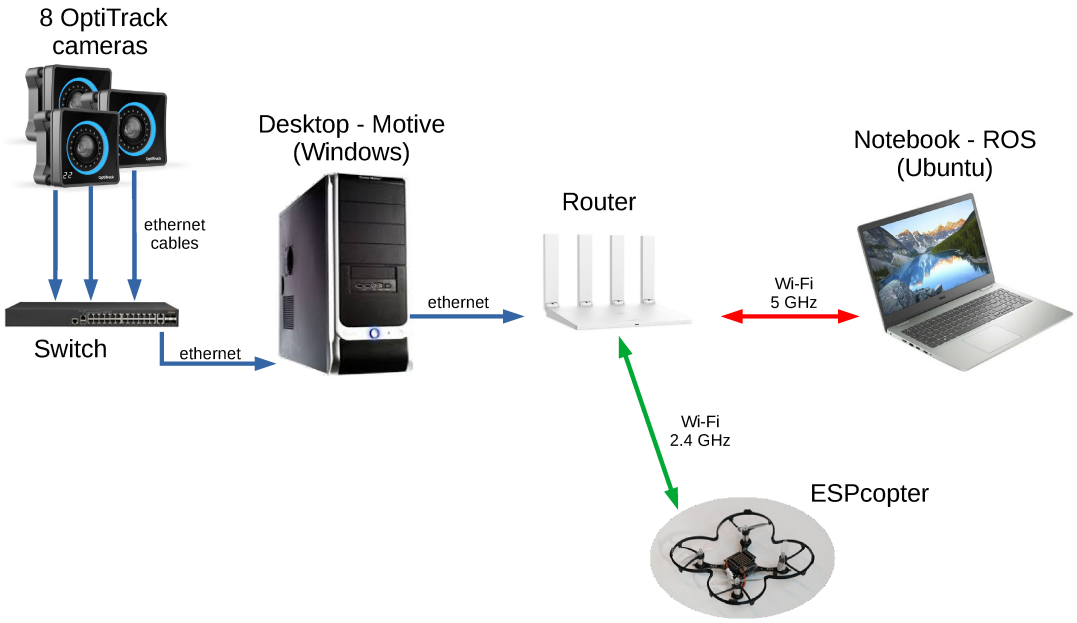

Figure 5.11: Setup for the ESPcopter experiment.

markers are attached to the ESPcoter frame, so that it can be detected by the 8 OptiTrack cameras, as depicted in Figure 5.11. The cameras are connected to a desktop computer that runs the software Motive[1], on a Windows operating system. The software provides the position and orientation of the quadcopter at a frequency of 180Hz. This measurement is streamed in a local network provided by a router. A notebook Dell running ROS under Ubuntu 18.04 is connected to the network through 5GHz Wi-Fi. It receives the pose information (from the motion capture system), estimates the vehicle's velocity, and computes the thrust and angular speed commands that are sent to the ESPcopter. This computer has a 5^{th} generation Intel® Core™ i7 processor, and the controller code was implemented in C++. The ESPcopter connects to the network through Wi-Fi 2.4GHz, from which it receives Acro commands at a frequency of 90Hz. The ESPcopter also provides telemetry data through the network, information that is logged by ROS in the notebook.

The firmware that runs in the ESPcopter counts with a low level controller responsible to impose the Acro rate commands to the vehicle. Basically, at frequency of 1kHz this controller computes the PWM values for each of the four motors. We adopted a PID plus feedfoward approach to compute the desired torques, which together with the total trust, are converted into the desired propulsion force for each motor. The PWMs are computed by using the propulsion model of the pairs motors/propellers, estimated empirically in the laboratory. Details of this controller are out of the scope of this thesis. The firmware is also able to communicate with ROS, which enables the reception of commands and telemetry. We believe that future improvements of this firmware can highly improve

[1] https://www.optitrack.com/software/motive/

the quality of the results presented here. This includes a better lower level controller and a better comunication protocol, with smaller delays. The current communication delay is in the order of 5 ms.

The robot velocity was set to $v_r = 0.8$ m/s. The gain of the function G(D) was set to $k_f = 2.5\ m^{-1}$. The other gains of the quadcopter controller were set empirically as $k_v = 8.0\ \ s^{-1}$ and $k_\beta = 4.0$ rad/s, while the mass of the used ESPcopter is $m = 0.0395$ kg. The considered curve is an ellipse described by the following parametric equation

$$\mathbf{r}(s,t) \equiv \mathbf{r}(s) = \begin{bmatrix} -1.2\cos(s) - 0.3 \\ 0.75\sin(2s) + 0.1 \\ 0.6 \end{bmatrix}, \tag{5.7}$$

in which $0 \leq s \leq 2\pi$ and the units are in meters..

Figure 5.12 shows the ellipse in black and, in red, the trajectory executed by the robot during 57 seconds. The initial condition was $\mathbf{x}(0) = [-0.45,\ 0.78,\ 0.02]^T$, in meters. It corresponds to the drone resting in the floor. The controller naturally makes the drone take off and start following the curve. Since the curve is static, again $v_m = 0$m/s. By comparing the control inputs and the measurements of the vehicle's gyro and IMU we can estimate $\delta_\omega(t)$ and $\delta_\tau(t)$. If we consider $\Delta_\omega = \max(\delta_\omega)$ and $\Delta_\tau = \max(\delta_\tau)$, the set $\mathcal{I}_D$ is excessively big. Thus we consider Δ_ω and Δ_τ as the standard deviations of $\delta_\omega(t)$ and $\delta_\tau(t)$, respectively. Considering this approach, we achieve $\Delta_\omega = 0.65$ rad/s and $\Delta_\tau = 0.008$ N. Using Proposition 5, we compute $\mathcal{I}_D = \{\mathbf{x} \in \mathbb{R}^3 \mid D(\mathbf{x}) \leq 0.21\ m\}$. The invariant set $\mathcal{I}_D$ is illustrated in Figure 5.12 by the blue tube around the curve $\mathbf{r}(s)$.

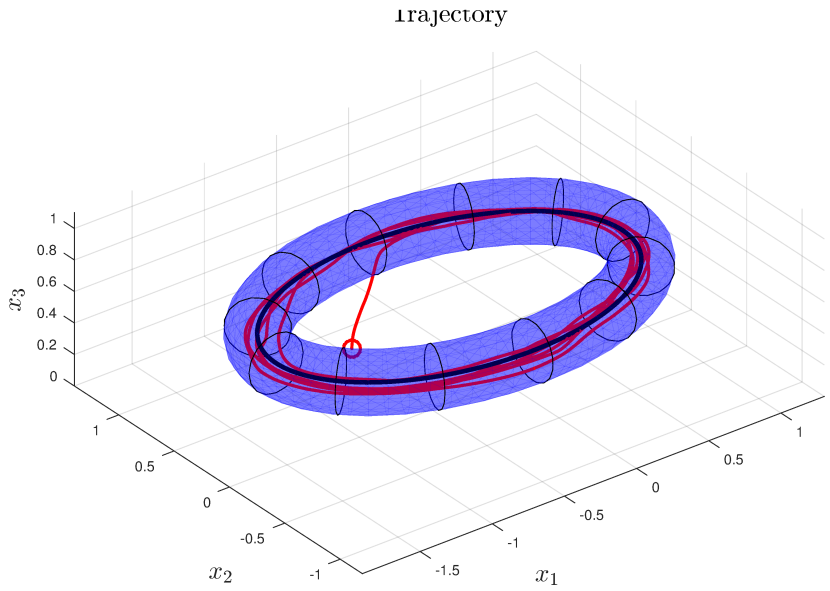

Figure 5.12: Trajectory, in red, of the ESPcopter following the ellipse, in black. The blue tube represents the positive invariant set $\mathcal{I}_D$.

Note that, even considering Δ_ω Δ_τ as the standard deviations of the errors (not the maximums), the computed ultimate bound is a conservative estimation of the maximum

position error of the quadcopter. A future work may be the evaluation of this ultimate bound with a stochastic approach, which could provide less conservative bounds.

In Figure 5.13, the evolution of the distance function D is presented. The relatively high errors in this experiment, up to 13 cm, are justified by imperfections of the lower level controllers. The ultimate bound for D is represented by the red line at the level of 0.21 m. In Section 5.3.2, we show another experiment with the motion capture system and the EspeleoRobô. This robot has a simpler dynamics and better lower level controllers, which will result in smaller errors.

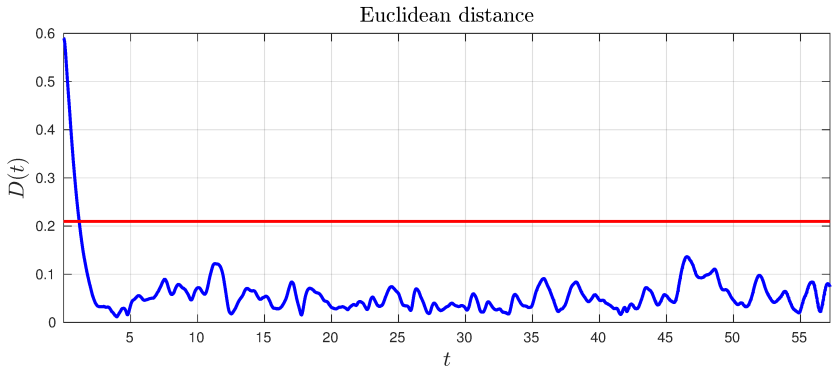

Figure 5.13: Distance function (blue) associated to the trajectory in Figure 5.12 and its ultimate bound (red).

A video with the recording of the ESPcopter experiment presented here is available at https://youtu.be/Uverw8ySvL0.

5.2.5 Drone racing

The Euclidean distance vector field described in Chapter 3 together with the controller described in Chapter 4, were used to control a quadcopter in a drone racing competition, the AlphaPilot Challenge, already commented in Section 1.1.2. In this section, we present some results obtained in the FlightGoggles Simulator[2], presented in [73]. The simulated quadcopter has low-level PID controllers that enable it to be controlled with total thrust and angular velocity commands. Thus, the closed-loop system can be well represented by the model in (4.1). The objective is to make the drone (quadcopter) fly through a sequence of gates in a simulated environment. The solution considers a curve $\mathcal{C}$ defined by a concatenation of polynomial segments, computed with the method proposed in [68]. For the simulation presented here, we considered the ground truth signals of $\mathbf{p}$,

[2] https://flightgoggles.mit.edu/

$\mathbf{v}$, and R_b^w. More details on the path planning, out of the scope of the present work, can be found in [66].

Figure 5.14 illustrates the results obtained in the execution of the task. The black rectangles correspond to the position of each gate. The red line is the path planned to cross the center of the gates. The trajectory executed by the drone is in blue. The vector field controller's parameter was settled as $k_f = 1.0\ m^{-1}$ and $v_r = 7.0$ m/s. The controller gains were adjusted as: $k_v = 1.0\ s^{-1}$, $k_\beta = 2.5$ rad/s. The drone mass was $m = 1.5$ kg and the drag force was modeled as $f_d = -k_d\mathbf{v}$ with $k_d = 0.1$ Ns/m. The controller code runs at 120 Hz.

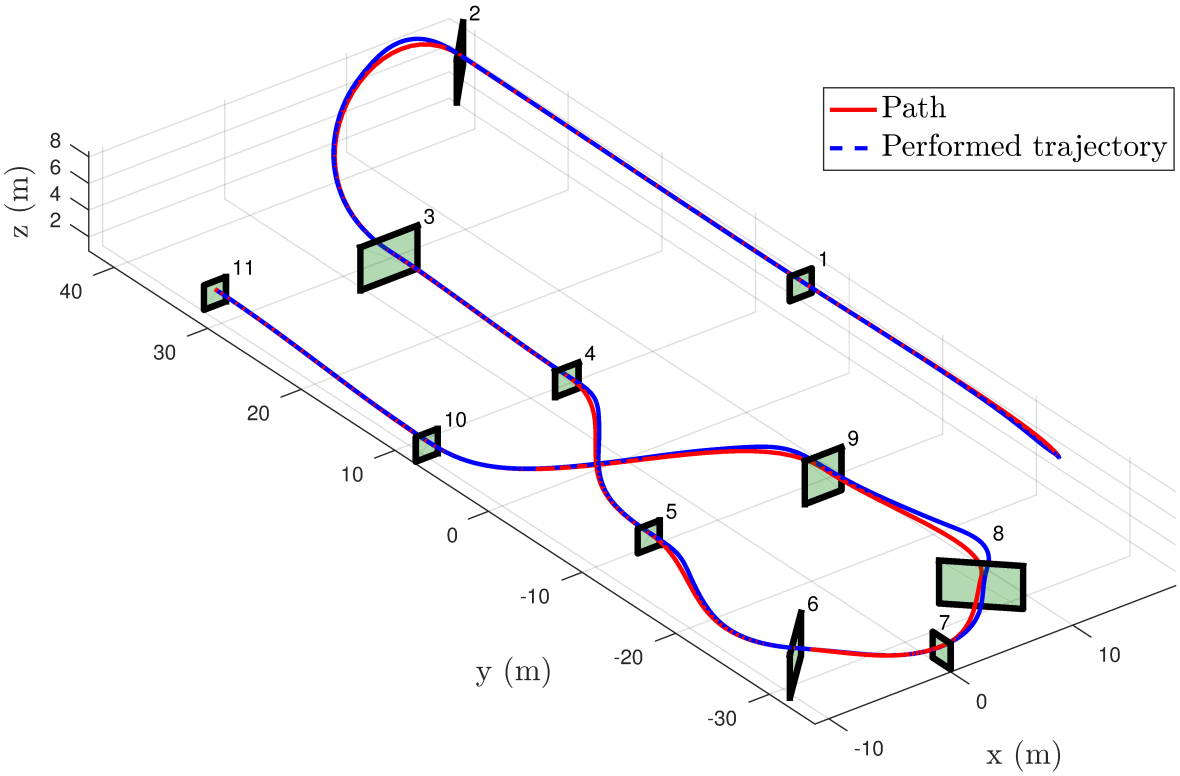

Figure 5.14: Obtained result in the FlightGoggles Simulation: planned path in red and performed trajectory in blue.

Figure 5.15 presents a comparison between the reference velocities given by the vector field (in red) and the ones performed by the drone (in blue), during the trajectory of Figure 5.14. In Figure 5.16, the red lines represent the reference signals for orientation (angles roll, pitch, and yaw) and the blue lines are the performed angles. The reference Euler angles are obtained from the matrix R_r^w only to plot the graph, they are not used by the controller. In both Figures, the performed velocities and angles are close to the references. The reference for yaw, ψ_r, was defined such that the quadcopter's camera points in the direction defined by the vector field.

In Figure 5.17, we show the computed control signals, ω_x, ω_y, ω_z, and τ, in red. The performed angular velocities and total thrust, obtained from a simulated IMU are shown in blue. Note that the references are not perfectly tracked by the inner loop

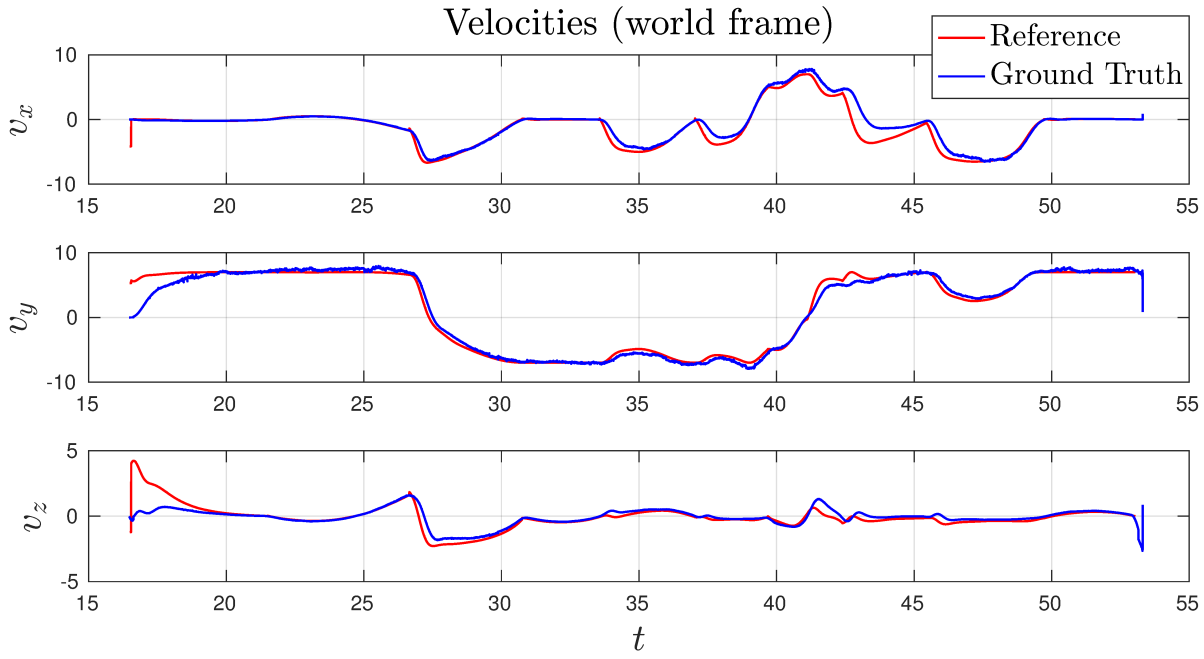

Figure 5.15: Reference for velocity $\mathbf{\Phi}(\mathbf{p})$ in red and performed ones in blue.

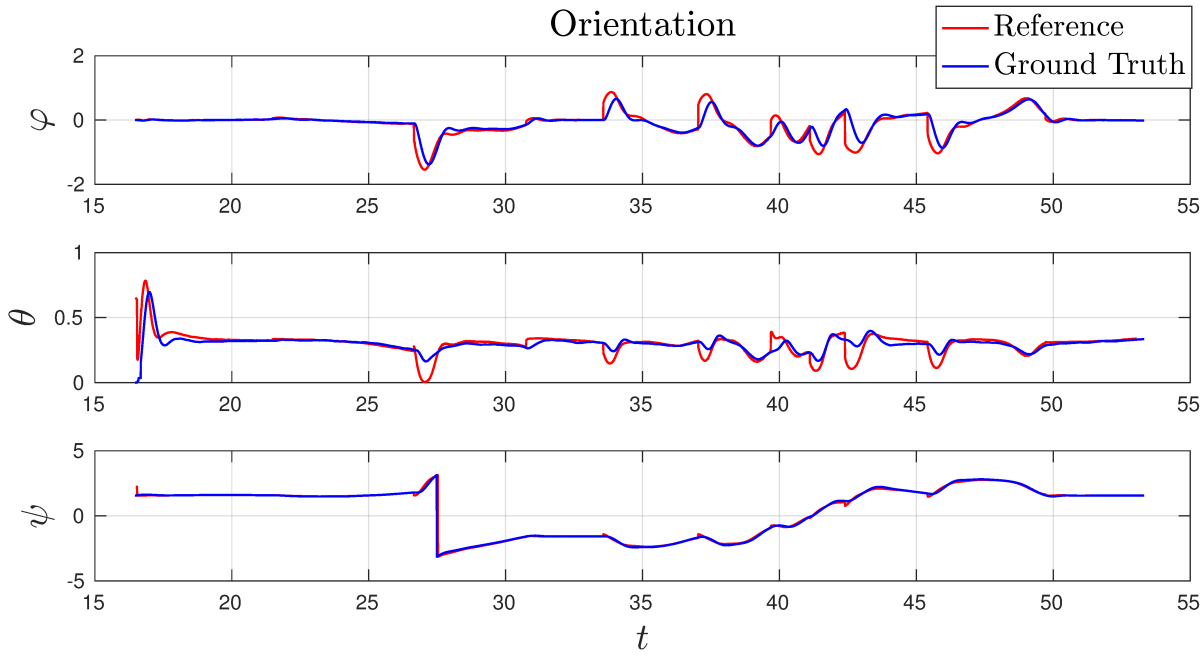

Figure 5.16: Reference for angles (roll, pitch and yaw) in red and performed ones in blue.

controllers. One important fact is that the planned path consists of a concatenation of polynomials. These sub-paths have continuity C^1 on the transitions (end of a polynomial and beginning of the next one), thus are not in agreement with the requirement of 3^{rd} order continuity made in Section 4.1. The lack of a continuity of higher order is responsible for the discontinuities observed in the red signals in Figures 5.16 and 5.17. The consideration of a path with a higher order continuity may decrease the high position errors after gates 2 and 8, in Figure 5.14.

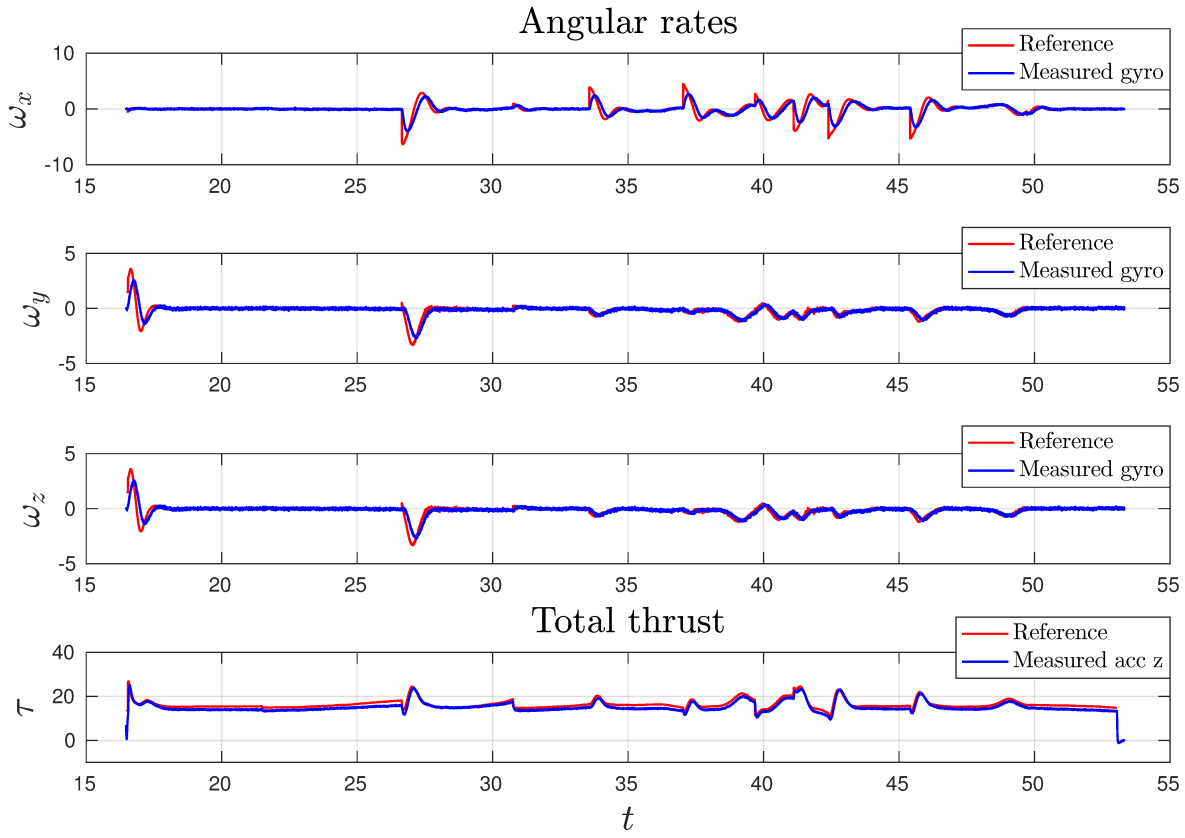

Figure 5.17: Control signals ω and τ commanded for the robot in red. In blue are the executed actions obtained from the IMU.

5.3 Results with other robots

Now we present some results that exemplify how the vector field designed in Chapter 3 can be applied to other robot types. First, we consider the simple integrator robot. Then, we show experiments with the EspeleoRobô. Finally, we consider a robotic manipulator with 6 DOF.

5.3.1 Simulations with the simple integrator model

5.3.1.1 Generic curve shapes

In order to demonstrate the ability of our vector field theory to deal with generic curve shapes, we first considered four curves in the form of the letters U, F, M and G.

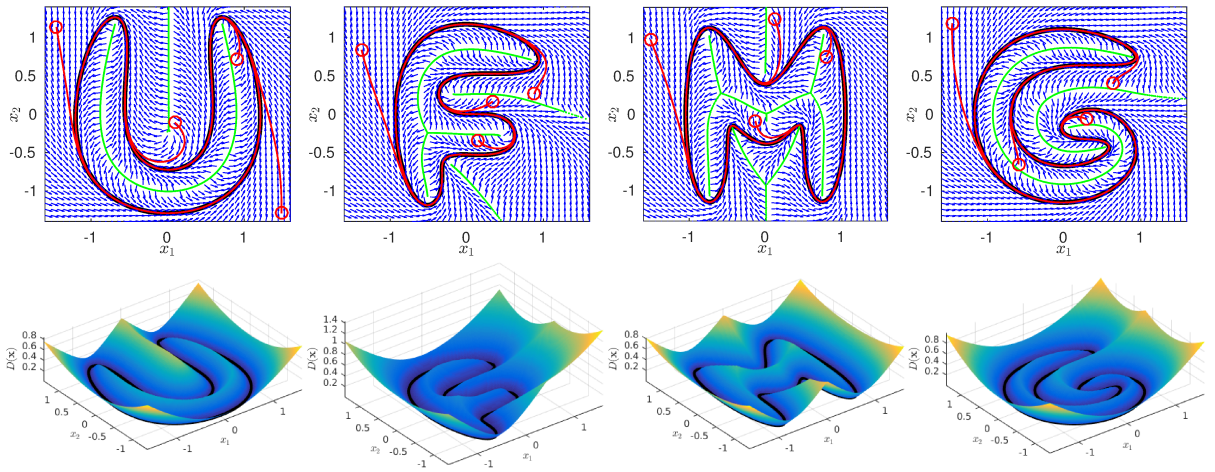

Figure 5.18: Four different curve shapes in the form of the letters U, F, M, and G. In the top the curves are in black, the vector field is depicted in blue and in green are the set of points in which there are more than one point closest to the curve or, equivalently, points not in $\mathcal{U}$. Note that none of the trajectories (in red) reaches the green lines. In the bottom, we show the corresponding distance functions.

We considered each curve was represented by a sequence of 1000 points and used the numerical strategy presented in Section 5.1. The result is presented in Figure 5.18. In the top, we show the 2D static vector fields in blue. In red we show some trajectories of the system. In green we show the set of "equidistant points", those that do not belong to the set $\mathcal{U}$. In the bottom we depict the associated distance functions. In order to validate the analysis presented in Appendix A, we assumed the G function defined in equation (3.45). Note that no trajectory reaches the green region. To compute this field we used MATLAB R2016a, in a computer with a third generation Intel® Core™ i7 processor. On average, each arrow on the grids of Figure 5.18 took 45 μs to be computed.

5.3.1.2 Self intersecting curves

In order to validate the methodology proposed in Section 3.3.3, we considered a trefoil curve. In the center of this curve, it intersects itself 3 times. The original vector field $\mathbf{\Phi}$ can not be applied to make the robot navigation in this case, since the curve breaks Assumption 1. For $0 \leq s < 2\pi$ the trefoil parametric equation is given by

$$\mathbf{r}(s) = \begin{bmatrix} \cos(s) + \cos(2s) \\ \sin(s) - \sin(2s) \end{bmatrix}. \tag{5.8}$$

The simulation considered the vector field in equation (3.53) together with the propagation law $\dot{s}_v$ in equation (3.56). We considered $k_f = 0.5\ m^{-1}$ and $k_s = 1\ m^{-2}$. The initial position of the system was $\mathbf{x}(0) = [-3,\ -1]$, in meters.

In a first simulation, we considered the initial condition $s_V(0)$ in (3.54). The result is shown in Figure 5.19. From left to right we have tree different instants of the simulation. The trefoil curve is in black. The current robot position is the red circle and the past is in red. The blue cross corresponds to the point $\mathbf{x}_V$.

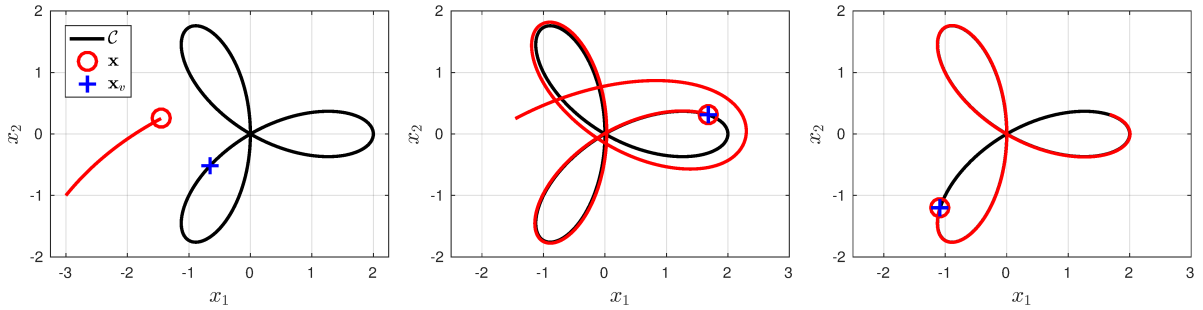

Figure 5.19: Simulation of of the augmented vector field $\mathbf{\Phi}_V$. Consideration of initial virtual variable $s_V(0) = s^*$.

Figure 5.20 shows two distance functions associated with the trajectory in Figure 5.19. In blue we show the distance D, which is the shortest distance to the curve, a global optimum. When the trajectory crosses the curve we see that D drops to zero, and increases again. This happens because the field is not using the global optimum anymore, but a local one. The drop on the distance D caused no harm in the trajectory in Figure 5.19. The function D_V, which corresponds to a local minimum of the distance to the curve, does not suffer the same falls. The crossing of the curve is basically neglected by the function D_V, which is based on the new state variable s_V.

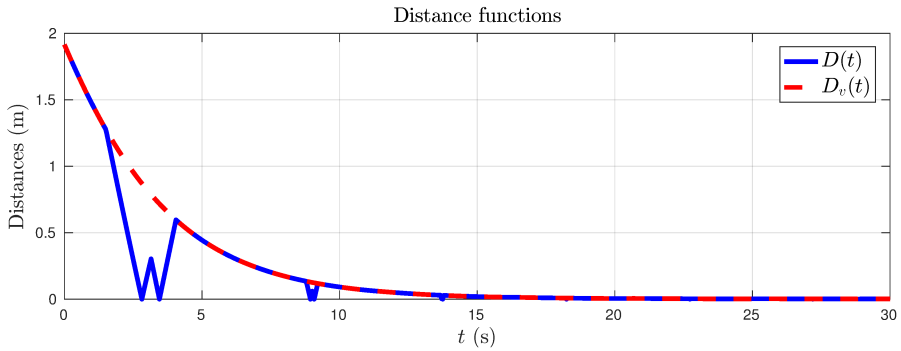

Figure 5.20: Distances associated with the trajectory in Figure 5.19.

In a second simulation, we considered a value for $s_V(0)$ that does not correspond to any local optimal of the distance function to the curve. With this simulation, we aim to show the ability of the controller proposed for s_V to make it converge to a local minimum of the distance to the curve. The result is shown in Figure 5.21. In the left image, the blue square shows the point $\mathbf{x}_V$ correspondent to $s_V(0)$. Note that the trajectory is different from the one in Figure 5.19, since the system converged to a different initial local optimum. We can say that $s_V(0)$ was in a different basin of attraction.

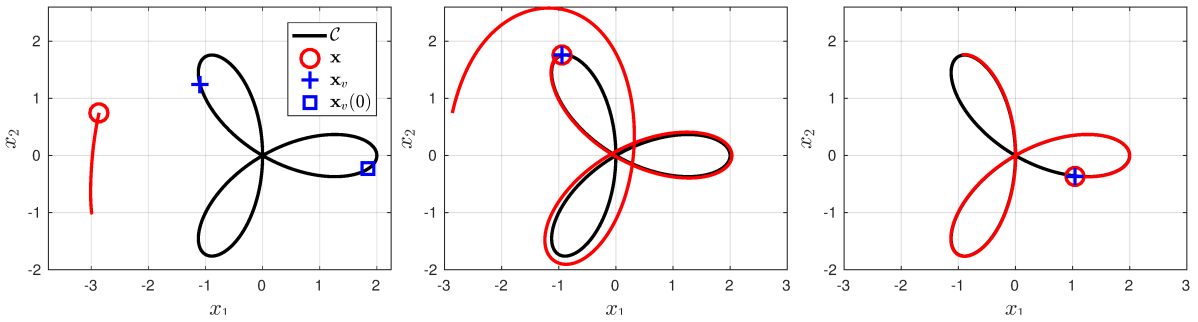

Figure 5.21: Simulation of of the augmented vector field $\mathbf{\Phi}_{\mathrm{V}}$. Consideration of initial virtual variable $\mathrm{s_V}(0) \neq \mathrm{s}^*$.

Figure 5.22 shows two distance functions associated with the trajectory in Figure 5.21. Note that, initially, $\mathrm{D_V}$ is significantly larger than D. After approximately 2 seconds, $\mathrm{D_V}$ was already in agreement with the optimal distance D, thus, corresponding to a local optimum. It shows that the propagation law in (3.56) is able to make $\mathrm{s_V}$ evolve properly even if $\mathrm{s_V}(0) \neq \mathrm{s}^*$.

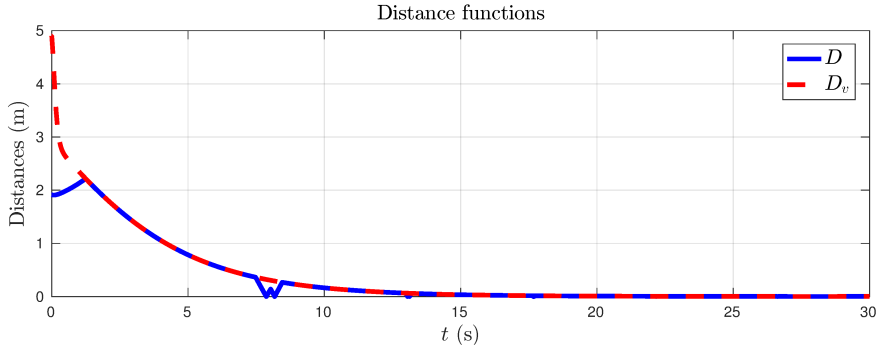

Figure 5.22: Distances associated with the trajectory in Figure 5.19.

5.3.2 Experiments with the EspeleoRobô

In this section, we present some experimental results obtained with the EspeleoRobô described in Section 1.2.3. Here we show the parametric equation of the curves, however, the implementation of the controller was made upon an equivalent sequence of points. We considered the function $\mathrm{G} = (2/\pi)\,\mathrm{atan}(\mathrm{k_f D})$ in all experiments. Also, the vector field was used along with Feedback Linearization technique recalled in Section 2 (equation 2.1). Based on [20], to convert the linear $\mathrm{v_{rob}}$ and angular ω_{rob} speeds to the

right and left wheels speeds, v_{right} and v_{left} respectively, we consider the following model

$$\begin{bmatrix} v_{right} \\ v_{left} \end{bmatrix} = \begin{bmatrix} 1 & \frac{a^2+b^2}{2b} \\ 1 & -\frac{a^2+b^2}{2b} \end{bmatrix} \begin{bmatrix} v_{rob} \\ \omega_{rob} \end{bmatrix}, \tag{5.9}$$

in which a is the distance between the front and back wheels and b is the distance between the right and left wheels. For the EspeleoRobô $a = 0.43$ m and $b = 0.36$ m.

5.3.2.1 Experiment with precise pose measurement

In a first experiment, the motion capture system of the laboratory was used to control the robot to perform a small ellipse. The main objective is to show the accuracy of the vector field strategy when a precise pose measurement is available. The precision of the OptiTrack system is 1 mm. The ellipse has semi axes $a = 1.7$ m and $b = 1.0$ m. We considered $d = 0.3$ m, $v_r = 0.3$ m/s and $k_f = 3.0$ m. Figure 5.23 shows the trajectory executed by the control point used by the feedback linearization technique (see Figure 2.5). In fact, the position used to compute the vector field is the position of this point, not the one of the robot's geometric center.

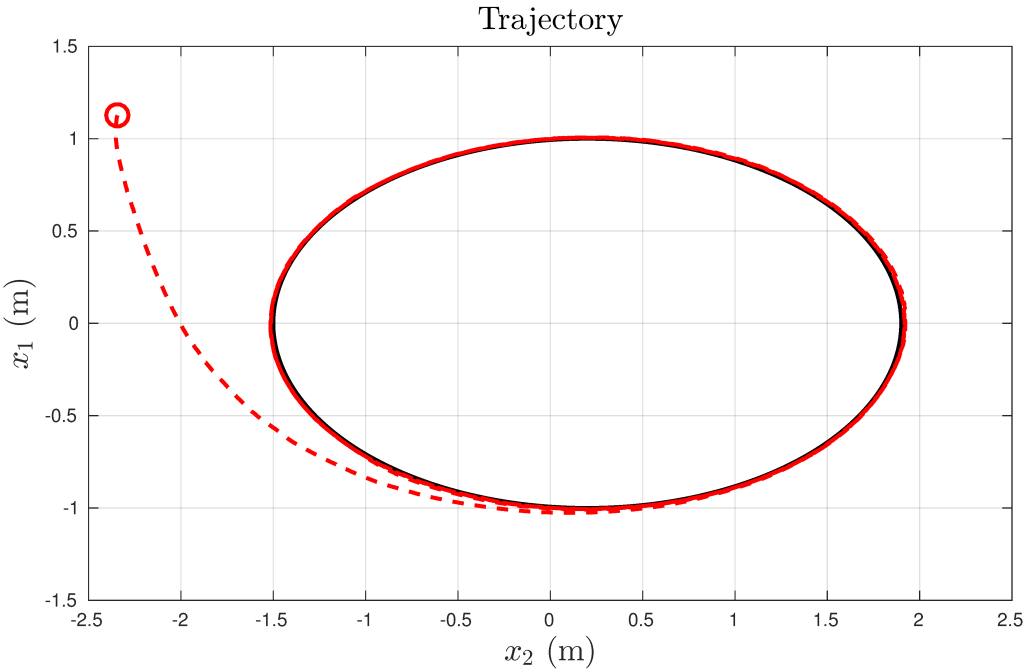

Figure 5.23: Result of the ellipse experiment with the EspeleoRobô when using the pose from motion capture system.

Figure 5.24 presents the evolution of the distance function D. After the trajectory reaches the curve, the error was kept bellow 0.02 m, or 2 cm. This result contrasts with the ones achieved with the ESPcopter, which had errors up to 13 cm. It shows that the accuracy of the lower level controllers are fundamental to have a good path following result.

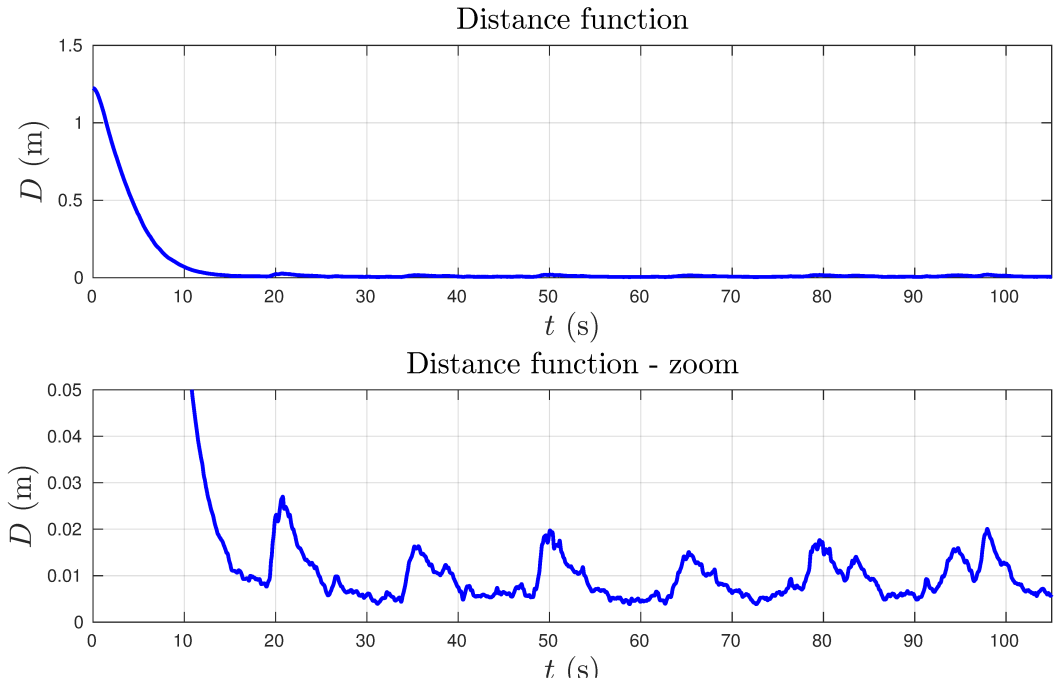

Figure 5.24: Result of the ellipse experiment with the EspeleoRobô when using the pose from motion capture system. The bottom graph is a zoom of the top one.

A video with the recording of the experiment in which the EspeleoRobô followed an ellipse using the motion capture system is available at https://youtu.be/lkiSWZKHR4c.

5.3.2.2 Experiment with self intersecting curve

Another experiment with the EspeleoRobô was performed in the hall of the Engineering School, also at the Campus in Pampulha. This time, the localization was performed with the LeGO-LOAM algorithm fed with the data of the Ouster sensor. The considered curve is a lemniscate, which has an 8 like shape. For $0 \leq s < 2\pi$ the lemniscate's parametric equation is given by

$$\mathbf{r}(s,t) = \mathbf{r}(s) = \begin{bmatrix} a\dfrac{\cos(s)}{1+\sin(s)^2} \\ 2b\sqrt{2}\dfrac{\cos(s)\sin(s)}{1+\sin(s)^2} \end{bmatrix}, \tag{5.10}$$

in which a = 4 m and b = 1.5 m correspond to half of the extension of the curve in the x_1 and x_2 directions. The vector field velocity was set as $v_r = 0.5$ m/s, while the vector field convergence gain as $k_f = 3\ m^{-1}$. The distance d for the feedback linearization was d = 0.15 m. The lemniscate has a self-intersection in the point [0, 0]. In order to deal with this type of curve, in the implementation we considered the numerical method shown in equation (5.1).

Figure 5.25 presents a superposition of nine frames of a recording of the experiment. It is possible to see the robot's position in different moments of the experiment and to observe the shape of the lemniscate. The red numbers indicate the chronological sequence. Figure 5.26 presents a 2D plot of the trajectory performed by the robot. The black curve is the lemniscate and the red line is the position estimated with the robot's localization system.

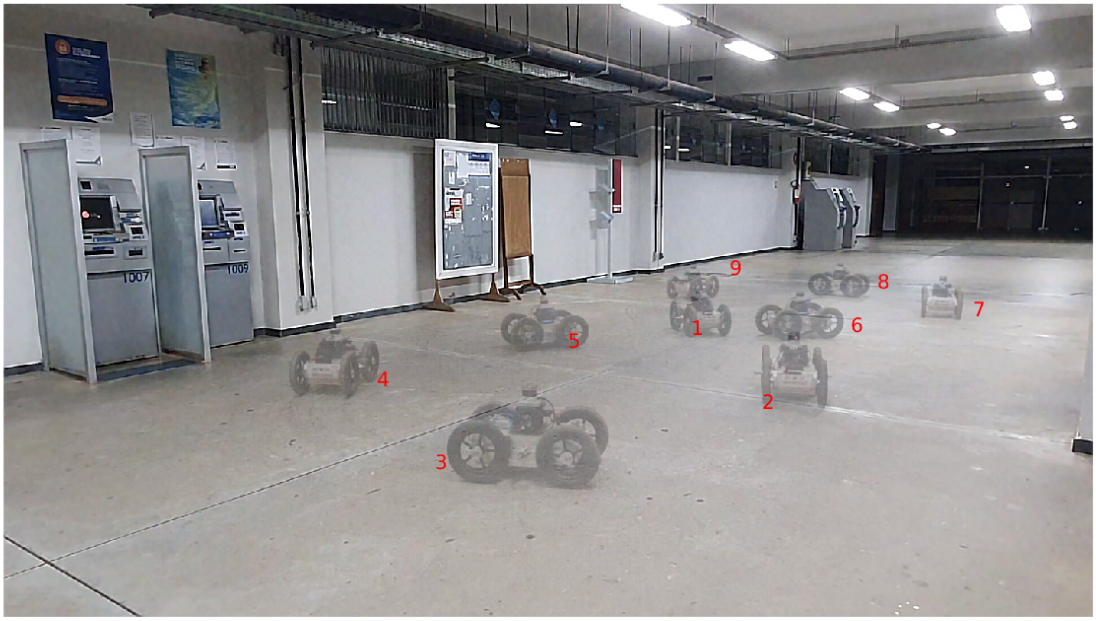

Figure 5.25: Superposition of frames of the recording of the lemniscate experiment.

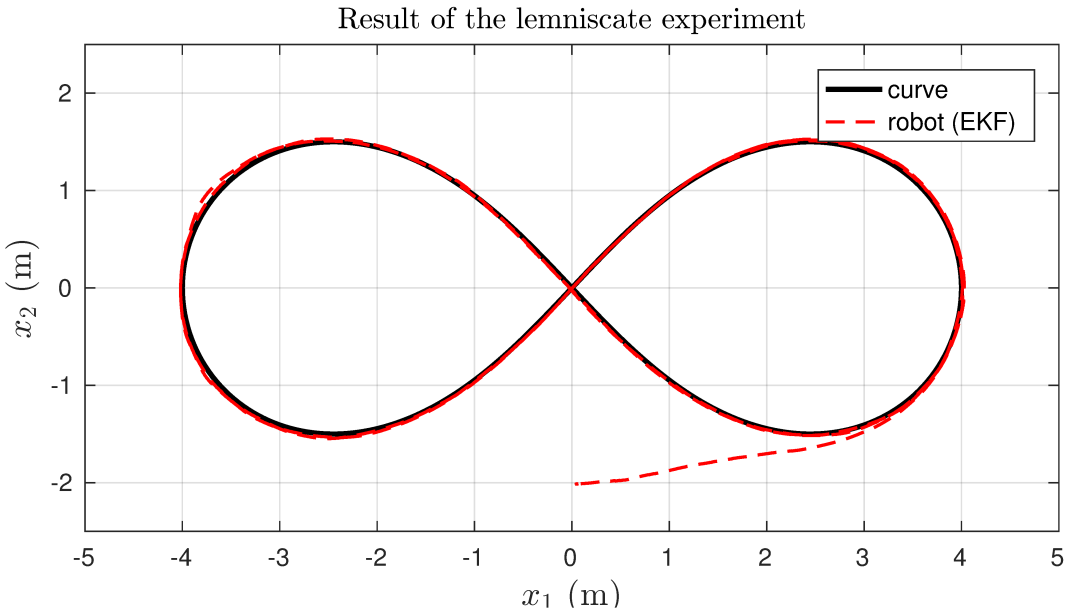

Figure 5.26: Result of the lemniscate experiment using the LeGO-LOAM for localization.

5.3.2.3 Experiment with obstacle deviation

In a second experiment in the hall of the Engineering School of the UFMG, the EspeleoRobô was again controlled to follow a lemniscate curve. Now, there are two differences in the setup. First, the localization was performed with the gmapping algorithm fed with the Hokuyo data. Second, we considered the vector field described in Section 3.4.2, to enable the robot to deviate from obstacles. While the robot was following the curve, traffic cones were placed in different positions of the workspace to evaluate the system's efficiency in deviating from them. We keep with the use of the strategy in (5.1) to compute the field. The closest point of the obstacle set was obtained from the Hokuyo data.

The curve parameters were set as a = 4.0 m and b = 0.8 m, see equation (5.10). We set v_r = 0.5 m/s, d = 0.3 m and k_f = 2.5 m^{-1}. The obstacle deviation parameters were D_{in}^0 = 1.2 m, D_{in} = 1.2 m, and λ = 0.7 m. Note that $D_{in}^0 = D_{in}$ suppresses the transition region that merges $\mathbf{\Phi}$ and $\mathbf{\Psi}$. Figure 5.27 shows, from top to bottom, three different stages of the experiment's execution. On the left we show a frame of a recording of the experiment correspondent to the plot on the right. In the plots, the curve is shown in black and the robot's past trajectory in red. The cones are represented by the orange circles. Note that, in the top, when there were no obstacles, the trajectory converges to the lemniscate path. In the middle, two cones were placed in the robot's path. The first one was contoured clockwise, while the second one counter-clockwise. As show in Section

5.2.2, the distance to the curve increases when the robot is deviating from the obstacles. However, it returns to decrease as the obstacles are left behind.

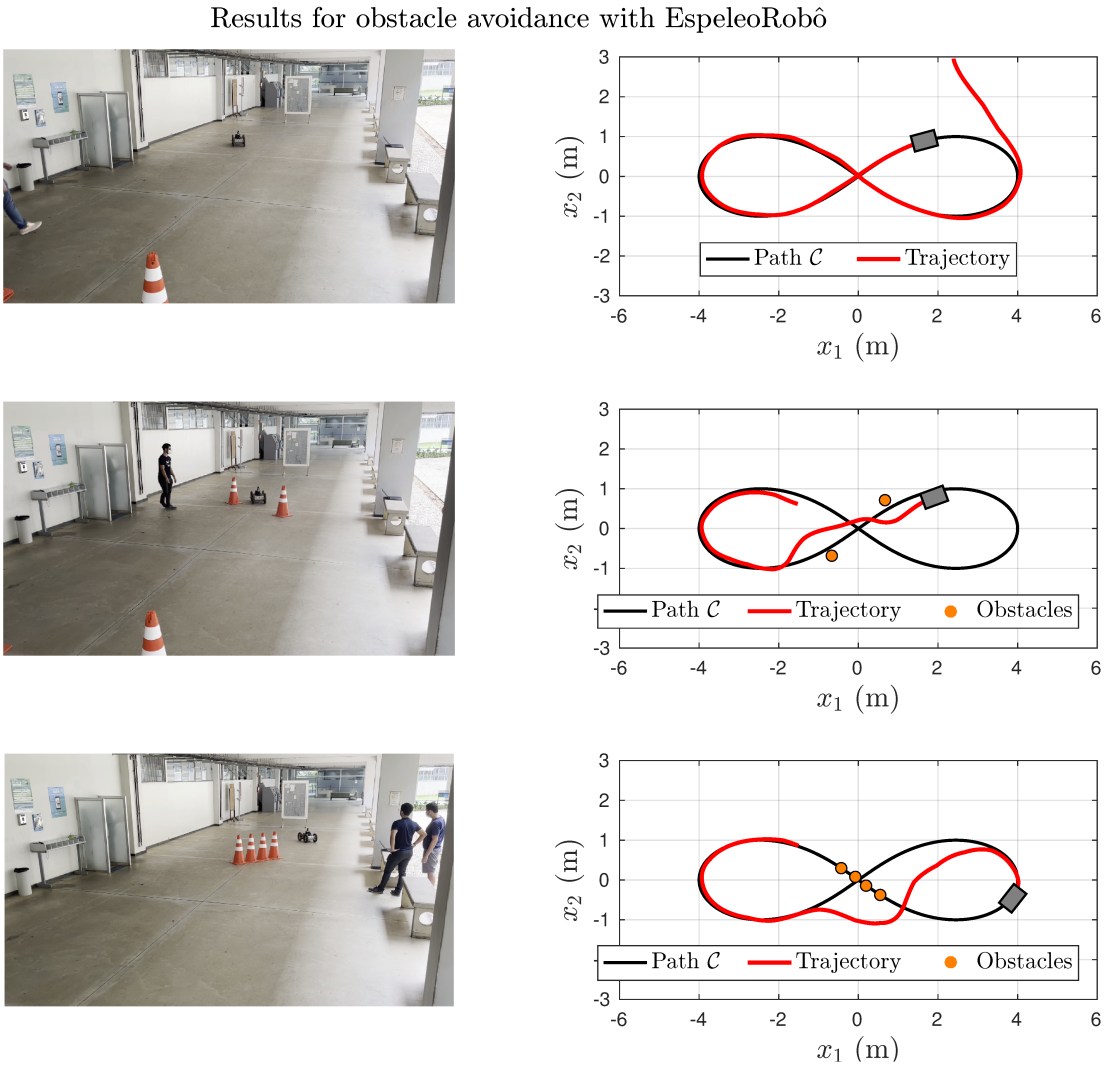

Figure 5.27: From top to bottom in the left, three frames of the recording of the experiment of the EspeloRobô following an lemniscate curve (in black) while deviating from obstacles (in orange). On the left we show respective plots of the robot's past trajectory (in red).

A video with the recording of the experiment in which the EspeleoRobô followed the lemniscate and deviated from the obstacles is available at https://youtu.be/hQuCXnLEOd8.

5.3.2.4 Experiment outdoors

The vector field was also tested in an outdoor experiment. In this case, the localization method was based on the Ouster sensor and the LeGO-LOAM algorithm. The considered curve was an ellipse that has the semi-axes with 8 and 12 meters. The velocity was $v_r = 0.65$ m/s, the convergence gain was $k_f = 3$ m^{-1}, and the parameter of the Feedback Linearization was $d = 0.2$ m. It was performed in a rough terrain close to the *Restaurante Setorial I* and the Fourth UFMG's entrance gate (*Portaria 4*) at the Campus in Pampulha. Figure 5.28 presents, from two different points of view, the map created by the LeGO-LOAM algorithm (see Section 1.2.3) and the trajectory executed by the EspeleoRobô in blue.

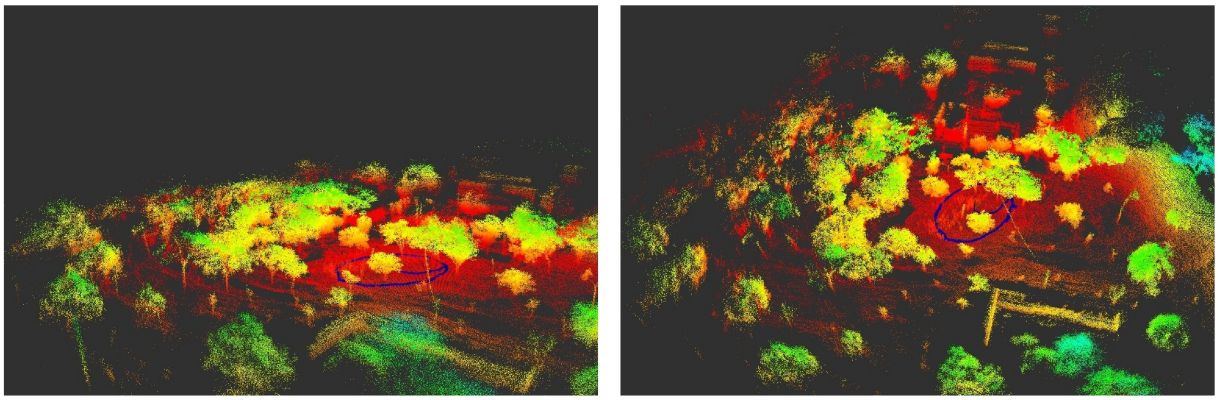

Figure 5.28: Result of the outdoor experiment obtained with the EspeleoRobô following the Euclidean Distance Vector field.

As can be seen in Figure 5.28, the environment counts with many trees. During the experiment, there was no tree in the robot's path. Thus, in this experiment we did not consider the vector field proposed in Section 3.4, but the original one proposed in Section 3.2.

The rough terrain, of ground and grass, made the EspeleoRobô slip during several moments of the experiment. Sometimes, although the wheels were rotating, the robot remained stopped for a few seconds. Anyways, this problem caused no harm in the path following. Thus, this experiment shows, in practice, one of the advantages of a path controller over traditional trajectory controllers, as discussed in Section 2.1.1.

5.3.3 Simulations with a manipulator

Up to now, we have only considered the vector field designed in Chapter 3 for problems in 2 and 3 dimensions. We now apply the methodology for a problem in 6 dimensions and in a non Euclidean space. We consider the control of a 6 degrees of freedom Kuka KRL4 robot to draw a circle orthogonal to the ground with the tool in the end-effector parallel to the ground. The vector field was applied directly to the robot's joint space. This implementation requires a simple trick, presented in Appendix B, to compute the distance function, originally designed for the Euclidean space. In the case of the Kuka robot, we can compute the curve $\mathcal{C}$ in the joint space by solving an inverse kinematic problem. This can be done by using numerical methods.

It may be easier to consider the field in the workspace and use an inverse Jacobian approach to control the manipulator. However, this would insert those recurrent problems associated to singular configurations. The application of the field in the joint space does not suffer with that. Moreover, the intention is to show how the vector field can be applied in dimensions higher than three.

To simulate the application of the vector field in the control of the manipulator taking into account its dynamics, we used the CoppeliaSim simulator[3]. We also used an interface that allows the user to submit a reference joint speed computed using the distance field, which is internally converted into torque by a control loop, an SGC structure.

The target curve $\mathcal{C}$, mapped into $\mathbb{R}^6$ using the replication trick (Appendix B), was defined so that the last link of the robot's kinematic chain performs a circle in a plane orthogonal to the ground. Also, the manipulator's tool should be orthogonal to the circular face defined by the circle. This can be interpreted as a task of welding a circular shape in a plane surface or drawing a circle on a whiteboard with a marker pen (either way, the tool must be aligned with the surface's normal).

Figure 5.29 shows, on the left, the CoppeliaSim scene used in the experiment. The ciano trajectory was the one performed by the robot's tool. On the right, we show a plot where we can observe that the tool of the end-effector (blue axis) is indeed parallel to the ground, aligned with the x axis. The initial condition had all joints with a zero value. It corresponds to the arm stretched out in a singular configuration. Since our vector field was applied directly on the joints, and no inverse kinematics was necessary to track the vector field, this singular configuration caused no harm to the task.

[3] https://www.coppeliarobotics.com/

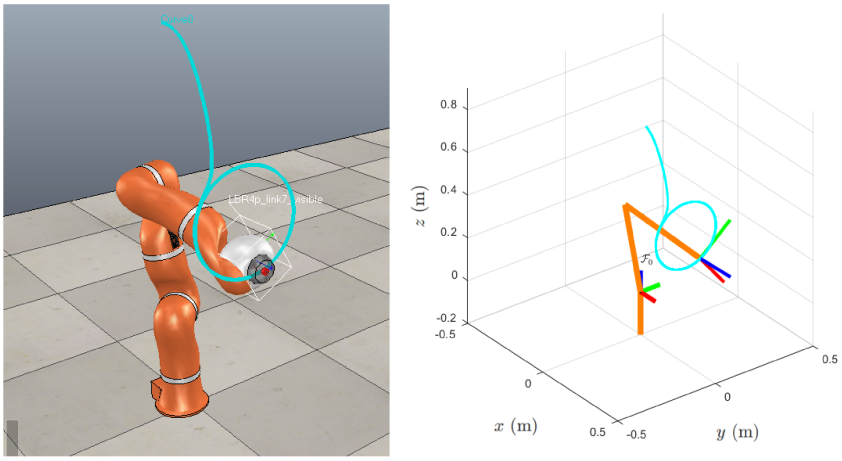

Figure 5.29: Scene in CoppeliaSim with the KUKA robot following a curve in the joints space (left). Matlab plot of the simulation (right).

It is important to emphasize that the circle in Figure 5.29 is not the curve $\mathcal{C}$, it is only the position of the last link when the joints follow the path in the joints space. It is a projection of the real curve $\mathcal{C}$ by the direct kinematics of the manipulators' tool. The curve $\mathcal{C}$ lies in six dimensions, thus it can not be directly visualized. In fact, in our example, the robot completes 2 laps on the circle while completing only one lap on the curve $\mathcal{C}$. In order to show the error in tracking the curve in the joints space, Figure 5.30 shows the function D, computed according to the closest point map given in equation (B.6).

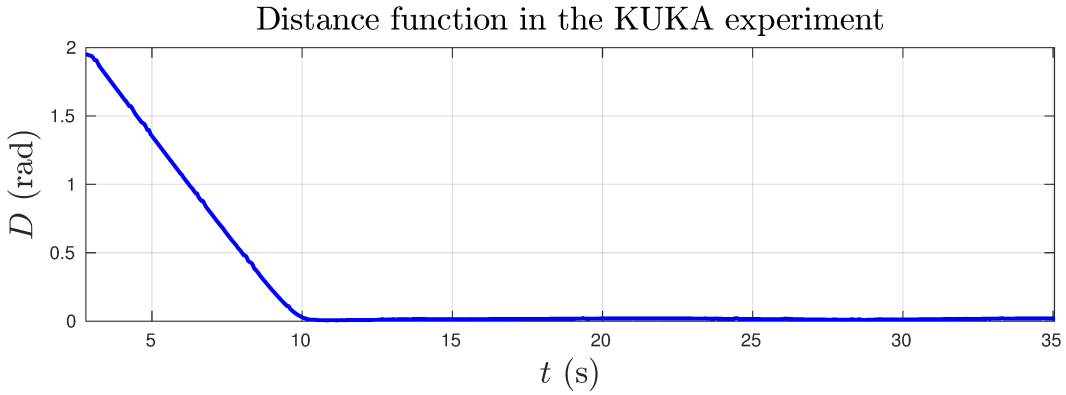

Figure 5.30: Distance function in the joints space in the manipulator example.

A video of the CoppeliaSim simulation with the KUKA manipulator performing the circular movement is available at https://youtu.be/8NUOtM_FP6g.

Chapter 6

Conclusion and future work

In this thesis, we tackled two problems in the field of robot navigation, the guidance problem and the control problem. First, we presented a novel vector field based methodology to solve the guidance problem. Then, we focused on a specific robot platform, a quadcopter, and designed a controller that enables it to follow the proposed vector field.

The guidance solution generates vector fields in n dimensions that converge to and follow time-varying curves. The strategy is based on the Euclidean distance function. We recall four novel important properties/advantages of the vector fields constructed by our method in comparison to previous works: (i) it can be easily constructed from a parametric representation of the curve or even by a curve represented by a sequence of points; (ii) its convergence pattern is isotropic on space, which facilitates the calibration of the convergence gain of the field; (iii) it has a constant norm even for time-variant cases, a novel result regarding fields with asymptotic convergence; and (iv) it does not contain equilibrium points. All of these important theoretical and practical features emerge from the use of the Euclidean distance as an error measurement function, instead of a general analytic function. Disturbance in the assumed simple integrator model was taken into account in order to evaluate the performance of the technique in real robot applications, in which imperfections on the lower level controllers and other disturbances are always present. The proposed field is not single-valuated in a region of measure zero in $\mathbb{R}^n$. Nevertheless, the simulations and experiments show that this is not a problem in practice.

Slightly modifications on the proposed vector field allowed to incorporate two features. The first is the ability to consider curves with self intersections. The second is the ability to deviate from obstacles while the curve is being followed. These theories are less mathematically rigorous, and open a branch for future research.

Although the designed vector field methodology can be used in the guidance of several robot platforms, we focused on the control of a quadcopter. A controller to make this vehicle follow the designed vector field is then designed. The control laws were first developed by considering a second-order integrator and then the control laws for the quadcopter were derived. The whole system has a cascade-like connection, and the asymptotic stability was proven via the property of input-to-state stability of each individual loop. This was possible due to the consideration of disturbances in the velocity

field, which is also a system with the ISS property. The designed path following controller for the quadcopter incorporates features such as: (i) path controller based on artificial vector fields; (ii) formal convergence proofs for generic time-varying curve shapes; and (iii) consideration of disturbed control inputs.

Both the vector field methodology and the quadcopter controller were validated in simulations and experiments performed with three real robots, both indoors and outdoors. The simulations exemplify the behavior of the controller at high speeds while the real robot experiments validate the framework in a real scenario. The vector field methodology was also tested in a simulation with a robotic manipulator with 6 degrees of freedom. The experiments with the ground robot, EspeleoRobô, some using a completely onboard localization system and performed in unstructured environments, corroborate with the robustness properties argued along the text.

Future work

The vector field proposed in Sections 3.1 and 3.2 has a strong mathematical formulation. However, the augmented field (Section 3.3) and the obstacle deviation strategy (Section 3.4) do not count with mathematical convergence proofs with the same level of rigor. These modified versions of the original vector field are extensions that proved to be effective in practice. However, they lack a better theory to support them. In fact, those extensions of the vector field point to the future research that can be made upon this thesis. In the following, we suggest possible directions for this future research and present some ideas.

- The vector field $\boldsymbol{\Phi}_{\mathrm{v}}$ presented in Section 3.3 showed to behave well both in simulation and real robot experiments. However, we did not provide a formal convergence proof. Our arguments rely on the similarity of the field $\boldsymbol{\Phi}_{\mathrm{v}}$ with the original field $\boldsymbol{\Phi}$ and that we initialize the parameter s_{v} with the optimal s^*. A future work may seek for a rigorous convergence proof even when $\mathrm{s}_{\mathrm{v}}(0) \neq \mathrm{s}^*$.

- The vector field $\boldsymbol{\Phi}_{\mathrm{v}}$ was constructed to avoid singularity problems when s^* is not unique. Although it serves this purpose, it is not globally free of singularities. We showed that there is a problem in the propagation of s_{v} when $\Omega_{\mathrm{v}} = 0$. A future research may seek for a different definition of the augmented field, still based on the Euclidean distance, that is truly singularity free, *i.e.* for all $\mathbf{x} \in \mathbb{R}^{\mathrm{n}}$.

- The field with the incorporation of obstacle avoidance property presented good

results in practice. However, we did not provide convergence proofs for this variant of the method either. In fact, before presenting convergence proofs, a convergence definition must be made for the case in which there are obstacles in the curve, or close to it. The standard asymptotic stability concept can not be used, since the robot will eventually need to move away from the curve to avoid a collision. The statement of a proper convergence definition when obstacles are present is also a topic for future research.

- In possession of a proper definition of convergence in the case obstacles are present, we may attempt to prove the convergence of the navigation proposed in Section 3.4. We believe that a proof for the generic n-dimensional case may be difficult to achieve. A good starting point is the bidimensional case. Possibly, a proof based on the convergence proofs of the bug algorithms may be possible to achieve. Seeking for the proof, new insights in the definition of the composite vector field may also appear.

- A better investigation of the presence of singularities in the vector field that deviates from the obstacle is necessary. This may include a better evaluation of the strategy to switch between the fields $\mathbf{\Phi}$, which converges to the curve $\mathcal{C}$, and $\mathbf{\Psi}$, which contours the obstacle.

Now, regarding the theory presented in Chapter 4, future research can also be done to improve the performance of the vector field based quadcopter controller we propose. The improvement could be the consideration of a more complete vehicle model, which does not neglect the rotational dynamics. This model may also include, for instance, the gyroscopic effects of the propellers and other effects. Incorporation of integral action may also be a good improvement to make the system reject constant disturbances. Regarding the experiments with the ESPcopter, the setup can be improved to allow higher speeds in the curve following. The improvement of the lower level PID controllers, responsible for imposing the acro rate commands, may be a good starting point. These new experiments may be a good way to better evaluate the performance of the controller and inspire new ideas to improve it.

References

[1] B. Abdurahman, A. Savvaris, and A. Tsourdos. A comparison between guidance laws for AUVs using relative kinematics. In *OCEANS 2017 - Aberdeen*, pages 1–6, June 2017.

[2] A Pedro Aguiar, João P Hespanha, and Petar V Kokotović. Performance limitations in reference tracking and path following for nonlinear systems. *Automatica*, 44(3):598–610, 2008.

[3] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. De Lellis, and A. Pironti. Path generation and tracking in 3-D for UAVs. *IEEE Transactions on Control Systems Technology*, 17(4):980–988, July 2009.

[4] T. S. Andersen and R. Kristiansen. Quaternion path-following in three dimensions for a fixed-wing UAV using quaternion blending. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1597–1602, August 2018.

[5] Oualid Araar and Nabil Aouf. Visual servoing of a quadrotor UAV for autonomous power lines inspection. In *22nd Mediterranean Conference on Control and Automation*, pages 1418–1424. IEEE, 2014.

[6] H. J. Asl, T. Narikiyo, and M. Kawanishi. Prescribed performance velocity field control of robotic exoskeletons with neural network. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2704–2709, Dec 2017.

[7] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton University Press, 2012.

[8] Marcel Berger and Bernard Gostiaux. *Differential Geometry: Manifolds, Curves, and Surfaces*, volume 115. Springer Science & Business Media, 2012.

[9] Sylvain Bertrand, Nicolas Guénard, Tarek Hamel, Hélène Piet-Lahanier, and Laurent Eck. A hierarchical controller for miniature VTOL UAVs: Design and stability analysis using singular perturbation theory. *Control Engineering Practice*, 19(10):1099 – 1108, 2011.

[10] M. Breivik and T. I. Fossen. Principles of guidance-based path following in 2D and 3D. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 627–634, 2005.

[11] D. Brescianini and R. D'Andrea. Tilt-prioritized quadrocopter attitude control. *IEEE Transactions on Control Systems Technology*, 28(2):376–387, 2020.

[12] D. Cabecinhas, R. Cunha, and C. Silvestre. Rotorcraft path following control for extended flight envelope coverage. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 3460–3465, 2009.

[13] D. Cabecinhas, R. Cunha, and C. Silvestre. A globally stabilizing path following controller for rotorcraft with wind disturbance rejection. *IEEE Transactions on Control Systems Technology*, 23(2):708–714, 2015.

[14] David Cabecinhas, Rita Cunha, and Carlos Silvestre. A nonlinear quadrotor trajectory tracking controller with disturbance rejection. *Control Engineering Practice*, 26:1 – 10, 2014.

[15] Daniel N. Cardoso, Sergio Esteban, and Guilherme V. Raffo. A robust optimal control approach in the weighted Sobolev space for underactuated mechanical systems. *Automatica*, 125:109474, 2021.

[16] Nicola Ceccarelli, Mauro Di Marco, Andrea Garulli, and Antonio Giannitrapani. Collective circular motion of multi-vehicle systems. *Automatica*, 44(12):3025–3035, 2008.

[17] Namhoon Cho, Youdan Kim, and Sanghyuk Park. Three-dimensional nonlinear differential geometric path-following guidance law. *Journal of Guidance, Control, and Dynamics*, 38(12):2366–2385, 2015.

[18] Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor, and Wolfram Burgard. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[19] Venanzio Cichella, Ronald Choe, S. Bilal Mehdi, Enric Xargay, Naira Hovakimyan, Isaac Kaminer, and Vladimir Dobrokhodov. A 3D path-following approach for a multirotor UAV on SO(3). *IFAC Proceedings Volumes*, 46(30):13 – 18, 2013. 2nd IFAC Workshop on Research, Education and Development of Unmanned Aerial Systems.

[20] Sedat Dogru and Lino Marques. An improved kinematic model for skid-steered wheeled platforms. *Autonomous Robots*, 45(2):229–243, 2021.

[21] Petar Durdevic, Daniel Ortiz-Arroyo, Shaobao Li, and Zhenyu Yang. Vision aided navigation of a quad-rotor for autonomous wind-farm inspection. *IFAC-PapersOnLine*, 52(8):61–66, 2019.

[22] David Eberly. Quaternion algebra and calculus. *Magic Software Inc*, 26, 2002.

[23] M. Faessler, A. Franchi, and D. Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2018.

[24] E. W. Frew and D. Lawrence. Tracking expanding star curves using guidance vector fields. In *2012 American Control Conference (ACC)*, pages 1749–1754, June 2012.

[25] Eric W. Frew and Dale Lawrence. Tracking dynamic star curves using guidance vector fields. *Journal of Guidance, Control, and Dynamics*, 40(6):1488–1495, 2017.

[26] Eric W. Frew, Dale A. Lawrence, and Steve Morris. Coordinated standoff tracking of moving targets using lyapunov guidance vector fields. *Journal of Guidance, Control, and Dynamics*, 31(2):290–306, 2008.

[27] V. M. Gonçalves, L. C. A. Pimenta, C. A. Maia, B. C. O. Dutra, and G. A. S. Pereira. Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4):647–659, Aug 2010.

[28] V. M. Gonçalves, L. C. A. Pimenta, C. A. Maia, G. A. S. Pereira, B. C. O. Dutra, N. Michael, J. Fink, and V. Kumar. Circulation of curves using vector fields: Actual robot experiments in 2D and 3D workspaces. In *2010 IEEE International Conference on Robotics and Automation*, pages 1136–1141, May 2010.

[29] V. M. Gonçalves, B. V. Adorno, A. Crosnier, and P. Fraisse. Stable-by-design kinematic control based on optimization. *IEEE Transactions on Robotics*, 36(3):644–656, 2020.

[30] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Robotics Research*, pages 235–252. Springer, 2017.

[31] Alexander Jahn and Luciano C.A. Pimenta. Sampling based path planning and vector fields for curve tracking by UAVs. In *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pages 223–228, 2016.

[32] I. Kaminer, O. Yakimenko, A. Pascoal, and R. Ghabcheloo. Path generation, path following and coordinated control for timecritical missions of multiple UAVs. In *2006 American Control Conference*, pages 4906–4913, 2006.

[33] I. Kamon, E. Rivlin, and E. Rimon. A new range-sensor based globally convergent navigation algorithm for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 429–435 vol.1, 1996.

[34] Yuri A. Kapitanyuk, Anton V. Proskurnikov, and Ming Cao. A guiding vector-field algorithm for path-following control of nonholonomic mobile robots. *IEEE Transactions on Control Systems Technology*, 26(4):1372–1385, 2018.

[35] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[36] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. *arXiv preprint arXiv:2006.05768*, 2020.

[37] Hassan K Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[38] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[39] Inyoung Ko, Beobkyoon Kim, and Frank Chongwoo Park. Randomized path planning on vector fields. *The International Journal of Robotics Research*, 33(13):1664–1682, 2014.

[40] J. Kwon, C. Kim, and D. Chwa. Vector field trajectory tracking control for wheeled mobile robots. In *2009 IEEE International Conference on Industrial Technology*, pages 1–6, Feb 2009.

[41] Dale Lawrence, Eric Frew, and William Pisano. Lyapunov vector fields for autonomous UAV flight control. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 6317, 2007.

[42] Kian Seng Lee, Mark Ovinis, T Nagarajan, Ralph Seulin, and Olivier Morel. Autonomous patrol and surveillance system using unmanned aerial vehicles. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1291–1297. IEEE, 2015.

[43] T. Lee, M. Leok, and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Dec 2010.

[44] Y. Liang, Y. Jia, J. Du, and J. Zhang. Vector field guidance for three-dimensional curved path following with fixed-wing UAVs. In *2015 American Control Conference (ACC)*, pages 1187–1192, July 2015.

[45] Vladimir J Lumelsky and Alexander A Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1):403–430, 1987.

[46] M. M. Marinho, B. V. Adorno, K. Harada, and M. Mitsuishi. Dynamic active constraints for surgical robots using vector-field inequalities. *IEEE Transactions on Robotics*, 35(5):1166–1185, Oct 2019.

[47] A. Martínez, B. Lawson, C. Durrough, and M. Goldfarb. A velocity-field-based controller for assisting leg movement during walking with a bilateral hip and knee lower limb exoskeleton. *IEEE Transactions on Robotics*, 35(2):307–316, April 2019.

[48] Timothy McLain, Randall W Beard, and Mark Owen. Implementing Dubins airplane paths on fixed-wing UAVs. 2014.

[49] A McNabb. Comparison theorems for differential equations. *Journal of Mathematical Analysis and Applications*, 119(1):417 – 428, 1986.

[50] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, May 2011.

[51] José Luis Mendoza-Soto, José J Corona-Sánchez, and Hugo Rodríguez-Cortés. Quadcopter path following control. A maneuvering approach. *Journal of Intelligent & Robotic Systems*, 93(1):73–84, 2019.

[52] Cun-Xiao Miao and Jian-Cheng Fang. An adaptive three-dimensional nonlinear path following method for a fix-wing micro aerial vehicle. *International Journal of Advanced Robotic Systems*, 9(5):206, 2012.

[53] Alain Micaelli and Claude Samson. *Trajectory tracking for unicycle-type and two-steering-wheels mobile robots*. PhD thesis, INRIA, 1993.

[54] I. B. P. Nascimento, A. Ferramosca, L. C. A. Pimenta, and G. V. Raffo. NMPC strategy for a quadrotor UAV in a 3D unknown environment. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 179–184, 2019.

[55] P. Niermeyer, V. S. Akkinapalli, M. Pak, F. Holzapfel, and B. Lohmann. Geometric path following control for multirotor vehicles using nonlinear model predictive control and 3D spline paths. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 126–134, 2016.

[56] J. L. G. Olavo, G. D. Thums, T. A. Jesus, L. C. de Araújo Pimenta, L. A. B. Torres, and R. M. Palhares. Robust guidance strategy for target circulation by controlled UAV. *IEEE Transactions on Aerospace and Electronic Systems*, 54(3):1415–1431, June 2018.

[57] A. Ollero and G. Heredia. Stability analysis of mobile robot path tracking. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 3, pages 461–466 vol.3, 1995.

[58] Guilherme A. S. Pereira and Elias J. R. Freitas. Navigation of semi-autonomous service robots using local information and anytime motion planners. *Robotica*, 38(11):2080–2098, 2020.

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

[60] Leonardo A. A. Pereira, Arthur H. D. Nunes, Adriano M. C. Rezende, Vinicius M. Gonçalves, Guilherme V. Raffo, and Luciano C. A. Pimenta. Collision-free vector field guidance and MPC for a fixed-wing UAV. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 176–182, 2021.

[61] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press, 2007.

[62] Guilherme V Raffo and Marcelino M de Almeida. Nonlinear robust control of a quadrotor UAV for load transportation with swing improvement. In *2016 American Control Conference (ACC)*, pages 3156–3162. IEEE, 2016.

[63] A. M. C. Rezende, V. M. Gonçalves, A. H. D. Nunes, and L. C. A. Pimenta. Robust quadcopter control with artificial vector fields. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6381–6387, 2020.

[64] A. M. C. Rezende, V. M. Gonçalves, G. V. Raffo, and L. C. A. Pimenta. Robust fixed-wing UAV guidance with circulating artificial vector fields. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5892–5899, Oct 2018.

[65] A. M. C. Rezende, G. P. C. Júnior, R. Fernandes, V. R. F. Miranda, H. Azpúrua, G. Pessin, and G. M. Freitas. Indoor localization and navigation control strategies for a mobile robot designed to inspect confined environments. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1427–1433, 2020.

[66] A. M. C. Rezende, V. R. F. Miranda, H. N. Machado, A. C. B. Chiella, V. M. Gonçalves, and G. M. Freitas. Autonomous system for a racing quadcopter. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 547–552, 2019.

[67] Adriano M. C. Rezende, Vinicius M. Goncalves, and Luciano C. A. Pimenta. Constructive time-varying vector fields for robot navigation. *IEEE Transactions on Robotics*, pages 1–16, 2021.

[68] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016.

[69] Ashton Roza and Manfredi Maggiore. Path following controller for a quadrotor helicopter. In *2012 American Control Conference (ACC)*, pages 4655–4660. IEEE, 2012.

[70] Bartomeu Rubí, Ramon Pérez, and Bernardo Morcego. A survey of path following control strategies for UAVs focused on quadrotors. *Journal of Intelligent & Robotic Systems*, Nov 2019.

[71] I. Sanchez, A. D'Jorge, A. Ferramosca, G. Raffo, and A. H. Gonzlez. Path following and trajectory tracking model predictive control using artificial variables for constrained vehicles. In *2019 XVIII Workshop on Information Processing and Control (RPIC)*, pages 198–203, 2019.

[72] Ignacio Sánchez, Agustina D'Jorge, Guilherme V Raffo, Alejandro H González, and Antonio Ferramosca. Nonlinear model predictive path following controller with obstacle avoidance. *Journal of Intelligent & Robotic Systems*, 102(1):1–18, 2021.

[73] Thomas Sayre-McCord, Winter Guerra, Amado Antonini, Jasper Arneberg, Austin Brown, Guilherme Cavalheiro, Yajun Fang, Alex Gorodetsky, Dave McCoy, Sebastian Quilter, Fabian Riether, Ezra Tal, Yunus Terzioglu, Luca Carlone, and Sertac Karaman. Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[74] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[75] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[76] Roger Skjetne, Thor I. Fossen, and Petar V. Kokotović. Robust output maneuvering for a class of nonlinear systems. *Automatica*, 40(3):373 – 383, 2004.

[77] S. Su. Path following control of non-minimum phase VTOL aircraft via minimum distance projection method. In *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pages 708–712, 2014.

[78] I. Sánchez, A. Ferramosca, G. Raffo, A. H. González, and A. D'Jorge. Obstacle avoiding path following based on nonlinear model predictive control using artificial variables. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 254–259, 2019.

[79] Maxime Vaidis and Martin J-D Otis. Toward a robot swarm protecting a group of migrants. *Intelligent Service Robotics*, pages 1–16, 2020.

[80] Daniel KD Villa, Alexandre S Brandao, and Mário Sarcinelli-Filho. A survey on load transportation using multirotor UAVs. *Journal of Intelligent & Robotic Systems*, pages 1–30, 2019.

[81] C. Wu, J. Chen, D. Jeltsema, and a. C. Dai. Guidance vector field encoding based on contraction analysis. In *2018 European Control Conference (ECC)*, pages 282–287, June 2018.

[82] Weijia Yao, Hector Garcia de Marina, and Ming Cao. Mobile robot path following control in 2D using a 3D guiding vector field: Singularity elimination and global convergence. *arXiv preprint arXiv:2003.10012*, pages 1–9, 2020.

[83] Weijia Yao, Héctor Garcia de Marina, Bohuan Lin, and Ming Cao. Singularity-free guiding vector field for robot navigation. *IEEE Transactions on Robotics*, 37(4):1206–1221, 2021.

[84] F. Yousefi and S. B. Monfared. Cascade feedback linearization controller with actuators dynamic for trajectory tracking of flying robot. In *2018 8th Conference of AI Robotics and 10th RoboCup Iranopen International Symposium (IRANOPEN)*, pages 46–51, April 2018.

[85] Y. Yu and X. Ding. A global tracking controller for underactuated aerial vehicles: Design, analysis, and experimental tests on quadrotor. *IEEE/ASME Transactions on Mechatronics*, 21(5):2499–2511, Oct 2016.

[86] S. Zhao, X. Wang, Z. Lin, D. Zhang, and L. Shen. Integrating vector field approach and input-to-state stability curved path following for unmanned aerial vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–8, 2018.

[87] D. Zhou and M. Schwager. Vector field following for quadrotors using differential flatness. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6567–6572, May 2014.

Appendix A

Approachability of singular points analysis

Proposition 2 establishes asymptotic stability. However, the field is only defined in $\mathcal{U}$, because outside this set there is no single definition for the vector field, and therefore a natural question is whether or not the trajectory $\mathbf{x}(\mathrm{t})$ approaches these points eventually. It will be established that for the time-invariant case, under some assumptions, no point outside $\mathcal{U}$ is approachable.

Henceforth, we consider the time-invariant vector field $\mathbf{\Phi}(\mathbf{x})$ in which we assume, without loss of generality, that $\mathrm{v_r} = 1$. To handle approachability to points of discontinuity of the vector field, the following convenient definition will be used.

Definition 18. *Given a normalized vector field* $\mathbf{f}(\mathbf{x})$, $\mathbf{f} : \mathbb{R}^{\mathrm{n}} \mapsto \mathbb{R}^{\mathrm{n}}$ *the point* $\mathbf{x}$ *is said to be* approachable *by the direction* $\mathbf{d}$, $\|\mathbf{d}\| = 1$, *if the limit*

$$\lim_{\epsilon \to 0+} \mathbf{f}(\mathbf{x} - \epsilon \mathbf{d}) \tag{A.1}$$

exists and is equal to $\mathbf{d}$.

Figure A.1 shows the intuition behind the result that will be derived in this subsection. Let us assume, for the sake of simplicity, that the vector field $\mathbf{\Phi}$ has only the convergent component ($\mathrm{G} = 1$, $\mathrm{H} = 0$) and therefore always points towards the closest point. If point $\mathbf{x}_{\mathrm{P}} \notin \mathcal{U}$ is approachable, the derivative of the trajectory just before reaching the point should be equal to one of the possible values of $\mathbf{\Phi}(\mathbf{x}_{\mathrm{P}})$. Each one of these possible values is attached to a point in $\mathbf{x}^*(\mathbf{x}_{\mathrm{P}})$. Let $\mathbf{m}$ be one of these points and $\mathbf{\Phi}(\mathbf{x}_{\mathrm{P}}; \mathbf{m})$ the corresponding vector. The situation described is shown in the left figure. However, the right figure shows that the situation shown in the left is a contradiction: if we move backward a little in the trajectory (in the opposite direction of $\mathbf{\Phi}(\mathbf{x}_{\mathrm{P}}; \mathbf{m})$), we see that the corresponding vector field is attached to *another* point in $\mathbf{x}^*$: $\mu(\mathbf{x}_{\mathrm{P}}; \mathbf{m})$, because when we move backward a little we become closer to this point than to $\mathbf{m}$. As we will show, this different point has an associated $\mathbf{\Phi}$ which is different from the one related to $\mathbf{m}$. Therefore, the trajectory on the left could not have been created by following the vector field. As the picture shows, this is always the case when $\mathrm{G} = 1$. The analysis that will

be presented in this section will establish sufficient conditions to affirm that these points are not approachable for the case in which G ≠ 1.

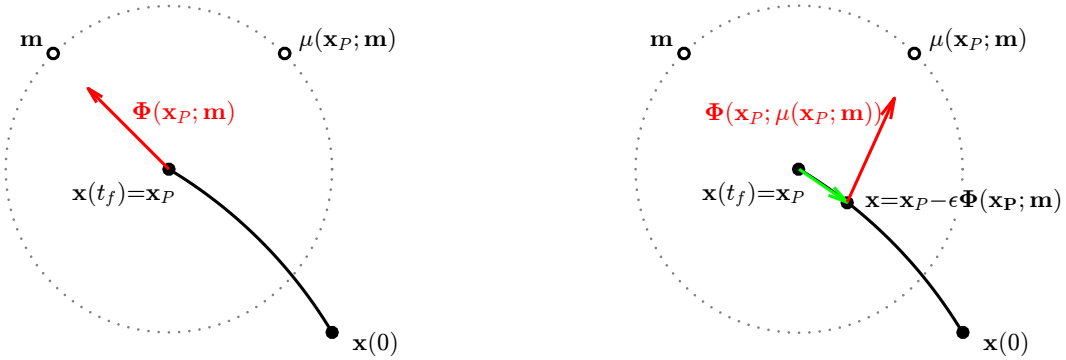

Figure A.1: Intuition behind our approachability analysis. In the figure, we assume, for the sake of simplicity, that the vector field $\mathbf{\Phi}$ has only the convergent component (H = 0) and therefore always points towards the closest point.

If the point $\mathbf{x}_\mathrm{P}$ is approachable by the direction $\mathbf{d}$, then by following the vector field, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, it is possible to have a trajectory $\mathbf{x}(\mathrm{t})$ and a *finite* time $\mathrm{t_P}$ such that $\mathbf{x}(\mathrm{t_P}) = \mathbf{x}_\mathrm{P}$ and $\lim_{\mathrm{t}\to\mathrm{t_P}^-} \dot{\mathbf{x}}(\mathrm{t}) = \mathbf{d}$. In other words, there is a trajectory that approaches the point $\mathbf{x}_\mathrm{P}$ from the direction $\mathbf{d}$. If $\mathbf{f}(\mathbf{x})$ is continuous in $\mathbf{x}$, then it is trivially approachable by the direction $\mathbf{d} = \mathbf{f}(\mathbf{x})$, and only from this direction. However, if it is not continuous, it may or may not be approachable. For instance, it is easy to see using the definition that for $\mathbf{f}_1(\mathbf{x}) = -\mathbf{x}/\|\mathbf{x}\|$ the point $\mathbf{x} = \mathbf{0}$ is approachable by any $\mathbf{d}$ (it is possible to reach the point $\mathbf{0}$ by following the vector field), whereas $\mathbf{f}_2(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$ is not approachable by any $\mathbf{d}$: the point $\mathbf{x} = \mathbf{0}$ is "unstable". Thus, the inaccessibility of a point in a vector field can be proved by establishing that it is not approachable by any direction.

We will need the following definition.

Definition 19. *Given a normalized vector field* $\mathbf{f}(\mathbf{x})$*, the* limit set *at a point* $\mathbf{x}$*,* $\mathcal{R}_\mathbf{f}(\mathbf{x})$*, is the set of all different values the vector field* $\mathbf{f}(\mathbf{x})$ *can have when we approach* $\mathbf{x}$ *from all the possible directions. That is,* $\mathcal{R}_\mathbf{f}(\mathbf{x}) = \cup_{\mathbf{e}\in\mathcal{E}(\mathbf{x})} \lim_{\epsilon\to 0+} \{\mathbf{f}(\mathbf{x} - \epsilon\mathbf{e})\}$*, in which* $\mathcal{E}(\mathbf{x}) \in \mathbb{R}^\mathrm{n}$ *is the set of directions* $\mathbf{e}$ *such that the limit exists.*

The next result follows from our definitions.

Lemma 15. *If a point* $\mathbf{x}$ *is approachable from a direction* $\mathbf{d}$*, then* $\mathbf{d} \in \mathcal{R}_\mathbf{f}(\mathbf{x})$*.*

Proof. Follows from the definition of approachability: if $\mathbf{x}$ is approachable from a direction $\mathbf{d}$, $\lim_{\epsilon\to 0+} \mathbf{f}(\mathbf{x} - \epsilon\mathbf{d}) = \mathbf{d}$. Since $\mathcal{R}_\mathbf{f}(\mathbf{x})$ is the set of *all* vectors that can appear in this limit and $\mathbf{d}$ is equal to one of them, the result holds. □

Definition 20. *Consider the time-invariant vector field* $\mathbf{\Phi}(\mathbf{x})$*. For* $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$ *and* $\mathbf{y} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$*, we define*

$$\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{y}) = -\hat{\mathrm{G}}(\mathbf{x}_\mathrm{P}) \left(\frac{\mathbf{x}_\mathrm{P} - \mathbf{y}}{\mathrm{D}(\mathbf{x}_\mathrm{P})} \right) + \hat{\mathrm{H}}(\mathbf{x}_\mathrm{P}) \mathbf{T}^*(\mathbf{y}). \tag{A.2}$$

The vector $\mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y})$ is just one of the possible values of the vector field $\mathbf{\Phi}(\mathbf{x})$ when approaching $\mathbf{x}_\mathrm{P}$ in different directions, given that $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$. Thus

$$\mathcal{R}_{\mathbf{\Phi}}(\mathbf{x}_\mathrm{P}) = \bigcup_{\mathbf{y}\in\mathbf{x}^*(\mathbf{x}_\mathrm{P})} \{\mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y})\}. \tag{A.3}$$

equation (A.3) means that in the neighborhood of point $\mathbf{x}_\mathrm{P}$ the vectors in $\mathcal{R}_{\mathbf{\Phi}}(\mathbf{x}_\mathrm{P})$ are the ones computed by using each possible closest point projection $\mathbf{x}^*$. Clearly, if a point $\mathbf{x} \in \mathcal{U}$ we have that $\mathcal{R}_{\mathbf{\Phi}}(\mathbf{x}) = \{\mathbf{\Phi}(\mathbf{x})\}$.

Lemma 16. *Let $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$. Given the field $\mathbf{\Phi}(\mathbf{x})$, if the point $\mathbf{x}_\mathrm{P}$ is approachable, then it should be by a direction $\mathbf{d} = \mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y})$ for $\mathbf{y} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$.*

Proof. This comes directly from Lemma 15 and equation (A.3). □

Definition 21. *Let $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$ and $\mathbf{y}, \mathbf{z} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$. Let also the operator $[\mathrm{u}]_+ = \max(\mathrm{u}, 0)$. Define*

$$\Gamma(\mathbf{x}_\mathrm{P};\mathbf{y}) = \min_{\mathbf{z}\in\mathbf{x}^*(\mathbf{x}_\mathrm{P})} \frac{2\mathrm{D}(\mathbf{x}_\mathrm{P})[(\mathbf{z}-\mathbf{y})^\mathrm{T}\mathbf{T}^*(\mathbf{y})]_+}{\|\mathbf{y}-\mathbf{z}\|^2}. \tag{A.4}$$

Finally, define $\gamma(\mathbf{x}_\mathrm{P};\mathbf{y})$ as one of its minimizers.

Note that $\gamma(\mathbf{x}_\mathrm{P};\mathbf{y})$ is bounded because we can always choose $\mathbf{z} \neq \mathbf{y}$. This also implies that $\gamma(\mathbf{x}_\mathrm{P};\mathbf{y}) \neq \mathbf{y}$.

Lemma 17. *Let $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$. Suppose*

$$\frac{\hat{\mathrm{G}}(\mathbf{x}_\mathrm{P})}{\hat{\mathrm{H}}(\mathbf{x}_\mathrm{P})} > \Gamma(\mathbf{x}_\mathrm{P};\mathbf{y}) \tag{A.5}$$

for $\mathbf{y} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$ and that $\mathbf{x}_\mathrm{P}$ is approachable.

Then the optimization problem

$$\min_{\mathbf{z}\in\mathbf{x}^*(\mathbf{x}_\mathrm{P})} \lim_{\epsilon\to 0+} \|(\mathbf{x}_\mathrm{P} - \epsilon\mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y})) - \mathbf{z}\|^2 \tag{A.6}$$

has a unique minimizer and $\mathbf{z} \neq \mathbf{y}$.

Proof. Unicity: First, note that equation (A.6) is basically the distance between the point $(\mathbf{x}_\mathrm{P} - \epsilon\mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y}))$ and the curve, because in this case, when $\epsilon \to 0+$, the closest point to the curve must lie on $\mathbf{x}^*(\mathbf{x}_\mathrm{P})$. Now, by definition, if $\mathbf{x}_\mathrm{P}$ is approachable, the limit $\mathbf{\Phi}(\mathbf{x}_\mathrm{P} - \epsilon\mathbf{d})$ must exist. This limit exists if and only if the limit $\mathbf{x}^*(\mathbf{x}_\mathrm{P} - \epsilon\mathbf{d})$ exists. The possible values for this limit are given by the minimizers in equation (A.6), therefore, there should be only one minimizer in order to the limit to exist.

Not $\mathbf{y}$: We will show that there is another $\mathbf{z}$ with a strictly smaller objective function. We will show that, given equation (A.5), with $\mathbf{z} = \gamma(\mathbf{x}_\mathrm{P};\mathbf{y}) \neq \mathbf{y}$ we obtain a smaller value in equation (A.6) than with $\mathbf{z} = \mathbf{y}$. This statement is equivalent to:

$$\|(\mathbf{x}_\mathrm{P} - \epsilon\mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y})) - \mathbf{y}\| > \|(\mathbf{x}_\mathrm{P} - \epsilon\mathbf{\Phi}(\mathbf{x}_\mathrm{P};\mathbf{y})) - \gamma(\mathbf{x}_\mathrm{P};\mathbf{y})\|. \tag{A.7}$$

This relation is obtained by comparing the cost function in (A.6) with $\mathbf{z} = \mathbf{y}$ and with $\mathbf{z} = \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})$. After some simplifications, this statement can be written as

$$(\mathbf{y} - \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}))^\mathrm{T} \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{y}) > 0. \tag{A.8}$$

Now, let $\mathrm{G_P}{=}\hat{\mathrm{G}}(\mathbf{x}_\mathrm{P})$, $\mathrm{H_P}{=}\hat{\mathrm{H}}(\mathbf{x}_\mathrm{P})$, and $\mathrm{D_P}{=}\mathrm{D}(\mathbf{x}_\mathrm{P})$. Using Definition 20 and after some algebra, we can see that the previous equation is equivalent to

$$\mathrm{G_P} \frac{\|\mathbf{y} - \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})\|^2}{2\mathrm{D_P}} + \mathrm{H_P}(\mathbf{y} - \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}))^\mathrm{T} \mathbf{T}^*(\mathbf{y}) > 0. \tag{A.9}$$

To obtain this, we develop the expression $\|\gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}) - \mathbf{y} + \mathbf{y} - \mathbf{x}_\mathrm{P}\|^2$ and use $\|\mathbf{y} - \mathbf{x}_\mathrm{P}\| = \|\gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}) - \mathbf{x}_\mathrm{P}\|$ to obtain the equivalence $(\mathbf{x}_\mathrm{P} - \mathbf{y})^\mathrm{T}(\mathbf{y} - \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})) = -\|\mathbf{y} - \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})\|^2/2$. By noting that the first term in the left-hand side of (A.9) is strictly positive, we see that condition in (A.9) is equivalent to:

$$\frac{\mathrm{G_P}}{\mathrm{H_P}} > 2\mathrm{D_P} \frac{(\gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}) - \mathbf{y})^\mathrm{T} \mathbf{T}^*(\mathbf{y})}{\|\mathbf{y} - \gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})\|^2}. \tag{A.10}$$

This condition is guaranteed by equation (A.5). Note that if $(\gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}) - \mathbf{y})^\mathrm{T} \mathbf{T}^*(\mathbf{y}) < 0$, the condition is trivially satisfied since $\mathrm{G_P}/\mathrm{H_P} \geq 0$. This justifies the use of the $[\mathrm{u}]_+$ operator in the definition of $\Gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})$. □

Based on Lemma 17, we can define the following

Definition 22. *For $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$ and when equation (A.5) holds, we define $\mu(\mathbf{x}_\mathrm{P}; \mathbf{y})$ as the unique minimizer of equation (A.6).*

Lemma 18. *Let $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$. Suppose $\mathbf{x}_\mathrm{P}$ is approachable by a direction and*

$$\frac{\hat{\mathrm{G}}(\mathbf{x_P})}{\hat{\mathrm{H}}(\mathbf{x_P})} > \max\Big(\Gamma(\mathbf{x}_\mathrm{P}; \mathbf{m}), \Gamma(\mathbf{x}_\mathrm{P}; \mu(\mathbf{x}_\mathrm{P}; \mathbf{m}))\Big) \tag{A.11}$$

holds for a $\mathbf{m} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$. Then the point $\mathbf{x}_\mathrm{P}$ is not approachable by the direction $\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m})$.

Proof. Note that equation (A.11) implies equation (A.5) with $\mathbf{y} = \mathbf{m}$. This, together with the assumption that $\mathbf{x}_\mathrm{P}$ is approachable, implies that Lemma 17 holds.

From Lemma 17, we see that when we approach $\mathbf{x}_\mathrm{P}$ by the direction $\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m})$, the closest point to $\mathbf{x}_\mathrm{P}$ is not $\mathbf{m}$, but $\mu(\mathbf{x}_\mathrm{P}; \mathbf{m}) \neq \mathbf{m}$. Henceforth, we will write for the sake of simplicity $\mu = \mu(\mathbf{x}_\mathrm{P}; \mathbf{m})$. Therefore

$$\lim_{\epsilon \to 0+} \mathbf{\Phi}(\mathbf{x}_\mathrm{P} - \epsilon \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m})) = \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mu). \tag{A.12}$$

It remains to prove that $\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m}) \neq \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mu)$, *i.e.*, $\mathbf{x}_\mathrm{P}$ is not approachable by the direction $\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m})$. We will show by contradiction that $\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m}) = \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mu)$ does not

hold. Suppose it is true. Note that equation (A.11) implies equation (A.5) not only with $\mathbf{y} = \mathbf{m}$ but also with $\mathbf{y} = \mu$ as well. Consequently, Lemma 17 also holds when $\mathbf{y} = \mu$.

Using $\mathbf{y} = \mathbf{m}$ on Lemma 17, we conclude that μ is the unique minimizer for equation (A.6) with $\mathbf{y} = \mathbf{m}$. Furthermore, since we are supposing that $\mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m}) = \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mu)$, we conclude that μ is also a minimizer for equation (A.6) with $\mathbf{y} = \mu$.

But using $\mathbf{y} = \mu$ on Lemma 17, we conclude that the minimizer for equation (A.6) with $\mathbf{y} = \mu$ can not be μ. This is a contradiction. □

We then present a definition and the final result.

Definition 23. *We define, for* $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$

$$\mathrm{K}(\mathbf{x}_\mathrm{P}) = \max_{\mathbf{y} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})} \Gamma(\mathbf{x}_\mathrm{P}; \mathbf{y}). \tag{A.13}$$

Note that $\mathrm{K}(\mathbf{x}_\mathrm{P})$ is finite, since $\Gamma(\mathbf{x}_\mathrm{P}; \mathbf{y})$ is finite for all $\mathbf{y}$. We can then state our final result.

Proposition 6. *Suppose that*

$$\frac{\hat{\mathrm{G}}(\mathbf{x_P})}{\hat{\mathrm{H}}(\mathbf{x_P})} > \mathrm{K}(\mathbf{x}_\mathrm{P}) \quad \forall\ \mathbf{x}_\mathrm{P} \notin \mathcal{U}, \tag{A.14}$$

holds. Then the point $\mathbf{x}_\mathrm{P}$ *is not approachable by* any *direction.*

Proof. The proof follows by contradiction. Suppose it is approachable by a direction. Then it should be by a direction $\mathbf{d} = \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m})$ (see Lemma 16) for a $\mathbf{m} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$.

Equation (A.14) implies equation (A.11) for any $\mathbf{m} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$, because we are taking the maximum overall possible $\mathbf{y} \in \mathbf{x}^*(\mathbf{x}_\mathrm{P})$, including $\mathbf{m}$ and $\mu(\mathbf{x}_\mathrm{P}; \mathbf{m})$. Since $\mathbf{x}_\mathrm{P}$ is also assumed to be approachable, we can use Lemma 18 to conclude that this should not be by the direction $\mathbf{d} = \mathbf{\Phi}(\mathbf{x}_\mathrm{P}; \mathbf{m})$. This is a contradiction, and therefore the proof is established. □

For any $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$, equation (A.14) can be written, together with the fact that $\mathrm{G}^2 + \mathrm{H}^2 = 1$, as

$$\hat{\mathrm{G}}(\mathbf{x}_\mathrm{P}) > \frac{\mathrm{K}(\mathbf{x}_\mathrm{P})}{\sqrt{\mathrm{K}(\mathbf{x}_\mathrm{P})^2 + 1}}. \tag{A.15}$$

One could wonder whether or not (A.15) can always be achievable for any $\mathbf{x}_\mathrm{P} \notin \mathcal{U}$, since $\hat{\mathrm{G}}(\mathbf{x}_\mathrm{P}) \leq 1$. However, the righthand side of this inequality is always strictly less than 1, because $\mathrm{K}(\mathbf{x}_\mathrm{P})$ is finite, therefore, there always exists a G that achieves this condition.

Appendix B

Vector field in the Torus space

The state-space formed by the rotation of each joint of a robotic manipulator does not have a Euclidean topology: a rotation of θ degrees is the same as one of $\theta + 2\pi k$ for all $k \in \mathbb{Z}$. For this reason, the Euclidean distance is not an appropriate metric for this space. Indeed, a closed curve in the correct topological space of $\mathbf{x}$ (that is of the n-dimensional torus, $\mathbb{S}^n = \mathbb{S} \times \mathbb{S} ... \times ... \mathbb{S}$) can become open and bounded when mapped to the Euclidean space.

This problem can be addressed by a "replication" trick. Suppose we have the target curve $\mathcal{C}$, $\mathbf{r}(s)$, static in time, as an open, bounded curve in $\mathbb{R}^n$. Assume that $s = 0$ and $s = s_{end}$ are the starting and ending parametrization of the curve. To be a closed curve in the n-dimensional torus space, it must hold that for all $i = 1, 2, .., n$, it holds that $|r_i(s_{end}) - r_i(0)| = 2\pi k_i \equiv T_i$, for a nonnegative integer k_i (possibly different for all i). If the curve is differentiable, it also holds that $\frac{\partial r_i}{\partial s}(0) = \frac{\partial r_i}{\partial s}(s_{end})$. Thus, the target curve can be bounded in a hyperrectangle in $\mathbb{R}^n$ whose sides are integer multiples of 2π.

Suppose this curve is replicated across $\mathbb{R}^n$ by summing to $\mathbf{r}(s)$ constant vectors Δ in which Δ_i is an integer multiple of T_i. See Figure B.1 for an example of this replication. Now, the space $\mathbb{R}^n$ is filled with (disjoint) copies of $\mathcal{C}$, which we call $\mathcal{C}_{rep}$ and we can now apply the traditional Euclidean distance towards $\mathcal{C}_{rep}$ because it already takes into consideration characteristics of the original topology of the state-space (through replication). Note that, after replication, the curve will not be in accordance with Assumption 1, which is now relaxed to incorporate this new type of curve. We now consider curves $\mathcal{C}$ that consist of the union of disjoint curves of the previously considered types.

Of course, it is not necessary to consider directly the (Euclidean) distance towards $\mathcal{C}_{rep}$, which is formed by disjoint, non-bounded curves, to compute the (Euclidean) distance between a state $\mathbf{x}$ and $\mathcal{C}_{rep}$. We can compute it from the original version, $\mathcal{C}$, by solving the following optimization problem not only in the variable s but also in the integer variables k_i:

$$D(\mathbf{x})^2 = \min_{\substack{s \in [0, s_{end}] \\ k_i \in \mathbb{Z}}} \sum_{i=1}^{n} \Big(x_i - r_i(s) - k_i T_i \Big)^2. \tag{B.1}$$

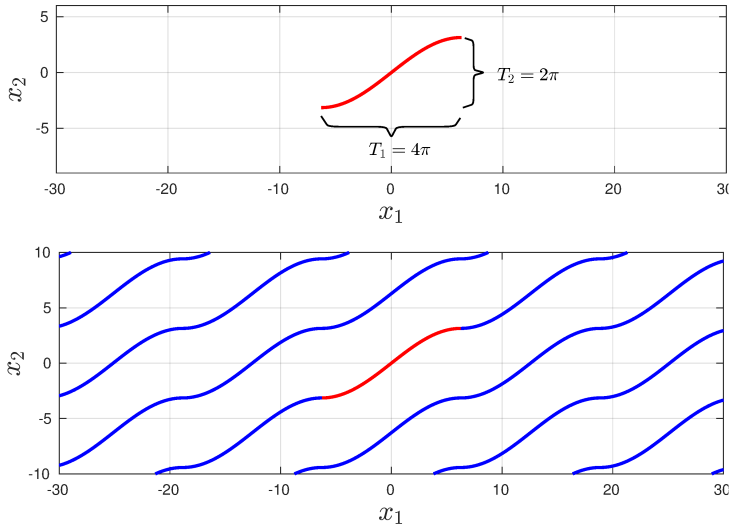

Figure B.1: Replication process for a bidimensional curve (top), bounded in a rectangle with sides $T_1 = 4\pi$ in x_1 and $T_2 = 2\pi$ in x_2. Note also that the derivatives agree at the end-points. These facts imply that the curve in the bidimensional torus $\mathbb{S}^2$ is closed and differentiable.

We can rewrite this problem as:

$$D(\mathbf{x})^2 = \min_{s\in[0,s_{end}]} \sum_{i=1}^{n} \min_{k_i\in\mathbb{Z}} \Big(x_i - r_i(s) - k_i T_i\Big)^2. \tag{B.2}$$

We now can easily solve the internal problems analytically. Let $\lfloor u \rfloor$ be the *floor* function and $\mathrm{rem}(\theta, T) = \theta - T\lfloor \theta/T \rfloor$ be the *remainder* function. Define the function $D_{\mathbb{S}}(\theta, T)$ as:

$$D_{\mathbb{S}}(\theta, T) = \min_{k\in\mathbb{Z}} |\theta - kT| = \begin{cases} \mathrm{rem}(\theta, T) & \text{if } \mathrm{rem}(\theta, T) \leq T/2, \\ T - \mathrm{rem}(\theta, T) & \text{otherwise.} \end{cases} \tag{B.3}$$

which is, by the way, a continuous function. Now, we can rewrite $D(\mathbf{x})$ as:

$$D(\mathbf{x})^2 = \min_{s\in[0,s_{end}]} \sum_{i=1}^{n} D_{\mathbb{S}}\Big(x_i - r_i(s), T_i\Big)^2. \tag{B.4}$$

Thus, we reduced our problem, a search in infinite unbounded curves, to a search in the original (bounded) curve $\mathcal{C}$. This construction also induces an expression for the closest point map $\mathbf{x}^*(\mathbf{x})$. Define the function $k^*_{\mathbb{S}}(\theta, T)$ as:

$$k^*_{\mathbb{S}}(\theta, T) = \arg\min_{k\in\mathbb{Z}} |\theta - kT| = \begin{cases} \lfloor \theta/T \rfloor & \text{if } \mathrm{rem}(\theta, T) < T/2, \\ \lfloor \theta/T \rfloor + 1 & \text{if } \mathrm{rem}(\theta, T) > T/2, \\ \text{either} & \text{if } \mathrm{rem}(\theta, T) = T/2. \end{cases} \tag{B.5}$$

Then, if $s^*_{red}(\mathbf{x})$ is the optimal s found by solving the optimization problem in equation (B.4), we can write each component of $\mathbf{x}^*(\mathbf{x})$, $x^*_i(\mathbf{x})$, as:

$$x^*_i(\mathbf{x}) = r_i\Big(s^*_{red}(\mathbf{x})\Big) + T_i k^*_{\mathbb{S}}\Big(x_i - r_i\Big(s^*_{red}(\mathbf{x})\Big), T_i\Big). \tag{B.6}$$

Note that this construction implies that the source of problematic points, points not in $\mathcal{U}$, are twofold: first, when $s^*_{red}(\mathbf{x})$ is not uniquely defined, and second, when $\mathrm{rem}(x_i - r_i, T_i) = T_i/2$, in which case $k^*_{\mathbb{S}}(x_i - r_i, T)$ has two possible choices, implying that there is more than one choice for the closest point.

Appendix C

Software registration

The following page is the Certificate of Computer Program Registration associated with the *vectorfield_stack* library described in Section 1.5. It is emitted by the Brazilian National Institute of Industrial Property (*Institudo Nacional da Propriedade Industrial*).

The authors of the software are listed below:

- Adriano Martins da Costa Rezende
- Victor Ricardo Fernandes Miranda
- Luciano Cunha de Araújo Pimenta
- Vinicius Mariano Gonçalves
- Gustavo Medeiros Freitas
- Héctor Azpúrua
- Gustavo Pessin

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022000695-2**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 22/12/2021, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: VectorField_stack - Controle de Navegação de Robôs Móveis por Campos Vetoriais Artificiais

Data de publicação: 22/12/2021

Data de criação: 12/11/2021

Titular(es): UNIVERSIDADE FEDERAL DE MINAS GERAIS - UFMG; VALE S.A.

Autor(es): GUSTAVO PESSIN; GUSTAVO MEDEIROS FREITAS; ADRIANO MARTINS DA COSTA REZENDE; VICTOR RICARDO FERNANDES MIRANDA; LUCIANO CUNHA DE ARAUJO PIMENTA; VINICIUS MARIANO GONÇALVES; HÉCTOR AZPÚRUA

Linguagem: PYTHON

Campo de aplicação: IN-02

Tipo de programa: AP-03; AT-01; SM-02

Algoritmo hash: SHA-512

Resumo digital hash:
435d679de1514d9c7a8129505caf7bc757b68e447a73e8574f88936dd03297f432958f8e5868d261adc49008204b09d1685b4c1f29629cda6c6e86320b50b7f7

Expedido em: 05/04/2022

Aprovado por:
Joelson Gomes Pequeno
Chefe Substituto da DIPTO - PORTARIA/INPI/DIRPA Nº 02, DE 10 DE FEVEREIRO DE 2021