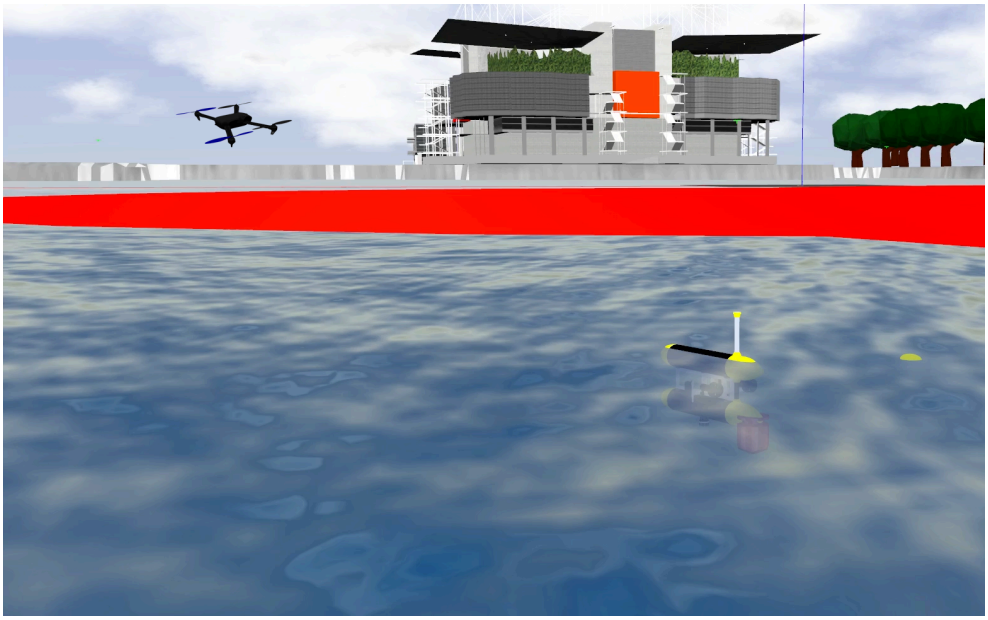
TÉCNICO LISBOA



Cooperative Motion Control of Aerial and Marine Vehicles for Environmental Applications

Marcelo Fialho Jacinto

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Rita Maria Mendes de Almeida Correia da Cunha
Prof. António Manuel dos Santos Pascoal

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Member of the Committee: Prof. João Manuel Lage de Miranda Lemos

December 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practises of the Universidade de Lisboa.

This thesis is dedicated to my parents.

Acknowledgements

This dissertation is the pinnacle of many years of continuous learning. I would like to dedicate this stepping stone in my life to my parents, who always made sure my education was their top priority, one they were never granted. Without their guidance I would have gotten loss countless times. Secondly I would like to thank to my close family for all their support throughout these years. Moreover, I would like to thank Maria de Sá who always cared for me like a grandmother.

I would like to express my gratitude to Nelson Alves for always giving me great advice during the car rides to Lisbon at Sunday nights and to my friend Alexandre Lopes for the long walks in those days I though I would not be able to finish this degree.

I remember like it was yesterday, my first semester at this institute. I was told that the 5 years ahead would be some of the most challenging years of my life - they were. In that same period of time I met three other brilliant gentleman with whom I shared blood, sweat and tears: João Silva, Rui Castro and Jorge Simões, it has been a hell of a ride.

I would like to thank professor António Pascoal for showing me, while still in my bachelor's, how beautiful and fun control theory can be and for introducing me to the DSOR group early on. I believe his enthusiasm for the area was one of the main reasons I chose this major in the first place. I would also like to thank professor Rita Cunha for embarking in this journey and always being available to provide thoughtful insights when something was not working as desired.

Moreover, I want to express my deep gratitude to the entire DSOR team: Luís Sebastião, Helena Santana, Manuel Rufino, João Botelho, and David Cabecinhas with a special mention to Nguyen Hung, João Quintas, João Cruz, David Souto and Francisco Rego who were always kind enough to spare some time to discuss new (some times lunatic) ideas, teach me new concepts, review this thesis and, help me improve and run the heavy simulations presented in this work.

As a final remark, I would like to thank Fundação para a Ciência e Tecnologia (FCT) through ISR under FCT [UIDP / 50009 / 2020] and through the FCT projects H2020 EU Marine Robotics Research Infrastructure Network [ID731103] and H2020 FET Proactive EU RAMONES [No.101017808] for partially funding this work.

"Só de pensar como isto começou, esta epopeia."

Someone in the 8^{th} floor of North Tower, IST

Resumo

Este trabalho aborda o problema do controlo de formação de um multirotor e um (ou mais) veículos marinhos que operam à superfície da água, com o objectivo final de circundar a fronteira de um derrame químico. A dissertação começa por introduzir os modelos matemáticos da classe de robôs marinhos Medusa e de robôs multirotores, seguidos da concepção de controladores de movimento que permitem a estes veículos seguir, de forma individual, um caminho definido por equações paramétricas, utilizando esquemas de ”inner-outer loop” acoplados a técnicas baseadas em Lyapunov. Numa segunda fase, é introduzido um controlador de coordenação distribuído que utiliza comunicações desencadeadas por eventos, permitindo aos veículos realizar o seguimento de trajectórias cooperativamente de acordo com uma formação geométrica pré-definida. Na etapa seguinte, é desenvolvido um algoritmo de planeamento de caminhos em tempo real que faz uso de uma câmera a bordo do multirotor, capaz de detectar nas imagens produzidas quais os pixels que codificam partes de um limite do derrame químico. Estes dados são utilizados para gerar e atualizar em tempo real um conjunto de percursos suaves modelados por B-splines. O desempenho do sistema é avaliado através do recurso a software de simulação 3-D, tornando possível a simulação visual de um derrame de um agente químico no mar. São também fornecidos resultados de testes reais para partes do sistema, onde dois veículos Medusa são obrigados a executar uma missão em que têm de seguir um caminho em forma de ”lawn-mowing”, de forma cooperativa, à superfície da água.

Palavras-Chave: Controlo de Quadrotor, Control de Veiculo Aquático, Controlo Cooperativo, Seguimento de Fronteira Ambiental

Abstract

This work addresses the problem of formation control of a quadrotor and one (or more) marine vehicles operating at the surface of the water with the end goal of encircling the boundary of a chemical spill. Firstly, the mathematical models of the Medusa class of marine robots, and quadrotor aircrafts are introduced, followed by the design of single vehicle motion controllers that allow these vehicles to follow a parameterized path individually. Inner-outer loop schemes coupled with Lyapunov based techniques are used for control design. At a second stage, a distributed coordination controller using event triggered communications is introduced, enabling the vehicles to perform cooperative path following missions according to a pre-defined geometric formation. In the next step, a real time path planning algorithm is developed that makes use of a camera sensor, installed on-board the quadrotor. This sensor enables the detection in the image of which pixels encode parts of a chemical spill boundary and use them to generate and update in real time a set of smooth B-spline based paths for all the vehicles to follow cooperatively. The performance of the complete system is evaluated by resorting to 3-D simulation software, making it possible to simulate visually a chemical spill. Results from real water trials are also provided for parts of the system, where two Medusa vehicles are required to perform a static lawn-mowing path following mission cooperatively at the surface of the water.

Keywords: Quadrotor control, Autonomous Surface Vehicle control, Cooperative Path Following, Environmental Boundary Following

Contents

List of Figures

List of Tables

Acronyms

DSOR	Dynamical Systems and Ocean Robotics Laboratory
ISR	Institute for Systems and Robotics
AUV	Autonomous Underwater Vehicle
ASV	Autonomous Surface Vehicle
UAV	Unmanned Aerial Vehicle
MAS	Multi-Agent Systems
ETC	Event-Triggered Communications
SNAME	Society of Naval Architects and Marine Engineers
CAD	Computer Aided Design
CAGD	Computer Aided Geometry Design
PF	Path Following
TT	Trajectory Tracking
DOF	Degrees of Freedom
LOS	Line of Sight
MPC	Model Predictive Control
NED	North-East-Down reference frame convention
PI	Proportional Integral
PD	Proportional Derivative
PID	Proportional Integral Derivative
IMU	Inertial Measurement Unit
AHRS	Attitude and Heading Reference System
DVL	Doppler Velocity Logger
DGPS	Differential Global Positioning System
CPF	Cooperative Path Following
LIDAR	Light Detection And Ranging

SLAM Simultaneous Localization and Mapping

MST Minimum Spanning Tree

EMST Euclidean Minimum Spanning Tree

BFS Breadth First Search

KKT Karush Kuhn Tucker

SQP Sequential Quadratic Programming

KF Kalman Filter

EKF Extended Kalman Filter

ROS Robots Operating System

OOP Object Oriented Programming

FPV First Person View

UDP User Datagram Protocol

TCP Transmission Control Protocol

Nomenclature

$\boldsymbol{\eta}$ Pose vector

$\mathbf{p}$ Position vector

$\mathbf{v}$ Velocity vector

$\boldsymbol{\tau}_{RB}$ Generalized vector of forces and torques

$\boldsymbol{F}_{RB}$ Forces vector expressed in body frame

$\boldsymbol{N}_{RB}$ Torques vector expressed in body frame

$\mathbf{v}_c$ Ocean current velocity vector measured in $\{U\}$

$\boldsymbol{v}_c$ Ocean current velocity vector measured in $\{U\}$ and expressed in $\{B\}$

$\boldsymbol{v}_d$ Virtual target progression speed

$\boldsymbol{v}_L$ Virtual target desired speed profile

$\boldsymbol{v}^{coord}$ Virtual target coordination speed

$\boldsymbol{\gamma}$ Curve parametric values vector / Virtual targets vector

J Inertia Matrix

$\mathbf{u}$ System input vector

$\mathbf{u}_d$ System desired input vector

m Mass

g Gravity acceleration

ψ Heading angle

$R(\cdot)$ Rotation matrix

$Q(\cdot)$ Angular velocity transformation matrix

$\{U\}$ Inertial reference frame

$\{B\}$ Body reference frame

$\{W\}$ Water-fixed reference Frame

$\{C\}$ Camera reference frame

$\sigma(\cdot)$ Saturation function

$S(\cdot)$ Skew-symmetric matrix

$\mathcal{G}$ Graph

$\mathcal{E}$ Edges of a graph

$\mathcal{V}$ Vertices of a graph

D Degree matrix

$\mathcal{A}$ Adjacency matrix

L Laplacian matrix

$C(\cdot)$ Parametric curve

$B(\cdot)$ B-Spline basis function

$\mathbf{P}$ B-Spline control points vector

D_X Total distance between a set of points

K Camera intrinsic parameters matrix

$\begin{matrix} C \\ U \end{matrix}[R|T]$ Camera extrinsic parameters matrix

Ω Camera transformation matrix

λ Camera transformation matrix

x, y Pixel coordinates

D_X Total distance between a set of points

N_C Number of curve segments

$\mathbf{p_s}$ Re-planning point

Chapter 1

Introduction

1.1 Motivation

The ocean covers around 361M km^2 of the Earth's surface [1] and there is evidence that it was at its bottom that the first primordial cells have formed, about 3 to 4 billion years ago [2]. According to LiveScience [3], it is estimated that to this day more than two third of the species that inhabit it are yet to be identified by humans. The ocean also plays a key role in our modern society, being a source of food and sustainable energy that powers millions of homes [4]. It is also key when it comes to the world's economy, being the path to many intercontinental transportation routes that are still used to this day.

Unfortunately, this vast habitat is also known for environmental disasters, some as a direct consequence of human behaviour, such as oil spills or ocean waste disposal and others as an indirect consequence, such as global warming and the rise of seawater levels. These catastrophes represent a major threat to wild life, and as a consequence a threat to humans. An example of such human interference activities are dredging operations, which are usually conducted to shape the shoreline, but can lead to short-term water pollution (Figure 1.1).

Figure 1.1: Dredging in Foz do Arelho, Portugal (08/2021)

In the case of oil spills or waste disposals, surveillance as well as cleanup missions must be carried out in order to restore these environments to their previous states. These operations are expensive to conduct and require the use of huge vessels with specialized staff on board to conduct them. In the case

of oil spills, these operations usually resort to skimmers used by boats in order to ”skim” the oil from the sea surface, together with chemical dispersants to break up the oil molecules.

On the other side of the spectrum, to adapt to the climate change, some aquatic species are migrating and finding new homes far away from their original habitats. This motivates the existence of organizations all over the globe, such as the Oceanic Society [5] that are dedicated to following, protecting and rescuing these endangered species. For monitorization missions to be carried out successfully, it is imperative that they do not disrupt or impact negatively the ecosystem.

Recent years have seen a huge development in computing power and miniaturization of sensors which have enabled the development of very efficient robots that can sweep through the sea at a relative small cost when compared to the current alternatives - large ships that are loud and very disruptive. These robots are usually known as Autonomous Underwater Vehicle (AUV) or/and Autonomous Surface Vehicle (ASV) when working only at the surface of water. In addition to these, there has been recently a growing interest on the development of miniaturized aircrafts denominated Unmanned Aerial Vehicles (UAV) which are usually equipped with camera sensors allowing them to have a top-down view of the environment. Together, these unmanned vehicles have a huge potential to decrease the cost of the previously mentioned operations, by having low-cost autonomous vehicles cooperating to achieve the same tasks of a huge vessel, without causing as much disruption in the ecosystem.

Motivated by all of those factors, the aim of this work is to develop a set of tools that allow an autonomous aerial vehicle (quadrotor) and multiple marine vehicles (ASVs) to perform a surveillance mission cooperatively, where the main goal is to detect and follow closely a dynamical environmental boundary[1], such as an oil spill, at the surface of the water, according to Figure 1.2.

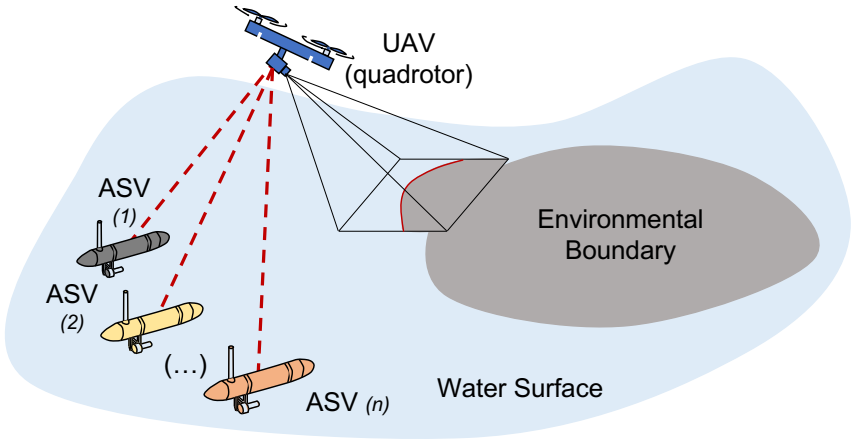


Figure 1.2: Cooperative path following on an environmental boundary

1.2 State of the Art

The goal of this section is to make a brief overview of the literature used as a backbone to all the work developed. As a first step, analysis on single vehicle motion control is made, in which a brief comparison between the two main techniques, Path Following (PF) and Trajectory Tracking (TT), is conducted. Notable contributions to solve the PF problem in 2-D are introduced and a high-level comparison between

[1]The term environmental boundary used in this context denotes any hazardous spread of contaminants, pollutants, etc. that generate anisotropic changes in the environment, for which a clear perimeter can be defined.

these algorithms is made. All the control laws reviewed are based on purely geometric concepts, only taking into consideration the kinematics of the vehicles.

A second step is to address the topic of information exchange in a vehicle network. This is a research area of growing interest in the recent years with emphasis on distributed approaches. This theme is of extreme relevance as it is the key that allows to have multiple vehicles performing PF missions in a synchronized manner. Therefore, on the scope of this work, a small overview on the problem of consensus of multiple agents is made.

Given that the broader scope of this work is to provide a mechanism to allow multiple robots to follow an environmental boundary in a pre-defined formation, an overview on motion path planning algorithms with the end goal of tracking and following an environmental boundary is also provided.

1.2.1 Inner-Outer Loop Control Structure

Developing controllers for vehicles such as ASVs and UAVs can be changeling, especially considering that these vehicles are usually under-actuated, i.e. vehicles with fewer actuators than Degrees of Freedom (DOF), and their dynamics are nonlinear. A very popular approach used to simplify the problem of vehicle motion control is therefore to consider an inner-outer loop control structure.

The goal of an outer-loop is to generate high level references such as desired orientation and speed based on a higher level goal - for example, following a path. On the other hand, the inner-loop controller is held responsible for computing the forces and torques that must be applied to the vehicle in order to follow the references generated by the outer-loop. Some benefits of the inner-outer loop approach are:

- The design of the PF algorithm considers only the outer-loop structure, meaning that its design can be very simple and not require an in depth knowledge of the vehicle's inner-dynamics;
- The same PF design methodologies can be applied to heterogeneous vehicles with the main changes being left for the inner-loop controls.

The main disadvantages is that stability analysis of the closed-loop system becomes non-trivial, but not impossible. In a more practical note, this may have a negative effect on the achievable performance, when compared with an approach that considers the inner-outer loop coupling.

An example of this approach applied to marine vehicles is detailed in Maurya et al. [6], where the author proposed an inner-loop structure to control an ASV heading motion and an outer-loop to generate the desired heading references to be tracked. This inner-outer loop structure is also commonly applied to quadrotors UAVs and discussed in detail in Mahony et al. [7].

1.2.2 Trajectory Tracking vs Path Following

The problem of single vehicle motion control has seen a huge amount of research in the recent years. Two of the main problems in motion control are TT and PF [8].

The TT can be summarized as a set of techniques that allow a vehicle to follow a predefined spacial path that is time parameterized. As mentioned in [9], there are already plenty of nonlinear algorithms

well described in textbooks applied to fully actuated systems to solve this problem. However, the problem becomes more challenging when under-actuated systems are considered. Typical examples of under-actuated systems can be, once again, some classes of ASVs and UAVs - the vehicles being considered in this project. Some nonlinear control algorithms based on Lyapunov theory have already been proposed for this kind of vehicles [10]. The problem becomes even more challenging when disturbances are considered in the design phase on these algorithms.

On the other side of the spectrum there is PF in which a vehicle is required to converge to and follow a desired path without imposing any time constraints. According to [9], the PF approach assumes that the vehicle's forward speed will track a speed profile (which can be specified as a function of the path) and that a controller acts on the vehicle in order to guide it to the reference path. There are multiple advantages of this method when compared to TT. One merit of this strategy is that typically there is a smoother convergence of the vehicle to the path using control signals that do not hit the actuator saturation so often. Another advantage of PF when compared to TT is when, in the presence of external disturbances, which might make it infeasible for the vehicle to fulfill the time requirements that TT imposes.

Due to the fact that quadrotor UAVs are typically very agile, able to perform a wide-range of manoeuvres at very high speeds, TT is often used as the main tool for motion control of these vehicles. On the other hand, UAVs and ASVs are usually much slower and less agile vehicles, when compared to their aerial counterparts. Therefore, PF is usually the preferred method of motion control for these vehicles [11]. Since the main goal of this work is to have a both a quadrotor and one or more ASVs following a boundary in a synchronised manner, the PF approach will be the one explored.

1.2.3 PF - Line of Sight (LOS)

One very popular approach to the PF problem applied to ASVs is the Line of Sight (LOS) algorithm. Its application to marine vehicles for both 2-D and 3-D cases is well detailed in Lekkas et al. [12]. This algorithm is very versatile and its applications can also be extended to the case of quadrotors, as demonstrated by A.T. Nugraha et al. [13]. The main goal of this control law is to point the vehicle to a given point in the path that is arbitrarily further ahead of the vehicle's projection on the path by controlling its heading angle. In Figure 1.3 a graphical overview of this method is presented.

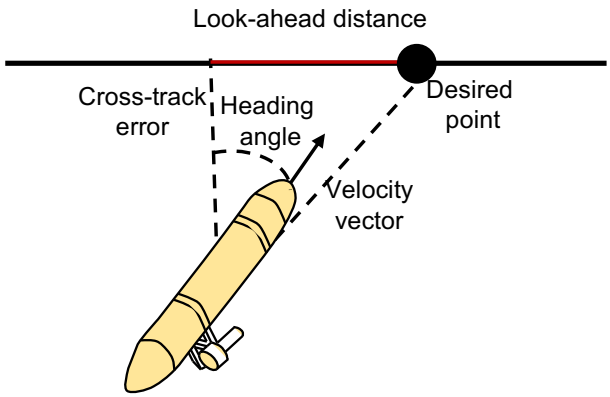


Figure 1.3: Line of Sight PF overview

The fixed look-ahead distance used in this algorithm works as ”tunning knob” that can make the convergence to the path slower or faster. For this particular control scheme the author provides some guarantees of convergence of the cross-track error to zero under some very mild conditions.

The main disadvantage of the LOS control scheme is that the look-ahead distance considered in this algorithm is fixed. Therefore, if the look ahead distance is too small, the vehicle will move more aggressively towards the path. On the other hand, if the look ahead distance is too big, the vehicle will take a long time to converge.

Simplified Line of Sight (P. Maurya et al. [6])

An alternative formulation for the LOS guidance law applied to ASVs is given in P. Maurya et al. [6] where the authors propose the use of an integral in the control law in order to reject disturbances caused by constant and non-rotational ocean currents, or other constant disturbances. In addition, a look ahead distance is no longer taken into consideration and only the distance of the vehicle to its projection on to the path is used, as well as its speed. The main advantage of this control scheme is the ability to reject constant disturbances caused by ocean currents while at the same time not having to define a fixed look ahead distance.

1.2.4 PF - Nonlinear Control using Lyapunov Theory

Proposal by C. Samson et al. [14]

A completely different approach to solving the PF problem, applied to differential drive vehicles, such as ASVs, was taken by A.Micaelli and C.Samson [14]. In this research paper the authors start by projecting the vehicle on to the path, with the orthogonal projection representing the closest point on the path to the vehicle. This point is also commonly known as a ”fixed rabbit” that the vehicle must follow. In addition a Frenet-Frame is associated to this projection point, with its origin coinciding with the projection point. The cross-track error is the Y-coordinate in the aforementioned Frenet-Frame of the center of mass of the vehicle, as depicted in Figure 1.4 a).

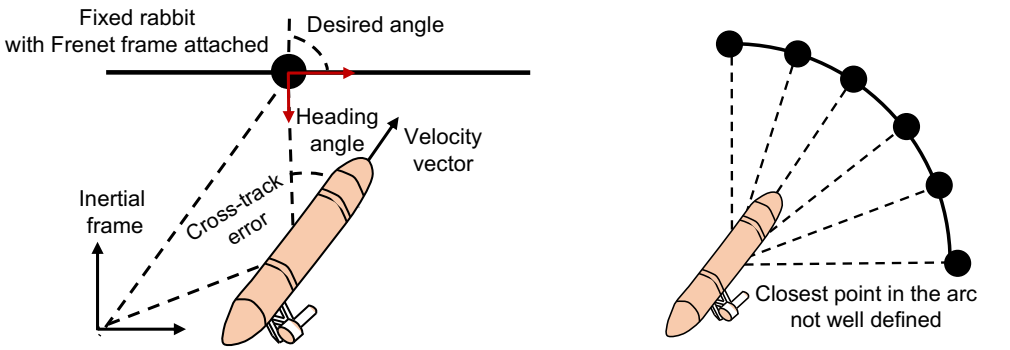
With this new formulation the authors take the two variables of interest - the cross-track error and heading error (given by the difference between the vehicle’s heading and the angle formed by the tangent to the curve on the computed projection point and the X-axis of the inertial frame) - and craft two candidate/Lyapunov functions, one being a function of the cross-track error and another a function of the heading error. Making use of Lyapunov nonlinear stability theory, the authors derive a control scheme for the heading rate of the vehicle.

One of the main advantages of this approach is the ability to tune the speed of convergence of cross-track error and heading error to zero separately. One key example of this feature is enabling the vehicle to give higher importance to the cross-track error when being further way from the path and high importance to the heading error when closer to the path, for a faster convergence of the algorithm while maintaining a smooth approach to the path. Another merit of this proposal is that it allows for the design of controllers that address not only the kinematics but also the dynamics of the vehicle by resorting to backstepping

techniques.

The main disadvantages of this new algorithm are:

- It does not provide global asymptotic stability guarantees. Therefore, the convergence to the path is only guaranteed if the vehicle's initial conditions are within a "tube of values".
- If the vehicle is on the concave side of a circular path, more precisely in the center of the curve, the closest point to the path is no longer well defined, according to Figure 1.4 b).



(a) Samson PF scheme

(b) Closest point to a circular path

Figure 1.4: Samson PF overview

Proposal by L. Lapierre et al. [9]

After the introduction of a new innovative approach of tackling the PF problem making use of a Serret-Frenet frame and Lyapunov nonlinear stability analisys by A.Micaelli and C.Samson [14] a constructive approach was taken by L. Lapierre, D. Soetanto and A. Pascoal (2006) [9]. The key difference between the proposed algorithms is that in this new approach instead of having a simple projection of the vehicle on the path representing the closest point on the path to the vehicle, a more versatile approach was taken - a movable virtual target.

The addition of a dynamic virtual target on the path, enables the control of the rate of progression of the point in the path, which in turn makes the problem more flexible relaxing the constraints imposed by Samson's algorithm in terms of initial conditions.

This new approach brings many advantages, namely:

- the initial conditions of the vehicle are no longer a constraint of the problem as we are able to prove global asymptotic convergence to the path;
- allows for the user to design a custom control scheme for the desired velocity, while still guaranteeing global stability.

The main disadvantage of methods such as the ones proposed by both C. Samson and L. Lapierre is the fact that they both rely on a Serret-Frenet frame fixed on the virtual target that moves along the path. This frame can have discontinuities when the concavity of a curve switches. A solution to this problem is to simply replace the Serret-Frenet frame by a Parallel-Transport frame. Still, in order to generalize the

above proposed controller methodologies to the 3-D space, a knowledge of both torsion and curvature of the path to be followed is required. Furthermore, even though robust, the application of these controllers is typically guided towards AUVs and fixed-wing UAVs as they define a set of control signals for the linear and angular velocities of the vehicles.

Proposal by P. Aguiar et al. [8]

Other Lyapunov based controller designs were provided in the works of P. Aguiar and, J. Hespanha [8] and F. Vanni [15] where the authors propose a global diffeomorphic coordinate transformation and define a position tracking error in the body reference frame of the vehicle itself, bypassing altogether the need for an additional frame fixed on a virtual target on the path.

One key advantage of this approach is that the backstepping techniques used in order to derive the path following controllers for underactuated AUVs is general enough to be applicable not only to fully-actuated AUVs and fixed-wing UAVs but also to quadrotor UAVs which assume a very different physical configuration when compared to torpedo-shaped vehicles.

A disadvantage of this method (and others previously presented) is that it is not trivial to embed system inputs and state constraints in the control laws.

1.2.5 PF - Model Predictive Control Approaches

Another completely different method to solve the PF problem is by resorting to optimal control theory. Model Predictive Control (MPC) in particular has the ability to explicitly handle vehicle's equality and inequality input and state constraints. In Alessandretti et. al. [16] the authors propose nonlinear control laws that can be used for both TT and PF in both 2-D and 3-D scenarios. In work by Hung. et al. [17], the authors propose the use of MPC to derive a global control scheme for the problem of multiple vehicle cooperative path following which explicitly handles vehicle input constraints applied to ASVs.

Even-though MPC provides a very intuitive way of taking into consideration the system limitations in the PF formulation, it overwhelmingly relies on extensive numerical computations, especially in the case of nonlinear systems and nonlinear constraints, making it much more computationally demanding than previously described methods. For this reason, this methodology will not be considered in this thesis.

1.2.6 Cooperative Control Contributions

The topic of cooperative motion control has seen increasing interest over the years. In order to design cooperative strategies to be successful in accomplishing a common goal several issues have to be taken into account. One of those issues is the convergence to a common value, also known as consensus or agreement problem as described in the literature.

Most common approaches to the consensus problem rely heavily on algebraic graph theory. Following this approach W. Ren and A. Ella [18] describe a distributed coordination scheme with local information exchange for multiple vehicle systems. In this research paper a first-order consensus protocol is summarized. In addition the authors introduce a second-order consensus protocol for double-integrator

systems along with the necessary and sufficient conditions under which consensus can be reached. In this work, several examples are provided in which the autonomous agents are required to converge to a static formation, while driving the formation error of the vehicles to zero. The results presented for double-integrator systems can be extremely relevant as UAV quadrotors dynamics can be linearized and expressed as double integrators, while the first-order consensus theory is extremely helpful when considering the synchronisation of path's parameters between different vehicles in the case of CPF.

In previous work developed by R. Ghabcheloo et al. [19] the authors propose a CPF control scheme that assumes a fixed vehicle network topology but considers the existence of communication losses and delays in the information exchange between the vehicles. On top of the work previously developed, there were efforts carried by A. Aguiar, A. Pascoal et al. [20], F. Rego et al. [21], [22], and N. Hung et al. [23] with the goal of achieving a general solution to the problem of consensus/synchronization for general classes of networked nonlinear Multi-Agent Systems (MAS) using a distributed control strategy with an Event-Triggered Communications (ETC) mechanism with a minimum number of inter-event communications between vehicles.

1.2.7 Perimeter Surveillance and Boundary Tracking

The problem of perimeter detection and boundary monitoring/tracking has been a widely researched topic with a variety of practical applications, ranging from monitoring of wildfire spreading [24], monitoring and control of the spread of oil spills [25], salinity distribution control and harmful invasive algae blooms [26] monitoring just to name a few. In this work, we focus our attention in the problem of detecting anomalies at the surface of the water for which a boundary curve can be defined, namely chemical spills or oil spills.

In water environments, chemical spills go through physical changes due to the interaction with the ocean environment which is not static. The two main phenomena that contribute to the transportation of chemicals, such as oil, over water are advection and diffusion. In the first, the oil is transported due to the flow of water while the second refers to the motion of the fluid caused by the existence of concentration gradients. One way of modelling the flow field of the incompressible fluid is by solving iteratively the convection-diffusion equations [27]. In Fahad et al. [25] the authors simulate an ASV equipped with sensors capable of measuring plume concentration and develop an observer for the spill diffusion coefficients and a gradient-based control law to steer the autonomous agent along the boundary of the plume. The main disadvantage of these chemical spill models is that they require either an a-priori knowledge of a set of constants, such as diffusion coefficients or an estimator for them. Furthermore, they require in depth knowledge of the dynamic behaviour of the anomaly being studied. Moreover, a monitoring control scheme developed on top of these models will not generalize well to other kinds of anomaly tracking problems.

Another approach taken by Saldaña et al. [28] is to consider that a general environmental boundary can be approximated by a closed curve that is slowly-varying over time and can be described by a parametric equation. Furthermore, the author proposes a model for the curve described spatially by a truncated Fourier Series that changes it's shape smoothly with time. The model is then defined in a matricial form, such that a recursive least squares problem can be formulated. In his work, it is assumed

that multiple vehicles are distributed equally around the chemical spill and every vehicle is capable of taking local measurements of the boundary (although the specific sensor used for that matter is not described). Those measurements are then used to update the shape of the closed curve using recursive least squares. Even though the techniques employed are interesting, the choice of a Truncated Fourier Series to represent a path for underactuated vehicles is a rather poor choice of function as we might get curves that self-intersect and have a huge amount of oscillations. Moreover, it does not take into consideration the physical limitations of the vehicles, as shown in the video provided by the author [29]. In order to lift the limitations imposed by this method, one could apply the same methodologies proposed by the author, but resorting to another type of parametric curves that are more stable, such as Bernstein polynomials or B-Splines [30].

Yet another attempt at solving this problem was taken in Pedrosa's master thesis [31], in which the author proposes a scheme where a quadrotor UAV equipped with a camera sensor flies high enough to capture an entire picture of the boundary of the spill and converts the pixels of the contour in the image to coordinates in a 2-D plane expressed in the inertial reference frame. After this step, the points expressed in the inertial frame are used to define an artificial potential field used by the control system of an ASV vehicle. This solution may not be suitable in practice since it does not take into consideration that an oil spill can span across several km^2 of area, implying that a quadrotor had to fly very high to get a complete picture of the boundary, which would affect the accuracy of the conversion between camera frame and inertial frame, or it had to take several pictures of the environment and construct a mosaic of the spill. If a mosaic approach is taken, given that the ocean environment is not static, as time progresses, the original boundary shape used for the path generation might no longer accurately describe the real chemical spill.

1.3 Objectives

The main goal of this work is to develop, simulate and test a set of control tools that allow one or more Medusa UAVs (operating in ASV mode) and one quadrotor UAV vehicle operating in a 2-D plane above the water to follow an environmental boundary, according to a pre-defined vehicle formation. In this setup, the quadrotor (the leader vehicle) is equipped with a camera sensor which is capable of detecting a local anomaly in the environment and define a local boundary for that anomaly. Given this information, the quadrotor is required to plan a path that can be followed closely by itself and the set of ASVs.

In order to achieve the goal of multiple non-homogeneous motion control, the work developed is split into multiple sub-problems:

- **Vehicle Modeling** - Develop a set of kinematic and dynamic models that describe the motion of an AUV operating at the surface of the water and the motion of a quadrotor UAV;
- **Single Vehicle Control** - Design, simulate and test a set of controllers that enable each individual vehicle to follow a pre-defined path individually;
- **Multiple Vehicle Coordination** - Derive a set of control laws for the velocity of each individual vehicle that enables them to maintain a desired formation. In addition develop experiments to

validate experimentally the theoretical results;

- **Online Path Planning** - Develop an algorithm to plan a path based on information extracted from a camera sensor attached to a quadrotor UAV in real time.

1.4 Main Contributions

The main contributions of the work developed are:

- Comprehensive study of PF techniques applied to ASVs, in particular the method proposed by A. Aguiar and F. Vanni and the extension of the rationale to quadrotor UAVs;
- Development of a new real time path planning mechanism using uniform cubic B-Splines;
- Setup of a realistic 3-D simulation environment for testing the proposed algorithms;
- Implementation and test of the PF and CPF algorithms using real Medusa vehicles.

1.5 Thesis Outline

The present document is divided in 10 chapters:

- **Chapter 2 (Background):** some preliminary notation and mathematical concepts are introduced. Furthermore, an overview on nonlinear control theory, graph theory, parametric equations and B-splines is provided;
- **Chapter 3 (Vehicle Models):** introduces kinematic and dynamic models of the vehicles that will be used later for the development of controllers;
- **Chapter 4 (Vehicle Motion Control):** presents the methodologies used for achieving low level control of an ASV and a quadrotor UAV;
- **Chapter 5 (Path Following):** derives a PF control scheme for each vehicle, making use of the adaptive virtual target concept;
- **Chapter 6 (Cooperative Path Following):** formulates the cooperative path following problem and establishes an event-triggered communication scheme used for inter-vehicle synchronisation;
- **Chapter 7 (Path Planning):** presents a sliding window–like online path planning approach based on the image feedback of a dynamic environmental boundary;
- **Chapter 8 (Implementation Details):** describes the implementation and simulation setup used to for testing the proposed algorithms;
- **Chapter 9 (Results):** a set of simulated and real results are presented;
- **Chapter 10 (Conclusion):** a summary of the work developed accompanied by a discussion of future work.

Chapter 2

Background

In this chapter the mathematical notation and preliminary theoretical concepts relevant to this dissertation are introduced. In section 2.1, the mathematical terminology used throughout this thesis is presented. In sections 2.2 and 2.3 a brief revision of concepts regarding nonlinear control theory and graph theory is provided. Furthermore a smooth projection operator useful in the development of estimators is introduced. Finally, in section 2.4 a review on B-splines, a set of parametric functions used to describe smooth curves is made, followed by a quick revision of convex optimization problems in section 2.5.

2.1 Mathematical Notation

The unit vectors $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ are defined as $\mathbf{e}_1 = [1,0,0]^T$, $\mathbf{e}_2 = [0,1,0]^T$ and $\mathbf{e}_3 = [0,0,1]^T$ respectively. Vectors are represented in lowercase bold. The notation $\mathbf{1}$ and $\mathbf{0}$ denote vectors with all elements equal to one or zero respectively. For a vector $\mathbf{x} \in \mathbb{R}^n$, the symbol x_i denotes the ith element of the vector and $|x_i|$ its absolute value. We shall use $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$ to denote the Euclidean norm of a vector and $sup\,\|\mathbf{x}\|$ the supremum norm. The notation $K \succeq 0$ is used to denote a matrix $K \in \mathbb{R}^{n\times n}$ that is positive semi-definite. The symbol I is used to denote the identity matrix. Let $\sigma(x) : \mathbb{R} \to \mathbb{R}$ define a differentiable saturation function that verifies the following properties:

1. $|\sigma(x)| \leq \sigma_{max}$, and $0 < \frac{\partial\sigma(x)}{\partial x} \leq \left(\frac{\partial\sigma(x)}{\partial x}\right)_{max}, \forall x \in \mathbb{R}$;
2. $x\sigma(x) > 0, \forall x \neq 0$;
3. $\sigma(0) = 0$;
4. $\sigma(-x) = -\sigma(x), \forall x \in \mathbb{R}$.

The vectorial equivalent of the saturation function is defined as

$$\begin{cases} \sigma(\mathbf{x}) = \sigma(\|\mathbf{x}\|)\frac{\mathbf{x}}{\|\mathbf{x}\|} \text{ with } \|\mathbf{x}\| \neq 0 \\ \sigma(\mathbf{0}) = \mathbf{0} \end{cases} . \tag{2.1}$$

The vectorial saturation function preserves the direction of the original vector $\mathbf{x}$. The inverse notation for the saturation function is defined as $\sigma^{-1}(\mathbf{x}) = [\sigma^{-1}(x_1), ..., \sigma^{-1}(x_N)]^T$. The saturation function that is adopted in this work is $\sigma(x) = tanh(x)$.

When considering an estimator for an unknown variable x, we use the hat nomenclature $\hat{x}$ to denote its estimate and $\tilde{x}$ when referring to the estimation error. The symbol $R(.)$ is used to denote a rotation matrix with properties: $R^T = R^{-1}$ and $det(R) = 1$. The map $S(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n\times n}, n = 2, 3$ yields a skew-symmetric matrix

$$S(\mathbf{x})\mathbf{y} = \mathbf{x} \times \mathbf{y}, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \tag{2.2}$$

which has the useful property

$$\mathbf{y}^T S(\mathbf{x})\mathbf{y} = 0, \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \text{ with } n = 2, 3. \tag{2.3}$$

The skew-symmetric matrix of vector $\mathbf{a} = [a_1, a_2, a_3]^T$ is given by

$$S(\mathbf{a}) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \tag{2.4}$$

while the skew-symmetric matrix $S(b), b \in \mathbb{R}$ is given by

$$S(b) = \begin{bmatrix} 0 & -b \\ b & 0 \end{bmatrix}. \tag{2.5}$$

The symbols $\lfloor x \rceil, x \in \mathbb{R}$ denotes x nearest integer, such that $\lfloor 10.2 \rceil = 10$ and $\lfloor 10.6 \rceil = 11$. The symbol $\lfloor x \rfloor$ denotes the floor of x, such that $\lfloor 10.2 \rfloor = 10$ and $\lfloor 10.6 \rfloor = 10$ and $\lceil x \rceil$ denotes the ceiling of x, such that $\lceil 10.2 \rceil = 11$ and $\lceil 10.6 \rceil = 11$. The norm of a matrix $A \in \mathbb{R}^{n\times m}$ can be computed according to

$$\|A\| = \sqrt{\lambda_{max}(A^T A)}, \tag{2.6}$$

where λ_{max} denotes the largest singular value of $A^T A$. Consider the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ given by

$$f = \mathbf{x}^T A \mathbf{x} = \mathbf{x}^T \left(\frac{A + A^T}{2} \right) \mathbf{x}. \tag{2.7}$$

If matrix A is symmetric the following property holds:

$$\lambda_{min}(A) \|\mathbf{x}\|^2 \leq \mathbf{x}^T A \mathbf{x} \leq \lambda_{max}(A) \|\mathbf{x}\|^2. \tag{2.8}$$

2.2 Nonlinear Control Theory

In this section a revision on stability of the equilibrium points of nonlinear autonomous and non-autonomous systems is conducted. The stated theorems and definitions originate from the book Nonlinear Systems by Khalil H. [32]. For proofs of the stated theorems, the reader is referred to the author's original work.

2.2.1 Lyapunov Stability

An equilibrium point of a system is said to be stable if all solution starting at nearby points stay nearby, otherwise it is unstable. It is asymptotically stable if all solutions starting at nearby points not only stay nearby, but also tend to the equilibrium point as time approaches infinity. Consider an autonomous nonlinear system defined by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \mathbf{x}(0) = \mathbf{x}_0. \tag{2.9}$$

Theorem 2.1. *Let $\mathbf{x} = 0$ be an equilibrium point for (2.9) and $D \subset \mathbb{R}^n$ be a domain containing $\mathbf{x} = 0$. Let $V : D \to R$ be a continuously differentiable function such that*

$$V(\mathbf{0}) = 0 \text{ and } V(\mathbf{x}) > 0 \text{ in } D - \{0\}, \tag{2.10}$$

$$\dot{V}(\mathbf{x}) \leq 0 \text{ in } D, \tag{2.11}$$

then $\mathbf{x} = \mathbf{0}$ is stable. Moreover, if

$$\dot{V}(\mathbf{x}) < 0 \text{ in } D - \{0\}, \tag{2.12}$$

then $\mathbf{x} = \mathbf{0}$ is asymptotically stable.

Theorem 2.2. *Let $\mathbf{x} = 0$ be an equilibrium point for (2.9). Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(\mathbf{0}) = 0 \text{ and } V(\mathbf{x}) > 0, \forall \mathbf{x} \neq \mathbf{0}, \tag{2.13}$$

$$\|\mathbf{x}\| \to \infty \Rightarrow V(\mathbf{x}) \to \infty, \tag{2.14}$$

$$\dot{V}(\mathbf{x}) < 0, \forall \mathbf{x} \neq 0, \tag{2.15}$$

then $\mathbf{x} = \mathbf{0}$ is globally asymptotically stable.

2.2.2 Input-to-State Stability

Definition 2.1. A continuous function $f(\mathbf{x})$ is said to be locally Lipschitz on a domain $D \subset \mathbb{R}^n$ if each point of D has a neighborhood D_0 and $\exists L$ (Lipschitz constant) such that $f(\cdot)$ satisfies

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|, \ \forall_{x,y} \in D_0. \tag{2.16}$$

Definition 2.2. A continuous function $\alpha : [0, a) \to [0, \infty)$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. It is said to belong to class $\mathcal{K}_\infty$ if $a = \infty$ and $\alpha(r) =\to \infty$ and $r \to \infty$.

Definition 2.3. A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ is said to belong to class $\mathcal{KL}$ if, for each fixed s, the mapping $\beta(r, s)$ belongs to class $\mathcal{K}$ with respect to r and, for each fixed r, the mapping $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$ as $s \to 0$.

Definition 2.4. Consider the system

$$\dot{\mathbf{x}} = f(t, \mathbf{x}, \mathbf{u}), \tag{2.17}$$

where $\mathbf{u}$ is the system's input vector and $f : [0, \infty) \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ a piecewise continuous function in t and locally Lipschitz in $\mathbf{x}$ and $\mathbf{u}$. The system is said to be input-to-state stable if there exists a class $\mathcal{KL}$

function β and a class $\mathcal{K}$ function γ such that for any initial state $\mathbf{x}(t_0)$ and any bounded input $\mathbf{u}(t)$, the solution $\mathbf{x}(t)$ exists for all $t \geq t_0$ and satisfies:

$$\|\mathbf{x}(t)\| \leq \beta(\|\mathbf{x}(t_0)\|, t - t_0) + \gamma\left(\sup_{t_0 \leq \tau \leq t} \|\mathbf{u}(\tau)\|\right). \tag{2.18}$$

Theorem 2.3. *Let $V : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}$ be a continuous differentiable function such that*

$$\alpha_1(\|\mathbf{x}\|) \leq V(t, \mathbf{x}) \leq \alpha_2(\|\mathbf{x}\|), \tag{2.19}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{x}} f(t, \mathbf{x}, \mathbf{u}) \leq -W(\mathbf{x}), \forall \|\mathbf{x}\| \geq \rho(\|\mathbf{u}\|) > 0, \tag{2.20}$$

$\forall (t, \mathbf{x}, \mathbf{u}) \in [0, \infty) \times \mathbb{R}^n \times \mathbb{R}^m$, where α_1 and α_2 are class $\mathcal{K}_\infty$ functions, $W(\mathbf{x})$ is continuous positive definite function and ρ is a class $\mathcal{K}$ function. Then the system given by (2.17) is input-to-state stable with $\gamma = \alpha_1^{-1} \circ \alpha_2 \circ \rho$ (with $\circ$ denoting the composition operation).

2.2.3 Smooth Projection Operator

When a nonlinear system operates under the presence of external disturbances that cannot be measured directly, it is commonplace to design a set of estimators used in conjunction with controllers to compensate for them. In the particular case of quadrotors, where the external disturbances can be caused not only by wind, but also by coriolis effects, rotor flapping, aerodynamic drag, etc. it is commonplace to use integral-like terms in the controllers as a simple way to reject constant disturbances. Straightforward estimators like these can lead to windup phenomena and make stability analysis non-trivial when used in conjunction with nonlinear controllers. To cope with these problems, Z. Cai et al. propose a smooth projection operator that preserves the properties of the control system while at the same time bounding the estimation parameter [33]. Consider the general system given by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}), \tag{2.21}$$

where $\mathbf{x} \in \mathbb{R}^n$ denotes the system state, $\mathbf{u} \in \mathbb{R}^m$ the system input vector and $\boldsymbol{\theta} \in \mathbb{R}^l$ a vector of unknown constant disturbances. Consider $\boldsymbol{\theta}$ to belong to the convex compact set $\Omega := \{\boldsymbol{\theta} : \|\boldsymbol{\theta}\| \leq \theta_0\}$, where θ_0 is a positive known constant. Furthermore, let $\tilde{\boldsymbol{\theta}} := \boldsymbol{\theta} - \hat{\boldsymbol{\theta}}$ be the estimation error of the unknown constant disturbance. The proposed operator used as an estimator for $\boldsymbol{\theta}$ is given by

$$\dot{\hat{\boldsymbol{\theta}}} = \text{Proj}(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}}) = \boldsymbol{\mu} - \frac{\eta_1 \eta_2}{4(\varepsilon^2 + 2\varepsilon\theta_0)^{n+1}\theta_0^2} \nabla \mathbf{p}_d(\hat{\boldsymbol{\theta}}), \tag{2.22}$$

where

$$p_d(\hat{\boldsymbol{\theta}}) = \hat{\boldsymbol{\theta}}^T \hat{\boldsymbol{\theta}} - \theta_0^2, \tag{2.23}$$

$$\eta_1 = \begin{cases} p_d^{n+1}(\hat{\boldsymbol{\theta}}), \text{ if } p_d(\hat{\boldsymbol{\theta}}) \geq 0 \\ 0, \text{ otherwise} \end{cases}, \tag{2.24}$$

$$\eta_2 = \frac{1}{2}\hat{\boldsymbol{\theta}}^T \boldsymbol{\mu} + \sqrt{\left(\frac{1}{2}\hat{\boldsymbol{\theta}}^T \boldsymbol{\mu}\right)^2 + \delta^2}, \tag{2.25}$$

and $\boldsymbol{\mu}(t) \in \mathbb{R}^p$ is a known, n times continuously differentiable (C^n) variable, ε and δ are arbitrary positive constants. This projection operator enjoys the following properties:

1. $\left\|\hat{\boldsymbol{\theta}}\right\| \leq \theta_0 + \varepsilon, \forall t \geq 0$;
2. $\tilde{\boldsymbol{\theta}}^T \text{Proj}(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}}) \geq \tilde{\boldsymbol{\theta}}^T \boldsymbol{\mu}$;
3. $\left\|\text{Proj}(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}})\right\| \leq \|\boldsymbol{\mu}\| \left[1 + ((\theta_0 + \varepsilon)/\theta_0)^2\right] + ((\theta_0 + \varepsilon)/(2\theta_0^2))\delta$;
4. $\text{Proj}(\boldsymbol{\mu}, \hat{\boldsymbol{\theta}})$ is C^n.

A practical implementation of this smooth projection operator applied to nonlinear quadrotor control can be found in Cabecinhas et al. [34].

2.3 Graph Theory

Cooperative path following requires vehicles in a network to exchange information about their state. In a general case, not every vehicle is able to communicate with each other, nor are all the inter-vehicle communications bi-directional. Therefore, a strong mathematical theory is needed to analyze this information exchange. Graph theory is the tool par excellence to model communication networks. In this section, the basic concepts of graph theory are introduced with the main theorems and definitions borrowed from Bullo et al. [35], W. Ren and R. Beard [36].

A weighted digraph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$ consists of a set of N vertices $\mathcal{V} = [V_1, ..., V_N]^T$, a set of directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and a weighted adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$ such that $a_{ij} > 0$ if the edge that connects vertex i to j belong to the graph and 0 otherwise. Self connected vertices, are not allowed, i.e. $a_{ii} = 0$. The set of in-neighbours of a vertex i is given by $\mathcal{N}_i^{in} = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ and the set of out-neighbours by $\mathcal{N}_i^{out} = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}$. The in- and out-degree matrices D^{in} and D^{out} are a set of diagonal matrices defined by

$$D^{in} = diag(d_i^{in}), \text{ with } d_i^{in} = \sum_{j \in \mathcal{N}_i^{in}} a_{ij}; \tag{2.26}$$

$$D^{out} = diag(d_i^{out}), \text{ with } d_i^{out} = \sum_{j \in \mathcal{N}_i^{out}} a_{ji}. \tag{2.27}$$

Remark: With the graph definition given above, we adopt the convention that an agent i can receive information from its neighbors in $\mathcal{N}_i^{in}$ and send information to its neighbors in $\mathcal{N}_i^{out}$.

The Laplacian matrix of of the graph $\mathcal{G}$ is given by

$$L = (D^{in} - \mathcal{A}). \tag{2.28}$$

The Laplacian matrix $L = [L_{ij}]$ satisfies the following conditions:

$$\begin{aligned} l_{ij} &\leq 0 \text{ , } i \neq j \\ \sum_{j=1}^{n} l_{ij} &= 0 \text{ , } i = 1, ..., N. \end{aligned} \tag{2.29}$$

A graph $\mathcal{G}$ is undirected if communication links are unidirectional. If $\mathcal{G}$ is an undirected graph, then $\mathcal{G}$ is also balanced, i.e. $D^{in} = D^{out} := D$ and its Laplacian matrix L is symmetric and positive semi-definite. A graph $\mathcal{G}$ is connected if there exists a walk between any two vertices and strongly connected if there exists a directed walk from any node to any other node. $\mathcal{G}$ is disconnected if not connected.

Theorem 2.4. *If $\mathcal{G}$ is an undirected and connected graph, L is symmetric and has a simple eigenvalue at zero associated with eigen vector $\mathbf{1}$ with the remaining eigen values positive. Moreover $L\mathbf{1} = \mathbf{0}$.*

Definition 2.5. A tree is an undirected, acyclic graph with the following property: there exists exactly only one path between two distinct nodes. If $\mathcal{G}$ is an undirected connected graph with N vertices and only $N - 1$ edges, then $\mathcal{G}$ is also a tree.

Definition 2.6. A Minimum Spanning Tree (MST) is a sub-set of edges from an undirected connected graph that connects all the vertices together, with minimal total edge weight and without forming any cycle. From this definition and the above theorem, it is possible to infer that every connected graph has a MST.

2.4 Parametric Curve Representations

There are two very popular methods for representing curves and surfaces in space: implicit equations and parametric functions. In the case of implicit equations, a curve $\mathbf{C}$ lying on xy plane has the form $f(x, y) = 0$, which describes directly the relationship between x and y coordinates of the points on the curve [30]. On the other hand, when using a parametric function, each point on the curve is represented by an explicit function on an independent parameter, such as:

$$\mathbf{C}(\gamma) = \begin{bmatrix} x(\gamma) \\ y(\gamma) \end{bmatrix}, \text{ with } \gamma \in [a, b], \tag{2.30}$$

where $a, b \in \mathbb{R}$ and γ is a parameterizing variable and $\mathbf{C} \in \mathbb{R}^2$ is continuous. The representation of curves in parametric form gives some useful properties that will be exploited throughout this work, such as:

- Possess a natural direction of transversal, i.e., from $\mathbf{C}(a)$ to $\mathbf{C}(b)$, leading to a very easy way of generating a sequence of points along the curve.
- It is easy to extend a curve represented using a parametric model to an N-dimensional space by just adding an extra set of coordinates dependent on the path parameter γ.
- Allow to express bounds in the curve segments through bounds on the path parameter interval.

Polynomial based parametric functions are very popular due to their easy implementation in software. However, when we consider the problem of interpolating or fitting a set of data using simple polynomials we quickly reach the conclusion that most of the times we need high degrees to represent the data accurately. Piecewise polynomial functions, i.e. splines are a convenient solution to those problems.

2.4.1 B-Spline Curves

Bernstein Polynomials and Bézier curves (usually used interchangeably) are parametric curves found in a wide variety of applications from Computer Aided Geometry Design (CAGD) to computer vision fields due to their efficient yet powerful way to describe continuous curves and surfaces through parametric equations using a finite set of parameters. Multiple Bézier curves can be combined to form Bézier-splines, usually referred to as B-splines [37]. Rather than having a high degree Bézier curve, we can have multiple low degree Bézier curves joined together. An unidimensional order $k+1$ B-spline is a piecewise polynomial function, formed by joining several pieces of polynomials of degree k [38]. For example, a cubic B-spline is of order four, as it requires four coefficients to specify a cubic polynomial [39].

A general B-spline curve is given by $n+1$ control points, consists of $n-k+1$ Bézier curves and is defined by the linear combination:

$$C(\gamma) = \sum_{i=0}^{n} B_{i,k}(\gamma) P_i, \tag{2.31}$$

where $P_i, i = 0, ..., n$ are a set of control points, $B_{i,k}(\gamma)$ the basis functions of fixed degree and $C(\gamma) \in \mathbb{R}$. It follows from the Cox-De Boor's recursive algorithm [30], that:

$$B_{i,0}(\gamma) = \begin{cases} 1, \text{ if } \gamma_i \leq \gamma \leq \gamma_{i+1} \\ 0, \text{ otherwise} \end{cases}, \tag{2.32}$$

$$B_{i,j}(\gamma) = \frac{\gamma - \gamma_i}{\gamma_{i+j} - \gamma_i} B_{i,j-1}(\gamma) + \frac{\gamma_{i+j+1} - \gamma}{\gamma_{i+j+1} - \gamma_{i+1}} B_{i+1,j-1}(\gamma), \tag{2.33}$$

where the values γ_i belong to knot vector defined as $\mathbf{U} = [\gamma_0, ..., \gamma_m]^T$ such that $\gamma \in [\gamma_0, \gamma_m]$, with the number of knots related to the degree of the curve and the number of control points by

$$m = k + n + 1. \tag{2.34}$$

A B-Spline is said to be uniform if its knots are equidistant, i.e the knots are given by

$$\gamma_i = (i-1)\Delta \ , \ i = 1, .., m \tag{2.35}$$

where $\Delta \in \mathbb{R}^+$ is a uniform step between consecutive knots. The B-Spline functions enjoy some useful properties, namely:

1. $B_{i,j}(\gamma)$ is a polynomial of degree j with joining points at $\gamma \in [\gamma_i, \gamma_{i+j+1})$;
2. **Non-negativity:** $B_{i,j}(\gamma)$ is always non-negative;
3. **Local Support:** The basis $B_{i,j}(\gamma)$ is a non-zero polynomial for $\gamma \in [\gamma_i, \gamma_{i+j+1})$. Moreover, on the span $[\gamma_i, \gamma_{i+1})$ at most $j+1$ degree j basis functions are non-zero, i.e. $B_u = [B_{i-j,j}, ..., B_{i,j}]$;
4. **Partition of unity:** From the recursive relation defined in (2.33), for any valid γ, the non-null B-spline functions are positive and add up to 1;
5. **Convex Hull:** The B-spline is contained in the convex hull of its control points;
6. **Continuity:** At a knot of multiplicity p, the basis functions $B_{ij}, i = 1, ..., n$ is $\mathcal{C}^{j-p}$ continuous. If we consider uniform B-Splines the knot multiplicity is $p = 1$, hence the basis functions B_{ij} are $\mathcal{C}^{j-1}$ continuous.

For an in-depth overview of the introduced properties, the reader is referred to [40] and to [41] for formal proofs.

2.4.2 Uniform Cubic B-Spline Curves (a practical overview)

Uniform cubic B-Splines are constructed on the assumption that each segment is given by cubic functions ($k = 3$). They constrain the points that joint the segments such that they meet the following continuity requirements:

- The final point on the Bézier curve i has the same coordinates as the first point on the Bézier curve $i + 1$ ($\mathcal{C}^0$ continuity).
- The first derivative at the end of the Bézier curve i is the same as the first derivative at the start of the Bézier curve $i + 1$ - no abrupt change in slope at transition points ($\mathcal{C}^1$ continuity).
- The second derivative at the end of the Bézier curve i is the same as the second derivative at the start of the Bézier curve $i + 1$ - no abrupt change in polarity at transition points ($\mathcal{C}^2$ continuity).

which can be verified according to property 6. Consider a unidimensional, uniform cubic B-spline ($k = 3$) with only 1 segment, hence only 4 control points and a knot vector given by:

$$\mathbf{U} = [0, 1, 2, 3, 4, 5, 6, 7]^T. \tag{2.36}$$

From (2.31) in conjunction with (2.33), and taking into consideration the local support property, we know that a unidimensional B-Spline curve, $C(\gamma) \in \mathbb{R}, \forall\gamma \in [0, 1)$ is given by

$$\begin{aligned} C(\gamma) &= B_{0,3}(\gamma)P_0 + B_{1,3}(\gamma)P_1 + B_{2,3}(\gamma)P_2 + B_{3,3}(\gamma)P_3 \\ &= \frac{1}{6}[(1-\gamma)^3P_0 + (3\gamma^3 - 6\gamma^2 + 4)P_1 + (-3\gamma^3 + 3\gamma^2 + 3\gamma + 1)P_2 + \gamma^3P_3] \\ &= \frac{1}{6}[(\gamma^3 + 3\gamma^2 - 3\gamma + 1)P_0 + (3\gamma^3 - 6\gamma^2 + 4)P_1 + (-3\gamma^3 + 3\gamma^2 + 3\gamma + 1)P_2 + \gamma^3P_3], \end{aligned} \tag{2.37}$$

which according to [42], [43] and [44], can be expressed in a matrix notation as:

$$C(\gamma) := \underbrace{\frac{1}{6}\begin{bmatrix}\gamma^3 & \gamma^2 & \gamma & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}}_{\begin{bmatrix}B_{0,3}(\gamma) & B_{1,3}(\gamma) & B_{2,3}(\gamma) & B_{3,3}(\gamma)\end{bmatrix}}\begin{bmatrix}P_0\\ P_1\\ P_2\\ P_3\end{bmatrix}. \tag{2.38}$$

This notation is particularly useful as it can be extended for the general case of $n - k + 1$ segments, such that:

$$C_i(\gamma) := \underbrace{\frac{1}{6}\begin{bmatrix}(\gamma-i)^3 & (\gamma-i)^2 & (\gamma-i) & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}}_{\begin{bmatrix}B_{i,3}(\gamma) & B_{i+1,3}(\gamma) & B_{i+2,3}(\gamma) & B_{i+3,3}(\gamma)\end{bmatrix}}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix}, \tag{2.39}$$

where $\gamma \in [0, n-k+1)$ and $i := \lfloor \gamma \rfloor$, such that $\gamma - i \in [0, 1)$. For example, if $\gamma = 0.8$, then $i = 0$ and the corresponding curve segment is defined only by the control points P_0, P_1, P_2 and P_3. On the other hand, if $\gamma = 1.2$, then $i = 1$ and the corresponding segment dictated by control points P_1, P_2, P_3 and P_4. This representation is very useful in practice as it allows us to have some values pre-computed in memory, making the evaluation of the basis functions very fast.

Let us now define a vector of unidimensional control points $\mathbf{P} = [P_0, ..., P_n] \in \mathbb{R}^{n+1}$, a vector of distinct curve parameters $\boldsymbol{\gamma} = [\gamma_0, ..., \gamma_q] \in \mathbb{R}^{q+1}$ that we wish to evaluate our curve at, and $\mathbf{C}(\boldsymbol{\gamma}) \in \mathbb{R}^{q+1}$ the points on the curve. Consider now (2.31), applied to $\boldsymbol{\gamma}$ expressed in vectorial form such that

$$B = \begin{bmatrix} B_{0,3}(\gamma_0) & \dots & B_{n,3}(\gamma_0) \\ \vdots & \ddots & \vdots \\ B_{0,3}(\gamma_q) & \dots & B_{n,3}(\gamma_q) \end{bmatrix}, \tag{2.40}$$

$$\mathbf{C}(\boldsymbol{\gamma}) = B(\boldsymbol{\gamma}) \cdot \mathbf{P}, \tag{2.41}$$

where, for each line of matrix $B(\boldsymbol{\gamma}) \in \mathbb{R}^{(q+1)\times(n+1)}$, only 4 basis are different then zero and computed according to (2.39). Using this matrix formulation we also improve computation speed performance due to the possibility of parallelization of operations.

Computing the derivative $\partial C/\partial \gamma$ also becomes trivial. Since the B-Spline results from a linear combination of basis functions, we just need to compute the derivative of each basis function with respect to γ. Considering the vectorial case just introduced, we know before hand that only 4 basis per line are different than zero, and their derivative can be computed according to:

$$\begin{bmatrix} \partial B_{i,3}(\gamma)/\partial\gamma \\ \partial B_{i+1,3}(\gamma)/\partial\gamma \\ \partial B_{i+2,3}(\gamma)/\partial\gamma \\ \partial B_{i+3,3}(\gamma)/\partial\gamma \end{bmatrix}^T = \frac{1}{6}\begin{bmatrix} 3(\gamma-i)^2 & 2(\gamma-i) & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \tag{2.42}$$

The same rationale can be applied to obtain the second partial derivative of $C(\gamma)$ with respect to γ.

Remark: For a brief explanation of how to expand this rationale to the 2-dimensional case, where $\mathbf{C}(\gamma) \in \mathbb{R}^2$, refer to appendix C.1.

2.5 Optimization Problems Overview

In this section, a small revision on convex optimization problems is provided with the main concepts borrowed from Boyd et al. [45]. Given a non empty set $\mathbf{X} \subset \mathbb{R}^n$ and an objective function $f : \mathbb{R}^n \to \mathbb{R}$, a constrained optimization problem can be defined by

$$\begin{aligned} &\underset{x\in\mathbf{X}}{\text{minimize}}\ f(x) \\ &\text{subject to } g_i(x) \leq 0,\ 1 \leq i \leq p \\ &\qquad\qquad h_j(x) = 0,\ 1 \leq j \leq q \end{aligned} \tag{2.43}$$

where $\mathbf{X}$ is the feasible region defined by a set of inequality and equality constraints. Consider also the following definition and theorems:

Definition 2.7. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if

$$f((1-\alpha)\mathbf{x} + \alpha\mathbf{y}) \leq (1-\alpha)f(\mathbf{x}) + \alpha f(\mathbf{y}), \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n \text{ and } 0 \leq \alpha \leq 1. \tag{2.44}$$

Theorem 2.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex. If $\mathbf{x}^*$ is a local minimum $\Rightarrow$ $\mathbf{x}^*$ is a global minimum.*

A notion that is usually forgotten when looking at theorem 2.5 is that a convex function may not have a global minimum, for example: $f(x) = x$. An optimization problem formulated according to (2.43) is said to be convex if $f(\mathbf{x})$ is convex, each h_i is affine (i.e. of the form $h_j(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} + b_i$) and g_i is convex.

Theorem 2.6. *Consider an optimization problem given by (2.43). Assume that f, h and g are $\mathcal{C}^1$. If $\mathbf{x}^*$ is a regular local minimum, there exists a $\boldsymbol{\lambda}^* \in \mathbb{R}^n$, $\boldsymbol{\mu}^* \in \mathbb{R}^m$ such that*

$$\begin{cases} \nabla f(\mathbf{x}^*) + \nabla h(\mathbf{x}^*)\boldsymbol{\lambda}^* + \nabla g(\mathbf{x}^*)\boldsymbol{\mu}^* = 0 \\ h(\mathbf{x}^*) = 0, g(\mathbf{x}^*) \leq 0 \\ \boldsymbol{\mu}^* \geq 0 \\ g(\mathbf{x}^*)^T\boldsymbol{\mu}^* = 0 \end{cases}, \tag{2.45}$$

where $\nabla h(\mathbf{x}^*) = [\nabla h_1(\mathbf{x}^*)...\nabla h_p(\mathbf{x}^*)]$ and $\nabla g(\mathbf{x}^*) = [\nabla g_1(\mathbf{x}^*)...\nabla g_m(\mathbf{x}^*)]$.

The Karush Kuhn Tucker (KKT) conditions are necessary conditions for an optimal solution. If the optimization problem is convex, then the KKT conditions are both necessary and sufficient for the optimal solution to be a global optimum.

2.5.1 Cubic B-Spline Fitting - Unconstrained Problem

Consider the problem of fitting a uniform cubic B-spline $\mathbf{C}(\gamma) \in \mathbb{R}^2$ to a set of points $\mathbf{X} := \{\mathbf{X}_m\}_{m=1}^M \in \mathbb{R}^2$. This problem can be formulated as an optimization problem given by:

$$\begin{aligned} &\underset{\gamma_1,...\gamma_M,P_0,...,P_n}{\text{minimize}} \sum_{m=1}^{M} \|\mathbf{C}(\gamma_m, \mathbf{P}) - \mathbf{X}_m\|^2 + F_r \\ &\text{subject to } 0 \leq \gamma_m \leq \gamma_{m_{max}}, \forall m = 1, ..., M \end{aligned} \tag{2.46}$$

where γ_{max} is a constant and F_r is a regularization term. Given the formulation of a cubic B-Spline introduced in (2.41), we can express the cost function in a vectorial form by concatenating the X and Y-coordinates of each point in a single vector (see appendix C.1), such that the optimization problem can be expressed in matricial form is given by

$$\begin{aligned} &\underset{\boldsymbol{\gamma},\mathbf{P}}{\text{minimize}} \ \|B(\boldsymbol{\gamma})\mathbf{P} - \mathbf{X}\|^2 + F_r \\ &\text{subject to } \mathbf{0} \leq \boldsymbol{\gamma} \leq \boldsymbol{\gamma}_{max} \end{aligned} \tag{2.47}$$

where $\boldsymbol{\gamma}_{max}$ is a constant vector. In the literature, it is common to find the F_r term to be given by

$$F_r = \lambda \int \left\| \frac{\partial \mathbf{C}(\gamma, \mathbf{P})}{\partial \gamma} \right\|^2 d\gamma + \beta \int \left\| \frac{\partial^2 \mathbf{C}(\gamma, \mathbf{P})}{\partial \gamma^2} \right\|^2 d\gamma, \tag{2.48}$$

where $\lambda > 0$ and $\beta > 0$ are positive constants. The main goal of the first term is to minimize the length of the curve by minimizing the L_2^2 norm of the first derivative, while the second term has the objective of minimizing bends on the curve. The constraints on the $\boldsymbol{\gamma}$ vector in (2.47) coupled with the attribution of a γ_m associated with each point $\mathbf{X}_m$ leads to an iterative approach to solve the minimization problem given by algorithm 1. In this algorithm, step 3 can be the most expensive to compute, as computing the

Algorithm 1 Iterative minimization approach

1: Define the number of B-spline segments to use;
2: Define a good initialization for $\mathbf{P}$;
3: Assign to each point $\mathbf{X}_m$ a parameter γ_m such that $\mathbf{X}_m$ is the closest point to $\mathbf{C}(\gamma_m)$;
4: Solve the optimization problem(2.47) with respect only to the control points $\mathbf{P}_0, ..., \mathbf{P}_n$;
5: **if** optimization error < threshold **then**
6: Stop the optimization;
7: **else**
8: Go to step 3.

closest point to the curve is in itself an optimization problem that is solved iteratively. Furthermore, this technique might not be feasible for all applications as the optimization of parameters γ_m associated to the corresponding $\mathbf{X}_m$ can get stuck in local minima leading to subpar solutions, especially in the presence of outliers in the data.

Another approach taken by Liu et al. [46] used to solve a similar optimization problem in real time for Simultaneous Localization and Mapping (SLAM) applications was to ditch step 2 altogether. In their works, the authors assume that the set of points is ordered, and from that they can assign γ_m for each $\mathbf{X}_m$ based on a normalized distance between the points themselves. Consider D_X to be the total distance between the points we wish to fit, given by

$$D_X := \sum_{m=2}^{M} \|\mathbf{X}_m - \mathbf{X}_{m-1}\|. \tag{2.49}$$

Then the vector of spline parametric values $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_m]^T$ is computed according to

$$\begin{cases} \gamma_1 = 0 \\ \gamma_m = \gamma_{m-1} + \frac{\|\mathbf{X}_m - \mathbf{X}_{m-1}\|}{D_X} \gamma_{max}, m = 2, ..., M \end{cases}, \tag{2.50}$$

where γ_{max} is defined by the number of control points that the target B-spline will have. This approach might not lead to the optimal solution, as $\mathbf{C}(\gamma_m)$ might not be the closest point to $\mathbf{X}_m$, but if obtaining very high accuracy fits is not a concern, then this method provides a good trade-off between time-complexity and accuracy. Regarding (2.48), the reader might be led to believe that the proposed integrals must be computed numerically in every iteration of the optimization problem. Fortunately, due to the representa-

tion introduced in (2.41), F_r can be computed in a very efficient way. Consider:

$$\int \|\mathbf{C}^m(\gamma)\|^2 \, d\gamma = \int \|B^m(\gamma)\mathbf{P}\|^2 \, d\gamma = \int \mathbf{P}^T B^m(\gamma) B^m(\gamma)^T \mathbf{P} d\gamma = \mathbf{P}^T R \mathbf{P}, \tag{2.51}$$

where m denotes the derivative of order m with respect to γ and R is a matrix that can be computed numerically a priori and stored in memory, as long as the number of B-spline segments is fixed between iterations (see appendix C.2).

Taking into consideration all the simplifications introduced, the unconstrained optimization fitting problem is now given by

$$\underset{\mathbf{P}}{\text{minimize}} \; \underbrace{\|B\mathbf{P} - \mathbf{X}\|^2 + \lambda \mathbf{P}^T R_1 \mathbf{P} + \beta \mathbf{P}^T R_2 \mathbf{P}}_{f(\mathbf{P})} \tag{2.52}$$

for which the closed form solution for the penalized least squares fit is given by

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}} f(\mathbf{P}) = [\underbrace{B^T B + \lambda R_1 + \beta R_2}_{W}]^{-1} B^T \mathbf{X}, \tag{2.53}$$

as long as matrix W is full-rank, hence invertible.

2.5.2 Cubic B-Spline Fitting - Constrained Problem

When considering a more general problem of fitting a set of points using a cubic B-Spline according to equality and/or inequality constraints we usually resort to an optimization technique named Sequential Quadratic Programming (SQP). This technique is based on Newton's method and it makes use of KKT conditions. For a more detailed overview of the inner-workings of this optimization algorithm, refer to [47]. Consider now that we want to solve a fitting problem given by:

$$\begin{aligned} &\underset{\mathbf{P}}{\text{minimize}} \; \underbrace{\|B(\gamma)\mathbf{P} - \mathbf{X}\|^2 + \lambda \mathbf{P}^T R_1 \mathbf{P} + \beta \mathbf{P}^T R_2 \mathbf{P}}_{f(\mathbf{P})} \\ &\text{subject to } A\mathbf{P} = \mathbf{Y} \end{aligned} \tag{2.54}$$

with $A \in \mathbb{R}^{n \times n}$ and $\mathbf{Y} \in \mathbb{R}^n$ a constant vector. This problem is well posed since the equality constraints are affine functions of the control points. Moreover, $f(\mathbf{P})$ is a convex objective function as it results from the sum of a norm of an affine function with two quadratics in which $R_1 \succeq 0$ and $R_2 \succeq 0$. In order to solve constrained optimization problems of this type we can resort to the, minimize, function provided in Scipy's python package [48] which implements a SQP solver.

Chapter 3

Vehicle Models

In this section a mathematical model is obtained for an AUV and a quadrotor UAV moving in a 3-D space, i.e. with 6 DOF. Since it is assumed that the AUV will be operating in ASV mode, i.e. only working on a 2-D plane, further simplifications to its model are introduced, achieving a final model with only 3 DOF.

3.1 Notation and Reference Frames

The terminology regarding coordinates and reference frames adopted for both vehicles is depicted in Figure 3.1. The *Inertial Reference Frame* $\{U\}$ is composed by $\{x_U, y_U, z_U\}$, it follows the North-East-Down reference frame convention (NED) and it can be "attached" to any fixed place on Earth. The *Body Reference Frame* $\{B\}$ is composed by $\{x_B, y_B, z_B\}$, it is attached to each vehicle's center of mass, and its axis correspond to the vehicle's principal axis of inertia.

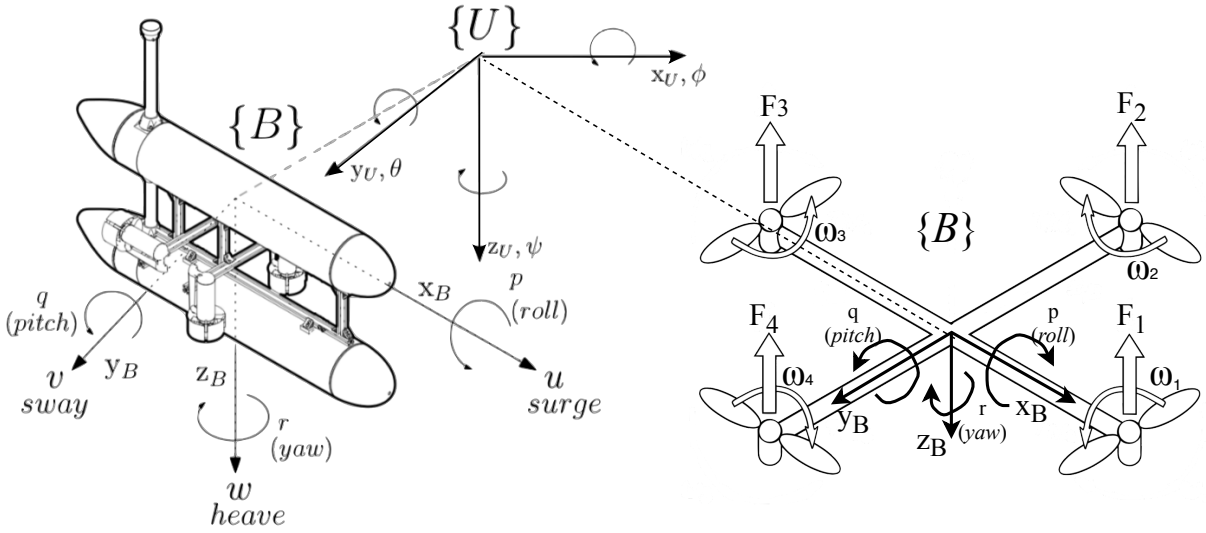


Figure 3.1: Adopted reference frames (adapted from Teixeira et al. [49] and Luukkonen T. [50])

To keep a consistent notation between both the AUV and UAV, the nomenclature adopted for both vehicles will follow the Society of Naval Architects and Marine Engineers (SNAME) convention to represent

the vehicle's pose, speed, torque and forces according to:

- $\{B\}$ - Body-fixed frame rigidly attached to the geometric center of mass of the vehicle;
- $\{U\}$ - Inertial reference frame;
- $\mathbf{v}_1 = [u, v, w]^T$ - Linear velocity of the origin of $\{B\}$ with respect to $\{U\}$ expressed in $\{B\}$;
- $\mathbf{v}_2 = [p, q, r]^T$ - Angular velocity of the origin of $\{B\}$ with respect to $\{U\}$ expressed in $\{B\}$;
- $\boldsymbol{\eta_1} = [x, y, z]^T$ - Position of the origin of $\{B\}$ measured in $\{U\}$;
- $\boldsymbol{\eta_2} = [\phi, \theta, \psi]^T$ - Orientation of $\{B\}$ with respect to $\{U\}$, expressed in ZYX Euler angles;
- $\boldsymbol{F}_{RB} = [X, Y, Z]^T$: External forces measured in $\{B\}$;
- $\boldsymbol{N}_{RB} = [K, M, N]^T$: External torques measured in $\{B\}$.

The table 3.1 summarizes the notation adopted.

Table 3.1: Notation adopted (adapted from Fossen et al. [51])

	forces and moments	linear and angular velocity	positions and Euler angles
motions in the X-direction (surge)	X	u	x
motions in the Y-direction (sway)	Y	v	y
motions in the Z-direction (heave)	Z	w	z
rotation about the X-axis (roll)	K	p	ϕ
rotation about the Y-axis (pitch)	M	q	θ
rotation about the Z-axis (yaw)	N	r	ψ

3.2 Kinematics

The kinematics describe motion of the vehicles based on purely geometric concepts, relating linear and angular velocities with position and orientation. Making use of the reference frames defined in Section 3.1, the kinematic equations can be described by

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix}}_{\boldsymbol{\eta}} = \underbrace{\begin{bmatrix} {}^U_B R(\boldsymbol{\eta}_2) & 0_{3x3} \\ 0_{3x3} & Q(\boldsymbol{\eta}_2) \end{bmatrix}}_{J(\boldsymbol{\eta})} \underbrace{\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{bmatrix}}_{\mathbf{v}}, \tag{3.1}$$

where $\boldsymbol{\eta} = [\boldsymbol{\eta_1}, \boldsymbol{\eta_2}]^T$. The rotation matrix ${}^U_B R(\boldsymbol{\eta}_2)$ is obtained by performing a series of rotations such as

$${}^U_B R(\boldsymbol{\eta}_2) = R_z(\psi) R_y(\theta) R_x(\phi). \tag{3.2}$$

The complete rotation matrix can be given by

$${}^U_B R(\boldsymbol{\eta}_2) = \begin{bmatrix} c\psi s\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}, \tag{3.3}$$

where c and s denote the trigonometric functions $cos(.)$ and $sin(.)$ respectively. The $Q(\boldsymbol{\eta}_2)$ is an angular velocity transformation matrix that represents the angular velocity transformation from $\{B\}$ to $\{U\}$ and is expressed as

$$Q(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}, \theta \neq \pm 90^\circ, \tag{3.4}$$

where t denotes the trigonometric function $tan(.)$. It is worth mentioning that matrix $Q(\boldsymbol{\eta_2})$ presents a singularity at $\theta \neq \pm 90^\circ$. In order to solve this issue, a quaternion formulation of the problem could be considered. Due to the nature of the problem being solved, it is assumed that the vehicle at the surface of the water will be operating far from this singularity ($\theta \approx 0^\circ$ and $\phi \approx 0^\circ$). Furthermore, it is assumed that the quadrotor vehicle will also be operating around its equilibrium point, thus eliminating the need to add an extra layer of complexity to the problem.

3.3 AUV Dynamics

The dynamics model will be used to study how forces and torques applied to a vehicle affect its motion. In order to model the dynamics of both vehicles, a Newton-Euler approach will be used. Let the rigid-body equation of the AUV be given by

$$M_{RB}\dot{\mathbf{v}} + C_{RB}(\mathbf{v})\mathbf{v} = \boldsymbol{\tau}_{RB}, \tag{3.5}$$

where M_{RB} is the rigid body inertia matrix, $C_{RB}(\mathbf{v})\mathbf{v}$ the Coriolis, centripetal and gyroscopic terms and $\boldsymbol{\tau}_{RB}$ the a generalized vector of external forces and torques expressed as

$$\boldsymbol{\tau}_{RB} = \begin{bmatrix} \sum \mathbf{F}_{RB}^T & \sum \mathbf{N}_{RB}^T \end{bmatrix}^T. \tag{3.6}$$

This vector of external forces and torques can be decomposed into

$$\boldsymbol{\tau}_{RB} = \boldsymbol{\tau} + \boldsymbol{\tau}_A + \boldsymbol{\tau}_D + \boldsymbol{\tau}_R + \boldsymbol{\tau}_{dist}, \tag{3.7}$$

where each term represents:

- $\boldsymbol{\tau}$ - Vector of forces and torques due to thrusters/surfaces, which can be viewed as the generalized control input;
- $\boldsymbol{\tau}_A$ - Vector of forces and torques due to the hydrodynamic added mass, given by

$$\boldsymbol{\tau}_A = -M_A\dot{\mathbf{v}} - C_A(\mathbf{v})\mathbf{v}; \tag{3.8}$$

- $\boldsymbol{\tau}_D$ - Hydrodynamics terms due to lift, drag, skin friction, etc. given by

$$\boldsymbol{\tau}_D = -D(\mathbf{v})\mathbf{v}; \tag{3.9}$$

- $\boldsymbol{\tau}_R$ - Restoring forces and torques due to gravity and fluid density, given by

$$\boldsymbol{\tau}_R = -g(\boldsymbol{\eta}); \tag{3.10}$$

- $\boldsymbol{\tau}_{dist}$ = Vector that represents external disturbances such as currents, wind, etc.

By replacing the terms in equation (3.7) by (3.8), (3.9) and (3.10) and then applying them to the dynamic equation in (3.5) yields

$$\underbrace{M_{RB}\dot{\mathbf{v}} + C_{RB}(\mathbf{v})\mathbf{v}}_{\textit{rigid-body terms}} + \underbrace{M_A\dot{\mathbf{v}} + C_A(\mathbf{v})\mathbf{v} + D(\mathbf{v})\mathbf{v}}_{\textit{hydrodynamic terms}} + \underbrace{g(\boldsymbol{\eta})}_{\textit{restoring term}} = \underbrace{\boldsymbol{\tau} + \boldsymbol{\tau}_{dist}}_{\textit{applied forces and torques}} . \tag{3.11}$$

3.4 AUV Simplified Equations of Motion

In the context of this work, the AUV is assumed to be underactuated and to only operate at the surface of the water which can be approximated by a 2-D plane. In this new case, the vehicle only has 3 DOF given by $[x, y, \psi]^T$ given that $\phi = 0$, $\theta = 0$ and $z = 0$, leading to a simplification in the transformation matrix defined in (3.1). Furthermore, we introduce at the kinematic level the ocean current velocities $\mathbf{v}_c = [v_{cx}, v_{cy}]^T$, assumed to be constant and irrotational, i.e. $\dot{\mathbf{v}}_c = 0$. Given this notion, the equations of motion are re-written and the surge u and sway v speeds are now defined as velocities of the vehicle with respect to a water-fixed frame $\{W\}$ moving at $\mathbf{v}_c$ with respect to $\{U\}$. The kinematics of the AUV operating in ASV mode with 3 DOF are now given by

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}}_{\dot{\mathbf{p}}} = \underbrace{\begin{bmatrix}cos(\psi) & -sin(\psi)\\ sin(\psi) & cos(\psi)\end{bmatrix}}_{R(\psi)} \underbrace{\begin{bmatrix}u\\ v\end{bmatrix}}_{\mathbf{v}} + \underbrace{\begin{bmatrix}v_{cx}\\ v_{cy}\end{bmatrix}}_{\mathbf{v}_c}, \tag{3.12}$$

$$\dot{\psi} = r. \tag{3.13}$$

Also neglecting motion in roll, pitch and heave the simplified dynamic equations for $[u, v, r]^T$ are now given by

$$\begin{aligned} m_u\dot{u} - m_v vr + d_u u &= \tau_u;\\ m_v\dot{v} + m_u ur + d_v v &= 0;\\ m_r\dot{r} - m_{uv}uv + d_r r &= \tau_r, \end{aligned} \tag{3.14}$$

where τ_u is the external force in surge (common mode), τ_r is the external torque about the Z-axis (differential mode) and

$$\begin{aligned} m_u &= m - X_{\dot{u}}, \quad & d_u &= -X_u - X_{|u|u}|u|,\\ m_v &= m - Y_{\dot{v}}, \quad & d_v &= -Y_v - Y_{|v|v}|v|,\\ m_r &= I_z - N_{\dot{r}}, \quad & d_r &= -N_r - N_{|r|r}|r|,\\ m_{uv} &= m_u - m_v, \end{aligned} \tag{3.15}$$

where m_u, m_v, m_r and m_{uv} represent the mass and hydrodynamic added mass and d_u, d_v and d_r the hydrodynamic damping effects. These equations of motion are used to model the MEDUSA class of underactuated vehicles described in detail in Appendix A.

3.5 UAV Quadrotor Dynamics

3.5.1 Assumptions taken into consideration

In order to simplify the modelling of the quadrotor UAV several assumptions were made, namely:

- The quadrotor is symmetric along the X-axis and Y-axis;
- Aerodynamic drag forces are negligible;
- Rotors flapping effect is ignored;
- All rotors have exactly the same specifications, hence the same constants.
- Other external disturbances are neglected (assumption lifted later on).

3.5.2 Quadrotor Actuator Dynamics

As shown in Figure 3.1, the angular velocity of each rotor ω_i generates a corresponding force F_i in the direction of the correspondent rotor axis. In addition, the angular velocity and acceleration of the rotor also creates torque τ_{Mi} around the rotor axis. It is known that for a quadrotor, the thrust and reaction to torque is approximately proportional to the square of the angular velocity of the rotor ω_i, such that:

$$\begin{aligned} F_i &= c_T \omega_i^2, \\ Q_i &= c_Q \omega_i^2 (-1)^{i+1}, \end{aligned} \tag{3.16}$$

where the parameters c_T and c_Q are constant values that can be identified experimentally using static thrust tests with varying payloads. By combining the forces F_i that each rotor generates we obtain a thrust Z in the direction of the Z-axis of the body reference frame. The applied forces in the $\{B\}$ frame are given by

$$\boldsymbol{F}_{RB} = \begin{bmatrix} 0 \\ 0 \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ c_T \sum_{i=1}^{4} \omega_i^2 \end{bmatrix}. \tag{3.17}$$

In addition a torque $\boldsymbol{N}_{RB}$ expressed in the body frame $\{B\}$ is also generated and it is computed as a function of the forces generated in each rotor by:

$$\boldsymbol{N}_{RB} = \begin{bmatrix} K \\ M \\ N \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_1 - F_3) \\ \frac{c_Q}{c_T}(F_1 - F_2 + F_3 - F_4) \end{bmatrix}, \tag{3.18}$$

where l is the arm-length of the quadrotor, measured from the center of mass to one of the rotors.

3.5.3 Quadrotor Rigid Body Dynamics

Similar to the AUV case, the quadrotor is a rigid body, which means that the Newton-Euler equations can also be used to deduce its dynamics. Therefore, the equation for the translational movement expressed

in the body reference frame $\{B\}$ is given by

$$\begin{aligned} \underbrace{mg_U^B R(\boldsymbol{\eta}_2)\mathbf{e}_3}_{\textit{gravitational force in \{B\}}} - \underbrace{\boldsymbol{F}_{RB}}_{thrust} &= m\underbrace{(\mathbf{v}_2 \times \mathbf{v}_1 + \dot{\mathbf{v}}_1)}_{\textit{acceleration in \{B\}}} \\ \Leftrightarrow m\dot{\mathbf{v}}_1 &= -mS(\mathbf{v}_2)\mathbf{v}_1 - Z\mathbf{e}_3 + mg_U^B R(\boldsymbol{\eta}_2)\mathbf{e}_3, \end{aligned} \tag{3.19}$$

where m denotes the mass of the vehicle, $\mathbf{e}_3$ a unit vector with all components equal to zero except the third, Z the total thrust and $S(\boldsymbol{\eta_2})$ a skew-symmetric matrix used as an alternative way of expressing the cross product, according to (2.2). The equation for the rotational dynamics of the vehicle expressed in the body reference frame $\{B\}$ are given by

$$\begin{aligned} J\dot{\mathbf{v}}_{\mathbf{2}} &= -\mathbf{v}_2 \times (J\mathbf{v}_2) + \underbrace{\boldsymbol{N}_{RB}}_{\textit{torque}} \\ \Leftrightarrow J\dot{\mathbf{v}}_{\mathbf{2}} &= -S(\mathbf{v}_2)J\mathbf{v}_2 + \boldsymbol{N}_{RB}, \end{aligned} \tag{3.20}$$

where J is the matrix of Inertia.

3.6 UAV Quadrotor Equations of Motion

In the previous section, the equations that modelled the dynamics of the quadrotor in the $\{B\}$ frame were presented. By taking into account (3.19) and representing it in the inertial reference frame $\{U\}$ we get

$$\begin{aligned} \ddot{\boldsymbol{\eta}}_{\mathbf{1}} &= g\mathbf{e}_3 - \frac{1}{m}{}_B^U R(\boldsymbol{\eta}_2)\boldsymbol{F}_{RB} \\ \Leftrightarrow \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \frac{1}{m}{}_B^U R(\boldsymbol{\eta}_2) \begin{bmatrix} 0 \\ 0 \\ Z \end{bmatrix}. \end{aligned} \tag{3.21}$$

Taking into consideration (3.20) and expanding it, the dynamics for rotational motion are given by

$$\begin{aligned} \dot{\mathbf{v}}_2 &= -J^{-1}(\mathbf{v}_2 \times J\mathbf{v}_2) + J^{-1}\boldsymbol{N}_{RB} \\ \Leftrightarrow \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} &= -J^{-1}\left[\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times J \begin{bmatrix} p \\ q \\ r \end{bmatrix}\right] + J^{-1}\begin{bmatrix} K \\ M \\ N \end{bmatrix}. \end{aligned} \tag{3.22}$$

From the kinematics equations defined in section 3.2 it is trivial to convert from $\mathbf{v}_2$ to $\boldsymbol{\eta}_2$ by taking

$$\begin{aligned} \dot{\boldsymbol{\eta}}_2 &= Q(\boldsymbol{\eta}_2)\mathbf{v}_2 \\ \Leftrightarrow \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \theta \neq \pm 90^\circ. \end{aligned} \tag{3.23}$$

The quadrotor vehicle adopted for simulation is described in detail in Appendix B.

Chapter 4

Vehicle Motion Control

In the previous chapter, a set of mathematical models were used to describe the behaviour of both an ASV operating at the surface of water, and a quadrotor UAV. As the development of commercial autonomous vehicles is becoming wide-spread, it is common for those vehicles to already provide a set of low-level control schemes that take into consideration the dynamics model of the vehicle. This allows the control designer to spend more time in the development of motion planning algorithms. Taking queues from this industry practice, the single-vehicle path following problem is solved by adopting an inner-outer loop control structure for both vehicles. At the inner-loop level, the controller is typically required to act on the thrusters/rotors of the vehicle in order to generate a set of forces and torques. On the other hand, at the outer-loop level, the controller is responsible for implementing a guidance law that steers the vehicle to a desired path by generating control references for the inner-loop system to follow. In this chapter we focus on the development of the inner-loop control systems.

4.1 ASV Inner-Loop Design

Due to the nature of the ASV vehicles, it is common practice for higher level controls to generate a set of references for the speed of the vehicle in surge u and the yaw angular rate r about the Z-axis of the rigid body frame $\{B\}$. For the sake of simplicity, two decoupled linear control laws are derived, given that the vehicle will operate at low speeds.

Problem 4.1. Consider the ASV with dynamics described by (3.14) and let $\mathbf{u}_d^\dagger = [u_d, r_d]^T \in \mathbb{R}^2$ denote the desired surge speed and yaw-rate respectively. Linearize the vehicle dynamics and design a linear control law for the force in surge τ_u and external torque about the Z-axis, τ_r such that $\mathbf{u}^\dagger = [u, r]$ converges to a desired set of surge and yaw-rate references $\mathbf{u}_d^\dagger$.

4.1.1 Surge Speed Control

In this section, the goal is to derive a control law that enables the vehicle to follow a desired surge speed reference. Taking into consideration the vehicle's operating conditions, the following assumption is taken:

Assumption 4.1. The sway-motion of the ASV is negligible, i.e $v \approx 0$.

Considering the equations introduced in (3.14) and assumption 4.1, the surge dynamics of the ASV are described by

$$\dot{u} = \frac{1}{m_u} \left[\tau_u - d_u u \right]. \tag{4.1}$$

Define a set of error dynamics of this sub-system:

$$\begin{cases} e(t) = u(t) - u_d(t) \\ \dot{e}(t) = \dot{u}(t) - \dot{u}_d(t) \end{cases}. \tag{4.2}$$

Given that the vehicle will be required to operate at an approximately constant nominal speed of 0.5 m/s, then we can consider $\dot{u}_d(t) \approx 0$. Replacing (4.1) in (4.2) yields:

$$\dot{e}(t) = \dot{u}(t) = \frac{1}{m_u} \left[\tau_u - d_u u \right]. \tag{4.3}$$

Proposal Proportional Integral (PI) control law with feed-forward term:

$$\tau_u = d_u u + m_u \left[-k_p e(t) - k_i \int_0^t e(\tau) d\tau \right], \text{ with } k_p, k_i > 0. \tag{4.4}$$

Substituting the proposed control law in equation (4.3) yields

$$\dot{e}(t) = -k_p e(t) - k_i \int_0^t e(\tau) d\tau. \tag{4.5}$$

Applying the Laplace transform to the above system and considering the existence of an external constant disturbance $w(t)$, the described feedback control can be resumed in Figure 4.1. By applying the separation theorem to the system described by equation (4.5) we get the following result:

$$U(s) = \frac{k_p s + k_i}{s^2 + k_p s + k_i} U_d(s) + \frac{s}{s^2 + k_p s + k_i} W(s). \tag{4.6}$$

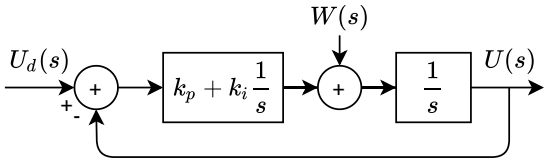


Figure 4.1: System analysis in the frequency domain

We can conclude by applying the Final Value Theorem that external constant disturbances represented by $w(t)$ are rejected by controller (4.4), that is,

$$\lim_{t \to \infty} u(t) = \lim_{s \to 0} sU(s) = \lim_{s \to 0} \frac{\omega_0 s}{s^2 + k_p s + k_i} = 0. \tag{4.7}$$

4.1.2 Heading Rate Control

Following a similar approach to the one taken in the design of the previous controller, a linear control law is also derived for the heading rate. Once again, taking into consideration the equations introduced in (3.14) and the assumption 4.1:

$$\dot{r} = \frac{1}{m_r}\left[\tau_r - d_r r\right]. \tag{4.8}$$

Defining the error dynamics:

$$\begin{cases} e(t) = r(t) - r_d(t) \\ \dot{e}(t) = \dot{r}(t) - \dot{r}_d(t) \end{cases}. \tag{4.9}$$

Considering that $\dot{r}_d(t) \approx 0$. Replacing (4.8) in (4.9) yields:

$$\dot{e}(t) = \dot{r}(t) = \frac{1}{m_r}\left[\tau_r - d_r r\right]. \tag{4.10}$$

Proposal PI control law with feed-forward term:

$$\tau_r = d_r r + m_r\left[-k_p e(t) - k_i \int_0^t e(\tau)d\tau\right], \text{ with } k_p, k_i > 0. \tag{4.11}$$

Replacing the proposed control law in (4.11) in (4.10):

$$\dot{e}(t) = -k_p e(t) - k_i \int_0^t e(\tau)d\tau. \tag{4.12}$$

which enjoys the same constant disturbance rejection properties as the designed surge speed controller.

Remark: In general, whenever an integrator is used in a control law it is good practice to implement an anti-windup scheme in order to guarantee that the integral term does not grow unbounded. In this work we do not go in detail on the inner-workings of this mechanism, but refer to [52] where our implementation is based upon.

4.1.3 Ocean Currents Observer

In order to develop a robust path following controller for an ASV, it is important to accurately measure the velocity of ocean currents. However, it is not possible to measure these variables directly with the sensors provided by the Medusa class of vehicles. Therefore, a viable alternative is to develop a currents observer.

Problem 4.2. Consider an ASV vehicle with kinematics given by (3.12), equipped with a Doppler Velocity Logger (DVL) system working in water lock mode, capable of providing the vehicle's relative velocity to the water, expressed in $\{B\}$ and a Differential Global Positioning System (DGPS) unit which provides measurements of the position of the vehicle $\mathbf{p}_m$, expressed in $\{U\}$. Furthermore, consider that it is possible to express the velocities provided by the DVL in $\{U\}$, by resorting to the rotation matrix ${}^U_B R(\psi)$, as $\mathbf{v}_m$. Develop an estimator for the ocean-currents $\mathbf{v}_c$ expressed in $\{U\}$ which are assumed to be constant and irrotational, i.e. $\dot{\mathbf{v}}_c = 0$.

In order to solve this problem, we borrow the results from Pascoal et al.[53] and Sanches et al.[54] where the authors propose a time-varying complementary filter structure, according to Figure 4.2.

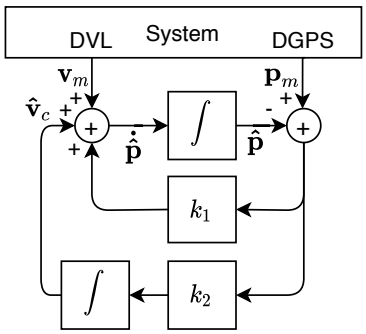


Figure 4.2: Currents estimator using a complementary filter structure

Proposition 4.1. *Consider the process model $\mathcal{M}_p$ given by*

$$\mathcal{M}_p := \begin{cases} \dot{\mathbf{p}} = {}^{U}_{B}R(\psi)\mathbf{v} + \mathbf{v}_c \\ \mathbf{v}_m = {}^{U}_{B}R(\psi)\mathbf{v} \\ \mathbf{p}_m = \mathbf{p} \end{cases}, \tag{4.13}$$

and the candidate complementary filter model described by

$$\mathcal{F} := \begin{cases} \dot{\hat{\mathbf{p}}} = k_1(\mathbf{p}_m - \hat{\mathbf{p}}) + \mathbf{v}_m + \hat{\mathbf{v}}_c \\ \dot{\hat{\mathbf{v}}}_c = k_2(\mathbf{p}_m - \hat{\mathbf{p}}) \end{cases}, \tag{4.14}$$

with k_1 and k_2 positive constants. The proposed complementary filter is asymptotically stable and solves problem 4.2.

The reader is referred to the original work by Pascoal et al. [53] for an in-depth proof of proposition 4.1 where a detailed stability analysis is provided for several complementary filters designs.

4.2 AUV Quadrotor Inner-Loop Design

Given the quadrotor thruster configuration, it is typical to have an inner-loop controller whose task is to follow attitude reference commands. Once again, due to the nature of the missions to be conducted and the required operating speeds, it suffices to have a linear inner-loop controller.

Problem 4.3. Consider the AUV quadrotor described by (3.22) and let $\boldsymbol{\eta}_{2d} = [\phi_d, \theta_d, \psi_d] \in \mathbb{R}^3$ denote the desired roll, pitch and yaw angles respectively. Linearize the vehicle dynamics and design a linear control law for the external torque $\boldsymbol{N}_{RB}$ about the X, Y and Z-axis, such that $\boldsymbol{\eta}_2$ converges to desired set of angle references $\boldsymbol{\eta}_{2d}$.

Assumption 4.2. The quadrotor's inertia matrix J is diagonal.

Assumption 4.3. The vehicle is working near its hover state, where $\phi \approx \theta \approx 0°$.

Consider (3.22) and assumption 4.2. Then the rotational dynamics of the quadrotor are given by

$$\begin{cases} \dot{p} = \frac{J_{yy}-J_{zz}}{J_{xx}} qr + \frac{K}{J_{xx}} \\ \dot{q} = \frac{J_{zz}-J_{xx}}{J_{yy}} pr + \frac{M}{J_{yy}} \\ \dot{r} = \frac{J_{xx}-J_{yy}}{J_{zz}} pq + \frac{N}{J_{zz}} \end{cases} . \tag{4.15}$$

By taking into consideration assumption 4.3, we know that near hover state:

$$sin(\phi) \approx \phi, sin(\theta) \approx \theta \text{ and } cos(\phi) \approx cos(\theta) \approx 1. \tag{4.16}$$

Replacing (4.16) in (3.23) yields:

$$\begin{cases} p = \dot{\phi} - \dot{\psi}\theta \\ q = \dot{\theta} + \dot{\psi}\phi \\ r = \dot{\psi} - \dot{\theta}\phi \end{cases} . \tag{4.17}$$

Considering that the higher order terms $\dot{\psi}\theta \approx \dot{\psi}\phi \approx \dot{\phi}\theta \approx 0$ and that $qr \approx pr \approx pq \approx 0$ we finally get that near hover state

$$\ddot{\boldsymbol{\eta}}_2 = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 1/J_{xx} & 0 & 0 \\ 0 & 1/J_{yy} & 0 \\ 0 & 0 & 1/J_{yy} \end{bmatrix} \begin{bmatrix} K \\ M \\ N \end{bmatrix} = J^{-1} N_{RB}, \tag{4.18}$$

such that $N_{RB} = J \cdot \mathbf{u}_d^{\ddagger}$, where $\mathbf{u}_d^{\ddagger}$ represents the control input. Defining a set of error dynamics:

$$\begin{cases} \mathbf{e}(t) = \boldsymbol{\eta}_2(t) - \boldsymbol{\eta}_{2d}(t) \\ \dot{\mathbf{e}}(t) = \dot{\boldsymbol{\eta}}_2(t) - \dot{\boldsymbol{\eta}}_{2d}(t) \\ \ddot{\mathbf{e}}(t) = \ddot{\boldsymbol{\eta}}_2(t) - \ddot{\boldsymbol{\eta}}_{2d}(t) \end{cases} . \tag{4.19}$$

Considering that $\ddot{\boldsymbol{\eta}}_{2d} = \mathbf{0}$, the proposed Proportional Derivative (PD) control law is given by:

$$\mathbf{u}_d^{\ddagger}(t) = -K_p \mathbf{e}(t) - K_d \dot{\mathbf{e}}(t) \text{ with } K_p, K_d \succeq 0. \tag{4.20}$$

Replacing the control law in the error dynamics yields:

$$\ddot{\mathbf{e}}(t) + K_d \dot{\mathbf{e}}(t) + K_p \mathbf{e}(t) = 0. \tag{4.21}$$

With the designed controller, given the desired thrust and attitude is it possible to use equations (3.17) and (3.18) to compute the angular velocity to apply to each individual rotor.

4.2.1 Generating Angle References from accelerations

Generally, when designing an auto-pilot for a quadrotor it is easier to consider that the vehicle is modeled by a double integrator system for which the main goal is to drive the position error to zero. In these scenarios, the yaw angle ψ_{des} of the vehicle is also left as a free variable controlled by the auto-pilot. A side note worth mentioning is that the rotational motion of the quadrotor about the Z-axis is typically slow as the moment the rotors are capable of producing around this axis is very limited.

Consider the double integrator model of the vehicle given by

$$\ddot{\mathbf{p}} := \ddot{\boldsymbol{\eta}}_1 = \mathbf{u}^{\diamond} = -\frac{Z}{m}\underbrace{{}^{U}_{B}R(\boldsymbol{\eta}_2)\mathbf{e}_3}_{\mathbf{r}_3} + g\mathbf{e}_3, \tag{4.22}$$

where $\mathbf{u}^{\diamond}$ is the output of the auto-pilot/outer-loop. It is necessary to develop a sub-system capable of computing the total thrust Z and the attitude associated with the matrix ${}^{U}_{B}R(\boldsymbol{\eta}_2)$ from the desired input of the auto-pilot. Consider expanding the previous equation:

$$\mathbf{u}^{\diamond} = -\frac{1}{m}R_z(\psi_{des})[R_y(\theta)R_x(\phi)Z\mathbf{e}_3] + g\mathbf{e}_3. \tag{4.23}$$

Furthermore, consider $\mathbf{u}^*$ to be given by

$$\mathbf{u}^* := R_y(\theta)R_x(\phi)Z\mathbf{e}_3. \tag{4.24}$$

Replacing (4.24) in (4.23) yields the relation:

$$\mathbf{u}^* = -mR_z^T(\psi_{des})(\mathbf{u}^{\diamond} - g\mathbf{e}_3). \tag{4.25}$$

From which it is possible to compute $\mathbf{u}^*$, from the desired output of the outer-loop. Consider the total thrust to be given by $Z := \|\mathbf{u}^*\|$. Combining with the previous equations, the yields

$$\frac{\mathbf{u}^*}{\|\mathbf{u}^*\|} = \begin{bmatrix} cos(\phi)sin(\theta) \\ -sin(\phi) \\ cos(\theta)cos(\phi) \end{bmatrix}. \tag{4.26}$$

From this relation, it is possible to conclude that:

$$\phi_{des} = arcsin\left(-\frac{u_2^*}{Z}\right), \tag{4.27}$$

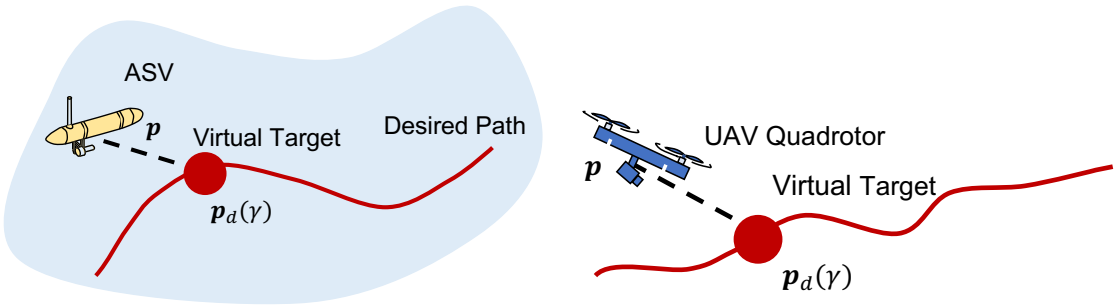
$$\theta_{des} = arctan\left(\frac{u_1^*}{u_3^*}\right), \tag{4.28}$$

such that $\mathbf{u}^* = [u_1^*, u_2^*, u_3^*]^T$, with u_3^* assumed to be different than zero, which is the typical case.

Chapter 5

Path Following

In this chapter, the path following problem applied to both ASVs and multirotor UAVs is addressed. In a path following problem, the vehicle is required to converge to and follow a path according to a desired speed profile. The key idea is to drive each vehicle to a virtual target that moves along the desired path, according to Figure 5.1. In order to solve the PF problem, the task is decomposed in two sub-tasks: *i)* steer the vehicle to remain inside a tube centered around the desired path; *ii)* assign a dynamic speed profile to the path to be followed.



(a) ASV path following

(b) UAV multirotor path following

Figure 5.1: Path following schematic

The first task should be solved individually for each vehicle, as the ASV only moves in a 2-D plane and the multirotor is able to move in a 3-D space. Moreover, the references generated by the outer-loop for each vehicle should also be different, as the ASV inner-loops track surge speed and yaw-rate references in $\{B\}$ while the quadrotor's track a set of desired accelerations in $\{U\}$.

For the ASV there are several PF algorithms that could be chosen, such as the one proposed by Samson et al. [14] or Lapierre et al. [55]. The main disadvantage of these algorithms is that it is not trivial to extend their rationale to quadrotor vehicles moving in a 3-D space. On the other hand, nonlinear controller proposals from Aguiar et al. [8] and Vanni et al. [56] provide a much more extensible framework that can be easily modified for applications to fully-actuated marine vehicles and aerial multirotor vehicles. For this reason, the PF controller adopted for the ASV vehicles follow the proposals of Aguiar and Vanni, while for the quadrotors a custom controller, inspired by the ASV controller design, is derived.

Common to both algorithms is the second task. Consider, for both vehicles that the desired speed

profile for a virtual target that moves along a desired path is given by:

$$v_d(\gamma, t) := v_L(\gamma) + v^{coord}(t) \text{ , with } |v_L(\gamma)| \leq v_L^{max}, \tag{5.1}$$

where $v_L(\gamma)$ is a desired speed profile as a function of the path, v_L^{max} a pre-defined speed upper-bound and $v^{coord}(t)$ a speed coordination term that will be used later for enabling CPF. The PF problem applied to ASVs is addressed in section 5.1 and for multirotor UAVs in section 5.2.

5.1 ASV Path Following

Problem 5.1. Consider the ASV with kinematics described by (3.12), and let $\mathbf{p}_d(\gamma) : [0, \infty) \to \mathbb{R}^2$ denote the desired path parameterised by a continuous variable $\gamma \in \mathbb{R}$ and $v_d(\gamma, t) \in \mathbb{R}$ be a desired speed profile for a virtual target moving along the desired path. Furthermore, consider $\mathbf{p}_d(\gamma)$ to be C^2 and have its first and second derivatives with respect to γ bounded. Moreover, the vehicle is equipped with an inner-loop controller that given a desired surge speed and yaw-rate $\mathbf{u}_d = [u_d, r_d]^T$, assumed to be bounded, computes a set of desired thrust and torques to apply to the vehicle. Design a control law for surge u_d, yaw-rate r_d and virtual target $\ddot{\gamma}$ such that:

- the vehicle's position converges to a tube around the desired position that can be made arbitrarily small, i.e. $\|\mathbf{p}(t) - \mathbf{p}_d(\gamma)\|$ converges to a neighbourhood of the origin;
- the speed of the virtual target moving along the path converges to the desired speed profile, i.e. $|\dot{\gamma} - v_d(\gamma, t)| \to 0$ as $t \to \infty$.

Following the approach proposed by Aguiar et al. [8], [15] and [57], consider the global diffeomorphic coordinate transformation which expresses the position error defined in the body-frame of the vehicle $\{B\}$ as

$$\mathbf{e}_p(t) := {}^B_U R(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma)). \tag{5.2}$$

and let the speed-tracking error be defined as

$$e_\gamma := \dot{\gamma} - v_d(\gamma, t). \tag{5.3}$$

The body-fixed position error dynamics can be given by:

$$\dot{\mathbf{e}}_p(t) = {}^B_U \dot{R}(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma)) + {}^B_U R(\psi)(\dot{\mathbf{p}}(t) - \dot{\mathbf{p}}_d(\gamma)). \tag{5.4}$$

Taking into account that the derivative of a rotation matrix can be expressed as the product of a skew-symmetric matrix with the transposed rotation matrix:

$${}^B_U \dot{R}(\psi) = -S(r) {}^B_U R(\psi), \tag{5.5}$$

where $S(r)$ is defined according to (2.5). Replacing (5.5) in (5.4) yields the position error dynamics expressed in the body-fixed frame become:

$$\dot{\mathbf{e}}_p(t) = -S(r)\underbrace{{}^B_U R(\psi)(\mathbf{p}(t) - \mathbf{p}_d(\gamma))}_{\mathbf{e}_p(t)} + \mathbf{v} + \boldsymbol{v}_c - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma}, \tag{5.6}$$

with

$$\boldsymbol{v}_c := {}^B_U R(\psi)\mathbf{v}_c. \tag{5.7}$$

Since there is no direct control in the sway motion, the goal is to generate surge speed and heading rate control references. Therefore we must make those references appear explicitly in the error expression. By introducing an offset $\boldsymbol{\delta} = [0, \delta]^T \in \mathbb{R}^2$ (with $\delta < 0$) in the standard position error, it is possible to re-write (5.6) as:

$$\begin{aligned}
\dot{\mathbf{e}}_p(t) &= -S(r)(\mathbf{e}_p - \boldsymbol{\delta}) - S(r)\boldsymbol{\delta} + \begin{bmatrix} u \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma} \\
&= -S(r)(\mathbf{e}_p - \boldsymbol{\delta}) - \begin{bmatrix} 0 \\ \delta r \end{bmatrix} + \begin{bmatrix} u \\ v \end{bmatrix} + \boldsymbol{v_c} - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma} \\
&= -S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & -\delta \end{bmatrix}}_{\Delta}\underbrace{\begin{bmatrix} u \\ r \end{bmatrix}}_{\mathbf{u}} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v_c} - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma} \\
&= -S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \Delta\mathbf{u} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\dot{\gamma}.
\end{aligned} \tag{5.8}$$

Proposition 5.1. *Consider the system described by the kinematics in (3.12) with outer-loop control laws given by*

$$\mathbf{u}_d := \Delta^{-1}\left(- K_p\boldsymbol{\sigma}(\mathbf{e}_p - \boldsymbol{\delta}) - \begin{bmatrix} 0 \\ v \end{bmatrix} - \boldsymbol{v}_c + {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}v_d(\gamma, t)\right); \tag{5.9}$$

$$\ddot{\gamma} := -k_\gamma e_\gamma + \dot{v}_d(\gamma, t) + (\mathbf{e}_p - \boldsymbol{\delta})^T {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}, \tag{5.10}$$

where $K_p \succeq 0$, $k_\gamma > 0$ and $\boldsymbol{\sigma}(\mathbf{e}_p)$ is a saturation function defined according to the definition in section 2.1. The proposed control laws solves problem 5.1.

Proof. Consider the candidate Lyapunov Function given by

$$V_1(\mathbf{e}_p) = \frac{1}{2}(\mathbf{e}_p - \boldsymbol{\delta})^T(\mathbf{e}_p - \boldsymbol{\delta}). \tag{5.11}$$

By taking the first derivative of (5.11) and replacing in (5.6) and (5.3):

$$\dot{V}_1(\mathbf{e}_p) = (\mathbf{e}_p - \boldsymbol{\delta})^T\left(- S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \Delta\mathbf{u} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi)\frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma}\big(e_\gamma + v_d(\gamma, t)\big)\right). \tag{5.12}$$

Taking into account the properties of the skew-symmetric matrix introduced in section 2.1, then

$$(\mathbf{e}_p - \boldsymbol{\delta})^T S(r)(\mathbf{e}_p - \boldsymbol{\delta}) = 0. \tag{5.13}$$

Replacing the equality and the control law (5.9) in $\dot{V}_1$ (assuming $\mathbf{u} = \mathbf{u}_d$) yields

$$\begin{aligned}\dot{V}_1(\mathbf{e}_p) &= (\mathbf{e}_p - \boldsymbol{\delta})^T \left(\Delta \mathbf{u}_d + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} \big(e_\gamma + v_d(\gamma, t)\big) \right) \\ &= -\underbrace{(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \sigma(\mathbf{e}_p - \boldsymbol{\delta})}_{W_1(\mathbf{e}_p)} - (\mathbf{e}_p - \boldsymbol{\delta})^T {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} e_\gamma.\end{aligned}$$

By taking a backstepping approach, consider a second candidate Lyapunov function

$$V_2(\mathbf{e}_p, e_\gamma) = V_1(\mathbf{e}_p) + e_\gamma^2. \tag{5.14}$$

Taking the first derivative and replacing in the control law (5.10) for the virtual target:

$$\begin{aligned}\dot{V}_2(\mathbf{e}_p, e_\gamma) &= \dot{V}_1(\mathbf{e}_p) + e_\gamma(\ddot{\gamma} - \dot{v}_d(\gamma, t)) \\ &= -W_1(\mathbf{e}_p) - (\mathbf{e}_p - \boldsymbol{\delta})^T {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} e_\gamma + e_\gamma(\ddot{\gamma} - \dot{v}_d(\gamma, t)) \\ &= -W_1(\mathbf{e}_p) - k_\gamma e_\gamma^2 < 0, \forall (\mathbf{e}_p - \boldsymbol{\delta}), e_\gamma \neq 0.\end{aligned}$$

Given that V_2 is radially unbounded and $\dot{V}_2 < 0$, the controlled system is globally asymptotically stable at the equilibrium point $(\mathbf{e}_p, e_\gamma) = (\boldsymbol{\delta}, 0)$. □

The proposed control laws in proposition 5.1 work under the assumptions that we have access to measurements of the currents velocity $\boldsymbol{v}_c$ and that the requested velocities $\mathbf{u}_d$ are applied to the system and are tracked exactly. In order to lift these assumptions, consider the currents estimation error and inner-loop tracking error given by

$$\begin{aligned}\tilde{\boldsymbol{v}}_c &:= \boldsymbol{v}_c - \hat{\boldsymbol{v}}_{\boldsymbol{c}}, \\ \tilde{\mathbf{u}} &:= \mathbf{u} - \mathbf{u}_d.\end{aligned} \tag{5.15}$$

Proposition 5.2. *Consider the system error dynamics described by equations (5.8) and (5.3), along with virtual target dynamics proposed in (5.10). Furthermore, consider the modified control law given by*

$$\mathbf{u}_d := \Delta^{-1} \left(-K_p \boldsymbol{\sigma}(\mathbf{e}_p - \boldsymbol{\delta}) - \begin{bmatrix} 0 \\ v \end{bmatrix} - \hat{\boldsymbol{v}}_c + {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} v_d(\gamma, t) \right), \tag{5.16}$$

with $K_p \succeq 0$. The closed-loop system is ISS with respect to $\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_{\mathbf{c}}$.

Proof. Consider the first derivative of the candidate Lyapunov function (5.14) and the newly proposed

control law (5.16), then

$$\begin{aligned}
\dot{V}_2 &= (\mathbf{e}_p - \boldsymbol{\delta})^T \left(-S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \Delta\mathbf{u} + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} \big(e_\gamma + v_d(\gamma, t)\big) \right) \\
&= (\mathbf{e}_p - \boldsymbol{\delta})^T \left(-S(r)(\mathbf{e}_p - \boldsymbol{\delta}) + \Delta(\mathbf{u}_d + \tilde{\mathbf{u}}) + \begin{bmatrix} 0 \\ v \end{bmatrix} + \boldsymbol{v}_c - {}^B_U R(\psi) \frac{\partial \mathbf{p}_d(\gamma)}{\partial \gamma} \big(e_\gamma + v_d(\gamma, t)\big) \right) \\
&= -(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \sigma(\mathbf{e}_p - \boldsymbol{\delta}) - k_\gamma e_\gamma + (\mathbf{e}_p - \boldsymbol{\delta})^T (\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c) \\
&\leq -(1-\theta)(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \sigma(\mathbf{e}_p - \boldsymbol{\delta}) - \theta(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \sigma(\mathbf{e}_p - \boldsymbol{\delta}) - k_\gamma |e_\gamma|^2 + \|\mathbf{e}_p - \boldsymbol{\delta}\| \, \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \,,
\end{aligned} \tag{5.17}$$

where $0 < \theta < 1$. The term

$$\begin{aligned}
&- \theta(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \boldsymbol{\sigma}(\mathbf{e}_p - \boldsymbol{\delta}) + \|\mathbf{e}_p - \boldsymbol{\delta}\| \, \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \\
&= -\theta(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \frac{\mathbf{e}_p - \boldsymbol{\delta}}{\|\mathbf{e}_p - \boldsymbol{\delta}\|} \boldsymbol{\sigma}(\|\mathbf{e}_p - \boldsymbol{\delta}\|) + \|\mathbf{e}_p - \boldsymbol{\delta}\| \, \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \,,
\end{aligned} \tag{5.18}$$

will be ≤ 0 if

$$\theta \lambda_{min}(K_p) \boldsymbol{\sigma}(\|\mathbf{e}_p - \boldsymbol{\delta}\|) \geq \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \,, \tag{5.19}$$

which in turn implies that

$$\|\mathbf{e}_p - \boldsymbol{\delta}\| \geq \boldsymbol{\sigma}^{-1} \left(\frac{1}{\theta \lambda_{min}(K_p)} \|\Delta\tilde{\mathbf{u}} + \tilde{\boldsymbol{v}}_c\| \right), \tag{5.20}$$

and

$$\dot{V}_2 \leq -(1-\theta)(\mathbf{e}_p - \boldsymbol{\delta})^T K_p \boldsymbol{\sigma}(\mathbf{e}_p - \boldsymbol{\delta}) - k_\gamma |e_\gamma|^2, \tag{5.21}$$

as the right side of inequality (5.20) can be made arbitrarily small through the choice of the gain matrix K_p. It follows directly from the application of Theorem 2.3 that the controlled system is ISS. □

5.2 UAV Quadrotor Path Following

Problem 5.2. Consider a quadrotor UAV with kinematics described by (4.22), and let $\mathbf{p}_d(\gamma) : [0, \infty) \to \mathbb{R}^3$ denote the desired path parameterised by a continuous variable $\gamma \in \mathbb{R}$ and $v_d(\gamma, t) \in \mathbb{R}$ be a desired speed profile for a virtual target moving along the desired path. Furthermore, consider $\mathbf{p}_d(\gamma)$ to be C^2 and have its first and second derivatives with respect to γ bounded. Moreover, the vehicle is equipped with a control structure that given a desired acceleration $\mathbf{u}_d^\diamond \in \mathbb{R}^3$, assumed to be bounded, computes an attitude and thrust reference to be followed by an inner-loop system. Design a control law for the quadrotor acceleration and virtual target such that:

- the vehicle's position converges to a tube around the desired position that can be made arbitrarily small, i.e. $\|\mathbf{p}(t) - \mathbf{p}_d(\gamma)\|$ converges to a neighborhood of the origin;
- the speed of the virtual target moving along the path converges to the desired speed profile, i.e. $|\dot{\gamma} - v_d(\gamma, t)| \to 0$ as $t \to \infty$.

Following a similar approach to the one proposed by F. Vanni and P. Aguiar, consider the position and velocity errors, but this time defined in the inertial frame $\{U\}$ as

$$\mathbf{e}_p := \mathbf{p}(t) - \mathbf{p}_d(\gamma), \tag{5.22}$$

$$\mathbf{e}_v := \dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t), \tag{5.23}$$

respectively and a virtual target speed tracking error to be defined by (5.3). Consider also new auxiliary error $\mathbf{z}$ defined as

$$\mathbf{z} := \mathbf{e}_v + K_1 \mathbf{e}_p, \tag{5.24}$$

where $K_1 \succeq 0$ is a gains matrix. The position and velocity error dynamics can be given by

$$\dot{\mathbf{e}}_p = \dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{\gamma}, \tag{5.25}$$

$$\dot{\mathbf{e}}_v = \ddot{\mathbf{p}} - \frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t) \right). \tag{5.26}$$

Furthermore, consider the time derivative introduced in (5.26), the desired virtual target speed function (5.1) and virtual target speed tracking error function (5.3), then we can expand the expression as

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t) \right) &= \frac{d}{dt}\left(\frac{\partial \mathbf{p}_d}{\partial \gamma} \right) v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \frac{d}{dt}\left(v_d(\gamma, t) \right) \\
&= \frac{\partial^2 \mathbf{p}_d}{\partial \gamma^2} \dot{\gamma} v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \frac{d}{dt}\left(v_L(\gamma) + v^{coord}(t) \right) \\
&= \frac{\partial^2 \mathbf{p}_d}{\partial \gamma^2} v_d(\gamma, t)(e_\gamma + v_d(\gamma, t)) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\left(\frac{\partial v_L(\gamma)}{\partial \gamma} \dot{\gamma} + \dot{v}^{coord}(t) \right) \\
&= \frac{\partial^2 \mathbf{p}_d}{\partial \gamma^2} v_d(\gamma, t)(e_\gamma + v_d(\gamma, t)) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \frac{\partial v_L(\gamma)}{\partial \gamma}\left(e_\gamma + v_d(\gamma, t) \right) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t) \\
&= \Bigg[\underbrace{\frac{\partial^2 \mathbf{p}_d}{\partial \gamma^2} v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \frac{\partial v_L(\gamma)}{\partial \gamma}}_{\mathbf{h}(\gamma)} \Bigg] (e_\gamma + v_d(\gamma, t)) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t) \\
&= \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t).
\end{aligned} \tag{5.27}$$

Replacing (4.22) and (5.27) in (5.26) yields

$$\dot{\mathbf{e}}_v = \mathbf{u}^\diamond - \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) - \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t). \tag{5.28}$$

Proposition 5.3. *Consider the double integrator system described by (4.22) with outer-loop control laws given by*

$$\mathbf{u}_d^\diamond := \mathbf{h}(\gamma) v_d(\gamma, t) + \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t) - K_1 \mathbf{e}_v - \mathbf{e}_p - K_2 \mathbf{z}; \tag{5.29}$$

$$\ddot{\gamma} := -k_\gamma e_\gamma + \dot{v}_d(\gamma, t) + \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} + \mathbf{z}^T \left(\mathbf{h}(\gamma) + K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma} \right), \tag{5.30}$$

where $K_1, K_2 \succeq 0$ and k_γ a positive gain. The proposed control laws solve the problem 5.2.

Proof. Consider the candidate Lyapunov function given by

$$V_1(\mathbf{e}_p) := \frac{1}{2} \mathbf{e}_p^T \mathbf{e}_p. \tag{5.31}$$

By taking the first derivative of (5.31) and replacing in (5.23) and (5.25), yields

$$\begin{aligned} \dot{V}_1(\mathbf{e}_p) &= \mathbf{e}_p^T \left(\dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{\gamma} \right) \\ &= \mathbf{e}_p^T \left(\underbrace{\dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma)}_{\mathbf{e}_v} - \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma \right) \\ &= -\mathbf{e}_p^T K_1 \mathbf{e}_p + \mathbf{e}_p^T \left(K_1 \mathbf{e}_p + \mathbf{e}_v - \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma \right). \end{aligned} \tag{5.32}$$

Replacing the auxiliary error (5.24) in $\dot{V}_1$ yields

$$\dot{V}_1 = -\underbrace{\mathbf{e}_p^T K_1 \mathbf{e}_p}_{W_1(\mathbf{e}_p)} + e_p^T \left(\mathbf{z} - \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma \right). \tag{5.33}$$

By resorting to a backstepping technique, consider a second candidate Lyapunov function

$$V_2(\mathbf{e}_p, \mathbf{e}_v) := V_1(\mathbf{e}_p) + \frac{1}{2} \mathbf{z}^T \mathbf{z}. \tag{5.34}$$

Replacing (4.22), (5.25) and (5.28) in $\dot{V}_2$ yields:

$$\begin{aligned} \dot{V}_2 = &-W_1(\mathbf{e}_p) + \mathbf{e}_p^T \mathbf{z} - \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma \\ &+ \mathbf{z}^T \left[\mathbf{u}^\diamond - \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) - \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t) + K_1 \left(\underbrace{\dot{\mathbf{p}} - \frac{\partial \mathbf{p}_d}{\partial \gamma} v_d(\gamma, t)}_{:= \mathbf{e}_v} \right) - K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma \right]. \end{aligned} \tag{5.35}$$

Replacing the control input (5.29) in the previous equation (assuming $\mathbf{u}^\diamond = \mathbf{u}_d^\diamond$):

$$\begin{aligned} \dot{V}_2 &= -W_1(\mathbf{e}_p) - \mathbf{z}^T K_2 \mathbf{z} - \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma - \mathbf{z}^T \left(\mathbf{h}(\gamma) + K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma} \right) e_\gamma \\ &= -W_2(\mathbf{e}_p, \mathbf{z}) - \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma - \mathbf{z}^T \left(\mathbf{h}(\gamma) + K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma} \right) e_\gamma. \end{aligned} \tag{5.36}$$

Consider a third candidate Lyapunov obtained by backstepping:

$$V_3 := V_2 + \frac{1}{2} e_\gamma^2. \tag{5.37}$$

Take it's derivative with respect to time:

$$\dot{V}_3 = -W_2(\mathbf{e}_p, \mathbf{e}_v) - \mathbf{e}_p^T \frac{\partial \mathbf{p}_d}{\partial \gamma} e_\gamma - \mathbf{z}^T \left(\mathbf{h}(\gamma) + K_1 \frac{\partial \mathbf{p}_d}{\partial \gamma} \right) e_\gamma + e_\gamma (\ddot{\gamma} - \dot{v}_d(\gamma, t)). \tag{5.38}$$

Replacing the virtual target control law (5.30) in $\dot{V}_3$ yields:

$$\dot{V}_3 = -W_2(\mathbf{e}_p, \mathbf{z}) - k_\gamma e_\gamma^2 < 0, \forall \mathbf{e}_p, \mathbf{z}, e_\gamma \neq 0. \tag{5.39}$$

Given that V_3 is radially unbounded and $\dot{V}_3 < 0$, the controlled system is globally asymptotically stable at the equilibrium point $(\mathbf{e}_p, \mathbf{z}, e_\gamma) = \mathbf{0}$. □

Up to this point, the presence of any external disturbances such as wind and drag forces acting on the multirotor frame have been disregarded. In order to take those into consideration, consider the new vehicle motion model given by

$$\ddot{\mathbf{p}} := \mathbf{u}^\diamond + \mathbf{d}, \tag{5.40}$$

where $\boldsymbol{d} \in \mathbb{R}^3$ represents unmeasured external constant disturbances acting on the vehicle, such that

$$\|\mathbf{d}\| \leq d_{max}, \tag{5.41}$$

where d_{max} is a known, positive constant. Let the disturbance estimation error be given by

$$\tilde{\mathbf{d}} := \mathbf{d} - \hat{\mathbf{d}}. \tag{5.42}$$

Given the new vehicle motion model given in (5.40), the dynamics of $\dot{\mathbf{e}}_v$ obtained previously in (5.28) are now given by

$$\dot{\mathbf{e}}_v = \mathbf{u}^\diamond + \mathbf{d} - \mathbf{h}(\gamma)(e_\gamma + v_d(\gamma, t)) - \frac{\partial \mathbf{p}_d}{\partial \gamma} \dot{v}^{coord}(t). \tag{5.43}$$

Unlike with the ASVs, where current estimates were given by a simple, yet quite effective complementary filter, in the case of a quadrotor a different direction is taken. The main reason lies on the fact that in this model the disturbances are included in the accelerations (and do not comprise only wind). In Cabecinhas et al. [34] the authors propose the use of a simple observer based on a smooth projection operator for disturbance rejection on a quadrotor. Consider the following disturbance observer:

$$\dot{\hat{\mathbf{d}}} := K_d \text{Proj}(\mathbf{z}, \hat{\mathbf{d}}) \text{ with } K_d \succeq 0, \tag{5.44}$$

where Proj$(\cdot)$ denotes a smooth projection operator introduced in section 2.2.3 and K_d denotes a diagonal gains matrix. The estimation error dynamics are given by

$$\dot{\tilde{\mathbf{d}}} = -\dot{\hat{\mathbf{d}}}. \tag{5.45}$$

Moreover, consider the inner-loop tracking error given by:

$$\tilde{\mathbf{u}}^{\diamond} := \mathbf{u}^{\diamond} - \mathbf{u}_d^{\diamond}, \tag{5.46}$$

which is assumed to be bounded.

Proposition 5.4. *Consider the system described by (5.40), the disturbance estimator dynamics given by (5.44) and the inner-loop tracking error given by (5.46). Furthermore, consider the modified control law given by*

$$\mathbf{u}_d^{\diamond} := -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - K_1\mathbf{e}_v - \mathbf{e}_p - K_2\mathbf{z}. \tag{5.47}$$

For sufficiently small initial position and velocity errors ($\mathbf{e}_p$, $\mathbf{e}_v$), and a sufficiently large separation between the time-scales of the inner and outer loop systems, it can be guaranteed that the system error converges to a neighbourhood of zero.

Proof. Consider the candidate Lyapunov function

$$V_4 = V_3 + \frac{1}{2}\tilde{\mathbf{d}}^T K_d^{-1}\tilde{\mathbf{d}}. \tag{5.48}$$

It's time derivative is now given by

$$\dot{V}_4 = -\mathbf{e}_p^T K_1 \mathbf{e}_p - \mathbf{z}^T K_2 \mathbf{z} - k_\gamma e_\gamma^2 + \underbrace{\tilde{\mathbf{d}}^T(\mathbf{z} - \mathsf{Proj}(\mathbf{z},\hat{\mathbf{d}}))}_{\leq 0} + \mathbf{z}^T\tilde{\mathbf{u}}^{\diamond}. \tag{5.49}$$

Making use of property 2 of the smooth projection operator (see section 2.2.3), it is possible to derive a bound for the derivative

$$\dot{V}_4 \leq -W(\mathbf{e}_p, \mathbf{e}_v, e_\gamma) + \mathbf{z}^T\tilde{\mathbf{u}}^{\diamond}. \tag{5.50}$$

By applying (4.22) to (5.46), yields

$$\begin{aligned}\tilde{\mathbf{u}}^{\diamond} &= \mathbf{u}^{\diamond} - \mathbf{u}_d^{\diamond} \\ &= \left(-\frac{Z}{m}\mathbf{r}_3 + g\mathbf{e}_3\right) - \left(-\frac{Z}{m}\mathbf{r}_{3d} + g\mathbf{e}_3\right) = \frac{Z}{m}\underbrace{(\mathbf{r}_{3d} - \mathbf{r}_3)}_{\tilde{\mathbf{r}}_3}.\end{aligned} \tag{5.51}$$

From the definition of Z introduced in section 4.2.1 and (5.47), the total thrust can be bounded by

$$\begin{aligned}Z = \|\mathbf{u}^*\| &= \left\|-mR_z^T(\psi)(\mathbf{u}^{\diamond} - g\mathbf{e}_3)\right\| \\ &= \left\|-mR_z^T(\psi)(-\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial\gamma}\dot{v}^{coord}(t) - K_1\mathbf{e}_v - \mathbf{e}_p - K_2\mathbf{z} - g\mathbf{e}_3)\right\| \\ &\leq m\left(\left\|\hat{\mathbf{d}}\right\| + \|\mathbf{h}(\gamma)v_d(\gamma,t)\| + \left\|\frac{\partial \mathbf{p}_d}{\partial\gamma}\dot{v}^{coord}(t)\right\| + \|K_1\|\|\mathbf{e}_v\| + \|\mathbf{e}_p\| + \|K_2\|\|\mathbf{z}\| + g\right).\end{aligned} \tag{5.52}$$

Therefore, it becomes trivial that the inner-loop tracking error is bounded by

$$\|\tilde{\mathbf{u}}^{\diamond}\| \leq \|\tilde{\mathbf{r}}_3\|\left(\left\|\hat{\mathbf{d}}\right\| + \|\mathbf{h}(\gamma)v_d(\gamma,t)\| + \left\|\frac{\partial \mathbf{p}_d}{\partial\gamma}\dot{v}^{coord}(t)\right\| + \|K_1\|\|\mathbf{e}_v\| + \|\mathbf{e}_p\| + \|K_2\|\|\mathbf{z}\| + g\right). \tag{5.53}$$

Replacing the inequality (5.53) in (5.50) makes the quadratic term $\|\mathbf{z}\|^2$ appear. Therefore, it is only possible to conclude that as long as the position and velocity errors are small, and the inner-loop of the system is much faster than the outer-loop (guaranteeing that $\|\tilde{\mathbf{r}}_3\|$ is small and as consequence also $\tilde{\mathbf{u}}$), then the system is able to converge to a neighborhood of the desired position and velocity references.

□

In order to make the designed control law $\mathbf{u}_d^{\diamond}$ more clear, consider the following algebraic manipulation:

$$\begin{aligned}
\mathbf{u}_d^{\diamond} &= -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - K_1\mathbf{e}_v - \mathbf{e}_p - K_2\mathbf{z} \\
&= -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - K_1\mathbf{e}_v - \mathbf{e}_p - K_2\mathbf{e}_v + K_1K_2\mathbf{e}_p \\
&= -\hat{\mathbf{d}} + \mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t) - \mathbf{e}_v\underbrace{(K_1+K_2)}_{K_v} - \mathbf{e}_p\underbrace{(I+K_1K_2)}_{K_p} \\
&= -\hat{\mathbf{d}} + \underbrace{\mathbf{h}(\gamma)v_d(\gamma,t) + \frac{\partial \mathbf{p}_d}{\partial \gamma}\dot{v}^{coord}(t)}_{\text{acceleration term}} - \underbrace{\mathbf{e}_v K_v}_{\text{derivative term}} - \underbrace{\mathbf{e}_p K_p}_{\text{proportional term}}.
\end{aligned} \tag{5.54}$$

From here, it is possible to conclude that the control law is essentially a Proportional Integral Derivative (PID) controller with a feed-forward term for acceleration. The integral effect results from the disturbance estimation term being interpreted as a bounded integral term.

Chapter 6

Cooperative Path Following

In the previous chapter, a set of control algorithms that allow ASV and quadrotor UAV vehicles to perform PF missions individually were introduced. As discussed previously, one key advantage of the PF control laws introduced is that they allow for the design of a virtual target speed synchronization law that is independent of the PF controller itself.

In order to perform the synchronization between virtual targets of multiple vehicles, both centralized or distributed approaches could be taken. A centralized controller would require that all vehicles were able to communicate directly with a central entity (which might not always be feasible) and would imply that this central entity could never lose its connection to the network. A better and more robust approach is to consider a distributed control problem for the synchronization of the multiple autonomous vehicles/agents, which is the one considered in this work. Particularly, in this chapter we address the problem of CPF. The end goal is to have an algorithm that allows one quadrotor and multiple ASV vehicles to perform a path following mission cooperatively, in a leader-follower network like structure. The vehicles are required to execute their mission according to a fixed geometric configuration. To this effect, it is considered that there exists inter-vehicle communication and that motion related information is accessible between the vehicles in a communication network. In the first section we start by formulating the CPF problem as a consensus/synchronization problem and then propose a solution to it in a two step fashion: i) design of a coordination control scheme that assumes a continuous exchange of information between vehicles; ii) design an ETC mechanism that copes with limitations imposed by real environments where inter-vehicle communications are discrete.

6.1 Synchronization Problem with Continuous Communications

Consider a group of $N \in \mathbb{R}^+ \setminus \{1\}$ autonomous vehicles/agents in a network that can be described mathematically by a digraph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})$, consisting on N vertices, a set of directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where the edge ε_{ij} represents the flow of information from agent i to agent j, and a weighted adjacency matrix $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$. Furthermore, each vehicle i is able to receive information from its neighbours in $\mathcal{N}_i^{in}$ and send information to its neighbours in $\mathcal{N}_i^{out}$. Let the state vector of the system be composed

by the path parameter of each individual vehicle $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_N]^T$. The CPF problem is formulated in problem 6.1.

Problem 6.1. For each agent i, with $i = 1, ..., N$ derive a consensus protocol for the speed correction term $\mathbf{v^{coord}} = [v_1^{coord}, ..., v_N^{coord}]^T$ such that $\lim_{t\to\infty} |\gamma_i - \gamma_j| = 0$, $\forall j \in N_i^{in}$, and the formation of vehicles achieves the desired speed assignment $\mathbf{v_L}(\boldsymbol{\gamma}) = [v_{L1}, ..., v_{LN}]^T$ as $t \to \infty$.

In order to solve problem 6.1 the following simplifying assumptions are taken:

Assumption 6.1. The communications are continuous between vehicles.

Assumption 6.2. The communication topology of the vehicles is fixed, i.e. the Laplacian matrix L associated to the graph $\mathcal{G}$ is constant. $\mathcal{G}$ is also undirected, i.e. $N_i^{in} = N_i^{out}$, and connected.

Let the synchronization error vector be defined as $\boldsymbol{\varepsilon} = [\varepsilon_1, ..., \varepsilon_N]^T$, such that

$$\varepsilon_i := \sum_{j \in \mathcal{N}_i^{in}} a_{ij}(\gamma_i - \gamma_j), \tag{6.1}$$

which can also be expressed in it's vectorial form as

$$\boldsymbol{\varepsilon} := L\boldsymbol{\gamma}, \tag{6.2}$$

where L is the Laplacian matrix of graph $\mathcal{G}$ and ε_i denotes the coordination error between vehicle i and its neighbours. Consider as well, the coordination error dynamics of the multi vehicle system to be given by

$$\dot{\boldsymbol{\varepsilon}} := L\dot{\boldsymbol{\gamma}}. \tag{6.3}$$

Note that, according to the previously developed PF controllers, for each vehicle i, $|\dot{\gamma}_i - v_d(\gamma, t)| = 0$ is only guaranteed as $t \to \infty$. Having this fact in mind, an extra naive assumption is made:

Assumption 6.3. The speed progression of a virtual targets along the desired path is always assumed to be modelled by a single integrator system, which can be expressed in vectorial form according to

$$\dot{\boldsymbol{\gamma}} = \mathbf{v_d}(\boldsymbol{\gamma}, t) = \mathbf{v_L}(\boldsymbol{\gamma}) + \mathbf{v^{coord}}. \tag{6.4}$$

According to Ren. W and Attkins E. [18], a first-order consensus protocol that could stabilize the system would be given by

$$\mathbf{v_d}(\boldsymbol{\gamma}, t) := -\boldsymbol{\varepsilon} = -L\boldsymbol{\gamma}, \tag{6.5}$$

where the final consensus value was given by $\varepsilon^* = \sum_{i=1}^{N} \alpha_i \varepsilon_i(0)$, with $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_N]^T$ a non-negative (left) eigenvector of $-L$ associated with the eigenvalue 0. Unfortunately, this control law cannot be applied directly as we do not control $\mathbf{v_d}$ but rather $\mathbf{v^{coord}}$. However, both vectors are related by (6.4).

Proposition 6.1. *Consider the distributed control protocol applied to each vehicle* i

$$v_i^{coord} := -k_\varepsilon \sum_{j \in \mathcal{N}_i^{in}} a_{ij}(\gamma_i - \gamma_j), \tag{6.6}$$

also expressed in vectorial form as

$$\mathbf{v}^{\mathbf{coord}} := -k_\varepsilon L\boldsymbol{\gamma}, \text{ with } k_\varepsilon > 0. \tag{6.7}$$

The system described by (6.3), (6.4) with distributed control law given by (6.7) is globally asymptotically stable, and $\boldsymbol{\varepsilon} \to \mathbf{0}$ and $\boldsymbol{\gamma} \to \mathbf{v_L}(\boldsymbol{\gamma})$ with $t \to \infty$ as long as $\mathbf{v_L}(\boldsymbol{\gamma}) = v_L\mathbf{1}$, i.e. all the virtual targets have the same desired speed profile.

Proof. Consider the following equality:

$$L\underbrace{\left[\mathbf{v}_1 \;\middle|\; \mathbf{v}_2 \;\middle|\; \dots \;\middle|\; \mathbf{v}_N\right]}_{V} = \underbrace{\left[\mathbf{v}_1 \;\middle|\; \mathbf{v}_2 \;\middle|\; \dots \;\middle|\; \mathbf{v}_N\right]}_{V}\underbrace{\begin{bmatrix}\lambda_1 & \dots & 0\\ \vdots & \ddots & \vdots\\ 0 & \dots & \lambda_N\end{bmatrix}}_{\Lambda}, \tag{6.8}$$

where $\mathbf{v}_i, i = 1, ..., N$ denotes the eigen vectors of L while $\lambda_i, i = 1, ..., N$ denotes the associated eigen values. By applying theorem 2.4 to (6.8), the Laplacian matrix expressed in Jordan form, according to

$$L = V\Lambda V^{-1} \Leftrightarrow L = \left[\mathbf{1_N} \;\middle|\; V_2\right]\left[\begin{array}{c|c} 0 & 0\\ \hline 0 & \Lambda_2\end{array}\right]\left[\begin{array}{c} \mathbf{1_N}^T\\ \hline V_2^T\end{array}\right]. \tag{6.9}$$

By replacing in (6.4) and (6.7) in (6.3) yields

$$\begin{aligned}\dot{\boldsymbol{\varepsilon}} &= L(\mathbf{v_L}(\boldsymbol{\gamma}) - k_\varepsilon L\boldsymbol{\gamma})\\ &= v_L\cancelto{0}{L\mathbf{1}} - k_\varepsilon L\boldsymbol{\varepsilon} = -k_\varepsilon L\boldsymbol{\varepsilon}.\end{aligned} \tag{6.10}$$

Consider the change of variables

$$\bar{\boldsymbol{\varepsilon}} = V^{-1}\boldsymbol{\varepsilon}. \tag{6.11}$$

Replacing (6.9) and (6.11) in (6.10) yields:

$$V^{-1}\dot{\boldsymbol{\varepsilon}} = -k_\varepsilon V^{-1}(V\Lambda V^{-1})\boldsymbol{\varepsilon} \Rightarrow \dot{\bar{\boldsymbol{\varepsilon}}} = -k_\varepsilon\Lambda\bar{\boldsymbol{\varepsilon}}. \tag{6.12}$$

With this new error dynamics, it becomes trivial that:

$$\bar{\boldsymbol{\varepsilon}}(t) = \left[\begin{array}{c|c} 1 & 0\\ \hline 0 & e^{-k_\varepsilon\Lambda_2 t}\end{array}\right]\bar{\boldsymbol{\varepsilon}}(0) \Rightarrow \boldsymbol{\varepsilon}(t) = V\left[\begin{array}{c|c} 1 & 0\\ \hline 0 & e^{-k_\varepsilon\Lambda_2 t}\end{array}\right]V^{-1}\boldsymbol{\varepsilon}(0) \tag{6.13}$$

From this equality, it is possible to conclude that

$$\lim_{t\to\infty}\boldsymbol{\varepsilon} = \mathbf{1}\cancelto{0}{\mathbf{1}^T L\boldsymbol{\gamma}}(0) = 0, \tag{6.14}$$

and as a consequence $\dot{\boldsymbol{\gamma}} \to \mathbf{v_L}(\boldsymbol{\gamma})$ as $t \to \infty$. □

6.2 Synchronization Problem with Discrete Communications

In the previous section a distributed control law that relied on the continuous flow of information among neighbour agents was proposed. However, in a physical world, there are bandwidth limitations that each vehicle must cope with. In this section a distributed control scheme with ETC is presented, based on previous work developed by A. Aguiar and A. Pascoal [58] and N. Hung and F. Rego [59]. In their work, the authors propose a scheme where each agent i has a set of estimators $\hat{\gamma}_j, j \in \mathcal{N}_i^{in}$ for the true state of each in-neighbour virtual target γ_j. In addition, each agent i has an estimator for its own state $\hat{\gamma}_i$ which is reset whenever vehicle i broadcasts its true state γ_i. The other estimators are reset whenever agent i receives the true state from its in-neighbours $j \in \mathcal{N}_i^{in}$. In their research paper, Hung et al. [59] propose two different broadcast/triggering conditions: i) a time-dependent triggering condition; ii) a state dependent triggering condition. In this work we will focus on the first proposal.

Proposition 6.2. *Consider the re-written distributed control law in (6.6) now given by*

$$v_i^{coord} := -k_\varepsilon \sum_{j \in \mathcal{N}_i^{in}} a_{ij}(\gamma_i - \hat{\gamma}_j), \tag{6.15}$$

where k_ε is still a sufficiently large positive constant and $\hat{\gamma}_j$ is vehicle's i estimate of vehicle's j real state/path parameter. Consider also that the bank of estimators that each vehicle i is running follows the dynamics equation

$$\dot{\hat{\gamma}}_i := v_L(\hat{\gamma}_i). \tag{6.16}$$

Based on the broadcast/triggering condition defined previously, at any time instant t, under negligible transmission delays, the vehicle's j self-state estimate $\hat{\gamma}_j$ is equal to vehicle's i estimate of $\hat{\gamma}_j$, which allows us to express the new distributed control law and estimator dynamics using vectorial notation as

$$\dot{\hat{\boldsymbol{\gamma}}} := \mathbf{v_L}(\hat{\boldsymbol{\gamma}}), \tag{6.17}$$

where, $\hat{\boldsymbol{\gamma}} = [\hat{\gamma}_1, ..., \hat{\gamma}_N]^T$ is the self-estimate of the state of each vehicle. Let $\tilde{\boldsymbol{\gamma}} = [\tilde{\gamma}_1, ..., \tilde{\gamma}_N]^T$ denote the local estimation errors of each vehicle, such that $\tilde{\boldsymbol{\gamma}} = \boldsymbol{\gamma} - \hat{\boldsymbol{\gamma}}$. Then $\mathbf{v^{coord}}$ is given by

$$\begin{aligned} \mathbf{v^{coord}} &:= -k_\varepsilon[D\boldsymbol{\gamma} - \mathcal{A}\hat{\boldsymbol{\gamma}}] \\ &= -k_\varepsilon[D\boldsymbol{\gamma} - \mathcal{A}(\boldsymbol{\gamma} - \tilde{\boldsymbol{\gamma}})] \\ &= -k_\varepsilon(\boldsymbol{\varepsilon} + \mathcal{A}\tilde{\boldsymbol{\gamma}}). \end{aligned} \tag{6.18}$$

where D is a diagonal matrix and A the graph adjacency matrix, introduced in section 2.3. Consider as well, a triggering function used to define when to broadcast the state of each vehicle, defined as

$$\begin{cases} \delta_i(t) := |\tilde{\gamma}_i(t)| - g_i(t) \\ \tilde{\gamma}_i(t) = \hat{\gamma}_i(t) - \gamma_i(t) \end{cases}, \tag{6.19}$$

where $\tilde{\gamma}_i(t)$ is the local estimation error of agent i and $g_i(t)$ is a threshold function that is time dependent, such that if the estimation error exceeds this threshold, i.e. $\delta_i(t) \geq 0$, vehicle i broadcasts its state to the out-neighbours $\mathcal{N}_i^{out}$ and resets its local estimator. Furthermore, consider $g_i(t)$ to belong to a class of

non-negative functions, given by

$$g_i(t) = c_i + b_i e^{-\alpha_i t}, \tag{6.20}$$

with c_i, b_i and α_i positive constants and $\mathbf{g}(t) = [g_1, ..., g_N]^T$ the collection of functions for each individual vehicle. Consider also that now $\mathbf{v_L}(\boldsymbol{\gamma}) = v_L\mathbf{1} + \tilde{\mathbf{v}}_\mathbf{L}$, where $\tilde{\mathbf{v}}_\mathbf{L}$ is a bounded and arbitrarily small term that accounts for a transient period in which the vehicles are on different sections of the path, with slightly different desired speed profiles. Then, the system is ISS with respect to the error vector $\boldsymbol{\varepsilon}$ and the inputs $\tilde{\boldsymbol{\gamma}}$ and $\tilde{\mathbf{v}}_\mathbf{L}$, under the assumptions 6.2 and 6.3.

Proof. Replacing (6.18) in (6.3) yields

$$\begin{aligned}
\dot{\boldsymbol{\varepsilon}} &= L(\mathbf{v_L}(\boldsymbol{\gamma}) - k_\varepsilon(\boldsymbol{\varepsilon} + \mathcal{A}\tilde{\boldsymbol{\gamma}})) \\
&= v_L \cancelto{0}{L\mathbf{1}} + L\tilde{\mathbf{v}}_\mathbf{L} - k_\varepsilon L(\boldsymbol{\varepsilon} + \mathcal{A}\tilde{\boldsymbol{\gamma}}) \\
&= -k_\varepsilon L(\boldsymbol{\varepsilon} + \mathbf{d}) \text{ with } \mathbf{d} = \frac{\tilde{\mathbf{v}}_\mathbf{L}}{k_\varepsilon} + \mathcal{A}\tilde{\boldsymbol{\gamma}},
\end{aligned} \tag{6.21}$$

where $\mathbf{d}$ is a disturbance that results from combining the terms dependent on $\tilde{\mathbf{v}}_\mathbf{L}$ and $\tilde{\boldsymbol{\gamma}}$. Considering the change of variables introduced in (6.11) and applying it to (6.21) yields

$$\dot{\bar{\boldsymbol{\varepsilon}}} = -k_\varepsilon \Lambda(\bar{\boldsymbol{\varepsilon}} + \bar{\mathbf{d}}), \text{ with } \bar{\mathbf{d}} = V^{-1}\mathbf{d}. \tag{6.22}$$

It is possible to decompose the above equality according to the following notation

$$\left[\begin{array}{c} \dot{\bar{\varepsilon}}_1 \\ \hline \dot{\bar{\boldsymbol{\varepsilon}}}_2 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline -k_\varepsilon \Lambda_2(\bar{\boldsymbol{\varepsilon}}_2 + \bar{\mathbf{d}}_2) \end{array}\right], \tag{6.23}$$

where the first half of the vector denotes the term that depends on the null eigen value of the Laplacian while the second term is a vector that depends only on the positive eigen values of the Laplacian. Consider now the candidate Lyapunov function

$$V_{\bar{\varepsilon}_2} = \frac{1}{2}\bar{\boldsymbol{\varepsilon}}_2^T \bar{\boldsymbol{\varepsilon}}_2, \tag{6.24}$$

and it's time derivative given by

$$\begin{aligned}
\dot{V}_{\bar{\varepsilon}_2} &= -k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2(\bar{\boldsymbol{\varepsilon}}_2 + \bar{\mathbf{d}}_2) \\
&= -(1-\theta)k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2 \bar{\boldsymbol{\varepsilon}}_2 - \theta k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2 \bar{\boldsymbol{\varepsilon}}_2 - k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2 \bar{\mathbf{d}}_2
\end{aligned} \tag{6.25}$$

where $0 < \theta < 1$. The term

$$-\theta k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2 \bar{\boldsymbol{\varepsilon}}_2 - k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2 \bar{\mathbf{d}}_2 \tag{6.26}$$

will be ≤ 0 if

$$\|\bar{\boldsymbol{\varepsilon}}_2\| \geq \frac{1}{\theta} \left\|\bar{\mathbf{d}}_\mathbf{2}\right\|, \tag{6.27}$$

and

$$\dot{V}_{\bar{\varepsilon}_2} \leq -(1-\theta)k_\varepsilon \bar{\boldsymbol{\varepsilon}}_2^T \Lambda_2 \bar{\boldsymbol{\varepsilon}}_2. \tag{6.28}$$

The term $\|\tilde{\gamma}\|$ can be made arbitrarily small by controlling the gains that dictate the broadcasting scheme. Moreover, the term $\tilde{\mathbf{v}}_{\mathbf{L}}$ can be dominated by a proper choice k_ε. Hence $\|\mathbf{d}\|$ can be made arbitrarily small and so does $\left\|\bar{\mathbf{d}}_2\right\|$. It follows from the application of theorem 2.3 that the controlled system is ISS with respect to the error vector ε and the inputs $\tilde{\gamma}$ and $\tilde{\mathbf{v}}_{\mathbf{L}}$. □

The proposed consensus protocol used for achieving CPF using ETC is summarized in algorithm 2.

Algorithm 2 Event Triggered Communication for vehicle i

1: At every time instant t, each vehicle i follows the procedure:
2: **procedure** COORDINATION AND COMMUNICATION
3: **if** $\delta_i(t) \geq 0$ where δ_i is computed using (6.19) and (6.20) **then**
4: Broadcast $\gamma_i(t)$;
5: Reset the estimator $\hat{\boldsymbol{\gamma}}_{\boldsymbol{i}}$;
6: **if** Receive a new message from agent j **then**
7: Reset $\hat{\gamma}_j(t)$;
8: Run the estimators according to (6.16);
9: Update the first order control protocol $\boldsymbol{u}_i$ using (6.15);

6.3 Final Architecture

Taking into consideration the control structures introduced in chapters 4, 5 and 6 for both ASVs and a quadrotor UAV, it is possible to summarize the entire control system in Figure 6.1.

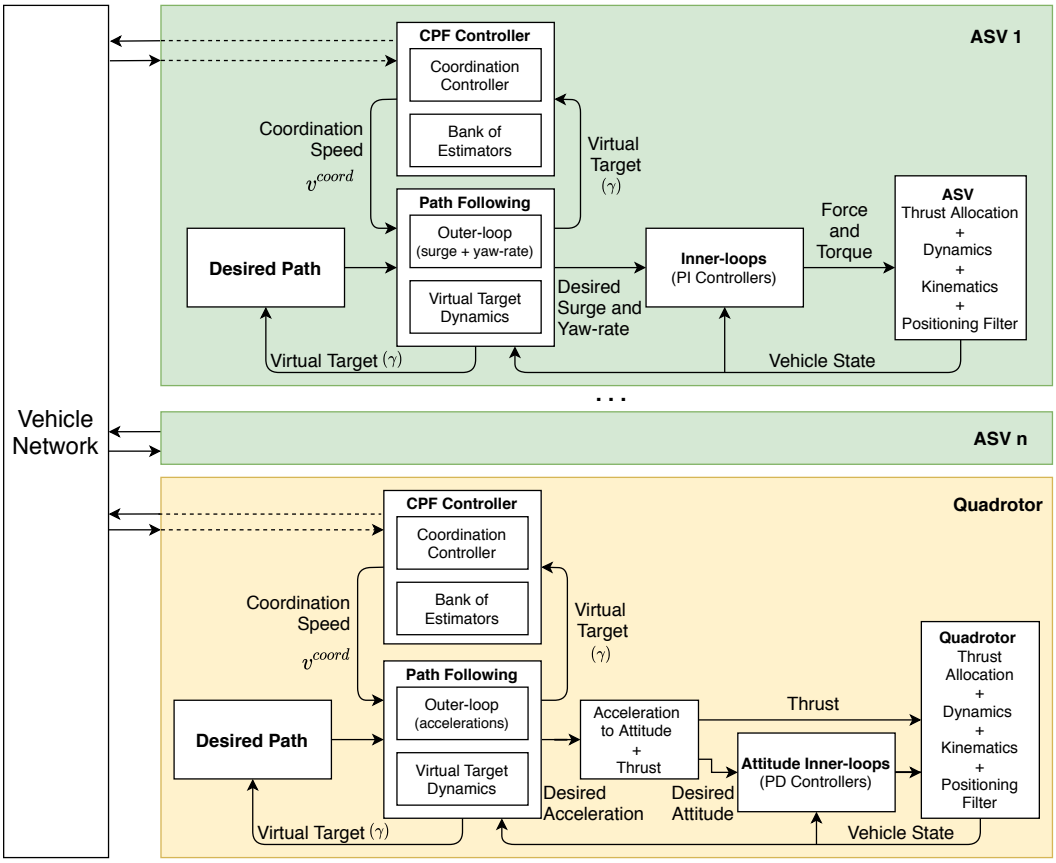


Figure 6.1: Complete system architecture - UAV quadrotor and multiple ASVs

Chapter 7

Path Planning

The previous chapter focused on the development of a consensus protocol that allowed a network of vehicles to perform CPF missions using a distributed ETC scheme. In this chapter the attention is shifted towards the problem of generating a set of smooth and planar reference paths for each individual vehicle to follow. In order to generate such paths, problem 7.1 is formulated.

Problem 7.1. Consider an UAV (quadrotor) flying over a body of water at a pre-defined fixed altitude, equipped with a camera sensor pointing downwards with a fixed pitch angle relative to the vehicle's body reference frame $\{B\}$. Consider also that the vehicle is capable of detecting environmental boundaries in the 2-D image provided by the camera sensor. Furthermore, consider that at the surface of the water, one or more ASV vehicles are required to follow the quadrotor according to a pre-defined vehicle formation.

As the quadrotor detects an environmental boundary in the 2-D image, generate a set of smooth and planar reference paths for each individual vehicle (quadrotor and ASVs), such that they encircle the boundary according to the pre-defined formation.

In order to solve this problem a few assumptions are made:

Assumption 7.1. The environmental boundary is located at the ocean's surface assumed to be a 2-D plane at $Z = 0$ in the inertial frame of reference $\{U\}$.

Assumption 7.2. The quadrotor has a navigation system that can track the vehicle's pose with a "good enough" accuracy.

Assumption 7.3. The quadrotor has a limited vision of the environment, i.e, the camera sensor might not be able to capture the entire boundary, but rather sections of it, according to Figure 7.1 a).

Assumption 7.4. The detection of the pixels that encode the boundary in the image frame is a sub-system that we assume to be already developed and readily available.

Furthermore, it is important to develop a rigorous mathematical definition for what type of environmental boundary we are considering in this work. As discussed previously in section 1.2.7, there are several possibilities for modelling such boundaries, which can be more or less useful considering the set of sensors the vehicles have onboard. In this work, the definition/model of a dynamic boundary is

borrowed from the work of Saldaña et al. [28], which in contrast to convection-diffusion based models, does not require in-depth knowledge of the dynamics of the fluid. Consider the following definitions and assumptions:

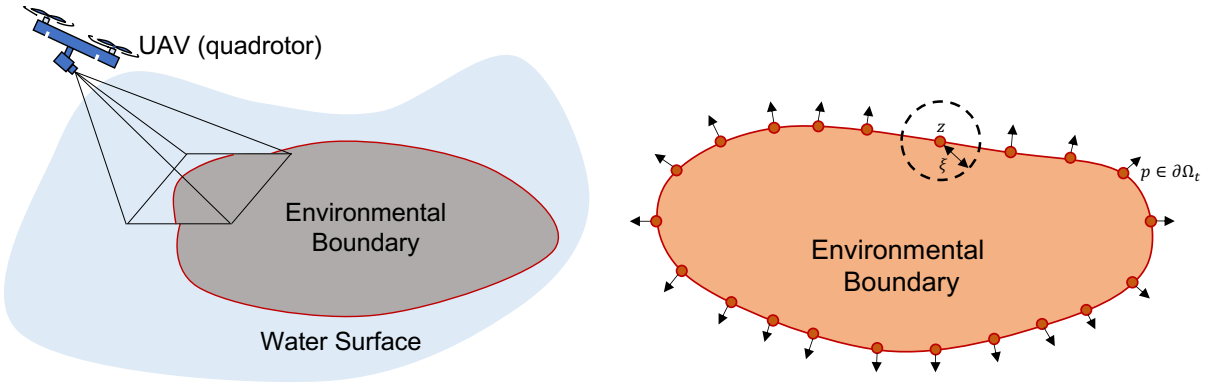
Definition 7.1. A dynamic boundary is a set of planar points Ω_t such that $\forall z \in \Omega_t$, and for any $\xi > 0$, the open disk centered at point z with radius ξ contains points of Ω_t and its complement set Ω_t^C.

Definition 7.2. A dynamic boundary can be approximated by a parametric closed curve (Jordan Curve) $\mathbf{C}(\gamma, t) : [0, \infty) \times [0, \infty) \to \mathbb{R}^2$, mapped by a parameter $\gamma \in \mathbb{R}_0^+$ and time $t \in \mathbb{R}_0^+$. The curve is continuous with no self-intersecting points.

Assumption 7.5. A dynamic boundary, modelled by the parametric curve $\mathbf{C}(\gamma, t)$ is assumed to change smoothly with respect to both time t and parameter γ, such that first and second derivatives exist and are continuous.

Assumption 7.6. The speed of any point $p \in \Omega_t$ is upper-bounded by a maximum speed value, as shown in Figure 7.1 b), such that:

$$\left\| \frac{\partial \mathbf{C}(\gamma, t)}{\partial t} \right\| \leq v_{max}. \tag{7.1}$$



(a) Quadrotor vision

(b) Environmental boundary definition

Figure 7.1: Environmental Boundary schematic

Given the previous definitions and assumptions 7.1 and 7.2, one can consider acceptable the conversion of the pixels that represent the boundary in the image frame to a 2-D point cloud expressed in the inertial frame of reference $\{U\}$. Moreover, we can expect that points representing regions of the boundary that are closer to the quadrotor will have a higher accuracy than points that represent sections that are further away.

Taking into consideration assumption 7.3 it is trivial to conclude that using a static planning algorithm for a vehicle to follow closely the chemical spill will not suffice as it would require the quadrotor to either: i) have a complete overview of the boundary; ii) construct a mosaic of the environment from the 2-D image stream. The first is infeasible if the anomaly at the surface spans across several kilometres, which would require the quadrotor to fly very high to get a complete picture of the scene and, as a consequence, small

errors in the vehicle's pose would generate huge errors in the conversion to a 2-D point cloud, rendering assumption 7.2 irrealistic. The second is also infeasible, as the environment is assumed to be dynamic, therefore even if the UAV was capable of taking several shots of the boundary, stitch them together and plan a path in a small amount of time, according to assumptions 7.5 and 7.6, the real boundary would keep morphing and the planned path would not be adequate after a small period of time.

A possible solution to problem 7.1 is to develop an online path planning mechanism which is actively re-planning the path at a given frequency f, as the vehicles move along it and more up-to-date data is acquired by the vision system. In this chapter we propose an online path planning algorithm that given a stream of pixels corresponding to a boundary to be followed:

1. uses the data provided by the quadrotor navigation system to convert the pixels to a 2-D point cloud expressed in the inertial frame;
2. removes outliers and does some pre-processing on the 2-D point cloud;
3. fits the data with open B-splines by formulating an online optimization problem;
4. generates a path for each vehicle to follow based on the generated B-spline;
5. repeats the process once new data is available.

In section 7.1 we introduce the camera model adopted and the conversion of pixels in the image frame to coordinates in a 2-D plane. In section 7.2 we describe in detail the multi-stage optimization algorithm developed to fit the set of points with a curve that can be used by the PF systems. In the last section we provide a way to generate individual desired paths for each vehicle, according to a pre-defined vehicle formation by introducing a new frame transformation.

7.1 Camera Model

In order to convert pixels in an image frame to a 2-D point cloud, we must first introduce the camera model adopted. For this purpose, we consider a camera to be characterized by: i) a set of extrinsic parameters, which model the conversion between coordinates expressed in the world/inertial reference frame $\{U\}$ and the camera reference frame $\{C\}$; ii) intrinsic parameters which describe how a set of points in $\{C\}$ are represented in the image frame, according to Figure 7.2.

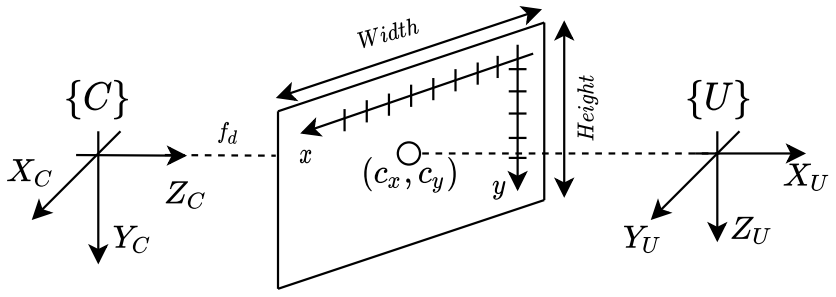


Figure 7.2: Camera model and reference frames

The intrinsic parameters consist of the focal distance f_d, the scale factors (s_x, s_y) in the X and Y-axis respectively, and (c_x, c_y) which correspond to the offset of the focal point in the image plane. These

parameters are fixed and can be obtained a priori by resorting to a camera calibration process, described in detail in [60]. The extrinsic parameters can be organized in a matrix that encodes both rotation and translation between reference frames. Combining together the matrices of intrinsic parameters K, also known as the full-rank calibration matrix, and the matrix of external parameters ${}^{C}_{U}[R|T]$ and expressing the inertial frame coordinates as homogeneous coordinates, we get the linear system described by

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_d s_x & 0 & c_x \\ 0 & f_d s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} {}^{C}_{U}R & {}^{C}_{U}\boldsymbol{T} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}}_{{}^{C}_{U}[R|T]} \begin{bmatrix} X_U \\ Y_U \\ Z_U \\ 1 \end{bmatrix}, \tag{7.2}$$

where x and y denote the coordinates in the image frame and λ is a scale factor. It is important to mention that ${}^{C}_{U}[R|T]$ results from a series of successive rigid-body transformations (rotations and translations), given by

$${}^{C}_{U}[R|T] = {}^{C}_{B}[R|T]{}^{B}_{U}[R|T], \tag{7.3}$$

where ${}^{B}_{U}[R|T]$ denotes the conversion of coordinates expressed in the inertial frame $\{U\}$ to the quadrotor's body frame $\{B\}$, provided by the navigation system of the quadrotor and ${}^{C}_{B}[R|T]$ is a matrix known a priori, assuming the camera attached to the vehicle is fixed. Furthermore, we can aggregate the intrinsic and extrinsic parameters in a matrix Ω according to

$$\Omega = K \cdot {}^{C}_{U}[R|T]. \tag{7.4}$$

In order to convert a given set of pixels (x, y) in the image frame to a point cloud expressed in the inertial frame we need depth information about the scene. In the scenario considered, we are very limited on the sensors available today. It is not possible to use a Light Detection And Ranging (LIDAR) camera to extract depth data in water environments nor is it possible to extract useful data from stereo depth cameras at more than $5 - 10$m from objects. Another alternative would be to consider the existence of landmarks that could be identified in the image and whose inertial position would be known a priori. The later is a very limiting assumption in sea environments, for which fixed natural landmarks are scarce. Taking into consideration assumption 7.1 we can assume that all the points in the inertial frame will lie on the plane described by $Z_U = 0$, which solves the depth requirement. From here it is possible to define the system

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{\lambda} \underbrace{\begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_3 & \Omega_4 \\ \Omega_5 & \Omega_6 & \Omega_7 & \Omega_8 \\ \Omega_9 & \Omega_{10} & \Omega_{11} & \Omega_{12} \end{bmatrix}}_{\Omega} \begin{bmatrix} X_U \\ Y_U \\ 0 \\ 1 \end{bmatrix}. \tag{7.5}$$

Making use of assumption 7.2 we can assume that the linear system of equations is well defined and can be inverted such that for each pixel representing the environmental boundary, X_U and Y_U are

extracted from

$$\frac{1}{\lambda}\begin{bmatrix} X_U \\ Y_U \\ 1 \end{bmatrix} = \begin{bmatrix} \Omega_1 & \Omega_2 & \Omega_4 \\ \Omega_5 & \Omega_6 & \Omega_8 \\ \Omega_9 & \Omega_{10} & \Omega_{12} \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{7.6}$$

It is important to stress that this methodology is far from perfect and small estimation errors in roll, pitch and yaw angles as well as altitude of the vehicle can lead to errors of several meters in the converted point cloud. Synchronizing the orientation data provided by the positioning system and the camera images can also prove challenging especially when the quadrotor is under high angular velocities (and the orientation of the vehicle is changing very fast). In order to avoid these data synchronization problems, a naive step is added to the process, such that if $\|\mathbf{v}_2\| \geq \omega_{max}$, the image data is discarded in that iteration.

7.2 Path Planning

In the previous section a model for converting pixels in an image to a 2-D point cloud expressed in $\{U\}$ was introduced. In this section, the problem of developing an online path planning algorithm that given a set of points generates a desired path for the quadrotor (the leader vehicle) to follow is addressed. In section 7.2.1 a set of pre-processing stages are developed, followed by sections 7.2.2 and 7.2.3 where a B-Spline fitting problem is formulated and an overview of the entire multi-step algorithm is presented.

7.2.1 Pre-processing point cloud data

Start by considering the example in Figure 7.3 where the vision system of the quadrotor produces a point cloud, representing the boundary to be followed, at an arbitrary time-step t_k. In the point cloud, some points represent the environmental boundary in a region close to the vehicle (with outliers) - the region of interest. Other points represent regions of the boundary that were partially occluded and, therefore,

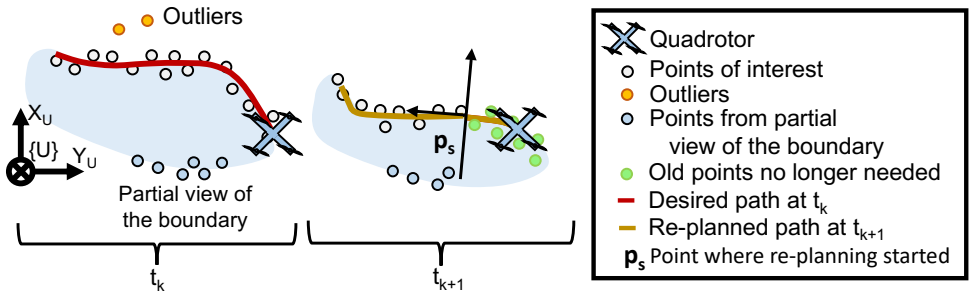


Figure 7.3: Pre-processing stage

seem disconnected from the main cluster of points. The goal is for the vehicle to follow a path (depicted in red) which fits only points in the region close to itself. This requires that at the pre-processing stage the cluster of points that are further way from the vehicle, as well as outliers, are ignored.

Another concern worth taking into consideration is that the re-planning of the path should never start exactly at the vehicle's current position on the path at instant t_k, but rather a little further ahead at an

arbitrary point $\mathbf{p}_s$. This helps avoiding jittery behavior that can result from the re-planning algorithm not being instantaneous and working independently from the PF controller. Moreover, if the camera sensor attached to the quadrotor is fixed, then the quadrotor's desired orientation should be given by the tangent to the path, allowing it to never lose sight of the boundary being followed.

Consider that at time-step t_{k+1} the vision system produces a new point cloud in which some of the points overlap a region where the quadrotor has already flown by. Since this section of the path should not be re-planned, those points should be discarded as well as other points that are "behind" $\mathbf{p}_s$ (depicted in green).

Motivated by this simple example, the following pre-processing steps are introduced:

- Remove unused points;
- Order a set of points and remove outliers;

1) Remove unused points

Consider $\mathbf{p}_s \in \mathbb{R}^2$ to be the point at which the path re-planning starts (to be defined later on), arbitrarily further ahead of the vehicle's position on the current path. In order to remove the points that are "behind" $\mathbf{p}_s$, consider that ψ_s is the tangent angle to the current path at $\mathbf{p}_s$. A coordinate transformation can be applied to the new points $\boldsymbol{X} := \{\boldsymbol{X}_m\}_{m=1}^{M} \in \mathbb{R}^2$, such that in a new reference frame, points that are behind $\mathbf{p}_s$ (points that should be ignored) have a negative X-coordinate. This coordinate transformation is given by

$$\boldsymbol{X}_m^{\circ} = R(\psi_s) \cdot (\boldsymbol{X}_m - \mathbf{p}_s), \forall m = 1, ..., M \tag{7.7}$$

where $\boldsymbol{X}_m^{\circ} = [\boldsymbol{X}_m^{\circ x}, \boldsymbol{X}_m^{\circ y}]^T$. Each point $\boldsymbol{X}_m$ is discarded if $\boldsymbol{X}_m^{\circ x} < 0$. The points that belong to set $\boldsymbol{X}$ and are not discarded, should be saved in a new set $\boldsymbol{X}^{\star} := \{\boldsymbol{X}_j\}_{j=1}^{J} \in \mathbb{R}^2$ with $J \leq M$. This process is summarized in algorithm 3.

Algorithm 3 Remove points "behind" the vehicle

1: At time step t_{k+1} obtain a new 2-D point cloud $\boldsymbol{X} := \{\boldsymbol{X}_m\}_{m=1}^{M} \in \mathbb{R}^2$;
2: Define $\boldsymbol{p}_s(\gamma)$ as the desired initial point for the re-planning to start;
3: Define ψ_s as the tangent angle to the path at t_k at $\boldsymbol{p}_s$;
4: Follow the procedure:
5: **procedure** REMOVE UNUSED POINTS($\boldsymbol{X}$, $\boldsymbol{p}_s$, ψ_s)
6: **for** $m = 1, ..., M$ **do**
7: Compute $\boldsymbol{X}_m^{\circ}$ according to (7.7);
8: **if** $\boldsymbol{X}_m^{\circ x} < 0$ **then**
9: Discard $\boldsymbol{X}_m$;
10: **return** the new (most likely smaller) set $\boldsymbol{X}^{\star} := \{\boldsymbol{X}_j\}_{j=1}^{J} \in \mathbb{R}^2$ with $J \leq M$.

One side effect of this method is that some points that are not outliers and belong to the region of interest can also end up being removed. Take for instance the example in Figure 7.4 where points in red are removed from the point cloud. At a first glance, removing points from this region might seem problematic, but it is not the case, as long as the re-planning frequency is fast enough. The point $\mathbf{p}_s$ is also changing as the vehicle moves along the path.

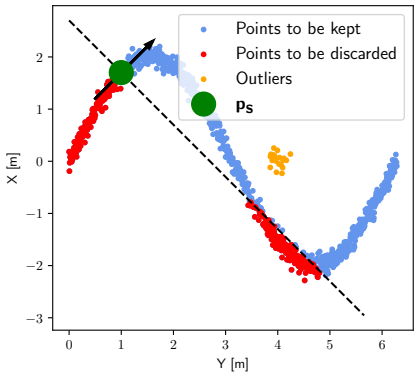
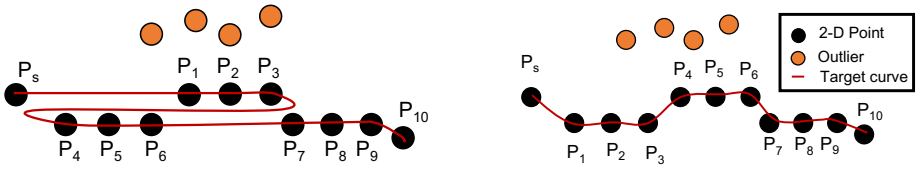


Figure 7.4: Side effect of the point removal algorithm

2) Order a set of points and remove outliers

In order to fit a set of points with a parametric curve, it is necessary to infer some natural ordering from the data. Unlike most path planning problems, we lack the knowledge of which points represent an end position goal, as well as in which order should the vehicle pass near each point. The only known variable is starting position, $\mathbf{p}_s$. Consider the simple scenario where one is required to fit 10 points of data (plus $\mathbf{p}_s$) with a parametric curve. In Figure 7.5 we can observe two distinct possible fitting results for which we associate $\mathbf{p}_s$ with $\gamma = 0$ and $\mathbf{p}_{10}$ with $\gamma = \gamma_{max}$, but assuming a different order at which each intermediate point gets associated with a corresponding curve parametric value γ.



(a) Unwanted ordering of points **(b)** Wanted ordering of points

Figure 7.5: Points ordering (A simple example)

In order to arrange the points in a consistent manner, the authors in [61] propose the construction of an Euclidean Minimum Spanning Tree (EMST) from the point cloud data. Before computing an EMST associated to the data we must first construct a graph from the set of points $\boldsymbol{X}^{\star}$ such that each vertex $\mathcal{V}$ of the graph represents a point, and each edge $\mathcal{E}$, with its associated weight A_{ij}, represents the Euclidean distance between each pair of points i and j. For computing a MST from a graph, there are two main algorithms described in the literature: Prim's and Kruskal's. Both are greedy algorithms and Kruskal's has a computational complexity of $\mathcal{O}(|\mathcal{E}|log|\mathcal{V}|)$ and Prim's $\mathcal{O}((|\mathcal{E}| + |\mathcal{V}|)log(|\mathcal{V}|))$, albeit Prim's requires an adjacency list graph representation in order to achieve that performance [62]. Kruskal's algorithm can generate forests for disconnected graphs, and achieves the best performance on sparse graphs.

If we consider that each vertex is connected to each other, the construction of the graph itself will have a computational complexity of approximately $\mathcal{O}(|\mathcal{V}|^2)$ due to the necessity of having to compute the euclidean norm between each pair of points. This is not suitable for real time applications nor to use with Kruskal's algorithm as the resulting graph will be dense. In order to simplify this problem we can consider that each point is only related to its nearest neighbour points.

To find the nearest neighbours for each point, we can resort to a very popular unsupervised learning

data structure proposed by Jon Bentley, the KDTree [63]. By defining a threshold distance N_J for the neighbors of each point, the computational cost of this nearest neighbor search for each individual point is on average $\mathcal{O}(2|\mathcal{V}|log(|\mathcal{V}|))$ [64]. Repeating this operation for all J points, we construct a sparse graph where each point has a limited set of neighbours. From there we can use Kruskal's algorithm to generate the EMST. In Figure 7.6 we can see the result of this algorithm applied to the simple example introduced previously.

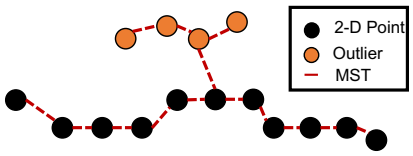


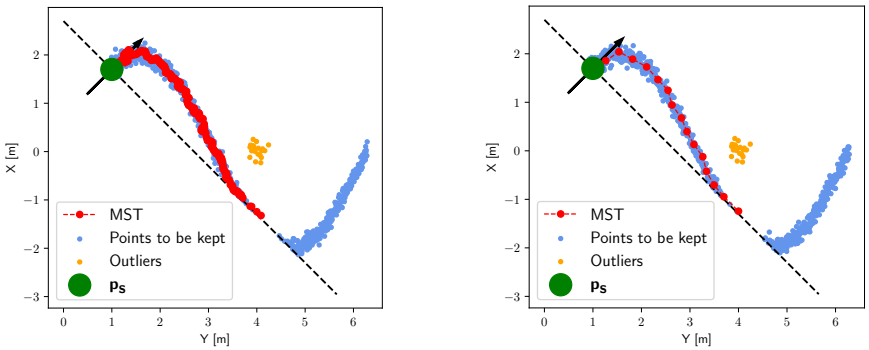
Figure 7.6: Minimum Spanning Tree (A simple example)

To get rid of outliers and define a coarse path to follow, Breadth First Search (BFS) can be applied to the points that form the MST, starting from $\mathbf{p}_s$. The resulting ordered list of points that forms the path with the highest number of points should be saved in a new ordered set $\boldsymbol{X}^\dagger := \{X_k\}_{k=1}^K \in \mathbb{R}^2$ with $K \leq J$. The algorithm for ordering the point cloud data is summarized in algorithm 4.

Algorithm 4 Order a set of 2-D points

1: Add the desired initial point for the path $\mathbf{p}_s$ to $\boldsymbol{X}^\star$ (with J points);
2: Define a threshold distance for the neighbours N_J;
3: Follow the procedure:
4: **procedure** ORDER POINTS($\boldsymbol{X}^\star$, N_J)
5: Construct a KDTree from $\boldsymbol{X}^\star$ and use N_J as a distance threshold;
6: Create a graph $\mathcal{G}$ with $J+1$ vertices and no edges;
7: **for** $\boldsymbol{X}_j$, $j = 1, ..., J+1$ **do**
8: Query the KDTree for the nearest neighbours of $\boldsymbol{X}_j$ and their corresponding distances;
9: Add the corresponding edges to the graph $\mathcal{G}$;
10: Compute the MST of the graph $\mathcal{G}$ starting from vertex corresponding to $\mathbf{p}_s$;
11: Find the path on the MST with the highest number of points, starting at $\mathbf{p}_s$ using BFS;
12: **return** the new set of points $\mathbf{X}^\dagger := \{X_k\}_{k=1}^K \in \mathbb{R}^2$ with $K \leq J$ which are part of the MST;

By applying algorithm 4 to the example introduced in Figure 7.4, the plot in Figure 7.7 a) is obtained. In this example the MST produced has a high density of points, noticeable by the wiggly red line produced. An (optional) step is to delete points that are within a pre-defined radius r, for each point in the ordered list of points $\boldsymbol{X}^\dagger$, according to Figure 7.7 b) - useful to improve the performance of the next step.



(a) Computing MST after point removal

(b) Pruning points from MST

Figure 7.7: Ordering the set of points to fit

7.2.2 Path Generation - Fitting data with a parametric curve

In order to have a suitable representation of a path that PF controllers can track, we are required to have parametric curves that are both smooth and at least $\mathcal{C}^2$. For this work, we resort to uniform cubic B-Splines which boast a set of useful properties introduced in section 2.4.

1) Define the number of segments

Consider now the ordered sequence of K points obtained via the application of algorithms 3 and 4 to the original point cloud data. In order to fit the ordered sequence of points with a parametric curve we are required to attribute to each point $\boldsymbol{X}_k \in \mathbb{R}^2$ a corresponding γ_k in the target curve (see section 2.5.1). This problem could be formulated as a nonlinear optimization problem (computationally demanding to solve for real-time applications). A good approximation proposed by [46] is to consider D_X to be the total distance between the points, given by

$$D_X := \sum_{k=2}^{K} \|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|, \tag{7.8}$$

and the corresponding vector of parametric values $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_k]^T$ to be given by

$$\begin{cases} \gamma_1 = 0 \\ \gamma_k = \gamma_{k-1} + \frac{\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|}{D_X}\gamma_{max}, k = 2, ..., K \end{cases}, \tag{7.9}$$

where γ_{max} is the maximum parameter value of the parametric curve. For cubic B-splines, this number depends directly on the number of control points that the target curve will have. The number of control points also dictates how many spline segments are actually used for the fitting problem. The optimal number of control points can also be obtained by solving yet another nonlinear optimization problem, but due to the real time nature of the problem this option is disregarded in this work. A uniform cubic B-spline must have at least 4 control points to define one segment. Given that a low number of sections can under-fit a long set of points and a high number lead to over-fitting issues, this number should not be a static constant. A non-optimal, yet dynamic way of defining the number of control points N_C is by taking:

$$N_C := max\left\{ \left\lfloor \frac{D_X}{\rho} \right\rceil, 4 \right\}, \tag{7.10}$$

with $(1/\rho) > 0$ a control points density (tunning parameter). A smaller ρ leads to a higher N_C.

2) Fit the points with a B-spline

For fitting the ordered set of points $\boldsymbol{X}^\dagger$ an optimization problem must be formulated. Consider the objective function given by

$$f(P_0, ..., P_{N_C-1}) := \underbrace{\sum_{k=1}^{K} \|\mathbf{C}(\gamma_k, \mathbf{P}) - \mathbf{X}_k\|^2}_{goal} + \underbrace{\lambda \int_0^{\gamma_{max}} \left\| \frac{\partial \boldsymbol{C}(\gamma, \mathbf{P})}{\partial \gamma} \right\|^2 d\gamma + \beta \int_0^{\gamma_{max}} \left\| \frac{\partial^2 \boldsymbol{C}(\gamma, \mathbf{P})}{\partial \gamma^2} \right\|^2 d\gamma}_{\text{regularization term}} \tag{7.11}$$

with $\alpha, \beta \geq 0$. The first term minimizes the distance between the target B-Spline curve and the set of points whilst the second is a regularization term. The integral of the L_2^2 norm of the first derivative penalizes the total length of the curve and while the integral of the L_2^2 norm of the second derivative penalizes bends in the path. This objective function can also be expressed using vector notation, according to

$$f(\mathbf{P}) = \underbrace{\|B(\boldsymbol{\gamma})\mathbf{P} - \mathbf{X}\|^2}_{goal} + \underbrace{\lambda \mathbf{P}^T R_1 \mathbf{P} + \beta \mathbf{P}^T R_2 \mathbf{P}}_{\text{regularization term}}, \tag{7.12}$$

where $\mathbf{P} = [P_0^x, ..., P_{N_{C-1}}^x, P_0^y, ..., P_{N_{C-1}}^y]^T$ denotes the vector of control points where the X- and Y-coordinates are concatenated such that each control point P_i is defined by the tuple $P_i = (P_i^x, P_i^y)$ (see appendix C.1). The vector $\mathbf{X} = [X_1^x, ..., X_K^x, X_1^y, ..., X_K^y]$ denotes points to fit, and R_1, R_2 are constant matrices that can be computed numerically (see section 2.5.1 and appendix C.2).

In order to guarantee $\mathcal{C}^2$ continuity between the current curve and the newly planned one, linear equality constraints could be imposed on the values of $\mathbf{C}(0)$, $\mathbf{C}'(0)$ and $\mathbf{C}''(0)$. Another robust approach is to take advantage of the local support property of B-splines, introduced in section 2.4.2. It is known that for the particular case of uniform cubic B-splines, each segment depends only on 4 control points that "slide" from one section to the next one. Take the example in Figure 7.8 where an initial curve $\mathbf{C}(\gamma)$ is made of two distinct spline segments such that $\gamma \in [0, 2)$. The first segment is only affected by control points P_0, P_1, P_2 and P_3 whilst the second is dictated by the control points P_1, P_2, P_3 and P_4. This powerful property can be exploited both to simplify the equality constraints that guarantee that the transition between the initial and newly planned curves is $\mathcal{C}^2$, but also to have a natural definition for the point $\mathbf{p}_s$, left undefined until now.

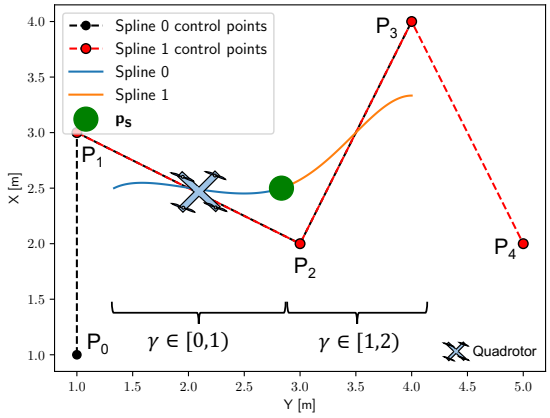


Figure 7.8: B-spline example of the local support property

A trivial method to define $\mathbf{p}_s$ is to consider the quadrotor's current virtual target γ_{t_k}, at time instant t_k, and define the starting point for the re-planning according to

$$\mathbf{p}_s := \mathbf{C}(\lceil \gamma_{t_k} \rceil). \tag{7.13}$$

This point corresponds to the transition between the spline the virtual target is "sitting on" and the next curve segment. For the particular example in Figure 7.8, $\gamma_{t_k} \in [0, 1)$ and $\mathbf{p}_s = \mathbf{C}(1)$, shown as a green circle.

Given a $\mathbf{p}_s$ dictated by (7.13), the curve segments that are described by parametric values such as $\gamma \geq \lceil \gamma_{t_k} \rceil$ should be discarded and replaced by a newer curve. Since each curve segment is defined by only 4 control points, then discarding those segments is equivalent to removing control points with indexes $i \geq \lceil \gamma_{t_k} \rceil + 3$ from the current control points vector. This operation results in a vector given by $\mathbf{P}^{old} = [P_0^x, P_1^x, ..., P_{\lceil \gamma_{t_k} \rceil}^x, P_{\lceil \gamma_{t_k} \rceil+1}^x, P_{\lceil \gamma_{t_k} \rceil+2}^x, P_0^y, P_1^y, ..., P_{\lceil \gamma_{t_k} \rceil}^y, P_{\lceil \gamma_{t_k} \rceil+1}^y, P_{\lceil \gamma_{t_k} \rceil+2}^y]^T$. For the example in Figure 7.8, spline 1 should be discarded as it is defined for parametric values of $\gamma \geq 1$, which in practice means removing control points with indexes $i \geq 1 + 3$ from the control points vector, i.e. $P_4 = (P_4^x, P_4^y)$. The resulting control points vector becomes $\mathbf{P}^{old} = [P_0^x, P_1^x, P_2^x P_3^x, P_0^y, P_1^y, P_2^y P_3^y]^T$.

Making use of the local support property once more, it is known that $\mathcal{C}^2$ continuity between the two consecutive spline segments is guaranteed, as long as the last 3 control points of the first segment coincide with the first 3 control points of the second segment. A trivial manner of generating a new B-Spline with guarantees of $\mathcal{C}^2$ continuity in the transition with the old curve, without explicitly defining equality constrains on the derivatives of the function, is to solve the following optimization problem:

$$\begin{aligned} \mathbf{P}^{new} = \underset{\mathbf{P}^{new}}{\text{argmin}} \ & \|B(\boldsymbol{\gamma})\mathbf{P}^{new} - \mathbf{X}\|^2 + \lambda {\mathbf{P}^{new}}^T R_1 \mathbf{P}^{new} + \beta {\mathbf{P}^{new}}^T R_2 \mathbf{P}^{new} \\ \text{subject to} \ & \begin{bmatrix} P_0^{x\,new} \\ P_1^{x\,new} \\ P_2^{x\,new} \\ P_0^{y\,new} \\ P_1^{y\,new} \\ P_2^{y\,new} \end{bmatrix} = \begin{bmatrix} P_{\lceil \gamma_{t_k} \rceil}^x \\ P_{\lceil \gamma_{t_k} \rceil+1}^x \\ P_{\lceil \gamma_{t_k} \rceil+2}^x \\ P_{\lceil \gamma_{t_k} \rceil}^y \\ P_{\lceil \gamma_{t_k} \rceil+1}^y \\ P_{\lceil \gamma_{t_k} \rceil+2}^y \end{bmatrix}, \end{aligned} \tag{7.14}$$

where $\mathbf{P}^{new} = [P_0^{x\,new}, ..., P_{N_C-1}^{x\,new}, P_0^{y\,new}, ..., P_{N_C-1}^{y\,new}]^T$. In order to keep track of old and new curves, it is possible to just concatenate the new control points vector $\mathbf{P}^{new}$ with the old control points vector $\mathbf{P}^{old}$, ignoring the first three control points, i.e. P_0^{new}, P_1^{new} and P_2^{new}, which are repeated as a result of the equality constraints imposed by (7.14). These series of procedures are summarized in algorithm 5.

Algorithm 5 Fitting the points - growing uniform cubic B-spline

1: Compute D_X, $\boldsymbol{\gamma}$ and N_C according to equations (7.8), (7.9) and (7.10) respectively

2: Consider γ_{t_k} as the value of the virtual target at the re-planning instant and the original control points vector:

$$\mathbf{P} = \left[P_0, P_1, ..., P_{\lceil \gamma_{t_k} \rceil}, P_{\lceil \gamma_{t_k} \rceil+1}, P_{\lceil \gamma_{t_k} \rceil+2}, P_{\lceil \gamma_{t_k} \rceil+3}, P_{\lceil \gamma_{t_k} \rceil+4} ..., P_n\right]^T; \tag{7.15}$$

3: Remove control points (corresponding to splines to be re-planned) from the original control points vector, such that:

$$\mathbf{P}^{old} = \left[P_0, P_1, ..., P_{\lceil \gamma_{t_k} \rceil}, P_{\lceil \gamma_{t_k} \rceil+1}, P_{\lceil \gamma_{t_k} \rceil+2}\right]^T; \tag{7.16}$$

4: Solve the optimization problem in (7.14) and obtain a new vector with N_C control points:

$$\begin{aligned} \mathbf{P}^{new} &= \left[P_0^{new}, P_1^{new}, P_2^{new}, ..., P_{N_C-1}^{new}\right]^T, \\ \text{with } P_0^{new} &= P_{\lceil \gamma_{t_k} \rceil}, P_1^{new} = P_{\lceil \gamma_{t_k} \rceil+1}, P_2^{new} = P_{\lceil \gamma_{t_k} \rceil+2}; \end{aligned} \tag{7.17}$$

5: Concatenate the new vector with the old vector (ignoring the first 3 control points which are repeated):

$$\mathbf{P}^{new} = \left[P_0, P_1, ..., P_{\lceil \gamma_{t_k} \rceil}, P_{\lceil \gamma_{t_k} \rceil+1}, P_{\lceil \gamma_{t_k} \rceil+2}, P_3^{new}, ..., P_{N_C-1}^{new}\right]^T. \tag{7.18}$$

Remark 1: For the sake of simplicity, the separation between X and Y-coordinates of the control points was omitted in both algorithm 5 and Figure 7.9.

Applying algorithm 5 to the example introduced previously, the result in Figure 7.9 is obtained. From here it is observable that at time instant t_k the vehicle was on top of spline 1, meaning that $\lceil \gamma_{t_k} \rceil = 1$. According to (7.16) control point P_4 (used only to define spline 2) is no longer needed, hence removed from the vector. After this step, the optimization problem (7.14) is solved, fitting an ordered set of points to a new curve with $N_C = 5$. In the end, only the last 2 new control points are concatenated into the previous control points vector, as the first 3 are repeated.

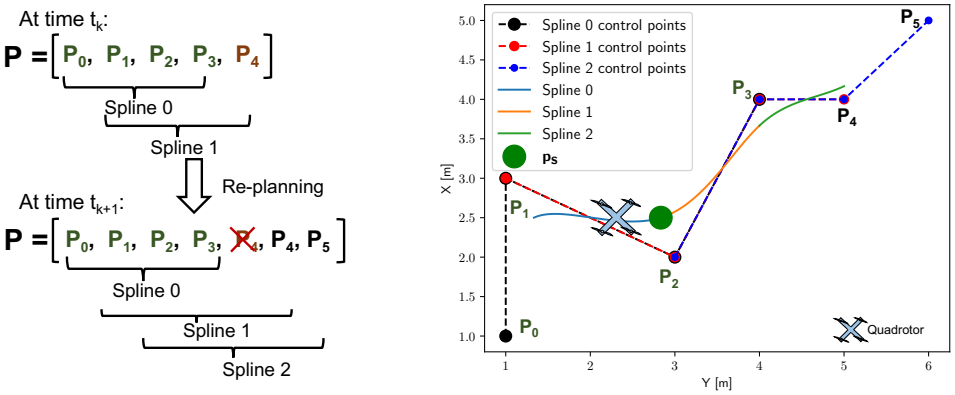


Figure 7.9: B-spline update with new control points

Remark 2: For the first iteration the same algorithms are used with 3 key differences:

- The position $\mathbf{p}_s$ and angle ψ_s are given by the vehicle's position and orientation;
- Steps 2, 3 and 5 in algorithm 5 are ignored;
- The linear constraint in (7.14) is now given by $B(0)\mathbf{P}^{new} = \mathbf{p}_s$, such that the initial path starts at the vehicle's position;

Remark 3: In practice, to solve the optimization problem, we resort to Scipy's python SQP solver. The control points vector is initialized by distributing N_C points uniformly across a straight line that connects $\boldsymbol{X}_1$ and $\boldsymbol{X}_K$.

Remark 4: The algorithms proposed are made publicly available as a Python library [65] and a brief performance analysis provided in appendix D.

7.2.3 Algorithm Overview

Considering the algorithm introduced in this chapter, it is possible to summarize the multiple steps of the proposed path planning algorithm in Figure 7.10. This constant re-planning of the path occurs at a pre-defined frequency f.

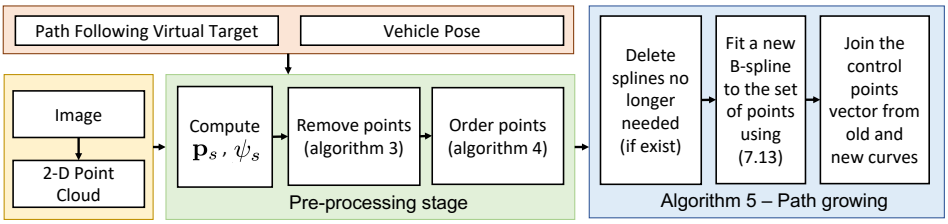
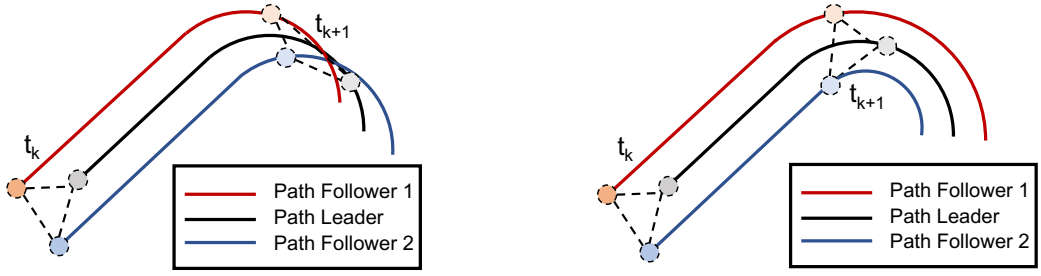


Figure 7.10: Resume of the path-planning algorithm

On a more practical side-note, this algorithm was devised to be executed in real time by an UAV or it's companion computer. This means that the path must be transmitted over the network to the fleet of vehicle's every time it gets updated. This can be done quite efficiently by only sharing the index at which the new control points vector starts along with the new points (only).

7.3 Multi Path Coordination

In the previous section, an algorithm for generating a desired path for the quadrotor (the leader vehicle) was developed. In order to perform CPF missions with a fleet of vehicles, it is necessary to define a path for each individual vehicle to follow. When considering a leader-follower topology, a naive approach would be to consider a path for each vehicle that results from a fixed offset of the original path. This approach is only viable for straight line paths, because for any other path with a curvature different than zero the curves produced by this method can intersect with each other, according to Figure 7.11 making the vehicles lose their required formation.



(a) Naive Path

(b) Trailor-like Path

Figure 7.11: Multi-Path Generation

A more robust approach, proposed by W. Xie [66], is to consider an auxiliar frame of reference attached to the virtual target on the path. The x-axis of this new reference frame $\{T\}$ is chosen to be aligned with the tangent vector to the path in γ_i. The unit vector $\mathbf{r}_1$ that describes the orientation of the x-axis is given by:

$$\mathbf{r}_1(\gamma_i) = \frac{\partial \mathbf{p}_d / \partial \gamma}{\|\partial \mathbf{p}_d / \partial \gamma\|}, \text{ with } \|\partial \mathbf{p}_d / \partial \gamma\| \neq 0. \tag{7.19}$$

Consider $\mathbf{r}_d$ to be a unit vector that is not parallel to $\mathbf{r}_1(\gamma_i)$, then it is possible to define

$$^{U}_{T}R(\gamma_i) = [\mathbf{r}_1(\gamma_i), \mathbf{r}_3(\gamma_i) \times \mathbf{r}_1(\gamma_i), \mathbf{r}_3(\gamma_i)], \tag{7.20}$$

where

$$\mathbf{r}_3(\gamma_i) = \frac{\mathbf{r}_d - (\mathbf{r}_d \cdot \mathbf{r}_1(\gamma_i))\mathbf{r}_1(\gamma_i)}{\|\mathbf{r}_d - (\mathbf{r}_d \cdot \mathbf{r}_1(\gamma_i))\mathbf{r}_1(\gamma_i)\|}. \tag{7.21}$$

Defining a formation vector denominated $\mathbf{d}_i \in \mathbb{R}^3$ for each vehicle, with each distance defined in the tangential reference frame $\{T\}$ it is possible to define

$$\mathbf{p}_{Fi}(\gamma_i) = \mathbf{p}_d(\gamma_i) + {}^{U}_{T}R(\gamma_i)\mathbf{d}_i, \tag{7.22}$$

where $\mathbf{p}_{Fi}$ is the desired path for the vehicle i, according to Figure 7.12. The introduction of the rotation matrix proposed, gives origin to a rigid formation between agents where each vehicle follows a virtual leader in a trailer-like approach. With this method, the virtual target progression stays the same, and just the desired position get's shifted.

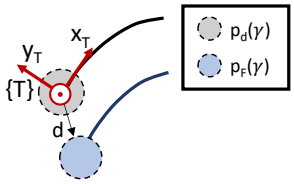


Figure 7.12: Formation vector overview

It is important to take into consideration that the ASVs can only operate at the surface of the water, hence a naive approach for dealing with this detail is to simply discard the Z-axis coordinate of $\mathbf{d}_i$ and $\mathbf{p}_{Fi}$. Moreover, since all the vehicle's will only be required to operate in a 2-D plane a trivial definition for one of the axis of the tangential frame $\{T\}$ is $\mathbf{r}_d = [0, 0, 1]^T$.

7.4 Desired Speed Assignment

Consider that the leader vehicle is required to move at a constant speed $V \leq V_{max}$. Consider also that in the PF strategies introduced previously, the desired velocity of the vehicle is not controlled directly, but rather the progression speed of the parametric variable $\dot{\gamma}$ by defining a desired speed profile $v_L(\gamma)$. Furthermore, the path might not be not parameterized by arc length and, for B-Splines in particular, each path segment i is such that $\gamma_i \in [0, 1)$. A trivial way to define the speed profile for the vehicle is to consider

$$v_L(\gamma) = \frac{V}{\|\mathbf{p}'_d(\gamma)\|}. \tag{7.23}$$

In Romulo et al. [67], the authors propose a more robust approach where each vehicle is required to slow down on sharper turns and approximate to the desired constant speed in straight lines, according to

$$v_L(\gamma) = \frac{V}{\|\mathbf{p}'_d(\gamma)\| \left(1 + K_c \|\mathbf{p}''_d(\gamma)\|\right)}, \tag{7.24}$$

where $K_c > 0$ is a constant gain that regulates the slow down of the vehicle on curves.

Chapter 8

Implementation Architecture

In this chapter, a detailed description of the code modules developed in the scope of this thesis is presented. In Section 8.1 an introduction to the simulation environment adopted for both the UAV quadrotor and Medusa AUV is conducted. The path manager, PF and CPF controllers implementation is addressed in section 8.2. Finally, in section 8.3 the setup adopted for performing real trials with two medusa vehicles is described.

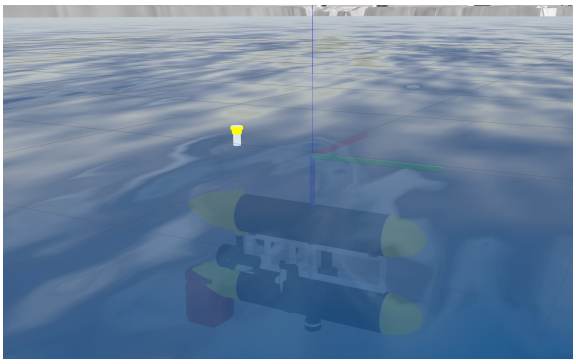
8.1 Simulation Architecture

To evaluate the performance of the proposed systems and in order to have a realistic simulation environment we resorted to the UUVSimulator Plugin [68] and PX4 SITL Gazebo Plugin [69] which are extensions of the Gazebo 9 simulator [70], widely used in the robotics community. The operating system used during development was Ubuntu 18.04LTS along with Robots Operating System (ROS) Melodic [71], which corresponds to the software version running on the real Medusa vehicles at the time of writing.

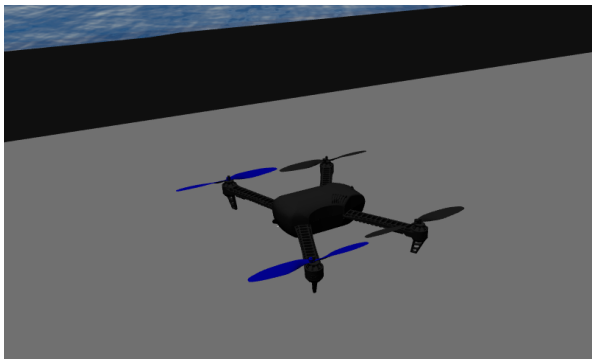
The UUVSimulator Plugin provides a structured environment for adding new AUV dynamics, thruster dynamics and vehicle Computer Aided Design (CAD) models to the Gazebo simulator. Furthermore, it provides a suite of sensors such as DVL, Attitude and Heading Reference System (AHRS) and DGPS that can be equipped in the virtual vehicle. Moreover it allows for the simulation of ocean currents, both physically and visually. With this plugin, it was possible to generate a virtual vehicle that closely follows the specifications in appendix A.

On the other hand, the PX4 SITL Gazebo Plugin already offers a set of quadrotor models that are very similar to the ones already available at Dynamical Systems and Ocean Robotics Laboratory (DSOR) lab. Therefore, only minor tweaks were made to the already provided Iris quadrotor in order to have it follow the spec-sheet in appendix B. A visual overview of the simulated vehicles is available in Figure 8.1.

Additionally, there was the necessity to have a realistic simulation environment that closely resembled Doca dos Olivais, Lisbon Portugal, which is the main sight where real water trials of DSOR AUV vehicles are conducted. For that matter, a 3-D real life-size CAD model of this area was imported into the simulator, according to Figure 8.2.



(a) Simulated Medusa vehicle



(b) Simulated Iris quadrotor vehicle

Figure 8.1: Simulated vehicles

Given the main goal of having a fleet of vehicles following an environmental boundary, is was necessary to modify the ocean mesh provided by the UUVSimulator in order to accommodate a simulated chemical spill. This required that the new surface mesh was exported twice: i) the first mesh corresponding to the chemical spill isolated; ii) the second mesh corresponding to the ocean itself. A limitation of this method it that the red mesh used to simulate the chemical spill inside gazebo is static and does not deform according to advection and diffusion processes.

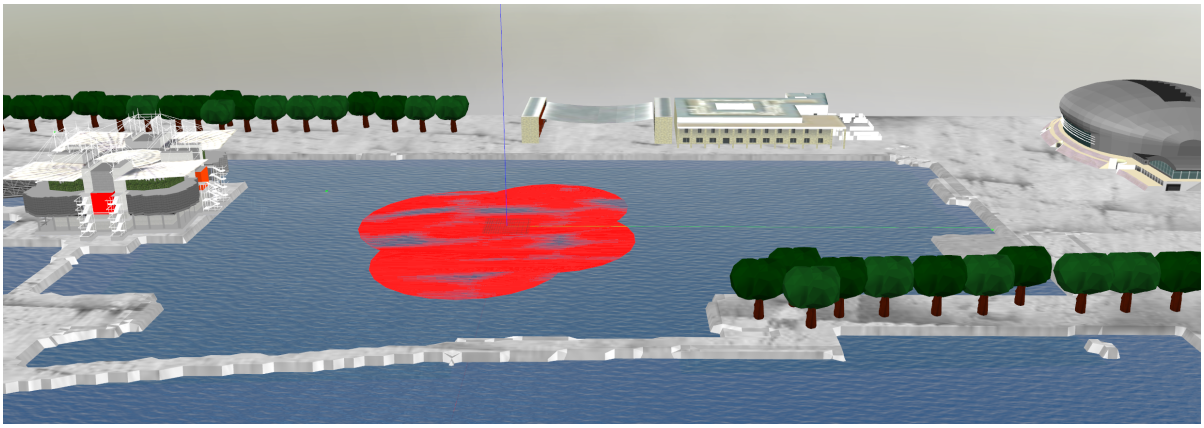


Figure 8.2: Simulated model of Doca dos Olivais

Since the development of an image processing algorithm responsible for the detection of the boundary region between the spill and the ocean surface was out of the scope of this project, the spill mesh was tinted with a bright red color. Resorting to the very popular computer vision library OpenCV [72] we use a mask to threshold red colours in the quadrotor camera feed, leaving us with a red spill in a black background (when its detected). After this step we apply Canny edge detection to the binary image in order to retrieve the pixels corresponding to the edge of the boundary. For every frame where boundary pixels are detected, we use the algorithm proposed in section 7.1 to generate a 2-D point cloud.

In Figure 8.3 a global overview of the entire simulated system composed of one quadrotor and one Medusa vehicle is provided, where each major sub-system is represented as an individual block. In order to allow the data provided by the simulator to be used by the already existing Medusa code framework, a new translation layer was developed, which converts actuator input and sensor data to a compatible

ROS format. The Medusa code framework already provides a Kalman Filter (KF) used for estimating the state of the vehicle and a set of inner-loops similar to the ones introduced in section 4.1.

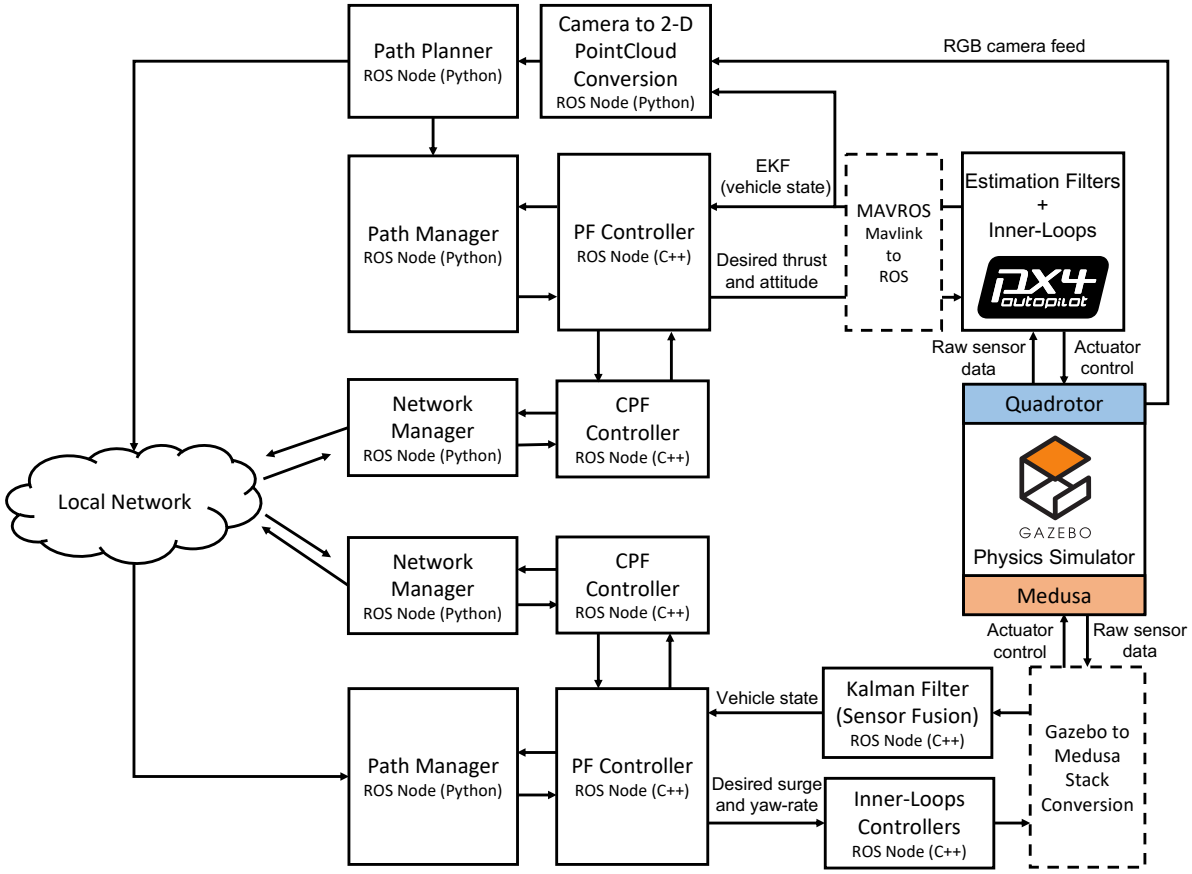


Figure 8.3: Simulation Architecture

On the other hand, the simulated quadrotor interfaces directly with the PX4 autopilot through the MAVLink protocol [73] and already provides an Extended Kalman Filter (EKF) and a set of inner-loops to control the vehicle. In order for the autopilot to interface with ROS 1 we resort to the translation layer MAVROS [74].

For executing CPF missions, four distinct ROS nodes run in each vehicle: a Path Manager, a PF controller, a CPF controller and a network manager. The Path Manager is responsible for saving the desired path for the vehicle to follow, and given a path parameterizing variable send the most up to date path data to the PF controller. The PF controller implements the actual control laws that allow the vehicle to follow the desired path and the CPF controller implements the speed coordination law. Each vehicle runs a Network Manager node responsible for receiving and broadcasting to the local network the virtual target state provided by the CPF controller according to the ETC scheme via User Datagram Protocol (UDP). The code running for the Network Manager, Path manager and CPF controller is vehicle agnostic and its the same for the quadrotor and the Medusa robots.

8.2 Path Following Code Structure

When developing the control algorithms to be employed in the real DSOR vehicles it was key to have a modular system that would enable fast switching between controllers and desired paths in real time, without having to restart the entire navigation system. With this requirement in mind, every ROS node developed follows a structure similar to the one presented in Figure 8.4 a), where the ROS middleware code (in green) is isolated from the logic code (in yellow). For the logic side of every node, we resort to Object Oriented Programming (OOP) principles. As a result, switching between controllers is as easy as instantiating a new object in memory.

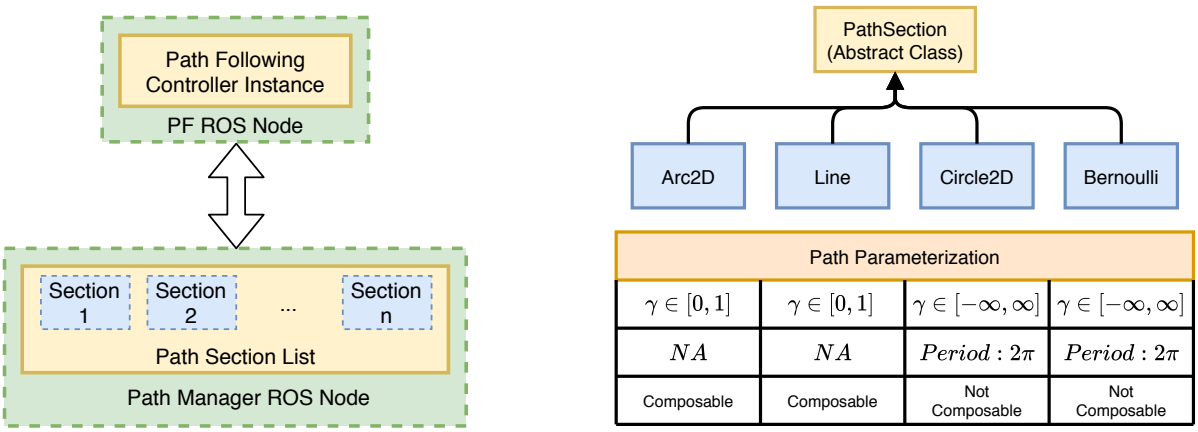
8.2.1 Path Manager

In order to represent the paths for each individual vehicle to follow, two path manager libraries were developed:

- A static path manager, written in C++, used to represent paths that do not change over time;
- A dynamic path manager, written in Python, used to represent B-spline paths for which the shape changes dynamically over time.

Representing Static Paths

When designing the path manager it was defined that each type of path should be a class that inherits an abstract *PathSection* class, according to Figure 8.4 b). This allows for the design of static paths to be flexible and at the same time follow a consistent code pattern.



(a) ROS node abstraction

(b) Paths library

Figure 8.4: Path following and static paths package

In the *PathSection* class code (see Appendix E.1), some functions that are common to all parametric curves already provide a default implementation. This is the case for computing the tangent angle to the path, the curvature, the norm of the derivative, etc. To add a new type of path to this code library, a control designer only needs to implement the parametric equations for computing the position, first and second derivatives of the path evaluated at a given parametric value.

Some of the path sections designed can be composed together to form more complex shapes, i.e. lines and arcs can be used to form lawn mowing paths (see Figure 8.4). Paths that are closed shapes,

such as Bernoulli Lemniscate, cannot be composed with other paths by default. The static path manager is only responsible for storing a list of path sections in memory, receiving from the PF controller the current parametric value γ and sending to the controller the necessary path data, such as position, derivatives, etc. (see Appendix E.1 - Listing C.2).

Representing Dynamic Paths

The dynamic path manager is solely dedicated to storing and updating B-splines. Each vehicle switches from the static path manager to the dynamic path manager when an environmental boundary is detected by the path planner. Upon a boundary detection, the path planner (that receives data from the vision system) starts planning in real time a new path, according to the algorithms described in chapter 7. The dynamic path manager is actively receiving new B-spline control points from the path planner. These points are used to replace the ones obtained in the previous time step. The dynamic path manager running on the quadrotor has direct access to the control points produced by the path planer. On the other hand, the path manager running on the ASVs receive them via Transmission Control Protocol (TCP).

8.2.2 ASV Path Following Controller

Following suit, the PF controller used for the ASVs has an abstract class *PathFollowing*, that every controller inherits from. Even though not necessary for this thesis, but also in the scope of the projects that partially funded this work (EUMarineRobotics and RAMONES), multiple PF algorithm were implemented in this module, according to Figure 8.5. A theoretical overview of these algorithms can be found in a survey paper by Hung et al. [11].

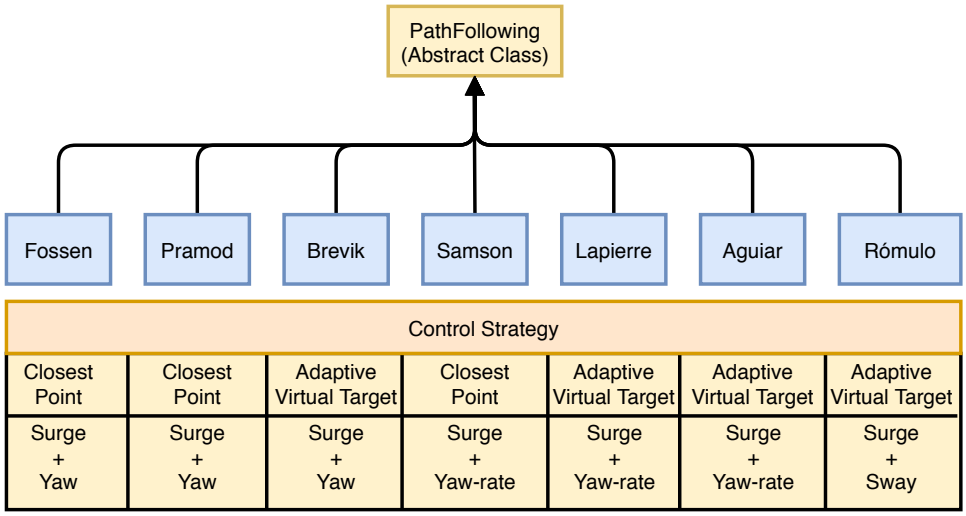


Figure 8.5: ASV path following library/package

Each controller class only has to implement 6 functions: *start*, *stop*, *reset*, *setPFGains*, *callPFController* and *publishPrivate* (see Appendix E.2). The first 3 only serve the purpose of initializing the controller, stop it or reset its gains to default values. The *callPFController* function is where the actual logic of the controllers is implemented and *publishPrivate* where the control values are made available to the inner-loops of the vehicle.

8.2.3 Quadrotor Path Following Controller

For the quadrotor, the PF controller implementation also follows an OOP philosophy. In this case, in order to make development faster, a modified version of the UAV C++ library developed by Oliveira et al. [75] was used, making it easier to access the state of the vehicle and to send command to the inner-loop controllers. The original trajectory tracking controller using a PID provided in the library was kept while an additional one using the nonlinear control law proposed in section 5.2 was created.

For the sake of simplicity, the inner-loops used in the simulated quadrotor were the ones already provided by PX4 which use as inputs a set of desired angles references and total thrust. The values of thrust received by the controller are normalized between $[0, 1]$, according to the thrust equation in appendix B.

8.3 Architecture for Real Water Trials with Medusa Vehicles

In the scope of this thesis and the projects that partially funded this work, it was also possible to access the performance of the proposed PF and CPF (using ETC communications) algorithms with two real Medusa ASVs. For that matter, the system architecture initially proposed in Figure 8.3 was slightly modified, according to Figure 8.6.

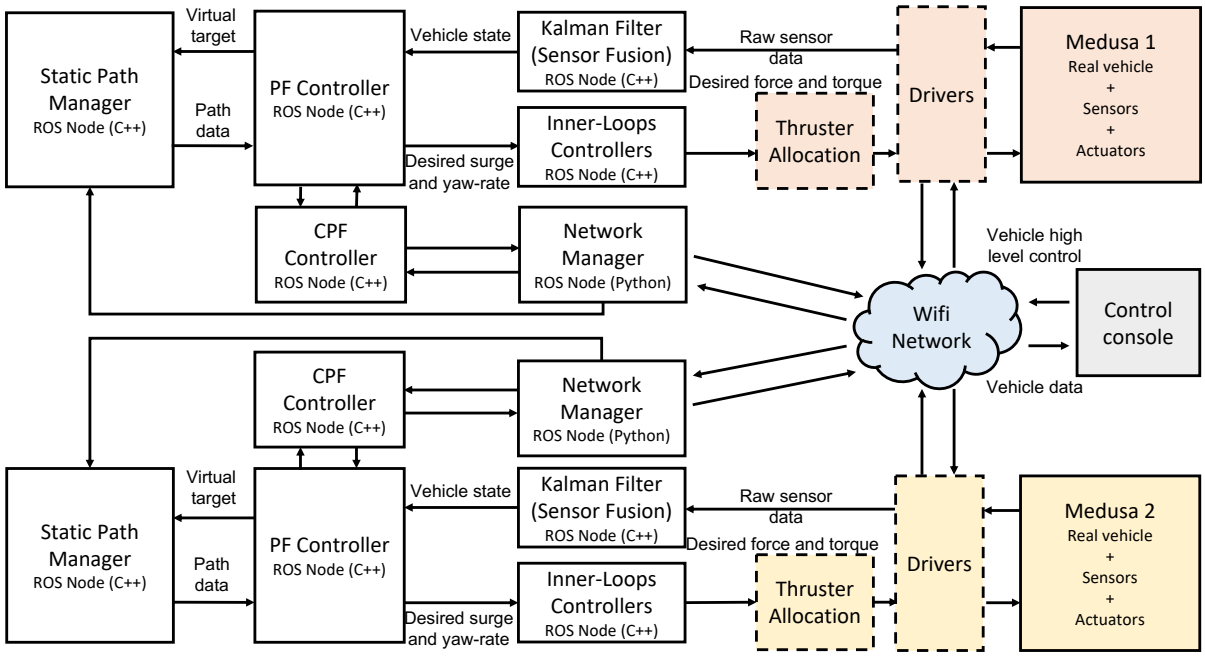


Figure 8.6: Real water trials architecture

The main absence in this architecture is the quadrotor and the real time path planning algorithm introduced in chapter 7.

Chapter 9

Results

In the previous chapter, a high-level description of the implementation architecture for the multi-vehicle system was provided. In this chapter, results emphasizing the different algorithms proposed are presented. First a set of simulations are conducted to access the performance of the ASV inner-loops, along with PF and CPF for both vehicles. Then, two experiments are carried where a quadrotor is required to follow a simulated environmental boundary in coordination with an ASV. Moreover real results for two real Medusa ASVs performing PF and CPF missions are provided. The controller gains and parameters adopted can be found in Appendix F.

9.1 Simulations Results

9.1.1 Medusa Inner-Loops

We start by presenting two simulations where PI inner-loops developed for ASVs were tested. It was required that the surge speed converged to a desired reference value. Then, the yaw-rate was required to converge to its reference values. From close examination of Figure 9.1, it is possible to conclude that both controllers exhibit good performance converging to their desired constant references in an acceptable amount of time.

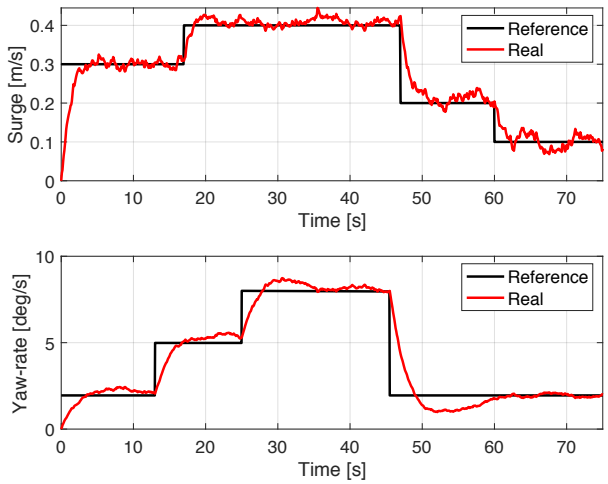
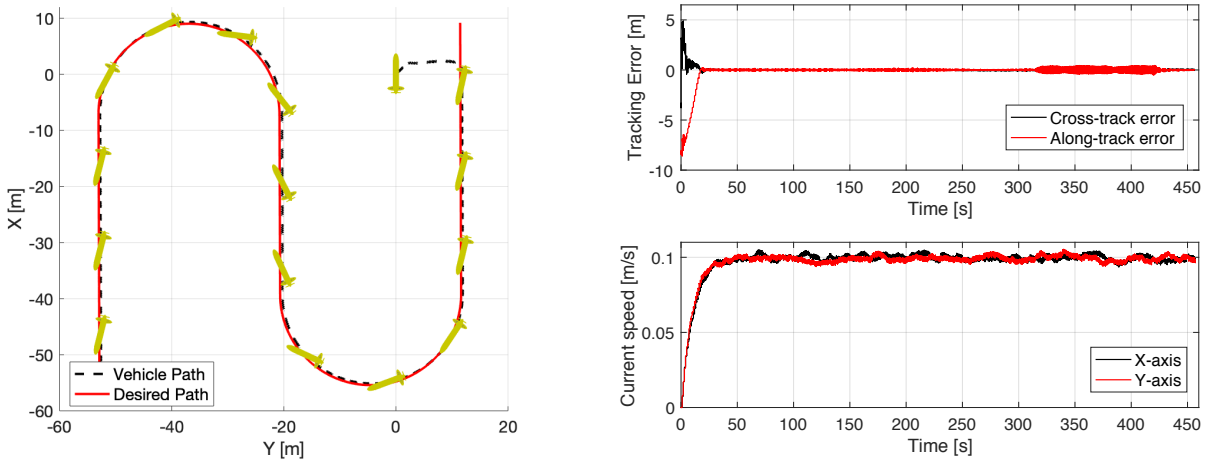


Figure 9.1: Medusa inner-loops performance (simulation)

9.1.2 Medusa Path Following

For the next simulation, a Medusa vehicle was required to execute a lawn-mowing path under the effect of oceans currents with speeds of $\mathbf{v_c} = [0.1, 0.1]^T$m/s. According to Figure 9.2 a) the vehicle started its mission approximately 10m away from the desired path, and the virtual target waited for the vehicle to converge to the path before it started progressing at its desired speed. From the results in Figure 9.2 b), it is observable that the cross-track and along-track errors converge to zero for straight lines and increase slightly in arc sections due to sway motion that was neglected in the control design phase. The currents observer also exhibited good performance converging to 0.1m/s in both X and Y-axis.

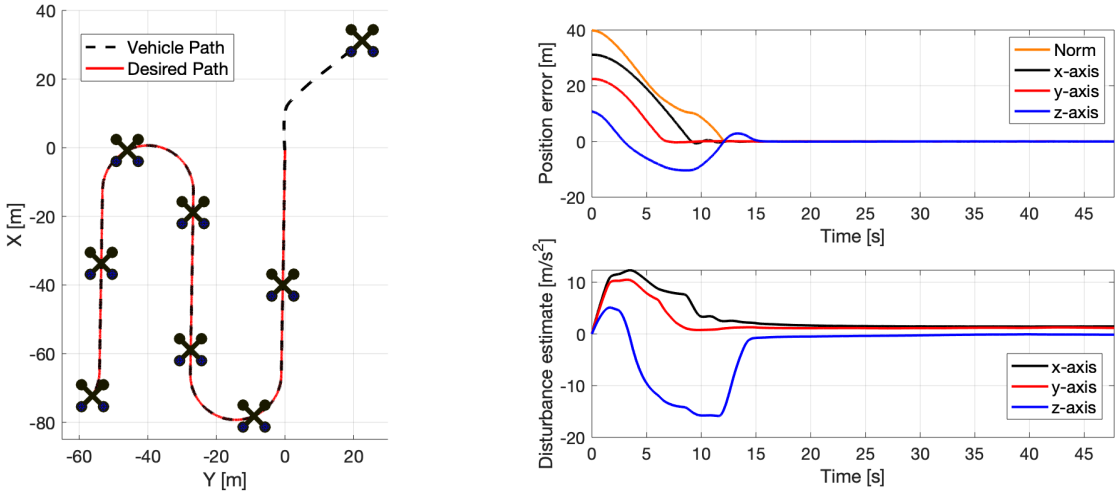


(a) XY view **(b)** Tracking error and currents estimates

Figure 9.2: Medusa path following performance (simulation)

9.1.3 Quadrotor Path Following

The next step was to test the performance of the quadrotor PF algorithm under the presence of winds with speeds of 1.0m/s in both X and Y-axis. In Figure 9.3 a) a top view of path executed by the drone is shown. For this mission, the UAV was required to perform a lawn-mowing manoeuvre at 30m of altitude



(a) XY view **(b)** Drone tracking error

Figure 9.3: Quadrotor path following performance (simulation)

with a fixed yaw angle of 0°. From the plots in Figure 9.3 b), it is observable that the error between the desired position of the quadrotor and the real one converged to approximately zero, while the disturbance observer converged to values near zero when the drone stabilized its altitude.

9.1.4 CPF with ETC between Quadrotor and 2 Medusa Vehicles

In the next experiment a CPF mission was performed where the quadrotor was required to follow a lawn-mowing trajectory with two Medusa ASVs, at a desired speed of 0.5m/s, according to a triangle formation. The aircraft was required to fly at an altitude of 30m and the formation vectors for the marine vehicles are given by $\mathbf{d}_1 = [-5, 5, 0]^T$m and $\mathbf{d}_2 = [-5, -5, 0]^T$m. In Figure 9.4 a 3-D view of the executed mission is provided whilst in Figure 9.5 a 2-D view is provided along with relevant performance metrics.

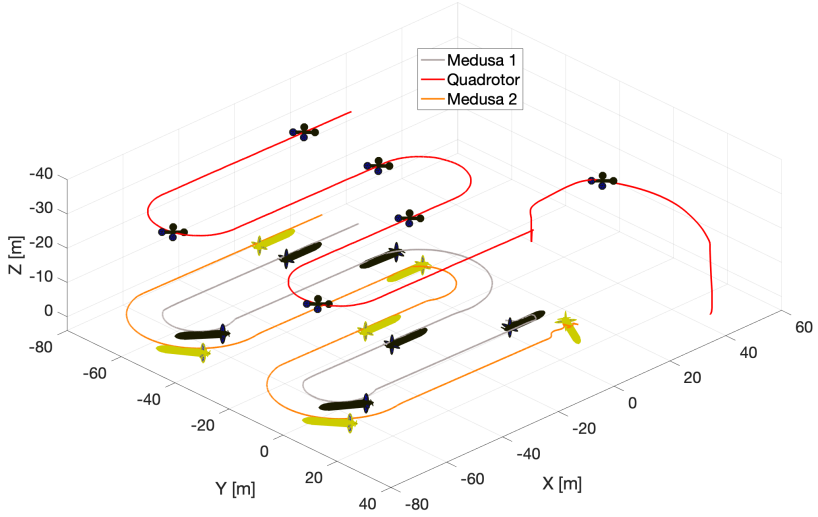
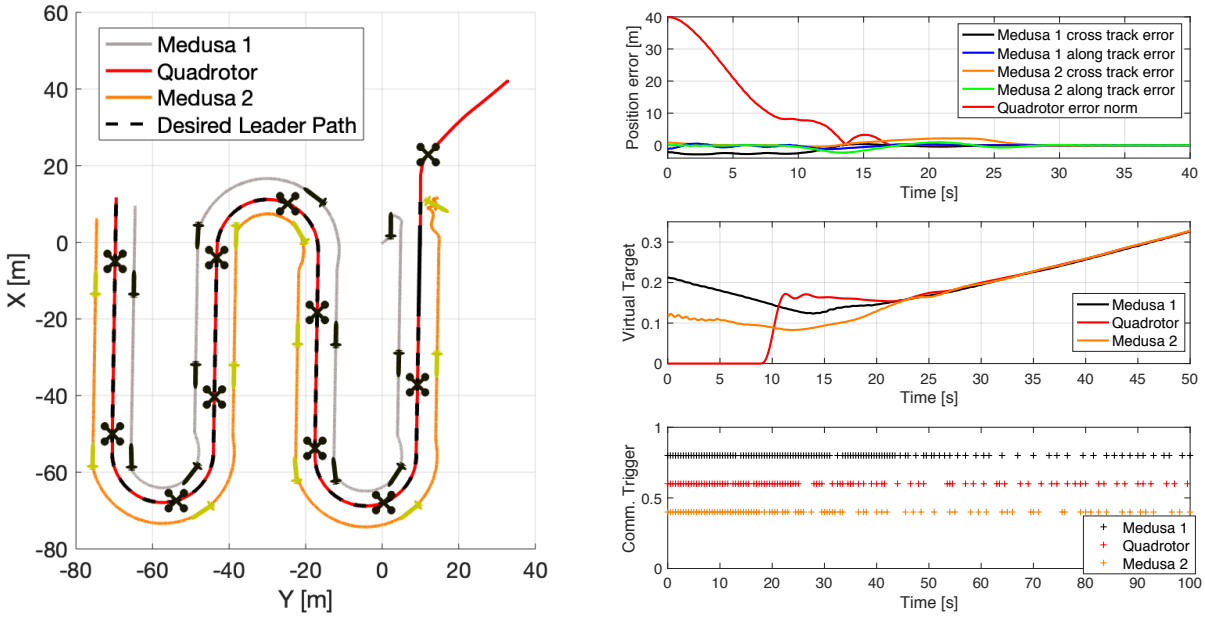


Figure 9.4: 3-D view of CPF betweeen quadrotor and 2 medusas (simulation)



(a) XY view

(b) Performance metrics

Figure 9.5: CPF between quadrotor and 2 medusas (simulation)

In this experiment there was bi-directional communication between the pairs of vehicles: (quadrotor, Medusa 1) and (quadrotor, Medusa 2). From the results obtained, it is observable that the vehicles converged to their desired formation after approximately 20s. After that period of time, the position error dropped to nearly zero for all vehicles, and the virtual target speeds converged to their desired constant value (along the line section). As a consequence, the number of communication events between the vehicles dropped as the bank of observers running in each vehicle could more accurately track the state of the virtual target of their peers.

9.1.5 Boundary Tracking with Quadrotor

Proven experimentally that the entire system was able to perform a CPF mission for a static pre-defined path, the next logical step was to test the path planning algorithm developed in chapter 7. In this scenario, the quadrotor was required to start the previous lawn-mowing PF mission, and as soon as it detected the environmental boundary, start re-planning in real time the desired path to follow, at a pre-defined height of 30m with a desired constant speed of 0.5m/s (Figure 9.6). The quadrotor was equipped with a camera locked at -45° in pitch angle, relative to it's body frame of reference. In Figure 9.7 a top view of the executed mission is provided along with the plots of the real distance of the vehicle to the boundary and the PF error.

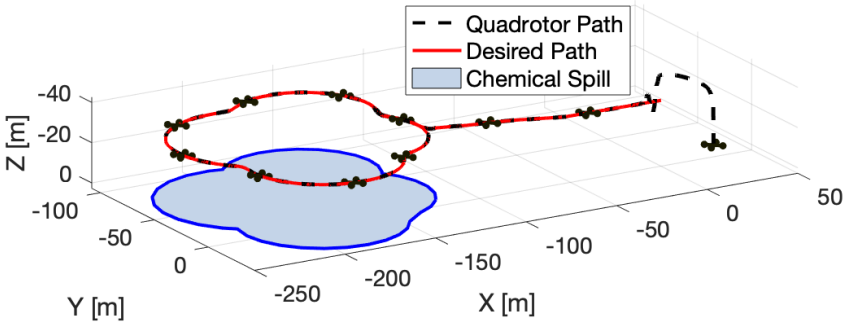
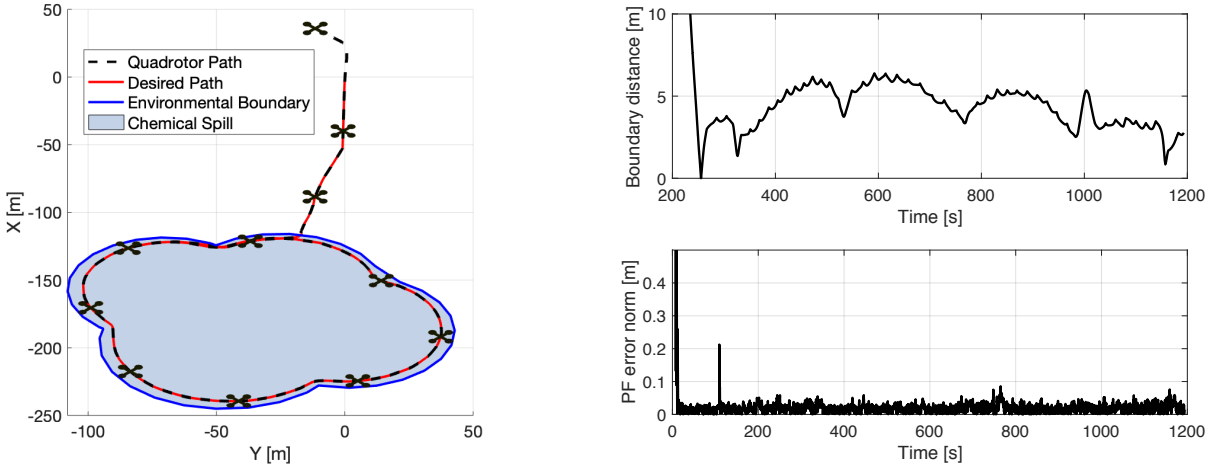


Figure 9.6: 3-D view of boundary tracking mission (simulation)



(a) XY view

(b) Performance metrics

Figure 9.7: Environmental boundary following (simulation)

In this simulation, the vehicle observed the boundary for the first time when at the position $\mathbf{p} = [-49.6, 0.1, -30]^T$m, where it started re-planing the path at a frequency $f = 1$Hz. As the desired path grew with the time, the drone was required to follow it, aligning its heading angle with the tangent to the path, guaranteeing that it never lost sight of the boundary being followed. From Figure 9.7 b) it is evident that the horizontal distance between the real vehicle position and the boundary is bounded by 7m. This result is to be expected due not only to the positioning filter of the aircraft not being perfect but also the simulated water not being located exactly at a constant plane $Z_U = 0$ as previously assumed (purposely done to add realism to the simulation). These errors were then propagated during the conversion of pixels in the image frame to a 2-D point cloud expressed in the inertial frame, leading to a point cloud that was offset by a few meters compared to the location of the real boundary. Moreover, small oscillations can be observed in the plot of the distance from the vehicle to the boundary. These originate from the simulated chemical spill not being smooth and having small creases resulting from the joining of small straight lines that compose the red mesh.

In Figure 9.8 a snapshot of the quadrotor's camera feed (used by the planning algorithm) is shown, when the vehicle was at position $\mathbf{p} = [-66, -4.3, -29.9]^T$m. On the other hand, in Figure 9.9 a plot of the point cloud generated by the algorithm is shown in two different time instants (red and blue dots), as well as the corresponding planned B-spline paths (black and green lines). Note that some of the red dots further away from the vehicle were discarded by the planning algorithm for being too far away from the main cluster of points.

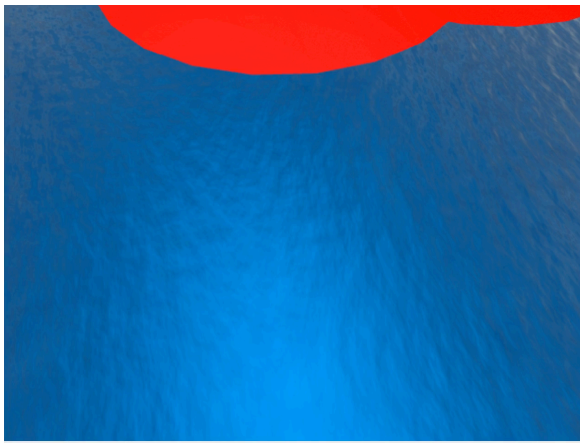


Figure 9.8: Camera image feed

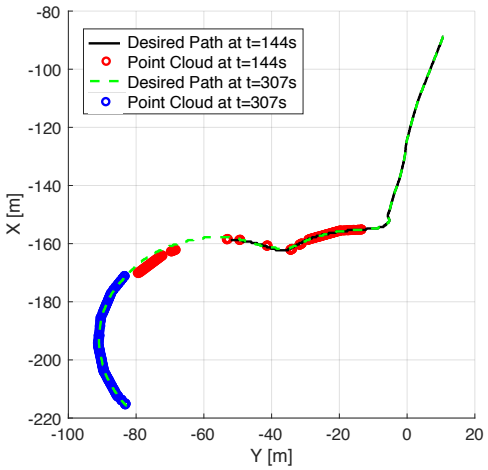


Figure 9.9: Real time path planning

9.1.6 Boundary Tracking with Quadrotor and a Medusa Vehicle

For the next experiment, it was required that the quadrotor performed the exact same mission while performing in parallel a CPF mission with a Medusa vehicle (Figure 9.10). From the previous results, it was observed that the path generated by the vehicle had an offset from the real boundary, such that the quadrotor was always moving "inside the chemical spill". In real life applications it might not be suitable for the marine vehicle to go inside the chemical spill, but rather around it. Moreover, it is desirable for the marine vehicle to always follow the aerial vehicle from behind and never in front of it, to guarantee

that the path further ahead can be generated. For that matter, a formation of vector $\mathbf{d} = [-5, 5, 0]^T$m was once again picked in order to manually compensate for the boundary estimation error. Communication was bi-directional between both simulated vehicles.

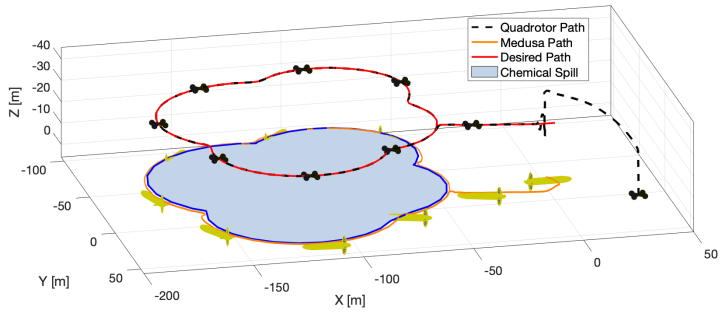
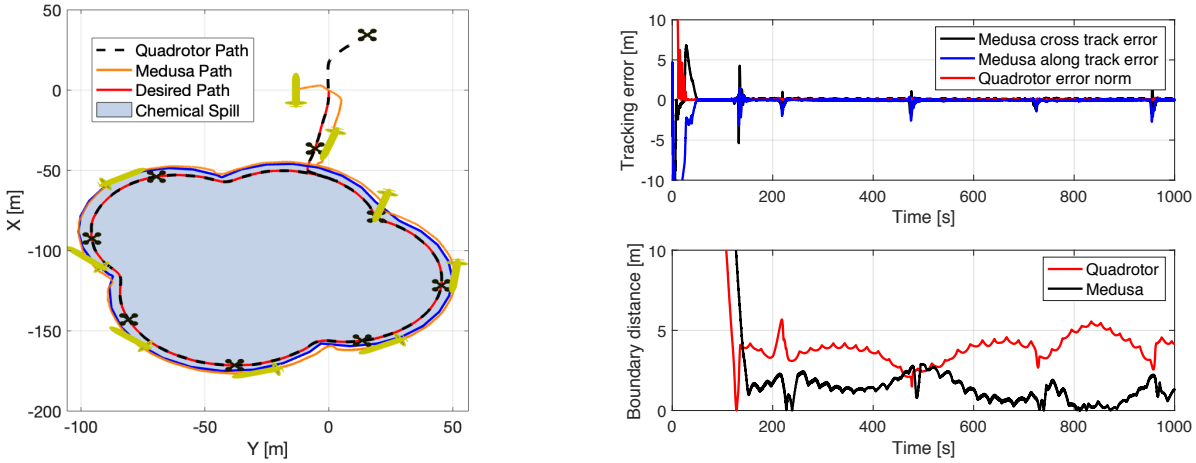


Figure 9.10: 3-D view of boundary following mission with Medusa vehicle (simulation)

In Figure 9.11 a) a top-down view of the executed mission is shown. In Figure 9.11 b) plots of the PF errors are provided along with the norm of the horizontal distance of each vehicle to the real boundary being followed. It is observable that the tracking error only increased in zones where the chemical spill had a crease. This is justified by the fact that the Medusa vehicle, when it has to perform tight turns, is not able to cope with its virtual target speed and slows down, leading to sudden spikes in along track error. These tracking errors were instantly compensated by the adaptive virtual target dynamics which attempted to minimize the distance between itself and the vehicle. It is also observable that the norm of the distance between the marine vehicle and the chemical spill is much lower, when compared to its aerial counterpart, with the Medusa always following the boundary from its outskirts.



(a) 2-D view

(b) Performance metrics

Figure 9.11: Environmental boundary following with Medusa vehicle (simulation)

Remark: A demonstration video of the conducted simulation is available online [76].

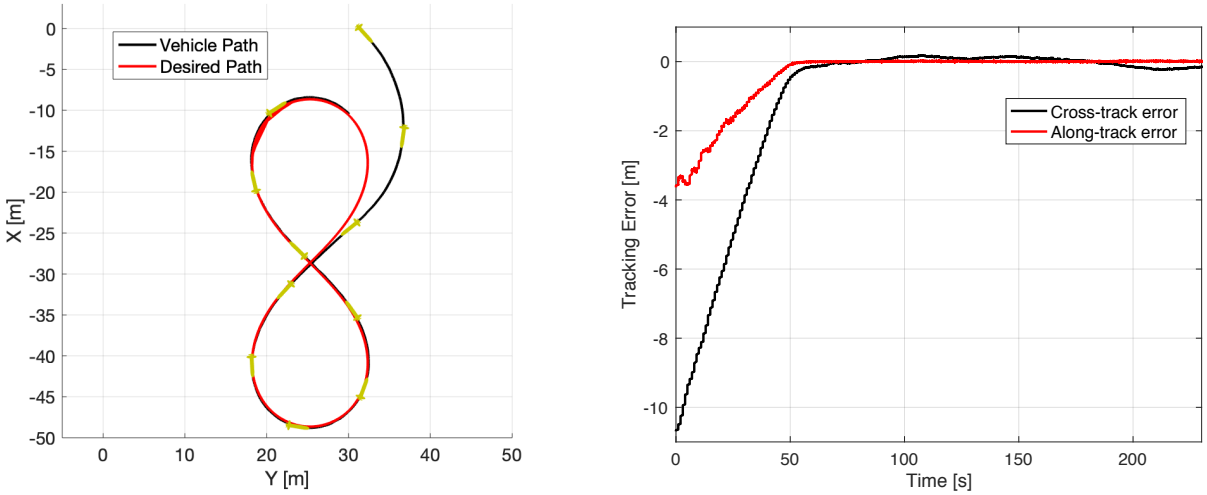
9.2 Real Trials

As described at the end of chapter 8, it was also possible to access the performance of the PF and CPF algorithms introduced in this work using two real Medusa vehicles. The following trials were conducted

at Doca dos Olivais (Lisbon, Portugal).

9.2.1 Medusa Path Following

In the first trial, it was requested for one vehicle to perform a Bernoulli's Lemniscate (Figure 9.12). From the results it is trivial to conclude that the vehicle exhibits a good performance, converging to the desired path. The cross-track and along-track errors converged to zero smoothly.



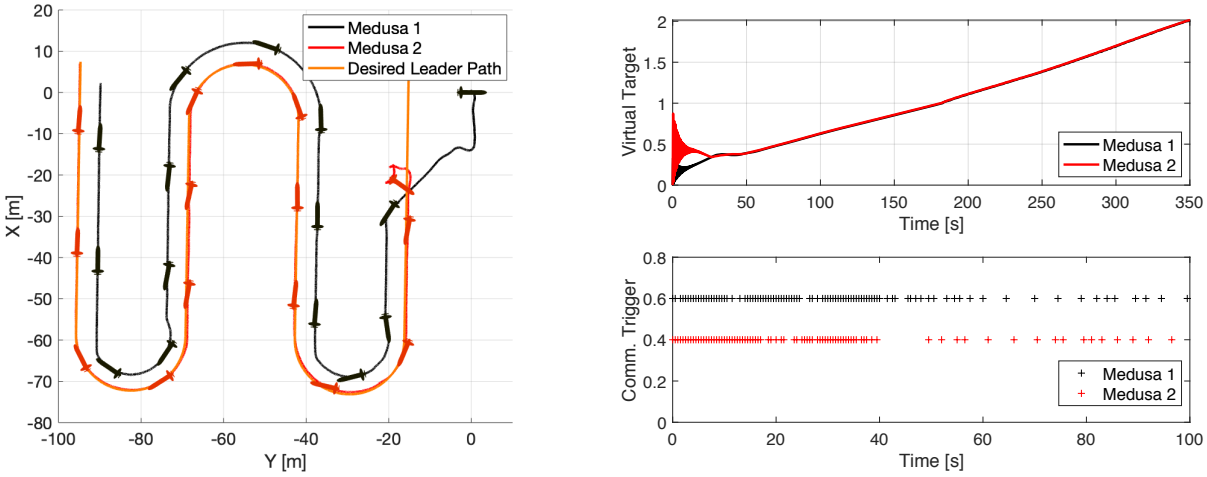
(a) Medusa path following XY view

(b) Tracking error

Figure 9.12: Medusa path following (real trial)

9.2.2 CPF with ETC between 2 Medusa Vehicles

For the second trial, it was requested that the two vehicles performed a lawn-mowing mission cooperatively (Figure 9.13). The black vehicle, Medusa 1, was required to follow a formation dictated by



(a) Medusa CPF XY view

(b) Triggering condition

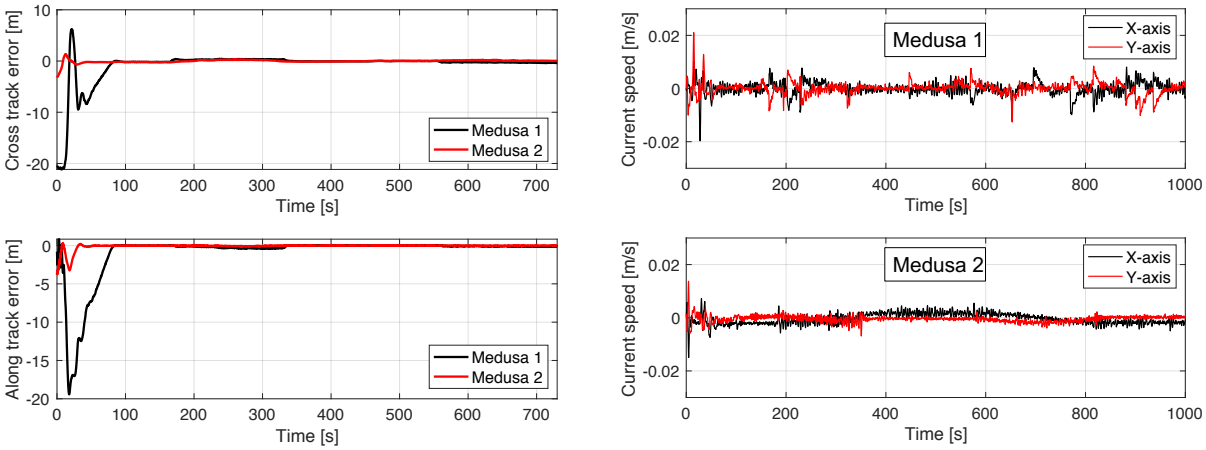
Figure 9.13: Medusa cooperative path following (real trial)

$\mathbf{d} = [-5, -5, 0]^T$m, with respect to the leader's path. There was bi-directional communication between both vehicles.

From the results obtained it can be seen that each virtual target rapidly increased to a value of approximately $\gamma \approx 0.4$ such that they got as close as possible to the original position of their respective vehicles.

Given that each curve segment was parameterized between $\gamma \in [0, 1]$, this corresponded roughly to the middle of the first line segment. Some oscillations were also observed in the beginning on the trial which resulted from the virtual targets finding an agreement between inter-vehicle alignment and intra-vehicle position error minimization. After approximately 50s, the vehicles aligned themselves into the required formation and started following the path at a constant speed of 0.3m/s. As a result, the rate of information exchange between the vehicle decreased after this period of time.

From the plots in Figure 9.14 a) it is possible to conclude that at $t = 100$s the cross-track and along-track errors of each vehicle converge to a neighbourhood of zero. During the day at which the trials were conducted there was negligible water current, which matches the observed results in Figure 9.14 b).



(a) Tracking error

(b) Currents Estimators

Figure 9.14: Tracking errors and currents observers (real trial)

Chapter 10

Conclusion

To sum up, this dissertation addressed the problem of tracking and following an environmental boundary caused by a chemical spill using a team of robots composed of one quadrotor and marine vehicles. A general overview of the state of the art regarding single vehicle path following and cooperative motion control was conducted. This literature overview was used to lay the foundations for the steps that followed: vehicle modelling, vehicle motion control, path following, cooperative path following and path planning.

In the vehicle modelling section, the notation and reference frames adopted for both the ASV and UAV were introduced. In the next section, the problem of vehicle inner-loop control was formulated and a set of linear control schemes were derived for both vehicles. Next, the PF problem was introduced, and a nonlinear control law derived for the ASV, according to the proposal by P. Aguiar and F. Vanni. Inspired by this control law, a new one was derived for a quadrotor following the same methodology with some key differences due to the nature of the aircraft.

For the section that followed, the CPF problem was formulated and a proposal to solve the problem was presented - a synchronization controller that is distributed and the same for all vehicles (aerial and marine). Borrowing from the works of A. Pascoal, N. Hung and F. Rego this last controller was generalized such that information exchange between vehicles would only be carried using even-triggered communications.

For the following chapter, a new real-time path planning algorithm was developed. This algorithm made use of a fixed camera sensor onboard of the quadrotor. This sensor was used to have a local view of an environmental boundary and generate a point cloud expressed in the inertial frame. This data was then used to solve an optimization problem which generated a B-spline based path that grows dynamically as the vehicle moves along the boundary and acquires more data. This path was then shared among all the ASV vehicles in the network.

Finally, the proposed algorithms were implemented in four main toolboxes by resorting to ROS, C++ and Python: a static path manager, a PF library, a CPF library and a dynamic path manager. These toolboxes were incorporated into the Medusa code base. Moreover, a 3-D virtual scenario that resembles Doca dos Olivais was also generated, allowing for realistic simulations of the proposed algorithms. In the end it was also possible to test experimentally the CPF algorithm using two real Medusa vehicles.

10.1 Future Work

In this work some problems were left unsolved. Some notable work that could be addressed includes:

- Considering event-triggered communication for CPF under network changing topologies and communication delays;
- Making the height at which the quadrotor operates dynamic;
- Developing a closed form solution for the B-spline fitting problem with linear constraints;
- Introducing curvature limits as inequality constraints of the fitting problems to cope with vehicle's limitations;
- Developing a model for controlling the orientation of the vehicle and camera that does not depend only on the tangent to the curve being followed;
- Introducing obstacle avoidance into the path planning problem;
- Testing the chemical spill tracking with a hybrid simulation where both the quadrotor and Medusa vehicle's are real, but the camera sensor is simulated in real time using 3-D simulation software.
- Developing a distributed path planning algorithm by taking advantage of sensors available in all vehicles.

Bibliography

[1] *Introduction to oceanography*. [Online]. Available: https://rwu.pressbooks.pub/webbooceanography/chapter/1-1-overview-of-the-oceans/ (visited on 10/08/2021).

[2] *Life's origins by land or sea - debate gets hot*. [Online]. Available: https://www.scientificamerican.com/article/lifes-origins-by-land-or-sea-debate-gets-hot/ (visited on 10/08/2021).

[3] T. Ghose, *Most ocean species remain undiscovered*. [Online]. Available: https://www.livescience.com/24805-undiscovered-marine-species.html (visited on 12/19/2020).

[4] Geoblueplanet, *Ocean resources*. [Online]. Available: https://geoblueplanet.org/ocean-resources/ (visited on 12/19/2020).

[5] O. society, *Our work*. [Online]. Available: https://www.oceanicsociety.org/our-work/ (visited on 12/19/2020).

[6] P. Maurya, A. P. Aguiar, and A. Pascoal, "Marine vehicle path following using inner-outer loop control", *IFAC Proceedings Volumes*, vol. 42, no. 18, pp. 38–43, 2009, 8th IFAC Conference on Manoeuvring and Control of Marine Craft, ISSN: 1474-6670.

[7] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor", *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[8] A. P. Aguiar and J. P. Hespanha, "Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty", *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1362–1379, 2007.

[9] L. Lapierre, D. Soetanto, and A. Pascoal, "Nonsingular path following control of a unicycle in the presence of parametric modelling uncertainties", *International Journal of Robust and Nonlinear Control*, vol. 16, pp. 485–504, Jul. 2006.

[10] Q. Zhang, J. Shippen, and B. Jones, "Robust backstepping and neural network control of a low-quality nonholonomic mobile robot", *International Journal of Machine Tools and Manufacture*, vol. 39, no. 7, pp. 1117–1134, 1999, ISSN: 0890-6955.

[11] N. Hung, F. Rego, J. Quintas, J. Cruz, M. Jacinto, L. Sebastiao, and A. Pascoal, "Theory, simulations, and experiments of path following guidance strategies for autonomous vehicles: Part I", *Robotics and Autonomous System (submitted)*, 2022.

[12] A. Lekkas and T. Fossen, “Line-of-sight guidance for path following of marine vehicles”, in *Advanced in Marine Robotics*. Lambert Academic Publishing, Jun. 2013, ch. 5, ISBN: 978-3659416897.

[13] A. T. Nugraha and T Agustinah, “Quadcopter path following control design using output feedback with command generator tracker LOS based at square path”, *Journal of Physics: Conference Series*, vol. 947, p. 012 074, Jan. 2018.

[14] A. Micaelli and C. Samson, “Trajectory tracking for two-steering-wheels mobile robots”, *IFAC Proceedings Volumes*, vol. 27, no. 14, pp. 249–256, 1994, Fourth IFAC Symposium on Robot Control, Capri, Italy, September 19-21, 1994, ISSN: 1474-6670.

[15] F. Vanni, A. P. Aguiar, and A. M. Pascoal, “Cooperative path-following of underactuated autonomous marine vehicles with logic-based communication”, *Proceedings Volumes*, vol. 41, no. 1, pp. 107–112, 2008, 2nd IFAC workshop on navigation, guidance and control of underwater vehicles, ISSN: 1474-6670.

[16] A. Alessandretti, A. P. Aguiar, and C. N. Jones, “Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control”, in *2013 European Control Conference (ECC)*, 2013, pp. 1371–1376.

[17] N. T. Hung, A. M. Pascoal, and T. A. Johansen, “Cooperative path following of constrained autonomous vehicles with model predictive control and event-triggered communications”, *International Journal of Robust and Nonlinear Control*, vol. 30, no. 7, pp. 2644–2670, 2020.

[18] W. Ren and E. Atkins, “Distributed multi-vehicle coordinated control via local information exchange”, *International Journal of Robust and Nonlinear Control*, vol. 17, pp. 1002–1033, 2007.

[19] R. Ghabcheloo, A. P. Aguiar, A. Pascoal, C. Silvestre, I. Kaminer, and J. Hespanha, “Coordinated path-following in the presence of communication losses and time delays”, *SIAM journal on control and optimization*, vol. 48, pp. 234–265, 2009.

[20] A. P. Aguiar and A. M. Pascoal, “Coordinated path-following control for nonlinear systems with logic-based communication”, in *2007 46th IEEE Conference on Decision and Control*, IEEE, 2007, pp. 1473–1479.

[21] F. C. Rego, A. P. Aguiar, and A. M. Pascoal, “A packet loss compliant logic-based communication algorithm for cooperative path-following control”, *IFAC Proceedings Volumes*, vol. 46, no. 33, pp. 262–267, 2013, 9th IFAC Conference on Control Applications in Marine Systems, ISSN: 1474-6670.

[22] F. Rego, H. Nguyen, C. Jones, A. Pascoal, and A. P. Aguiar, “Cooperative path-following control with logic-based communications: Theory and practice”, in IET Michael Faraday House, Stevenage, Apr. 2019, pp. 187–224, ISBN: 9781785613388.

[23] N. T. Hung and A. M. Pascoal, “Consensus/synchronisation of networked nonlinear multiple agent systems with event-triggered communications”, *International Journal of Control*, pp. 1–10, Nov. 2020.

[24] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain, “Cooperative forest fire surveillance using a team of small unmanned air vehicles”, *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[25] M. Fahad, N. Saul, Y. Guo, and B. Bingham, “Robotic simulation of dynamic plume tracking by unmanned surface vessels”, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2654–2659.

[26] L. Pettersson and D. Pozdnyakov, *Monitoring of harmful algal blooms*, First. Springer Science & Business Media, Jun. 2013, 309 pp, ISBN: 978-3-540-68209-7.

[27] A. Sukhinov, A. Chistyakov, A. Nikitina, A. Semenyakina, I. Korovin, and G. Schaefer, “Modelling of oil spill spread”, in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, IEEE, 2016, pp. 1134–1139.

[28] D. Saldaña, R. Assunção, M. A. Hsieh, M. F. M. Campos, and V. Kumar, “Cooperative prediction of time-varying boundaries with a team of robots”, in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, 2017, pp. 9–16.

[29] D. Saldaña, *Video of cooperative prediction of time-varying boundaries with a team of robots video*, Jul. 2017. [Online]. Available: `https://www.youtube.com/watch?v=Zwc6vNuUFDw` (visited on 08/23/2021).

[30] L. Piegl and W. Tiller, *The Nurbs Book*, 1st ed. Springer Science & Business Media, 1995, 646 pp, ISBN: 978-3-540-55069-3.

[31] D. C. Pedrosa, “Control-law for oil spill mitigation with a team of heterogeneous autonomous vehicles”, M.Sc. thesis, Instituto Superior de Engenharia do Porto, Jul. 2018.

[32] H. Khalil, *Nonlinear systems*, 3rd ed., ser. Always Learning. Pearson Education Limited, 2013, ISBN: 978-1-29203-921-3.

[33] Z. Cai, M. de Queiroz, and D. Dawson, “A sufficiently smooth projection operator”, *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 135–139, 2006.

[34] D. Cabecinhas, R. Cunha, and C. Silvestre, “A nonlinear quadrotor trajectory tracking controller with disturbance rejection”, *Control Engineering Practice*, vol. 26, pp. 1–10, May 2014.

[35] F. Bullo, *Lectures on network systems*, 1.5. Kindle Direct Publishing, 2021, ISBN: 978-1-98642-564-3.

[36] W. Ren and R. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control, Communications and Control Engineering*, 1st ed. Springer, London, 2008, vol. 27, ISBN: 978-1-84800-014-8.

[37] R. Winkel, “Generalized Bernstein polynomials and Bézier curves: An application of umbral calculus to computer aided geometric design”, *Advances in Applied Mathematics*, vol. 27, no. 1, pp. 51–81, 2001, ISSN: 01968858.

[38] *B-spline curves and surfaces*. [Online]. Available: `https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node15.html` (visited on 08/28/2021).

[39] *Types of splines: Ppform and b-form*. [Online]. Available: https://www.mathworks.com/help/curvefit/types-of-splines-ppform-and-b-form.html (visited on 10/21/2021).

[40] *B-spline basis functions: Important properties*. [Online]. Available: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-property.html (visited on 10/06/2021).

[41] C. de Boor, *A practical guide to splines*. Springer New York, Jan. 1978, vol. 27, ISBN: 978-0-387-95366-3.

[42] *Course 784 notes - The Ohio State University - CSE department*. [Online]. Available: https://web.cse.ohio-state.edu/~dey.8/course/784/notes.html (visited on 10/21/2021).

[43] *B-splines - online class*. [Online]. Available: https://www.youtube.com/watch?v=r6UcF0S0HvQ&list=PLcneaCnT6pzjDE15VPwYI33ve_a-YoAKf&index=26 (visited on 10/21/2021).

[44] *Uniform cubic b-spline curves - university of regina*. [Online]. Available: http://www2.cs.uregina.ca/~anima/408/Notes/Interpolation/UniformBSpline.htm (visited on 10/21/2021).

[45] S. Boyd, S. P. Boyd, and L. Vandenberghe, "Convex optimization", in Cambridge University Press, 2004, ISBN: 052-1-83378-7.

[46] M. Liu, S. Huang, G. Dissanayake, and S. Kodagoda, "Towards a consistent SLAM algorithm using B-splines to represent environments", in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2065–2070.

[47] D. Kraft, *A software package for sequential quadratic programming*, ser. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[48] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey,
bibinitperiodI. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python", *Nature Methods*, vol. 17, pp. 261–272, 2020.

[49] D. Teixeira, "Sensor-based cooperative control of multiple autonomous marine vehicles", M.Sc. thesis, Instituto Superior Técnico, Nov. 2019.

[50] T. Luukkonen, "Modelling and control of quadcopter", *Independent research project in applied mathematics, Espoo*, vol. 22, Aug. 2011.

[51] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*, 2nd ed. Wiley, Apr. 2021, ISBN: 978-1-119-57505-4.

[52] L. Trammell, *Building PID controls in software*. [Online]. Available: https://www.mstarlabs.com/apeng/techniques/pidsoftw.html.

[53] A. Pascoal, I. Kaminer, and P. Oliveira, “Navigation system design using time-varying complementary filters”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 4, pp. 1099–1114, 2000.

[54] G. Sanches, “Sensor-based formation control of autonomous marine robots”, M.Sc. thesis, Instituto Superior Técnico, Oct. 2015.

[55] L. Lapierre and D. Soetanto, “Nonlinear path-following control of an AUV”, *Ocean Engineering*, vol. 34, pp. 1734–1744, Aug. 2007.

[56] F. Vanni, A. P. Aguiar, and A. Pascoal, “Nonlinear motion control of multiple autonomous underwater vehicles”, *IFAC Proceedings Volumes*, vol. 40, no. 17, pp. 75 –80, 2007, 7th IFAC Conference on Control Applications in Marine Systems, ISSN: 1474-6670.

[57] A. P. Aguiar, R. Ghabcheloo, A. M. Pascoal, and C. Silvestre, “Coordinated Path-Following Control of Multiple Autonomous Underwater Vehicles”, International Ocean and Polar Engineering Conference, vol. All Days, Jul. 2007, ISOPE-I-07-006.

[58] A. P. Aguiar and A. M. Pascoal, “Coordinated path-following control for nonlinear systems with logic-based communication”, in *2007 46th IEEE Conference on Decision and Control*, IEEE, 2007, pp. 1473–1479.

[59] N. T. Hung, F. C. Rego, and A. M. Pascoal, “Event-triggered communications for the synchronization of nonlinear multi agent systems on weight-balanced digraphs”, in *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 2713–2718.

[60] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, Jan. 2011, vol. 5, ISBN: 978-1-84882-934-3.

[61] Y. Liu, H. Yang, and W. Wang, “Reconstructing B-spline curves from point clouds–a tangential flow approach using least squares minimization”, in *International Conference on Shape Modeling and Applications 2005 (SMI’ 05)*, IEEE, 2005, pp. 4–12.

[62] *Hackerearth - minimum spanning tree*. [Online]. Available: `https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/` (visited on 10/23/2021).

[63] J. L. Bentley, “Multidimensional binary search trees used for associative searching”, *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975, ISSN: 0001-0782.

[64] *Scikit learn - unsupervised learning, section 1.6.4.2. k-d tree*. [Online]. Available: `https://scikit-learn.org/stable/modules/neighbors.html` (visited on 08/31/2021).

[65] M. Jacinto, *Bsplinefit python library*. [Online]. Available: `https://github.com/MarceloJacinto/BSplineFit` (visited on 10/25/2021).

[66] W. Xie, D. Cabecinhas, R. Cunha, and C. Silvestre, “Cooperative path following control of multiple quadcopters with unknown external disturbances”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 1, pp. 667–679, 2020.

[67] R. T. Rodrigues, A. P. Aguiar, and A. Pascoal, “A coverage planner for AUVs using B-splines”, in *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, Nov. 2018, pp. 1–6.

[68] M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation", in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1–8.

[69] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Robot operating system (ROS): The complete reference (volume 1)", in A. Koubaa, Ed. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625, ISBN: 978-3-319-26054-9.

[70] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator", in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3.

[71] *ROS middleware*. [Online]. Available: https://www.ros.org (visited on 10/21/2021).

[72] G. Bradski, "The OpenCV library", *Dr. Dobb's Journal of Software Tools*, 2000.

[73] *Mavlink - micro air vehicle communication protocol*. [Online]. Available: https://mavlink.io/en/ (visited on 10/21/2021).

[74] *Mavros package*. [Online]. Available: http://wiki.ros.org/mavros (visited on 10/21/2021).

[75] T. Oliveira, P. Trindade, D. Cabecinhas, P. Batista, and R. Cunha, "Rapid development and prototyping environment for testing of unmanned aerial vehicles ", in *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2021, pp. 191–196.

[76] M. Jacinto, *Thesis demo video*. [Online]. Available: https://youtu.be/ax8q3wf5MYM (visited on 12/20/2021).

[77] J. Ribeiro, "Motion control of single and multiple autonomous marine vehicles", M.Sc. thesis, Instituto Superior Técnico, Oct. 2011.

[78] *Arducopter - Iris Quadcopter UAV*. [Online]. Available: http://www.arducopter.co.uk/iris-quadcopter-uav.html (visited on 08/31/2021).

[79] S. Flory, "Fitting B-spline curves to point clouds in the presence of obstacles", M.Sc. thesis, Vienna University of Technology, 2005.

Appendix A

MEDUSA-class vehicles

In this appendix, the Medusa vehicles are introduced. The main goal was to adapt the previously introduced motion models to this class of vehicles used for performing simulated and real water trials.

The Medusa autonomous marine vehicles are property of the Laboratory of Robotics and Systems in Engineering and Science (LARSyS)/Institute for Systems and Robotics (ISR) located at the Instituto Superior Técnico of Lisbon, Portugal. The physical dimensions of this class of robots are available in Figure A.1.

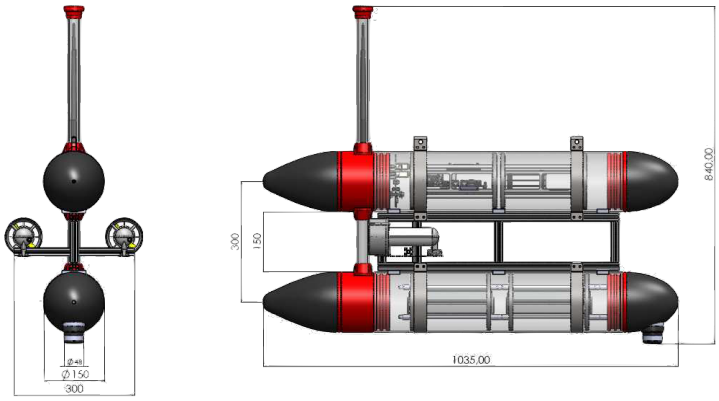


Figure A.1: MEDUSA-class of vehicles with dimensions in *mm* (from Ribeiro et al. (2011) [77])

As described in detail in Ribeiro et al. [77], each Medusa vehicle is composed of 2 acrylic tubes of sizes 0.15m by 1.035m (diameter x length) with aluminium end caps. These two tubes are attached to a central aluminium frame. The lower structure houses two packs of 7-cell lithium polymer batteries, thruster electronics, an acoustic modem and several sensors. The top unit contains the computer unit, along with a DGPS unit for absolute localization, an AHRS unit which provides precise angular velocity measurements as well as a DVL to acquire linear velocity measurements with respect to the water. It is also worth mentioning that these vehicles are highly modular and can accommodate different sensors, depending on the type of mission being executed.

Inter-vehicle communications are carried using Wi-Fi between different air or surface vehicles and via an acoustic modem network underwater.

For the surface version of the vehicles there are two thrusters, one on starboard and another on port side, which control directly the surge and yaw motions of the vehicle.

The identification of the model parameters for these vehicles was determined a priori by the research team at DSOR and can be consulted in table A.1.

Table A.1: MEDUSA-class of vehicles parameters (adapted from [49])

$X_{\dot{u}}$	-20 kg	$Y_{\dot{v}}$	-30 kg	$N_{\dot{r}}$	-0.5 kg m^2
X_u	-0.2 kg/s	Y_v	-55.1 kg/s	N_r	-4.14 kg m/s
$X_{\|u\|u}$	-25 kg/s	$Y_{\|v\|v}$	0.01 kg/m	$N_{\|r\|r}$	-6.23 kg m

The vehicle weights approximately $m = 30$kg and the moment of inertia about the z-axis is approximately $I_z = 1$Kgm^2.

The two thrusters can be used in a combination of common mode and differential mode to control the surge and yaw motion respectively. In particular the external force in surge τ_u and external torque about the Z-axis τ_r can be mapped by

$$\begin{aligned} \tau_u &= F_s + F_p; \\ \tau_r &= l(F_s - F_p), \end{aligned} \tag{A.1}$$

where F_s is the starboard relative force, F_p is the port side relative force and $l = 0.15$m is the length of the arm that connect the thrusters to the vehicle. Each thruster is also characterized by a thrust curve which follows the equation

$$F(F_{in}) = a|F_{in}|F_{in}, \tag{A.2}$$

with $a = 0.0036$, and the input $F_{in} \in [-100, 100]$ being normalized input used by the Medusa motor allocation driver. Moreover, each motor is characterized as a first order system (pole + delay), described by

$$G(s) = \frac{K_0}{s + K_0} e^{-s\tau}, \tag{A.3}$$

with $K_0 = 7.21$ a constant gain and $\tau = 0.346$s a delay, according to Figure A.2.

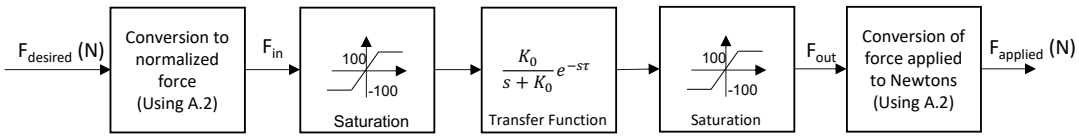


Figure A.2: Thruster Model

Appendix B

3DR Iris Quadrotor

In this appendix, the Iris quadrotor used in simulation is introduced. The 3DR Iris quadrotor in Figure B.1 is a commercial vehicle developed and manufactured by 3D robotics (3DR). Its CAD model is open source and a gazebo simulation model is available through PX4 SITL Plugin [69].

Figure B.1: Iris quadrotor (adapted from Arducopter [78])

The vehicle has an X-shaped frame with 4 rotors. Its housing contains several sensors such as: barometer, magnetometer, Inertial Measurement Unit (IMU) and DGPS. It weights approximately $m = 1.5$kg and its moments of inertia are $I_x = 0.029$kgm^2, $I_y = 0.029$kgm^2 and $I_z = 0.055$kgm^2, about the X, Y and Z-axis respectively. The simulated vehicle is also characterized by a total thrust curve which follows a quadratic of the form $T(T_{in}) = aT_{in}^2 + bT_{in}$, with $a = 34$, $b = 7.2$, with the input $T_{in} \in [0, 1]$ being a normalized input used by the quadrotors motor mixer.

In simulation, the vehicle was also equipped with a fixed First Person View (FPV) camera mounted 21mm below the vehicle's center of mass with a pitch angle of $-45°$, pointing downwards. This camera produces an image with a resolution of 640×480px. The camera intrinsic parameters (see Section 7.1) are given by

$$K = \begin{bmatrix} 381.4 & 0 & 320.5 \\ 0 & 381.4 & 240.5 \\ 0 & 0 & 1 \end{bmatrix}. \tag{B.1}$$

Appendix C

Uniform Cubic B-Splines

C.1 Expanding to the 2-Dimensional Case

Consider the unidimensional uniform cubic B-Spline model (with $n-k+1$ segments) introduced in section 2.4.2 given by

$$C_i(\gamma) := \underbrace{\frac{1}{6}\begin{bmatrix}(\gamma-i)^3 & (\gamma-i)^2 & (\gamma-i) & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}}_{\begin{bmatrix}B_{i,3}(\gamma) & B_{i+1,3}(\gamma) & B_{i+2,3}(\gamma) & B_{i+3,3}(\gamma)\end{bmatrix}}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix} \tag{C.1}$$

$$\Leftrightarrow C_i(\gamma) = \begin{bmatrix}B_{i,3}(\gamma) & B_{i+1,3}(\gamma) & B_{i+2,3}(\gamma) & B_{i+3,3}(\gamma)\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ P_{i+2}\\ P_{i+3}\end{bmatrix}$$

where $\gamma \in [0, n-k+1)$ and $i := \lfloor\gamma\rfloor$, such that $\gamma - i \in [0,1)$. An analogous representation for a 2-Dimensional curve, i.e. $\mathbf{C}(\gamma) \in \mathbb{R}^2$ is given by

$$\mathbf{C}(\gamma) := \underbrace{\begin{bmatrix}B_{i,3}(\gamma) & B_{i+1,3}(\gamma) & B_{i+2,3}(\gamma) & B_{i+3,3}(\gamma) & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & B_{i,3}(\gamma) & B_{i+1,3}(\gamma) & B_{i+2,3}(\gamma) & B_{i+3,3}(\gamma)\end{bmatrix}}_{B(\gamma)}\underbrace{\begin{bmatrix}P_i^x\\ P_{i+1}^x\\ P_{i+2}^x\\ P_{i+3}^x\\ P_i^y\\ P_{i+1}^y\\ P_{i+2}^y\\ P_{i+3}^y\end{bmatrix}}_{\mathbf{P}} \tag{C.2}$$

where the basis matrix $B(\gamma)$ is diagonal by blocks and the vector of control points $\mathbf{P}$ is structured such that the first half is given by the X-coordinates of each point and second half by the corresponding Y-coordinates. Let us now define a vector of control points $\mathbf{P} = [P_0^x, ..., P_n^x, P_0^y, ..., P_n^y]^T \in \mathbb{R}^{2(n+1)}$, a vector of distinct curve parameters $\boldsymbol{\gamma} = [\gamma_0, ..., \gamma_q]^T \in \mathbb{R}^{q+1}$ that we wish to evaluate our curve at, and $\mathbf{C}(\boldsymbol{\gamma}) = [C_0^x, ..., C_q^x, C_0^y, ..., C_q^y]^T \in \mathbb{R}^{2(q+1)}$ the points on the curve. Then the basis matrix becomes

$$B(\boldsymbol{\gamma}) = \begin{bmatrix} B_{0,3}(\gamma_0) & \dots & B_{n,3}(\gamma_0) & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ B_{0,3}(\gamma_q) & \dots & B_{n,3}(\gamma_q) & 0 & \dots & 0 \\ 0 & \dots & 0 & B_{0,3}(\gamma_0) & \dots & B_{n,3}(\gamma_0) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & B_{0,3}(\gamma_q) & \dots & B_{n,3}(\gamma_q) \end{bmatrix}, \tag{C.3}$$

a matrix diagonal by blocks. This representation is particularly useful due not only to its efficiency when implemented in practice, but also to the flexibility it provides for adding extra dimensions to the problem.

C.2 Integral Calculation

In this appendix section, an efficient method for computing the integral terms proposed in the regularization function (2.48) is shown. For the sake of simplicity, start by considering the simplest unidimensional uniform cubic B-spline curve $C(\gamma) \in \mathbb{R}$ with only one segment, such that $\gamma \in [0, 1)$ and described by:

$$C(\gamma) = \underbrace{\frac{1}{6}\begin{bmatrix}\gamma^3 & \gamma^2 & \gamma & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0\end{bmatrix}}_{B(\gamma)}\underbrace{\begin{bmatrix}P_0 \\ P_1 \\ P_2 \\ P_3\end{bmatrix}}_{\mathbf{P}}, \tag{C.4}$$

and with first derivative $C'(\gamma)$ given by:

$$C'(\gamma) = \underbrace{\underbrace{\begin{bmatrix}\gamma^2 & \gamma & 1 & 0\end{bmatrix}}_{\mathbf{T}(\gamma)}\frac{1}{6}\underbrace{\begin{bmatrix}3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0\end{bmatrix}}_{M}\begin{bmatrix}P_0 \\ P_1 \\ P_2 \\ P_3\end{bmatrix}}_{B'(\gamma)}. \tag{C.5}$$

Therefore, the term $\int_\gamma \|C'(\gamma)\|^2 d\gamma$ is computed according to:

$$\begin{aligned}\int_\gamma \|C'(\gamma)\|^2 d\gamma &= \int_\gamma (B'(\gamma)\mathbf{P})^T (B'(\gamma)\mathbf{P}) d\gamma \\ &= \int_0^1 \mathbf{P}^T B'(\gamma)^T B'(\gamma)\mathbf{P} d\gamma \\ &= \mathbf{P}^T \left[\int_0^1 B'(\gamma)^T B'(\gamma) d\gamma\right]\mathbf{P} \\ &= \mathbf{P}^T \left[\int_0^1 M^T \mathbf{T}(\gamma)^T \mathbf{T}(\gamma) M d\gamma\right]\mathbf{P} \\ &= \mathbf{P}^T M^T \left[\int_0^1 \mathbf{T}(\gamma)^T \mathbf{T}(\gamma) d\gamma\right] M\mathbf{P}.\end{aligned} \tag{C.6}$$

Further note that:

$$\mathbf{T}(\gamma)^T \mathbf{T}(\gamma) = \begin{bmatrix} t^4 & t^3 & t^2 & 0 \\ t^3 & t^2 & t & 0 \\ t^2 & t & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{C.7}$$

and as a consequence:

$$\int_0^1 \mathbf{T}(\gamma)^T \mathbf{T}(\gamma) d\gamma = Q = \begin{bmatrix} 1/5 & 1/4 & 1/3 & 0 \\ 1/4 & 1/3 & 1/2 & 0 \\ 1/3 & 1/2 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{C.8}$$

Hence, for the simplest case of a single B-spline segment it is known that

$$\int_\gamma \|C'(\gamma)\|^2 d\gamma = \mathbf{P}^T \underbrace{M^T Q M}_{R_1} \mathbf{P}. \tag{C.9}$$

The easiest way to extend this technique to a B-spline with n segments, it to consider the modified vector $\mathbf{T}(\gamma) = [(\gamma - i)^2, (\gamma - i), 1, 0]^T$, where $i = \lfloor \gamma \rfloor$, according to the notation introduced in section 2.4.2. Then, since $(\gamma - i) \in [0, 1)$, one can compute individually for each segment intermediate matrices R_1^i, calculated according to (C.9). Due to the locality property of B-splines, one can just "stack" these intermediate matrices to form the final matrix R_1, where the values that "overlap" are summed (Figure C.1).

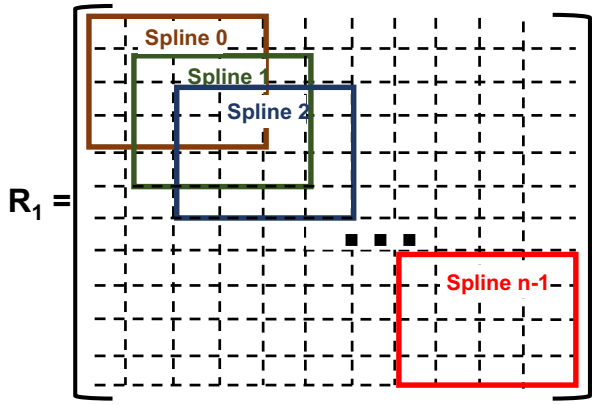


Figure C.1: Computing matrix R_1 for B-spline with n segments

An analogous rationale can be applied to compute the integral of the norm of the second derivative.

Remark: For a more general (but less efficient) formula that can be applied to B-Splines of any degree, the reader is referred to [79], pp.15.

Appendix D

Path Planning Performance

In order to access the performance of the path planning algorithm developed in chapter 7, the artificial test in Figure D.1 was developed. In this example, the time taken for the re-planning of the curve (with the end goal of fitting the points in red) is evaluated. By varying the number of points in red, the performance plot in Figure D.2 was generated. It is evident from the results obtained that the pre-processing stage consumes most of the planning time.

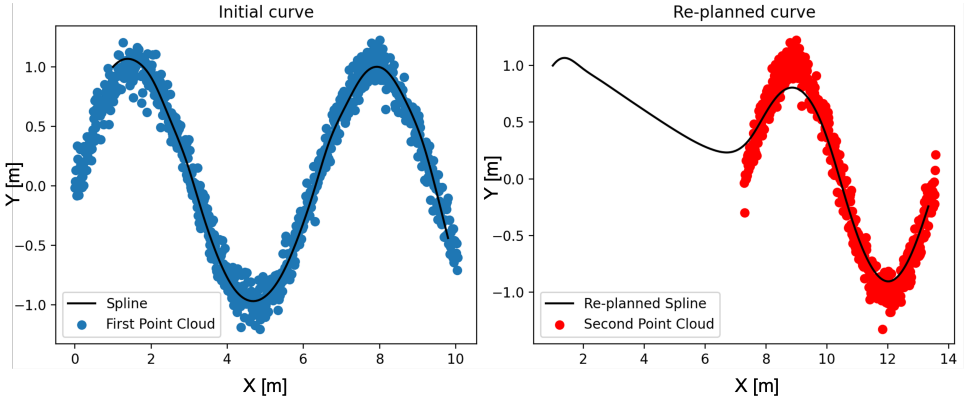


Figure D.1: Path used for statistics

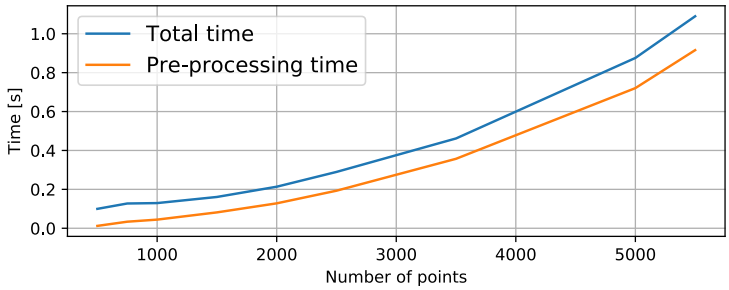


Figure D.2: Path planning algorithm performance

The results presented in this performance analysis were obtained using a MacBook Pro (early 2015 model) with a 2,7GHz Intel Core i5. The parameters used for the analysis are available in table D.1. For comparison, the typical point cloud produced by the quadrotor vision system had an average of 1500 points.

Table D.1: Parameters adopted for performance analysis

Path Planning	
Max. neighbour distance N_m	0.2 m
Number of control points	26
Regularization term λ	0.8
Regularization term β	1.2

Appendix E

Code API

E.1 Paths Package

```
class PathSection {
  public:
    /* Methods to get the position, first and second derivatives given the
    parameter gamma - Must be implemented by every path section individually */
    virtual Eigen::Vector3d eq_pd(double t) = 0;
    virtual Eigen::Vector3d eq_d_pd(double t) = 0;
    virtual Eigen::Vector3d eq_dd_pd(double t) = 0;

    /* Methods to get the tangent angle, curvature and derivative norm to the path
    at gamma */
    virtual double tangent(double t);
    virtual double curvature(double t);
    virtual double derivative_norm(double t);

    /* Method to get the closest point on the path given the vehicle's coordinates
    */
    virtual double getClosestPointGamma(Eigen::Vector3d &coordinate);
    bool can_be_composed();

    /* Auxiliar methods to retrieve path limits */
    double limitGamma(double t);
    double getMaxGammaValue();
    double getMinGammaValue();

    virtual ~PathSection();

  protected:
  /* Class constructor */
    PathSection(bool can_be_composed);

    /* Methods to define the path section limits */
    bool setMaxGammaValue(double gamma_max);
```

```
    bool setMinGammaValue(double gamma_min);

  private:

    /* Min and Max values for gamma */
    double max_value_gamma_{std::numeric_limits<double>::max() / 2};
    double min_value_gamma_{std::numeric_limits<double>::lowest() / 2};

  /* section can be used in a composition of sections */
    bool can_be_composed_{true};
};
```

Listing E.1: PathSection abstract class

```
  Header header           # Message time, ID and other metadata
  float64[3] pd           # Path Position
  float64[3] d_pd         # First Derivative
  float64[3] dd_pd        # Second Derivative
  float64 curvature       # Curvature of the Path
  float64 tangent_angle   # Angle of Tangent to the Path
  float64 derivative_norm # Norm of the derivative
  float64 vd              # Desired speed for virtual target
  float64 d_vd            # Desired acceleration for virtual target
  float64 vehicle_speed   # Desired vehicle speed
  float64 gamma_min       # Min Path parameter
  float64 gamma_max       # Max Path parameter
```

Listing E.2: Path data messages exchanged with Path Following Controller

E.2 Path Following Package

```
class PathFollowing {
  public:

    /* Class destructor for the abstract pathfollowing class */
    virtual ~PathFollowing();

    /* Method to update the path following control law */
    virtual void callPFController(double dt) = 0;

    /* Method to publish the control values given by the algorithm */
    void publish();
    virtual void publishPrivate() = 0;

    /* Method used to setup the algorithm in the first iteration */
    virtual void start() = 0;

    /* Method used to check whether we have reached the end of the path */
```

```
18      virtual bool stop() = 0;
19
20      /* Method used to reset the algorithm control parameters */
21      virtual bool reset() = 0;
22
23      /* Method to tune the gains while running in real time */
24      virtual bool setPFGains(std::vector<double> gains) = 0;
25
26      /* Method to update the current vehicle state */
27      void UpdateVehicleState(const VehicleState &vehicle_state);
28
29      /* Method to update the path data */
30      void UpdatePathState(const PathState &path_state);
31
32      /* Method used for publishing performance metrics */
33      void setPFollowingDebugPublisher(const ros::Publisher &pfollowing_debug_pub);
34
35   protected:
36
37      /* Variable to store the state of the vehicle */
38      VehicleState vehicle_state_;
39
40      /* Variable to store the state of the path */
41      PathState path_state_;
42
43      /* Variable to store the performance metrics */
44      PFollowingDebug pfollowing_debug_;
45      ros::Publisher pfollowing_debug_pub_;
46  };
```

Listing E.3: PathFollowing abstract class

Appendix F

Controller Gains and Parameters

The controller gains used to obtain the results in chapter 9 are presented in tables F.1, F.2, F.3 and F.4.

Table F.1: ASV controller gains

ASV		
Inner-loops		
Surge	k_p	0.6
	k_i	0.099
Yaw-rate	k_p	0.32
	k_i	0.088
Currents Observer		
k_1		2.0
k_2		0.2
Path Following		
K_p		$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
k_γ		0.5
δ		-1.0

Table F.2: UAV quadrotor controller gains

Quadrotor	
Path Following	
K_p	$\begin{bmatrix} 5.5 & 0 & 0 \\ 0 & 5.5 & 0 \\ 0 & 0 & 5.5 \end{bmatrix}$
K_v	$\begin{bmatrix} 4.5 & 0 & 0 \\ 0 & 4.5 & 0 \\ 0 & 0 & 4.0 \end{bmatrix}$
k_γ	0.5
Projection Operator	
K_d	$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$
ε	10.0
δ	10.0
d_{max}	8.0

Table F.3: CPF gains

Cooperative Path Following	
k_ε	1.0
c	0.001
b	5.0
α	1.0

Table F.4: Path planning parameters

Path Planning	
Max. Angular velocity ω_{max}	0.1s^{-1}
Max. neighbour distance N_m	0.6 m
Control points density $1/\rho$	4.0
Radius (optional step) r	0.3 m
Regularization term λ	0.05
Regularization term β	0.01
Desired altitude	-30 m
Re-planning frequency f	1Hz