M 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

COOPERATIVE MOTION CONTROL OF A FORMATION OF UAVs

RÔMULO TEIXEIRA RODRIGUES

DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
AUTOMAÇÃO E ROBÓTICA

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Cooperative Motion Control of a Formation of UAVs"

foi aprovada em provas realizadas em 24-07-2014

o júri

Presidente Professor Doutor António Paulo Gomes Mendes Moreira
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Manuel Fernandes dos Santos Silva
Professor Adjunto do Departamento de Engenharia Eletrotécnica da Instituto Superior de Engenharia do Porto

Professor Doutor António Pedro Rodrigues Aguiar
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Rômulo Teixeira Rodrigues

Faculdade de Engenharia da Universidade do Porto

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Cooperative Motion Control of a Formation of UAVs

Rômulo Teixeira Rodrigues

Dissertation

Advisor: Dr. António Pedro Aguiar

Co-advisor: Dr. Marcelo Becker

July 31, 2014

Abstract

The recent advances in electronics and communication networks have aroused interest in the research community towards cooperative motion control of multiple autonomous robotic vehicles. There are many scenarios where employing a fleet of small, scalable and inexpensive vehicles is more attractive than using a single expensive robot. In the literature the topic is addressed for mobile robots, autonomous underwater vehicles (AUVs), autonomous surface vehicles (ASVs), autonomous aerial vehicles (UAVs) and other robots. Specifically, the formation of UAVs is an asset, as the UAV technology grows stronger in society.

In this dissertation we address cooperative motion control problem for UAVs that unravels in two tasks: path-following and coordination control. The former requires the vehicle to converge and follow a desired path with no temporal constraints. The latter coordinates the elements in a fleet to travel on a desired pattern.

The strategy adopted for the path following unfolds the problem in a geometric and speed assignment task. The vehicle is permanently following a virtual target point (VTP), which moves on the desired path. Adjusting the speed of the target, synchronization is accomplished. Nonlinear techniques enable to explicitly take into account the nonlinearities inherent to the model. Graph theory describes the inter-vehicle communication topology.

In order to validate the adopted strategies, a guidance, navigation and control (GNC) evaluation environment for Flight Variables Management System (FVMS) is developed. The tool, herein developed, may be as well used to evaluate other GNC algorithms in a reliable manner.

Numerical simulations and software-in-the-loop (SiL) data evaluate the methods addressed. The results show that the nonlinear Lyapunov based control law proposed correctly solves the path following problem. The performance is comparable to other well-established solutions. Moreover, coordination in a switching topology communication is achieved.

Keywords: Coordination, Nonlinear path following, Cooperative formation, Unmanned aerial vehicles, Flight Variables Management System, Software-in-the-loop.

Resumo

Os avanços recentes na eletrónica e redes de computadores têm despertado interesse na comunidade científica quanto ao controlo de movimento cooperativo de múltiplos veículos robóticos autónomos. Existem várias situações em que a utilização de um conjunto de veículos pequenos, escaláveis e de baixo custo é vantajosa relativamente ao uso de um único robô de alto custo. Na literatura este tópico é abordado para robôs móveis, veículos submarinos autónomos (AUVs - Autonomous Underwater Vehicle), veículos de superfície autónomos (ASVs – Autonomous Surface Vehicle), veículos aéreos autónomos (UAVs – Unmanned Aerial Vehicle) e outros robôs. Particularmente, a formação de UAVs é um foco de interesse, uma vez que se tem observado uma expansão dos UAVs na sociedade.

Esta dissertação foca o problema de controlo de movimento cooperativo de UAVs que é dividido em duas tarefas: o seguimento de caminhos e o controlo de coordenação. O seguimento de caminhos tem por objetivo que o veículo convirja para um percurso desejado e o siga, sem quaisquer restrições temporais. A segunda tarefa trata de coordenar os elementos de um conjunto de veículos para que estes se movimentem de acordo com um padrão desejado.

A estratégia adotada para o seguimento de caminhos divide o problema em duas componentes, nomeadamente uma tarefa geométrica e uma tarefa de atribuição de velocidade. O veículo segue constantemente um alvo virtual (VTP – Virtual Target Point), que se move no percurso desejado. Ao ajustar a velocidade do alvo é possível obter sincronização. Técnicas não-lineares permitem considerar não-linearidades inerentes ao modelo de forma explícita. A topologia de comunicação entre veículos é descrita através de teoria de grafos.

A fim de validar as estratégias adotadas, um ambiente de seguimento, navegação e controlo (GNC – Guidance, Navigation and Control) é desenvolvido para um sistema de geral que relaciona as várias variáveis de voo (FVMS - Flight Variables Management System). A ferramenta desenvolvida pode também ser usada para avaliar outros algoritmos GNC de forma fidedigna.

Simulações numéricas e dados software-in-the-loop (SiL) são usados para avaliar os métodos abordados. Segundo os resultados, a lei de controlo baseada em Lyapunov não-linear proposta resolve corretamente o problema de seguimento de caminhos. Para além de a sua performance ser comparável à de outras soluções bem estabelecidas, é conseguida a coordenação numa topologia alternada no tempo.

Palavras-chave: Coordenação, Seguimento, Formação cooperativa, Veículos aéreos não-tripulados, Flight Variables Management System, Software-in-the-loop .

Acknowledgments

I am very grateful for all my friends and relatives who assisted me along the years. Special thanks to

Prof. Antonio Pedro Aguiar for offering me the chance to research beyond regular lectures. Despite the distance, he was always available to help and guide me.

Prof. Marcelo Becker and Rafael Coronel for letting me join their team. Their solicitude before and during this journey was a comfort.

LSTS/FEUP team for the support and field test invitation.

the friends I collected in these five years (hopefully our friendship will last for the next fifty): Tiago, Nuno, Gilbert, Antony, Hugo and, more recently, Rafael. Thanks for receiving me in your home and families. For sure, these years would not had been as enjoyable without our discussions, laughs and coffee breaks.

to Ana Rita for supporting and taking care of me through this journey. Also to her family for welcoming me in their home and sharing their time with me.

to my dad, mom and sisters whose unconditional love gives me strength for improving daily.

Contents

List of Figures

1. Introduction

This chapter introduces the topic of the dissertation which is cooperative control of a formation of unmanned aerial vehicles. Its relevance in the modern society and some open challenges are addressed in Section 1.1. Afterwards, in Section 1.2 the problem to be solved in the dissertation is formulated. Then, Section 1.3 presents a collection of previous work related to the topic and the contribution of this work. Section 1.4 introduces a brief scope of the document.

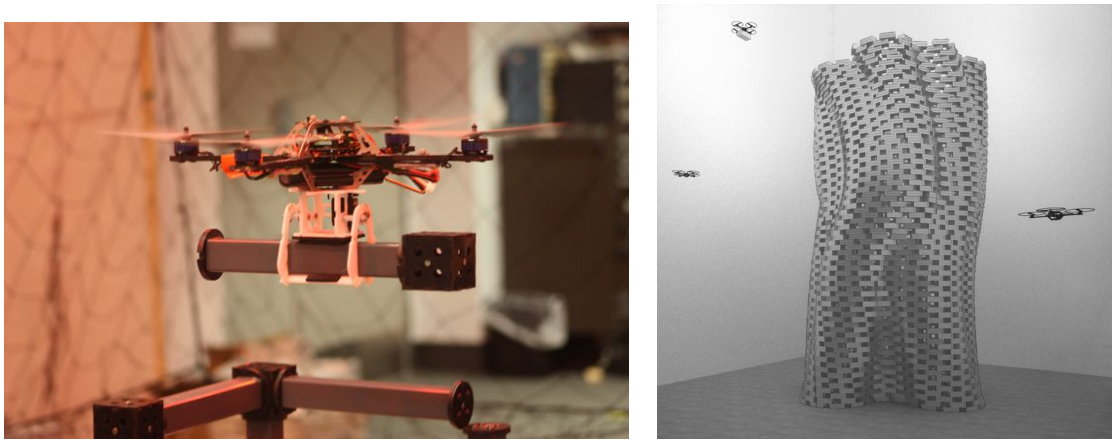
1.1 Motivation

Unmanned Aerial Vehicles (UAVs) are no longer science fiction but an expanding technology. It may be employed in a wide range of applications, from border patrolling and fire detection to aerobiological sampling and crop monitoring. These platforms do not only prevent human pilots from hazardous situations, but they are also a cheap and reliable solution when contrasted to other manned vehicles. It is not a fortuity, UAVs are robotics industry fastest growing market segment (Cheng and Kumar, 2008). The military UAV market will consume about $61.37 billion between 2011 and 2020 (Roundup, 2013). The civilian market as well is expect to increase, considering the growth of its usage among universities, environmental entities and media companies. In 2013, there were 57 countries producing more than 960 distinct UAVs (Roundup, 2013). Each year this number grows, as nations develop their own indigenous aircraft to comply with specific requirements.

The popularization of UAVs and recent advents in communication networks, renewed interest in UAV Coordinated Control. A single vehicle may be enough in a simple application. However, the success of more challenging missions requires the employment of multiple vehicles working in cooperation towards the same goal. This concept is based on the advantages of distributed systems, such as robustness, flexibility and scalability, which endow a fleet of simple and cheap vehicles to perform tasks that are not feasible for an expensive single unit. For instance, inexpensive sensors can be distributed along the vehicles and data fusion occurs periodically or sporadically. A set of possible applications is shown next.

Aerial Robotic Construction

Over the past few decades, land robots have populated modern industries. However, they have a limited working area due to their rigid body constraints. On the other hand, aerial vehicles can operate freely in space. Taking advantages of such system, a new field named *Aerial Robotic Construction* (ARC) emerged. It was first described in (Lindsey et al., 2011). In this pioneering work, a set of UAVs capable of assembling three-dimensional small scale structures, like the ones for building high-voltage towers and skyscrapers, were developed. The pieces were specially manufactured to be transported by quadcopters.



(a) Hummingbird quadcopter. source:(Lindsey et al., 2011) (b) Tower. source:(Willmann et al., 2012)

Figure 1.1: Humming bird quadcopter holding a structure module (a). The 6 meter high tower assembled by four UAVs (b).

A especially interesting ARC demonstration is discussed in (Willmann et al., 2012). The paper illustrates the process of a prototype 6 meter high tower with 1500-module built by a fleet of autonomous vehicles, exposed in Orléans, France. The authors highlight three important advantages of ARC. First, possibility to fly and mount the pieces directly in their position. Second, structures can be built with complex designs. Third, but not last, ARC work capacity is scalable. In other words, one or more vehicles may work on the same structure individually or cooperatively. In spite of that, the authors recognize that is yet not possible to realize full-scale aerial construction. The main reasons for that are the limitations of flying robots available in the market, such as payload and battery time. Nevertheless, ARC envisions a great potential.

Persistent Surveillance

The *24/7 Persistent Surveillance* is a clear example of a task that a single vehicle cannot achieve. It consists in guarantee that a certain target or region is being monitored uninterruptedly. The problem is well covered in (Valenti et al., 2007). In the referred paper, after running tests to measure the charging/discharging battery time for two specific quadcopters, the authors propose a minimum five vehicles fleet to assure continuous service. One should bear in my mind that the charging is much longer than discharging time. A

regular battery, in a four rotor vehicle, last in average 20 minutes, while the charging time is around an hour.

In the 24/7 Persistent Surveillance there may be only one vehicle in the air for each time instant, but there is a net of vehicles performing the same task. For instances, the vehicles may be disputing one or more charging stations. In this case a more complex cooperative control is required, inasmuch as the vehicles compete for a resource. The problem consists in assuring persistent surveillance and appropriating charging management.

Search and Rescue Operations

Search and Rescue Operations (SAR) represents a significant expense on the budget of a country. The U.S. Coast Guard spends 680 million dollars per year on SAR (Shimanski, 2009). Charging rescuers may not be the best option, since it will probably delay call for assistance. On the other way around, the price per operation can be drastically reduced with unmanned vehicles. Since no human pilot is required, UAVs are an attractive solution for flights over bad weather conditions or dangerous regions. They can also be employed with manned crafts in a cooperative fashion. In addition, the more vehicles, more likelihood the target will be detected in a short time.

In (Grocholsky et al., 2006) an integrated solution of aerial and ground vehicles is presented. Fixed wing aircraft are used for fast coverage of a large area, while ground vehicles are guided to potential interesting region to provide complementary information. Also, Australia holds annually UAV Challenge - Outback Rescue. In this competition, unmanned aircraft have to perform a search and rescue operation (Roberts and Walker, 2010). Although there are considerable effort towards the problem, it lacks practical results with safety, capability and technical quality. By 2013 no team, neither professional nor national agencies, completed the mission goals.

Figure 1.2: Outback Joe waiting for UAV rescue since 2007. source: (Joe, 2012)

Inter Vehicle Cooperation

There are different configurations of autonomous robots, like fixed-wing aircraft, quadcopters, submarines, boats, hovercraft and others. They are designed and built to operate in the more diversified environments like air, dry land, sea, dam and rivers. The integration of these systems is a vanguard field that has gained much interest of researchers in the last few years. For instance, consider the search for a specific target in the sea. UAVs carrying radars able to detect contact on the water surface are employed for a wide search. The information is transmitted to an Autonomous Surface Vessel (ASV) that contacts a Autonomous Underwater Vehicle (AUV) for direct inspection of the possible target. Operations in that scale are costly due to equipment and specialized crew. Cooperative robotic is a relatively cheap platform that will improve the success in large scale mission.

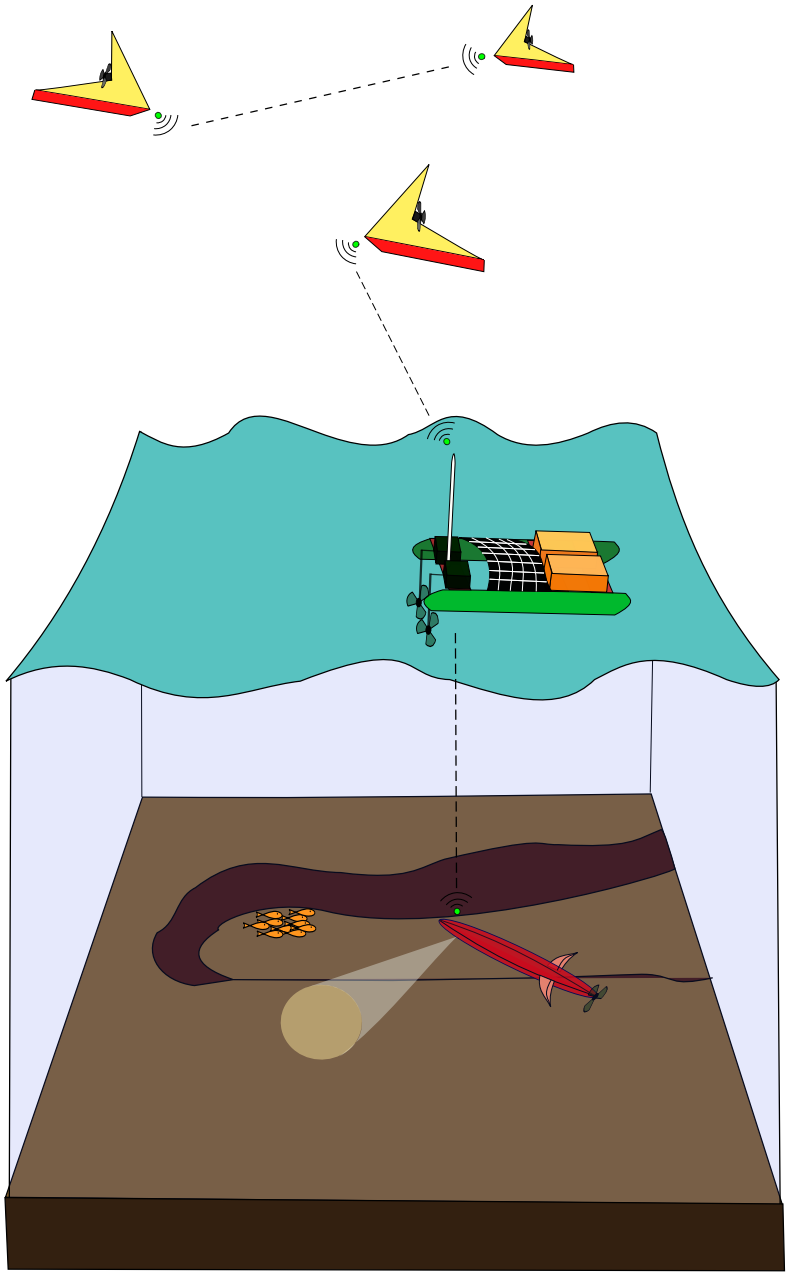


Figure 1.3: Multiple autonomous vehicle cooperation

For more applications the reader is referred to (Etter et al., 2011) and (Sahin, 2005) where a variety of multiple UAV tasks are explored.

1.2 Problem Statement

Many challenges arise in a multi-UAV scenario: data fusion, collaborative assignment, collision avoidance and coordination, just to name a few. The present dissertation focuses on coordinated path following, defined here as a desired inter-vehicle pattern where multiple vehicles are required to follow pre-specified paths.

In order to decrease the complexity, the problem was divided in two parts. The first topic is **path-following** (PF) motion control, where a single vehicle is required to converge and keep track of a pre-specified spatial path with a desired speed assignment without temporal requirements. The second topic is **coordinated control**, in which the vehicles are required to follow a desired inter-vehicle formation.

The path following problem task is a desired capability of an autonomous robot. It aims to bring the vehicle to a desired geometric path parametrized in space such as a straight line or a circle (typically called loiter). As stated in (Sujit et al., 2013), a satisfactory PF algorithm must steer the vehicle towards the path, follow it precisely and be robust against wind disturbances.

The coordinated control of autonomous vehicles is important in a multi-UAV scenario, relieving the burden over human operators that can focuses on high level decisions. The coordination task is particularly challenging, since the nature of the communication networks restricts the information exchanged among vehicles (Fax and Murray, 2004). No aircraft will possible be able to continuously communicate with every unit in the fleet during the entire mission. Furthermore, the coordination strategy must be robust against link failure amid the vehicles. These constraints limit the use of centralized control strategies, where a central unit (ex: ground station) process all information required to achieve a certain goal, for instance. In the last few years, decentralized strategies have come to forum to discuss multi-agent networks with application in engineering and science. This approach is more reliable in terms of flexibility, robustness and scalability.

In this framework, a decentralized multi-vehicle control structure, where the vehicles and communication topology constraints are taken into account is addressed. The presented solution focus on the kinematics, consequently it may be implement in different aerial vehicles morphologies. Numerical simulation and data collect from software-in-the-loop (SiL) experiments evaluate the performance of the methods.

1.3 Previous work and contributions

1.3.1 Path Following Motion Control

As a human pilot is not present to take decisions, the guidance system is a major component in UAV systems. The path following guidance problem requires that a vehicle converge to and follow a path without time constraints. For fully actuated system the

problem is well solved. However, it is usually expensive and impractical to fully actuate autonomous vehicle (Aguiar and Hespanha, 2003). Motivated by these limitations, huge effort has been directed towards underactuated autonomous vehicles. In an underactuated system the independent generalized coordinates outnumber the control inputs, e.g. wheeled robots, surface vessels, underwater vehicles, helicopters and flying wings.

The motion control problem for these systems are particularly interesting because underactuated vehicles are mostly not fully feedback linearizable and exhibit nonholonomic constraints. A traditional solution is linearize the model for a specific operating point and design a PID controller or apply any other classic control method. The performance could be low, except for the neighborhood of the operating point that the system was initially designed to. Gain scheduling solves the problem of a single operating point on the cost of increase complexity. In the scope of this dissertation, nonlinear control techniques are preferred to classic control strategies. As highlighted in (Vanni, 2007), this decision is based on the fact that nonlinear control explicitly takes into account the nonlinearities inherent to the model rather than opposing it.

Pioneering work on the PF problem for wheeled mobile robots is described in (Samson, 1992). The approach is further extended for the three-dimensional case in (Encarnacao and Pascoal, 2000) using Lyapunov based control laws for an *autonomous underwater vehicle* (AUV). In these works a virtual target point (VTP) is placed on the position of the path closest to the vehicle. The vehicle is projected on the desired path and a tangent frame associated to the projection point. This frame is called Serret-Franet $\{\mathcal{F}\}$. In this solution, the vehicle converges and remains inside a tube that involves the path. However, the radius of the tube must be less than the shortest curvature of the path, otherwise a singularity may arise. The work in (Lapierre et al., 2003) proposes an alternative solution to remove the singularity. The origin of the Serret-Frenet frame is not attached to the projection point, instead evolves in time according to a certain function.

In (Aguiar and Hespanha, 2007) an elegant solution by decomposing the problem in two tasks (geometric and dynamic) is presented. The geometric task aims to bring the vehicle and assures it remains inside a tube centered around the desired path. The control signals actuate on the vehicle's orientation. The dynamic assignment task assigns a speed profile to the path. In (Oliveira et al., 2013) the path following problem is generalized for the case of moving path following (MPF), in which the path is not stationary but dynamic. The authors evaluates the method for a fixed-wing UAV with hardware-in-the-loop (HiL) simulations.

As many authors adopt different approaches to solve the PF problem, the work in (Sujit et al., 2013) evaluates five different methods: carrot chasing, nonlinear guidance law, pure pursuit, vector field and linear quadratic regulator. Nonlinear guidance law described by (Park et al., 2007) consumes the least control effort with acceptable response under different wind conditions for straight and loiter path.

1.3.2 Coordinated Control

In the last years a large interest raised towards coordinated control of a fleet of autonomous vehicles. The work described by (Encarnacao and Pascoal, 2000) for a single vehicle is

extended for coordination control of fully actuated vehicles in (Encarnacao and Pascoal, 2001). The authors explore a scenario where an underwater vehicle (slave) must follow a surface vessel (master). The strategy adopted requires a large amount of information exchange between vehicles. Consequently, it cannot be fairly extended for more than two robots.

In (Egerstedt and Hu, 2001) a model-independent for multi-agent formation control is proposed. Each vehicle is assigned to follow a respective reference virtual target. If the tracking errors are bounded, a formal proof is given that states the formation error stabilizes. This allows to decouple the coordination problem into a planning and tracking problem. But the communication constraints of the inter-vehicle communication networks are not considered.

In theory, vehicles could share all internal and external information to improve coordination performance. However, such approach is heavily punished in terms of bandwidth and computational complexity. Moreover, the communication topology may vary over time due to link or even vehicle failure. A suitable communication constraints representation is a methodology based on a framework as addressed in (Fax and Murray, 2004). It relates the concept of Graph Laplacian to represent links between vehicles. Particularly, the work demonstrated in (Fax and Murray, 2002) explicitly shows how the Graph Laplacian associated to a formation interconnection structure plays a fundamental role in assessing stability of the behavior of the components in coordination.

In (Ghabcheloo et al., 2007) the authors borrow results from the fore mentioned work to explicitly address communication constraints for the problem of coordinated path following for a group of wheeled robots. Lyapunov techniques and graph theory are applied. (Aguiar and Pascoal, 2007) discusses a framework that takes into account the topology of the communication links, the logic based nature of communications and the cost of exchanging information. The strategy is further developed in (Ghabcheloo et al., 2009) for a three-dimensional robot. The authors consider two different communication topologies. In the first scenario, the communication graph is alternately connected and disconnected, therein called *brief connectivity losses*. The second scenario, named *uniformly connected in mean*, captures union of communication graphs connected over uniform intervals of time.

1.3.3 Contributions

In order to validate guidance algorithms, a robust and reliable platform is required. The Flight Management Variables System (FVMS), a SiL platform, was the chosen tool. In the scope of this project, a new guidance, navigation and control (GNC) module was developed for the platform. This platform was used to validate the algorithms reported in this work. It may be as well employed in different contexts to evaluate or teach a variety of GNC related algorithms.

1.4 Thesis Outline

The structure of this dissertation is as following

Chapter 2 introduces basic results about nonlinear stability and graph theory. The definitions and theorems presented support the remainder of this thesis.

Chapter 3 : discusses the kinematic and dynamic model of an aircraft. Focus is given towards the kinematic model, which is simplified to the horizontal plane.

Chapter 4 : addresses the path-following control strategy adopted.

Chapter 5 : the coordination formation problem is discussed. It is first introduced an ideal model, with continuous communication. Afterwards, a discrete communication topology is analyzed.

Chapter 6 : describes the simulator that was employed to validate the control laws. The Guidance module, designed in the context of this dissertation, is presented.

Chapter 7 : the main results to validate the control laws described in previous chapters are reported.

Chapter 8 : presents a summary of the results obtained and future work.

2. Mathematical Preliminaries

This chapter exposes the mathematical background that subsequent chapters are based on. In Section 2.1, the notations and basic definitions are settled. Next, Section 2.2 introduces nonlinear system theory and some tools concerned with stability of dynamic systems. In Section 2.3, a brief overview of graph theory is reported.

2.1 Basic definitions

2.1.1 Notations

The notations in this work are normalized as follow. In order to be easily distinguished in the text, mathematical variables are in italics. The scalars are typed in lower-case. Let γ be a scalar variable, then $|\gamma|$ represents its absolute value. Vectors are represented in lower case bold. The vector $\boldsymbol{x} = [x_1, ..., x_n]^T \in \mathbb{R}^n$ is a n-dimensional vector. The element in the ith position is referred as x_i. The norm of $\boldsymbol{x}$, denoted by $\|\boldsymbol{x}\|$, typically considered is the class of p-norms, given by:

$$\|\boldsymbol{x}\|_p = (|x_1|^p + ... + |x_n|^p)^{1/p}, \quad 1 \leq p \leq \infty$$

and

$$\|\boldsymbol{x}\|_\infty = \max_i |x_i|$$

The most commonly used p-norms are $\|\boldsymbol{x}\|_1$, $\|\boldsymbol{x}\|_2$, $\|\boldsymbol{x}\|_\infty$, but the basic properties are satisfied by any p-norm. It is possible to define an inequality that relates any two different p-norms. In the scope of this work it will be considered the Euclidean norm:

$$\|\boldsymbol{x}\|_2 = (|x_1|^2 + ... + |x_n|^2)^{1/2} = (\boldsymbol{x}^T\boldsymbol{x})^{1/2}$$

For the sake of simplicity, along the report, the 2-norm will be referred as $\|\boldsymbol{x}\|$. The following two norms support the concepts of *input-to-state stability* (ISS) that are introduced later in this chapter.

The supremum norm

$$\|\boldsymbol{x}_{[t_o,\infty]}\| = \sup_{t \geq t_0} \|\boldsymbol{x}(t)\|$$

and the asymptotic norm

$$\|\boldsymbol{x}_a\| = \lim_{t\to\infty} \|\boldsymbol{x}(t)\|$$

The vector $\mathbf{1}$ contains all elements equal to the unity. The same way, all the values of vector $\mathbf{0}$ are null. The length of the vector is implicit within the context.

Matrices are denoted in upper upper case. For instance, the element a_{ij} lies in the ith row and jth column of the matrix A. The most commonly used induced matrix norms are the p-norm, with p=1,2 and ∞ and the Frobenius norm, defined as follow:

Let $A_{m\times n} \in \mathbb{R}^{m\times n}$, then

- p_{norm}: $\|A\|_p = \max \|A\boldsymbol{x}\|_p : \|\boldsymbol{x}\|_p = 1$
- ∞_{norm}: $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$
- Frobenius: $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$

The Frobenius norm can be simplified as $\|A\|_F = \text{trace}(A^T A)$, where the function $trace(\cdot)$ is defined as the sum of the elements of the main diagonal. The transpose of a matrix, represented by A^T, contains the elements of A reflected over its main diagonal. A square matrix has the same number of rows and columns. A square matrix is said to be symmetric if it is equal to its transpose, $A = A^T$. The size of the matrix is implicit in the context where it belongs.

Definition 2.1. *A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is*

- ***positive definite*** *if and only if*

$$\boldsymbol{x}^T A\boldsymbol{x} > 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \mathbf{0}$$

  This fact is stated as $A \succ 0$

- ***semipositive definite*** *if and only if*

$$\boldsymbol{x}^T A\boldsymbol{x} \geq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \mathbf{0}$$

  This fact is stated as $A \succeq 0$

A skew-symmetric matrix S is equal to the negative of its transpose, $S = -S^T$.

Coordinate frames, sets and graphs are typed in calligraphic letters, except the number sets, which are represented with blackboard bold.

2.1.2 Sets

A subset $\mathcal{S} \subset \mathbb{R}^n$ is said to be *open* if for every vector $\boldsymbol{x} \in \mathcal{S}$ one can find an ε-neighbourhood of x:

$$N(\boldsymbol{x}, \varepsilon) = \{\boldsymbol{z} \in \mathbb{R}^n | \quad \|\boldsymbol{z} - \boldsymbol{x}\| < \varepsilon\}, \text{such that} \quad N(\boldsymbol{x}, \varepsilon) \subset \mathcal{S}$$

The set composed by all points inside a circle is an open set. On the other way round, a set is *closed* if and only if its complement in $\mathbb{R}^n$ is open. In other words, $\mathcal{S}$ is closed

if and only if every convergence sequence $\|\boldsymbol{x}_k\|$, with elements in $\mathcal{S}$, converges to a point within the set. The set composed of the points in the edge of a circle is a closed set. A set $\mathcal{S}$ is subject to range of properties which the following are brought out:

1. Boundness: a set is called bounded if there is a real positive number r such that $\|\boldsymbol{x}\| \leq r, \forall \boldsymbol{x} \in \mathcal{S}$.
2. Compact: a set is compact if it is both closed and bounded.
3. Connected: a set is said to be connected if its points can be linked by an arc lying within the set .

2.1.3 Continuous Function

A function f that maps $\mathbb{R}^n$ into $\mathbb{R}^m$, denoted by $f : \mathbb{R}^n \to \mathbb{R}^m$, is said to be continuous at a point $\boldsymbol{x}$ if $f(\boldsymbol{x}_k) \to f(\boldsymbol{x})$ as $\boldsymbol{x}_k \to \boldsymbol{x}$, that is, if given $\varepsilon > 0$ there is $\delta > 0$ such that

$$\|\boldsymbol{x} - \boldsymbol{y}\| < \delta \implies \|f(\boldsymbol{x}) - f(\boldsymbol{y})\| < \varepsilon$$

Definition 2.2. *A function is said to be*

- ***continuous*** *on a set $\mathcal{S}$ if it is continuous at every point of $\mathcal{S}$.*
- ***uniformly continuous*** *on $\mathcal{S}$ if given $\varepsilon > 0$ there is $\delta > 0$ such that*

$$\|\boldsymbol{x} - \boldsymbol{y}\| < \delta \Rightarrow \|f(\boldsymbol{x}) - f(\boldsymbol{y})\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}.$$

2.2 Nonlinear System

This section presents a summary of the most relevant concepts from nonlinear systems theory that are used in this work. The theorems and definitions here presented are borrowed from (Khalil, 2002). The reader finds in the cited reference proof and further details.

2.2.1 System Representation

Dynamical systems are described by first order differential equations:

$$\begin{aligned} \dot{x}_1 &= f_1(t, x_1, \ldots, x_n, u_1, \ldots, u_p) \\ &\vdots \\ \dot{x}_n &= f_n(t, x_1, \ldots, x_n, u_1, \ldots, u_p) \end{aligned}$$

where x_i, $\dot{x}_i$ and u_i to $i = 1, \ldots, n$ are the state variable, time derivative of x_i and input variable, respectively. It is possible to rewrite the n-differential equations as one n-dimensional first-order vector differential equation:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}, \quad f(t, \boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} f_1(t, \boldsymbol{x}, \boldsymbol{u}) \\ \vdots \\ f_n(t, \boldsymbol{x}, \boldsymbol{u}) \end{bmatrix}$$

In a compact form, the *state equation* is given by

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}, \boldsymbol{u}) \tag{2.1}$$

When there is no explicit dependence of the input, it is called *unforced state equation* or *nonautonomous system*:

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}) \tag{2.2}$$

When the system does not depended explicitly on time t, it is named as *time-invariant* or *autonomous system*. In this case the system is invariant to shifts in the time origin.

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \tag{2.3}$$

2.2.2 Lipschitz Function

The properties of existence and uniqueness are fundamental for a state equation correctly model a dynamic system. The Lipschitz condition, expressed by

$$\|f(t, \boldsymbol{x}) - f(t, \boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\| \tag{2.4}$$

is a tool to verify that the initial-value problem stated as

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}), \quad \boldsymbol{x}(t_0) = \boldsymbol{x_0} \tag{2.5}$$

has a unique solution.

Theorem 2.1. ***(Local Existence and Uniqueness):*** *Let $f(t, \boldsymbol{x})$ be a piecewise continuous function in t that satisfies the Lipschitz condition for all $(t, \boldsymbol{x})$ and $(t, \boldsymbol{y})$ in some neighbourhood of $(t_0, \boldsymbol{x_0})$. Then there exist some $\delta > 0$ such that the initial-value problem has a unique solution over the interval $[t_0, t_0 + \delta]$.*

According to the domain which the Lipschitz condition holds true the following classifications take place.

Definition 2.3. *A function $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ is*

- *Lipschitz in $\boldsymbol{x}$ on $[a, b] \times \mathcal{W}$ if it satisfies (2.4) with the same Lipschitz constant L*
- *Locally Lipschitz in $\boldsymbol{x}$ on $[a, b] \times \mathcal{D}$ if every point in $\boldsymbol{x} \in \mathcal{D}$ has a neighbourhood $\mathcal{D}_\prime$ that satisfies (2.4) with given Lipschitz constant L_0*
- *Globally Lipschitz if it is Lipschitz in $\mathbb{R}^n$*

2.2.3 Autonomous systems

Consider the autonomous system described by $\dot{x} = f(x)$, where f is locally Lipschitz on a domain $\mathcal{D}$ and suppose $\boldsymbol{x}^* \in \mathcal{D}$

Definition 2.4. *If the system starts with on $\boldsymbol{x} = \boldsymbol{x}^*$ at $t = t_0$ and remains there for all future time $t > t_0$, then $\boldsymbol{x}^*$ is called an **equilibrium point** of f.*

The equilibrium point can be shifted to any state-vector on the domain through a simple change of variables. Let $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{x}^*$, then

$$\dot{\boldsymbol{y}} = f(\boldsymbol{x}) = f(\boldsymbol{y} + \boldsymbol{x}^*) = g(\boldsymbol{y}), \quad \text{where } g(\mathbf{0}) = \mathbf{0} \tag{2.6}$$

Therefore, with no loss of generality, it is assumed the equilibrium point is always at the origin.

Definition 2.5. *The equilibrium point at $\boldsymbol{x} = \mathbf{0}$ of (2.3) is*

- ***stable*** *if for each $\varepsilon > 0$, there is a $\delta > 0$ such that $\|\boldsymbol{x}(0)\| < \delta$, implies $\|\boldsymbol{x}(t)\| < \varepsilon$ for all future time.*

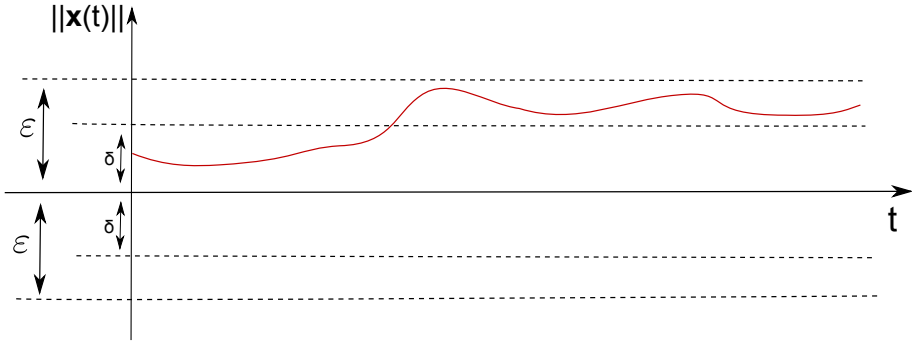


Figure 2.1: A stable system

- ***asymptotically stable*** *if it stable, but also converges to the zero as time approach infinity. ($t \to \infty$, $\|\boldsymbol{x}\| \to 0$).*

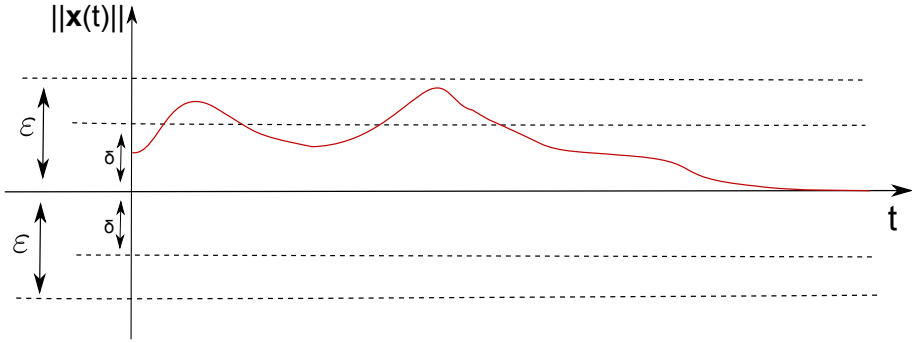


Figure 2.2: An asymptotically stable system

- ***unstable*** *if it is not stable*

The previous definition uses $\varepsilon - \delta$ requirement for stability. In many cases, it is challenging to provide a value of δ, given a ball ε. The generalized approach requires finding all the solutions of (2.3) for determining that the system once started at the δ neighborhood of the origin will never leave the specified ε neighborhood. Such task may be hard or not feasible. However, it is possible to determine stability without having to know explicitly the solution of (2.3) through the *Lyapunov stability theorem.*

Theorem 2.2. *Let $\mathcal{D}$ be a domain that contains $\boldsymbol{x} = \mathbf{0}$, an equilibrium point for (2.3). Let $V : \mathcal{D} \to \mathbb{R}$ be a continuously differentiable function, such that*

$$V(\mathbf{0}) = 0 \text{ and } V(\boldsymbol{x}) > 0 \text{ in } \mathcal{D} - \{\mathbf{0}\} \tag{2.7}$$

$$\dot{V}(\boldsymbol{x}) \leq 0 \text{ in } \{D\} \tag{2.8}$$

Then $\boldsymbol{x} = \mathbf{0}$ is stable. if

$$\dot{V}(\boldsymbol{x}) < 0 \text{ in } \mathcal{D} - \{\mathbf{0}\}$$

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

A continuously differentiable function $V(\boldsymbol{x})$ that holds (2.7) and (2.8) is named Lyapunov function. The Lyapunov function expresses the energy of the system. If there is an equilibrium point, the energy stored decreases until reaches the minimum value on that point. The following definition is introduced

Definition 2.6. *A function $V(\boldsymbol{x})$ is*

- *positive (negative) definite if $V(\boldsymbol{x}) > 0$ ($V(\boldsymbol{x}) < 0$)*
- *positive (negative) semidefinite if $V(\boldsymbol{x}) \geq 0$ ($V(\boldsymbol{x}) \leq 0$)*

Theorem 2.3. *Let $\boldsymbol{x} = \mathbf{0}$ be an equilibrium point for (2.3). Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(\mathbf{0}) = \mathbf{0} \text{ and } V(\boldsymbol{x}) > \mathbf{0}, \forall \boldsymbol{x} \neq \mathbf{0} \tag{2.9}$$

$$\|\boldsymbol{x}\| \to \infty \Rightarrow V(\boldsymbol{x}) \to \infty \tag{2.10}$$

$$\dot{V}(\boldsymbol{x}) < 0, \ \forall \boldsymbol{x} \neq \mathbf{0} \tag{2.11}$$

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

A function satisfying (2.10) is said to be *radially unbounded*.

2.2.4 Nonautonomous systems

Consider the nonautonomous system $\dot{\boldsymbol{x}} = f(t, \boldsymbol{x})$, where f is piece continuous in t, locally Lipschitz in $\boldsymbol{x}$ and its domain contains the origin $\boldsymbol{x} = \mathbf{0}$. The origin is an equilibrium point at $t = 0$ if

$$f(t, 0) = 0, \quad t \geq 0$$

As discussed for autonomous systems, a nonzero equilibrium point may be represented as an equilibrium point at the origin by means of an appropriate transformation. The notions of stability and asymptotic stability introduced in Definition 2.5 for autonomous systems are very much alike for nonautonomous systems. However, while the solution of an autonomous system depends only on $(t - t_0)$, a nonautonomous system relies on both t and t_0. Consequently, the stability behavior depends on the origin at the initial time t_0, that is, the constant δ is a function of t_0. As a result, Definition 2.5 is redefined.

Definition 2.7. *The equilibrium point $\boldsymbol{x} = \mathbf{0}$ is*

- ***stable*** *if, for each* $\varepsilon > 0$*, there is a* $\delta = \delta(\varepsilon, t_0) > 0$*, such that*

$$\|\boldsymbol{x}(t_0)\| < \delta \Rightarrow \|\boldsymbol{x}(t)\| < \varepsilon, \quad 0 \leq t \leq t_0$$

- ***uniformly stable*** *if, for each* $\varepsilon > 0$*, there is* $\delta = \delta(\varepsilon) > 0$*, independent of* t_0*, such that the stable condition is satisfied.*

- ***unstable*** *if it is not stable.*

- ***asymptotically stable*** *if it is stable and there is a positive constant* $c = c(t_0)$ *such that* $\boldsymbol{x}(t) \to \mathbf{0}$ *as* $t \to \infty$*, for all* $\|\boldsymbol{x}(t_0)\| < c$*.*

- ***uniformly asymptotically stable*** *if it is uniformly stable and there is a positive constant* c*, independent of* t_0*, such that for all* $\|\boldsymbol{x}(t_0)\| < c$*,* $\boldsymbol{x}(t) \to \mathbf{0}$ *as* $t \to \infty$ *uniformly in* t_0*; that is, for each* $\eta > 0$*, there is* $T = T(\eta) > 0$ *such that*

$$\|\boldsymbol{x}(t)\| < \eta, \, \forall t \geq t_0 + T(\eta), \, \forall \|\boldsymbol{x}(t_0)\| < c$$

- ***globally uniformly asymptotically stable*** *if it is uniformly stable,* $\delta(\varepsilon)$ *can be chosen to satisfy* $\lim_{\varepsilon \to \infty} \delta(\varepsilon) = \infty$*, and, for each pair of positive numbers* η *and* c*, there is a* $T = T(\eta, c) > 0$ *such that*

$$\|\boldsymbol{x}(t)\| < \eta, \, \forall t \geq t_0 + T(\eta, c), \, \forall \|\boldsymbol{x}(t_0)\| < c$$

The control problems derived in this dissertation requires a more convenient and transparent definition of uniform stability and uniform asymptotic stability. To this end we need to introduce the following classes of functions.

Definition 2.8. *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* $\alpha(r) \to \infty$ *as* $r \to \infty$*.*

Definition 2.9. *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* $s \to \infty$*.*

It is now possible to redefine uniform stability and uniform asymptotic stability.

Lemma 2.1. *The equilibrium point at* $\boldsymbol{x} = \mathbf{0}$ *of (2.2) is*

- ***uniformly stable*** *if and only if there exist a class* $\mathcal{K}$ *function* α *and a positive constant* c*, independent of* t_0*, such that*

$$\|\boldsymbol{x}(t)\| \leq \alpha(\|\boldsymbol{x}(t_0)\|), \, \forall t \geq t_0 \geq 0, \, \forall \|\boldsymbol{x}(t_0)\| < c$$

- ***uniformly asymptotically stable*** *if and only if there exist a class* $\mathcal{KL}$ *function* β *and a positive constant* c*, independent of* t_0*, such that*

$$\|\boldsymbol{x}(t)\| \leq \beta(\|\boldsymbol{x}(t_0)\|, t - t_0), \, \forall t \geq t_0 \geq 0, \, \forall \|\boldsymbol{x}(t_0)\| < c$$

- ***globally uniformly asymptotically stable*** *if and only if uniformly asymptotically stable definition holds for any initial state* $\boldsymbol{x}(t_0)$.

The class $\mathcal{KL}$ function $\beta(r,s) = kre^{-\lambda s}$ arises as special case of uniform asymptotic stability, as defined next.

Definition 2.10. *The equilibrium point* $\boldsymbol{x} = \mathbf{0}$ *of* (2.2) *is exponentially stable if there existe positive constants* c, k *and* λ *such that*

$$\|\boldsymbol{x}(t)\| \leq k\|\boldsymbol{x}(t_0)\|e^{-\lambda(t-t_0)}, \quad \forall\|\boldsymbol{x}(t_0)\| < c \tag{2.12}$$

and globally exponentially stable if the fore mentioned equation is satisfied for any initial state $\boldsymbol{x}(t_0)$.

Lyapunov theory for autonomous systems can be extended to nonautonomous systems.

Theorem 2.4. *Let* $\boldsymbol{x} = \mathbf{0}$ *be an equilibrium point for* (2.2) *and* $\mathcal{D} \subset \mathbb{R}^n$ *be a domain contains it. Let* $V : [0,\infty) \times \mathcal{D} \to \mathbb{R}$ *be a continuously differentiable function such that*

$$W_1(\boldsymbol{x}) \leq V(t,\boldsymbol{x}) \leq W_2(\boldsymbol{x}) \tag{2.13}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \boldsymbol{x}} f(t,\boldsymbol{x}) \leq \mathbf{0} \tag{2.14}$$

$\forall t \geq 0$ *and* $\forall \boldsymbol{x} \in \mathcal{D}$, *where* $W_1(\boldsymbol{x})$ *and* $W_2(\boldsymbol{x})$ *are continuous positive definite functions on* $\mathcal{D}$. *Then,* $\boldsymbol{x} = \mathbf{0}$ *is uniformly stable.*

Theorem 2.5. *Suppose the assumptions of Theorem (2.4) are satisfied with inequality* 2.14 *strengthened to*

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \boldsymbol{x}} f(t,\boldsymbol{x}) \leq -W_3(\boldsymbol{x})$$

$\forall t \geq 0$ *and* $\forall \boldsymbol{x} \in \mathcal{D}$, *where* $W_3(\boldsymbol{x})$ *is continuous positive definite functions on* $\mathcal{D}$. *Then,* $\boldsymbol{x} = \mathbf{0}$ *is uniformly asymptotically stable. Moreover, if positive constant* r *and* c *are chosen such that* $\mathcal{B}_r = \{\|\boldsymbol{x}\| \leq r\} \subset \mathcal{D}$ *and* $c < \min_{\|\boldsymbol{x}\|=r} W_1(\boldsymbol{x})$, *then every trajectory starting in* $\{\boldsymbol{x} \in \mathcal{B}_r | W_2(\boldsymbol{x}) \leq c\}$ *satisfies*

$$\|\boldsymbol{x}(t)\| \leq \beta(\|\boldsymbol{x}(t_0)\|, t - t_0),\ \forall t \geq t_0 \geq 0$$

for some class $\mathcal{KL}$ *function* β. *Finally, if* $\mathcal{D} = \mathbb{R}^n$ *and* $W_1(\boldsymbol{x})$ *is radially unbounded, then* $\boldsymbol{x} = \mathbf{0}$ *is globally uniformly asymptotically stable.*

Theorem 2.6. *Let* $\boldsymbol{x} = \mathbf{0}$ *be an equilibrium point for (2.2) and* $\mathcal{D} \subset \mathbb{R}^n$ *be a domain containing* $\boldsymbol{x} = \mathbf{0}$. *Let* $V : [0,\infty) \times \mathcal{D} \to \mathbb{R}$ *be a continuously differentiable function, such that*

$$k_1\|\boldsymbol{x}\|^a \leq V(t,\boldsymbol{x}) \leq k_2\|\boldsymbol{x}\|^a$$
$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \boldsymbol{x}} f(t,\boldsymbol{x}) \leq -k_3\|\boldsymbol{x}\|^a$$

$\forall t \geq 0$ *and* $\forall \boldsymbol{x} \in \mathcal{D}$, *where* k_1, k_2, k_3 *and* a *are positive constants. Then,* $\boldsymbol{x} = \mathbf{0}$ *is exponentially stable. If the assumptions hold globally, then the equilibrium point is globally exponentially stable.*

2.2.5 Boundedness

Lyapunov analysis can be used for showing boundedness of the solution of state equations, even when no equilibrium point at the origin exists. The bound gives a better estimate of the solution after the transient period finishes.

Definition 2.11. *The solutions of (2.2) are*

- ***uniformly bounded*** *if there exists a positive constance c, independent of $t_0 \geq 0$, and for every $a \in (0, c)$, there is $\beta = \beta(a)$, independent of t_0, such that*

$$\|\boldsymbol{x}(t_0)\| \leq a \Rightarrow \|\boldsymbol{x}(t)\| \leq \beta, \forall\, t \geq t_0 \tag{2.15}$$

- ***globally uniformly bounded*** *if it holds (2.15) for arbitrarily large a.*

- ***uniformly ultimately bounded*** *with ultimate bound b if there exist positive constants b and c, independent of $t_0 \geq 0$, and for every $a \in (0, c)$, there is $T = T(a, b) \geq 0$, independent of t_0, such that*

$$\|\boldsymbol{x}(t_0)\| \leq a \Rightarrow \|\boldsymbol{x}(t)\| \leq b, \forall\, t \geq t_0 + T \tag{2.16}$$

- ***globally uniformly ultimately bounded*** *if it holds (2.16) for arbitrarily large a.*

For autonomous system, the word "uniform" is not employed, as the solution depends exclusively on $(t - t_0)$.

2.2.6 Input-to-state stability

Consider the system described by $\dot{x} = f(t, \boldsymbol{x}, \boldsymbol{u})$, where f is piecewise continuous in t and locally Lipschitz in $\boldsymbol{x}$ and $\boldsymbol{u}$. Consider that the unforced system

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}, \boldsymbol{0})$$

has a globally uniformly asymptotically stable equilibrium point at the origin $\boldsymbol{x} = \boldsymbol{0}$. The system described by (2.1) can be viewed as a perturbation of the unforced system. If $\boldsymbol{u}$ is bounded, then it may be possible, in some cases, to show that $\boldsymbol{x}(t)$ is also bounded. This fact leads to the introduction of *input-to-state stability*.

Definition 2.12. *The system described by (2.1) is said to be input-to-state stable if there exist a class $\mathcal{KL}$ function β and a class $\mathcal{K}$ function γ such that for any initial state $\boldsymbol{x}(t_0)$ and any bounded input $\boldsymbol{u}(t)$, the solution $\boldsymbol{x}(t)$ exists for all $t \geq t_0$ and satisfies*

$$\|\boldsymbol{x}(t)\| \leq \beta(\|\boldsymbol{x}(t_0)\|, t - t_0) + \gamma(\|\boldsymbol{u}_{[t_0,t]}\|) \tag{2.17}$$

The inequality (2.17) ensures that for any bounded $\boldsymbol{u}$, the state $\boldsymbol{x}(t)$ will be bounded. Further, as t increases, the state $x(t)$ will be ultimately bounded by a class $\mathcal{K}$ function of $\|\boldsymbol{u}_{[t_0,\infty)}\|$. Moreover, if $\boldsymbol{u}(t)$ converges to zero as $t \to \infty$, so does $\|\boldsymbol{x}(t)\|$. The following theorems give sufficient conditions for input-to-state stability.

Theorem 2.7. *Let $V : [0,\infty) \times \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function such that*

$$\alpha_1(\boldsymbol{x}) \leq V(t,\boldsymbol{x}) \leq \alpha_2(\boldsymbol{x}) \tag{2.18}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial \boldsymbol{x}} f(t,\boldsymbol{x},\boldsymbol{u}) \leq -W_3(\boldsymbol{x}),\ \forall\, \|\boldsymbol{x}\| \geq \rho(\|\boldsymbol{u}\|) > 0 \tag{2.19}$$

$\forall\,(t,\boldsymbol{x},\boldsymbol{u}) \in [0,\infty) \times \mathbb{R}^n \times \mathbb{R}^m$, where α_1 and α_2 are class $\mathcal{K}_\infty$ functions ρ is a class $\mathcal{K}$ function and $W_3(\boldsymbol{x})$ is a continuous positive definite function on $\mathbb{R}^n$. Then, the system (2.17) is input-to-state stable with $\gamma = \alpha_1^{-1}\, o\, \alpha_2\, o\, \rho$

The conditions (2.18) and (2.19) are also necessary for autonomous systems. A function V that satisfies Theorem 2.7 is named ISS-Lyapunov function.

2.3 Graph theory

Graph theory is used in this work to model the communication between vehicles in a fleet. This tool allows modelling and studying the impact of different communication topologies on the performance of the coordinated system. This section introduces elementary concepts about digraphs. The notations and definitions presented for digraphs are borrowed from (Mesbahi and Egerstedt, 2010) and (Aguiar and Hespanha, 2007).

2.3.1 Basic concepts

A graph denoted by $\mathcal{G}(\mathcal{V},\mathcal{E})$ or simple $\mathcal{G}$ is composed of a set of *vertices* (nodes) $\mathcal{V} = \{v_1,..,v_n\}$ and a set $\mathcal{E} = \{(v_1v_2),(v_2v_3),..,(v_{n-1}v_n)\}$ that corresponds to its *edges* (lines). Let each node of $\mathcal{V}$ represent a vehicle in the fleet, the edges of $\mathcal{E}$ the data link and $\mathcal{G}$ the intervehicle communication network. If the edges are bidirectional, or equivalently, communication works both ways, then the graph is *undirected*, otherwise it is named *directed graph* or *digraph*. Different graphs are shown in Fig. 2.3. Henceforth, only the digraph is considered, as it is a generalized approach.

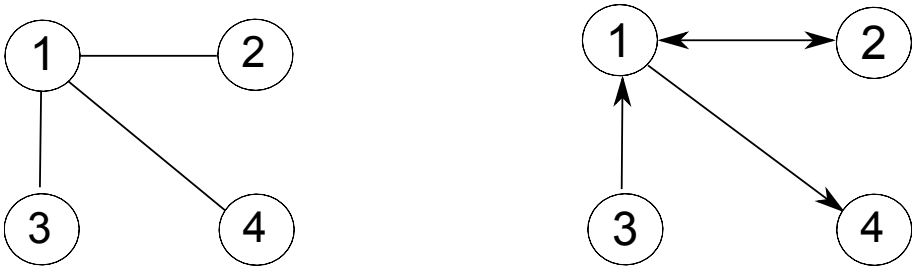


Figure 2.3: An undirected (left) and directed graph (right)

The oriented edges in a digraph are known as *direct edges* or *arcs*. The order of a graph denoted by $|\mathcal{G}|$ is equal to its number of vertices. If the ordered pair $(v_i,v_j) \in \mathcal{E}$, the nodes are called adjacent, with a (v_i,v_j) arc joining them. The first element of the ordered pair is said to be the tail of the arc and the second is its *head*. It is stated that the arc (v_i,v_j) points from v_i to v_j and the flow of information is directed from head (transmitter) to tail

(receiver). If for every arc in $\mathcal{E}$ the tail is different from the head and each element of $\mathcal{E}$ is unique, then the graph is *oriented*. The *in-degree* of a node v_i is the number of arcs with v_i as its head. Analogously, the *out-degree* of a node v_i is the number of arcs with v_i as its tail. A graph is said to be *complete* if all vertices are pairwise adjacent.

The operation of union and intersection are defined as following

Definition 2.13. *Let* $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$ *and* $\mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2)$

- $\mathcal{G}_1 \cup \mathcal{G}_2 = (\mathcal{V}_1 \cup \mathcal{V}_2, \mathcal{E}_1 \cup \mathcal{E}_2)$
- $\mathcal{G}_1 \cap \mathcal{G}_2 = (\mathcal{V}_1 \cap \mathcal{V}_2, \mathcal{E}_1 \cap \mathcal{E}_2)$

If $\mathcal{G}_1 \cap \mathcal{G}_2 = \emptyset$*, then the graphs are disjoint.*

2.3.2 Connectivity

A *path* of length m from a node v_i to v_j is a sequence of $m+1$ distinct nodes such that for $k = 0, 1, .., m-1$, v_k and v_{k+1} are adjacent. If a path links v_i to v_j, then v_i can access v_j and v_j is said to be *reachable* from v_i. If a node is reachable from any other node then it is *globally reachable*. If a graph $\mathcal{G}$ has a globally reachable node, it is called *quasi strongly connected* (QSC), that is, the opposite graph has a globally reachable node. If every node is globally reachable, then the graph is *strongly connected*. A graph whose disjoint sets of nodes are not linked with one another is called *disconnected*.

2.3.3 Algebraic graph theory

A matrix can be used to represent a graph. Let the *adjacency matrix* $\mathrm{A}(\mathcal{G})$, a square matrix of size $|\mathcal{G}|$, be defined as

$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

Remember that the relation $v_i v_j = v_j v_i$ is not necessarily true. The adjacency matrix defines a graph uniquely, as long as a given enumeration of its vertices is kept constant. The *degree matrix* of a directed graph $\mathcal{G}$, denoted by $D(\mathcal{G})$, is a square matrix whose elements of the main diagonal are the out-degrees of the respective node v_{ii}. The *Laplacian* of a graph is expressed as following

$$L = D - A \tag{2.20}$$

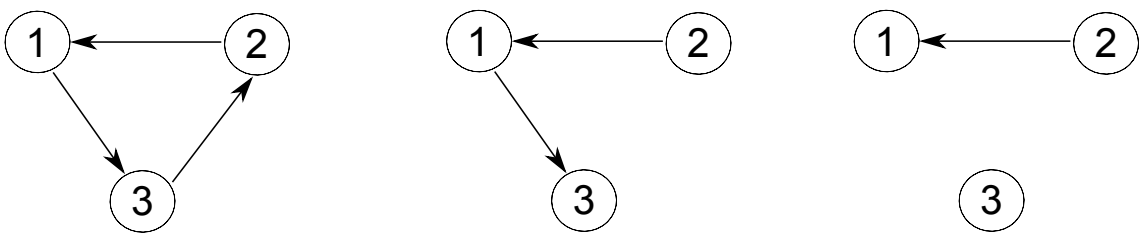


Figure 2.4: A strongly connected (left), a quasi strongly connected (center) and a disconnect (right) graph

By definition, each row of the matrix L sums up to zero, therefore the vector $\mathbf{1}$ belongs to its kernel $K(L)$. The matrix associated to the digraphs shown in Fig. 2.4 are:

$$A_1 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad D_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad L_1 = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad D_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad L_2 = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad D_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad L_3 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

2.3.4 Switching Topology

Let $\mathcal{G}$ be a complete graph, i.e., all possible arcs $1 \dots \bar{n}$ exist. Consider the piecewise continuous function $p_i(t) : [0, \infty) \to \{0, 1\}$, where $i = 1, \dots, \bar{n}$.

$$p_i(t) = \begin{cases} 1, & \text{existence of arc } i \text{ at time } t \\ 0, & \text{otherwise} \end{cases} \tag{2.21}$$

The switching signal is defined as the column vector $\boldsymbol{p}(t) = [p_i]_{\bar{n}\times 1}$. For each time instant, the graph $\mathcal{G}_{\boldsymbol{p}(t)}$ is defined by $(\mathcal{V}, \mathcal{E}_{\boldsymbol{p}(t)})$.

Consider that in a given interval of time T, there are q graphs defined, $\mathcal{G}_i$; $i = 1, \cdots, q$. Each graph has associated a Laplacian matrix L_i. The union graph, denoted as $\mathcal{G} = \cup_i \mathcal{G}_i$, is the graph whose arcs are the union of the arcs $\mathcal{E}_i$ of $\mathcal{G}_i$; $i = 1, \cdots, q$.

Definition 2.14. *A graph $\mathcal{G}_{\boldsymbol{p}(t)}$ is said to be **uniformly quasi strongly connected** (UQSC), if, for every $t_0 > 0$, there is a $T > 0$, such that the union graph $\mathcal{G}([t, t+T))$ is QSC.*

In Fig. 2.5, the graphs $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$, represented at the top left, right and bottom left, respectively, are disconnected. However, the union graph (bottom right) is uniformly quasi strongly connected.

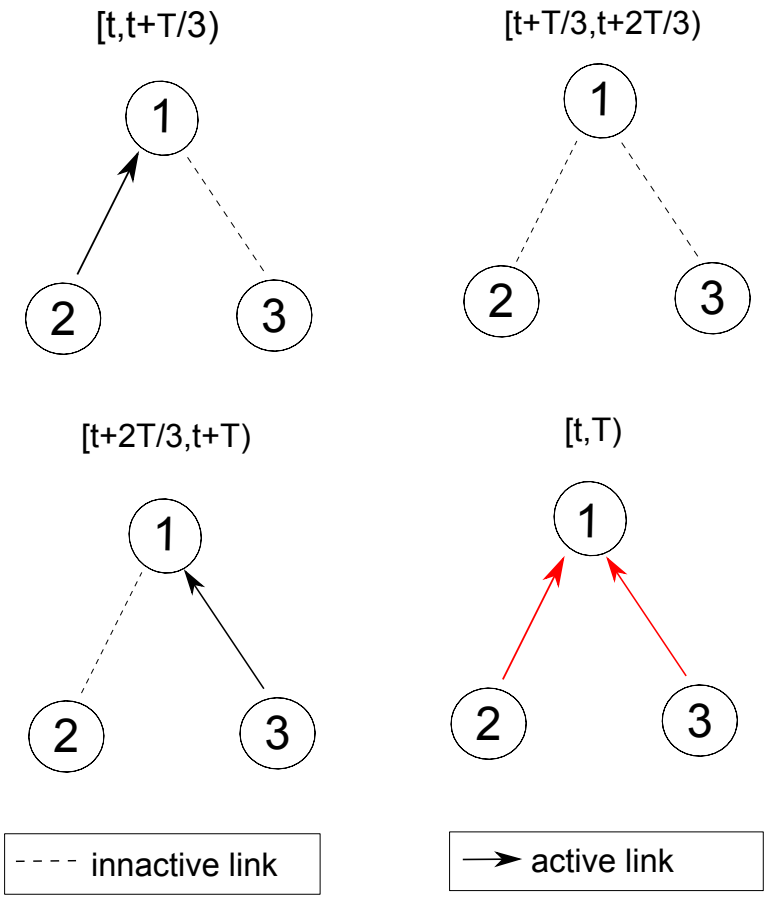


Figure 2.5: Switching topology. The graphs are disconnected, however, its union over a time interval (bottom right) is UQSC

3. UAV System Model

This chapter discusses the mathematical model of a UAV. In Section 3.1 the coordinate frames employed to describe the mathematical model are introduced. Then, the kinematic equations are derived, as well as a simplified model usually considered in practical UAV missions (Section 3.2). This simplified model is based on the assumption that the dynamics of small wing-fixed UAV can be built on the unicycle model (Jackson, 2011). This approach is also used to introduce wind perturbations in the model. The dynamics of the UAV and its basic working principles are presented in Section 3.3.

3.1 Coordination Frames

An aircraft mathematical model has six degrees of freedom (DoF), including three translational (vertical, horizontal and longitudinal) and three rotational movements (roll, pitch and yaw) usually described by Euler angles. To present the model, two coordinates frames are introduced. As illustrated in Fig. 3.1, it is common practice to place an inertial frame $\{\mathcal{I}\}$ on the ground, composed of the orthonormal axes $\{x_{\mathcal{I}}, y_{\mathcal{I}}, z_{\mathcal{I}}\}$. In this work, the origin of the inertial frame is placed at the center of earth, on the intersection of the equator (0^o latitude) and the prime meridian (0^o longitude). Its orientation follows the **N**orth, **E**ast,

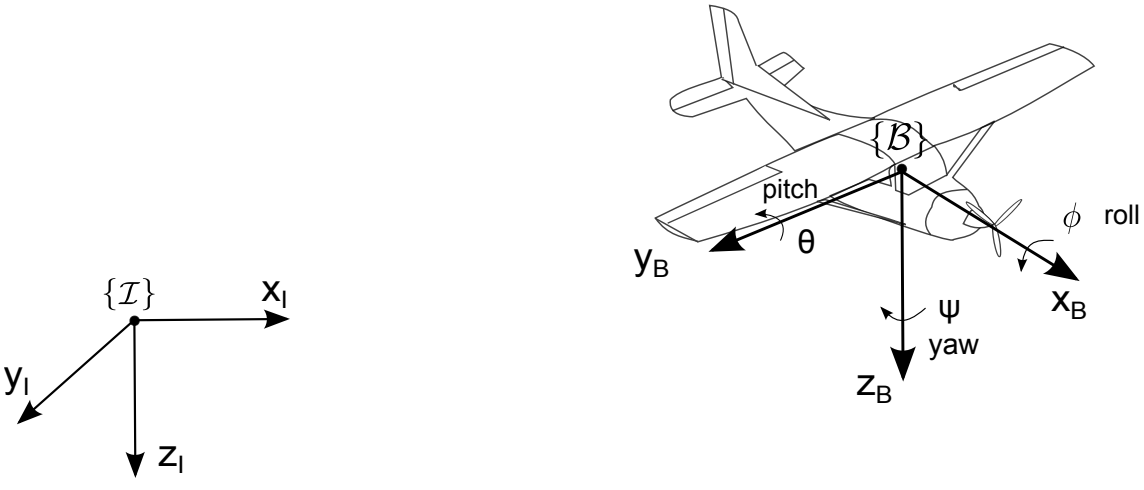


Figure 3.1: UAV Coordinate frame

Down (NED) system, as described next

- $\{x_{\mathcal{I}}\}$ pointing North;
- $\{y_{\mathcal{I}}\}$ pointing East;
- $\{z_{\mathcal{I}}\}$ pointing Down.

It is also considered a body fixed frame $\{\mathcal{B}\}$ composed of the orthonormal axes $\{x_{\mathcal{B}}, y_{\mathcal{B}}, z_{\mathcal{B}}\}$, according to the NED-system orientation. This frame can be fixed anywhere in the body. However, to simplify the equation model, its origin usually lies in the center of gravity (CG) of the vehicle, with two of its axes aligned with main inertial axes of the vehicle:

- $\{x_{\mathcal{B}}\}$ in the longitudinal axis, from the tail to the nose of the vehicle, parallel to the fuselage;
- $\{y_{\mathcal{B}}\}$ in the lateral axis, pointing to the right side, parallel to the right wing;
- $\{z_{\mathcal{B}}\}$ in the vertical axis, pointing downward, from top to bottom.

The position and orientation of the UAV are relative to the inertial reference frame $\{\mathcal{I}\}$. The linear velocities of the craft are measured relatively to $\{\mathcal{I}\}$, expressed in the body frame $\{\mathcal{B}\}$. The following notation is used:

- $\boldsymbol{p} = [x, y, z]^T$: The position of the origin of $\{\mathcal{B}\}$, expressed in $\{\mathcal{I}\}$;
- $\boldsymbol{\eta} = [\phi, \theta, \psi]^T$: The orientation of $\{\mathcal{B}\}$ with respect to $\{\mathcal{I}\}$;
- $\boldsymbol{v_{gs}} = [u, v, w]^T$: The linear velocities of the origin $\{\mathcal{B}\}$ in relation to $\{\mathcal{I}\}$, expressed in $\{\mathcal{B}\}$;
- $\boldsymbol{w} = [p, q, r]^T$: The angular velocities of $\{\mathcal{B}\}$ with respect to $\{\mathcal{I}\}$, expressed in $\{\mathcal{B}\}$

The orientation is also known as attitude of the aircraft. It is composed of the angles of roll (ϕ), pitch (θ) and yaw(ψ). The linear velocities vector contains the forward (u), lateral (v) and vertical (w) speed of the vehicle.

3.2 Equations of Motion

This section describes the kinematic equations of an unmanned aerial vehicle. The first model is based on underactuated autonomous vehicles moving in the 3D space. For instance, it represents the model of an aircraft or an underwater vehicle. The second model represents a vehicle that moves in a plane, for example a mobile robot. As explained in Section 3.2.2, in practical application, this is a valid model for UAVs at constant altitude.

3.2.1 Kinematic equations

According to the notation introduced in section 3.1, the velocity of the vehicle expressed in the body fixed coordinates $\boldsymbol{v_{gs}}$ relates to the inertial frame by the following equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R(\boldsymbol{\eta}) \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

where $R(\boldsymbol{\eta})$ is the matrix that performs roll, pitch and yaw rotations, respectively. The rotation sequence is important, as a different order alters the final result.

$$R(\boldsymbol{\eta}) = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\theta & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\theta & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{3.1}$$

The body-fixed angular rates $\boldsymbol{w}$ relate to the derivatives of Euler angles according to

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T(\boldsymbol{\eta}) \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

where the matrix $T(\boldsymbol{\eta})$ is given by

$$T(\boldsymbol{\eta}) = \begin{bmatrix} c\theta & s\theta s\phi & s\theta c\phi \\ 0 & c\theta c\phi & -c\theta s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{3.2}$$

The system may be compressed into the following form:

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} R(\boldsymbol{\eta}) & 0 \\ 0 & T(\boldsymbol{\eta}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v_{gs}} \\ \boldsymbol{w} \end{bmatrix} \tag{3.3}$$

3.2.2 Simplified kinematic equations

In practical UAV missions, an independent altitude controller is responsible for the displacement in the z-axis and the movement is restricted to the xy plane. Thus the kinematic model presented in Section 3.2.1 can be reduced to the 2-dimensional space. In this case, the model of an unicycle is a good approximation to the UAV motion.

The unicycle is well studied in the literature, being a foundation for more complex systems. Approximating the UAV to an unicycle, removing its dynamics, is fairly used in research topics that are primarily concerned with the aircraft position. In fact, most mid to high level controllers reduce the complexity of aircraft's dynamics (Jackson, 2011). The vehicle and the coordinate frame evolved are presented in Fig. 3.2. The equations for the unicycle model are as follow:

$$\begin{aligned} \dot{x} &= u\cos\psi \\ \dot{y} &= u\sin\psi \\ \dot{\psi} &= r \end{aligned} \quad \Rightarrow \quad \dot{p} = R(\psi) \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{3.4}$$

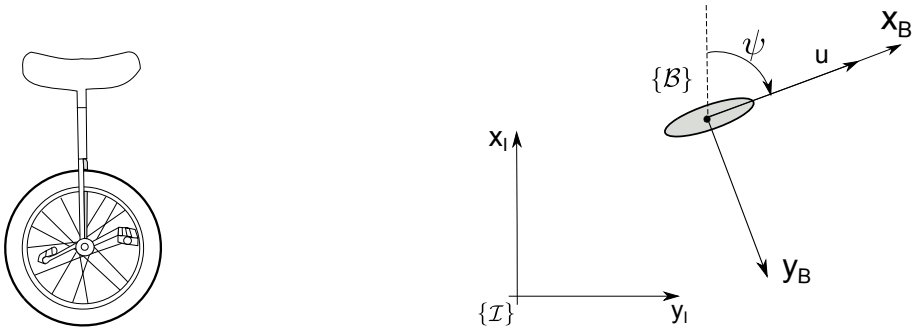


Figure 3.2: Unicycle kinematic model

Following the notation introduced in Section 3.1, u is the first component of the ground speed vector $\boldsymbol{v_{gs}}$. If the UAV is not subject to perturbations, then (3.4) is a sufficient model. However, taking advantage of the simplified 2D model, the wind component is introduced. The reader is referred to Fig. 3.3. The ground speed component may be decomposed as

$$\boldsymbol{v_{gs}} = \boldsymbol{v} + R^T(\boldsymbol{\psi})\boldsymbol{v_w} \tag{3.5}$$

where $\boldsymbol{v} = [v_a, 0]$ is the airspeed vector and $\boldsymbol{v_w} = [v_{w_x}, v_{w_y}]$ is the wind speed vector expressed in the inertial frame. Assume that the aircraft has approximately zero roll and pitch angles. The UAV kinematic equations yield

$$\begin{aligned}
\dot{x} &= v_a \cos\psi + v_{w_x} \\
\dot{y} &= v_a \sin\psi + v_{w_y} \\
\dot{\psi} &= r
\end{aligned}$$

or in a more compact fashion, the 2D model is expressed by

$$\begin{aligned}
\dot{\boldsymbol{p}} &= R(\psi)\boldsymbol{v} + \boldsymbol{v_w} \\
\dot{\psi} &= r
\end{aligned} \tag{3.6}$$

For the 3D kinematic model, with a slight abuse of notation, $r(t)$ and $q(t)$ are considered the yaw rate and pitch rate, respectively. Then, the system described by (3.3) can be rewritten in function of wind and airspeed by

$$\dot{\boldsymbol{p}} = R(\boldsymbol{\eta})\boldsymbol{v} + \boldsymbol{v_w} \tag{3.7a}$$
$$\dot{\theta} = q \tag{3.7b}$$
$$\dot{\psi} = r \tag{3.7c}$$

The size of the vectors must be consistent.

3.3 Dynamic modelling

This section borrows the dynamic models from (Roskam, 2001). The physical interpretation and mathematical analysis are not showed.

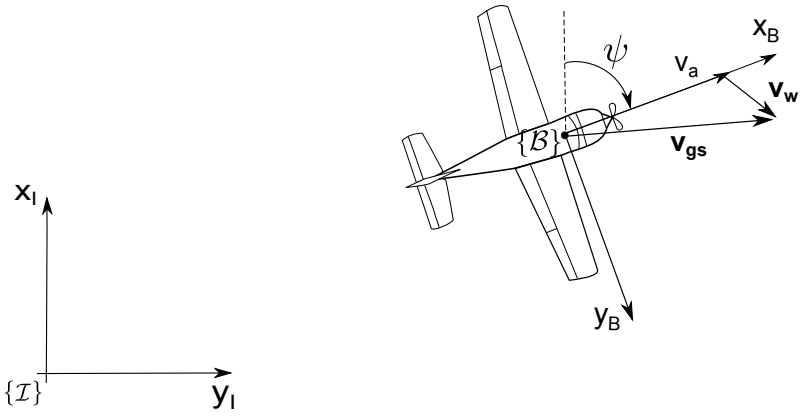


Figure 3.3: UAV Coordinate frame projected in the xy plane.

Neglecting the gyroscopic moments exerted by spinning rotors, the general linear and angular momentum equations in the body fixed axis $\{\mathcal{B}\}$ are given by:

$$F_x = m(\dot{u} - vr + wq) + mg\sin\theta + T \tag{3.8a}$$
$$F_y = m(\dot{v} + ru - pw) - mg\cos\theta\sin\phi \tag{3.8b}$$
$$F_z = m(\dot{w} + pv - qu) - mg\cos\theta\sin\phi \tag{3.8c}$$

$$M_x = I_x\dot{p} + I_{xy}\dot{r} + (I_z - I_y)qr + I_{xz}qp \tag{3.9a}$$
$$M_y = I_y\dot{q} + (I_x - I_z)pr + I_{xz}(r^2 - p^2) \tag{3.9b}$$
$$M_z = I_z\dot{r} + I_{xz}\dot{p} + (I_y - I_x)qp - I_{xz}qr \tag{3.9c}$$

where F_x, F_y and F_z are the linear aerodynamic forces on the vehicle, M_x, M_y and M_z are the momentum around the aircraft axis, m is the mass of the vehicle, I_x, I_y and I_z are the inertia relative to each axis and T the trust exerted by the engine.

3.3.1 Dynamics of Flight

In this work a Cessna C172SP Skyhawk, a wing-fixed aircraft, is employed. It contains three control surfaces know as *aileron, elevator* and *rudder* (see Fig. 3.4).

Ailerons lie on the main wings, near the flaps. They are in charge of rotating the wings, producing an angular moment over the longitudinal axis, known as rolling or banking rate. It works in a differentially manner, in order to roll the aircraft to the left, the right aileron lowers and the left aileron raises and vice-versa. Elevators are placed on the tail section. They produce angular moment on the lateral axis, being used to control the pitch angle. If the plane has to ascend, the elevators are raised. The rudder is responsible for the movement around the vertical axis, called heading or yaw rate. Tilting the rudder to one side or the other makes the plane turns in the same direction.

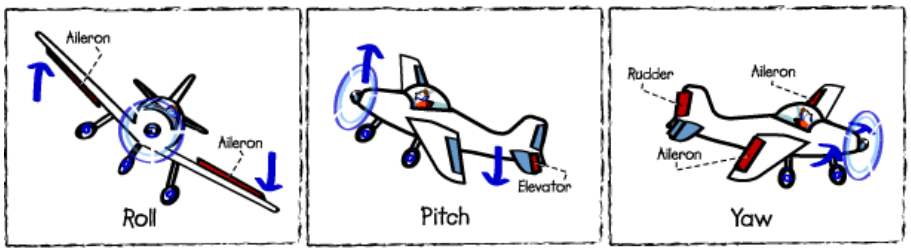


Figure 3.4: Roll, pitch and yaw on a wing-fixed airplane. Source: (Shaw, 2014)

In the framework of this dissertation the control surfaces available are ailerons and elevators. Therefore, to control the heading of the vehicle a technique known as *banked turn* or *coordinated turn* is employed. This concept is very used in aircraft and even ground vehicles applications, like high-speed railways and highways.

When an aircraft banks, the lift force F_z may be broken down in two components. The vertical component cancels out with the weight of the vehicle, so that the vehicle keeps a constant altitude. If the vertical forces are not equivalent, a possible drop in altitude occurs. The second component of the lift force acts in the radial direction. The aircraft tries to minimize the F_y to zero, since, in regular conditions, the plane do not drift. Assuming that the pitch angle is nearly zero ($\theta \approx 0$), the linear force (3.8b) on the y-axis becomes

$$0 = mrv_a - mg\sin\phi \tag{3.10}$$

The yaw rate is obtained from (3.2) assuming $\theta \approx 0$,

$$r = \dot{\psi}\cos\phi$$

Substituting in (3.10), the following equation is obtained:

$$m\dot{\psi}\cos\phi v_a = mg\sin\phi \tag{3.11}$$

$$\tan\phi = \frac{v_a\dot{\psi}}{g} \tag{3.12}$$

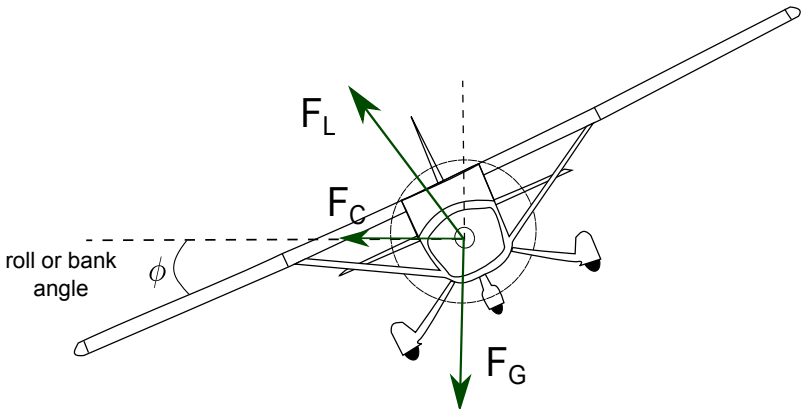


Figure 3.5: Important forces acting on a coordinated turn.

The equation described by (3.12) is known as *coordinated turn assumption*. For a more detailed reading and the acquaintance of high order model the reader is referred to (Jackson, 2011)

3.4 Cessna 172SP Skyhawk

The Cessna Skyhawk is the most popular aircraft in the world with more than 43,000 aircraft sold. It is a best-selling and the most-flown plane ever built (Cessna, 2013). It is well suitable for either students or professional pilots due to its agility, stability, robustness and high-strength construction. It is also widely used by military units all over the world for border patrolling and other purposes.

It requires a short runway for take-off/landing. The maximum cruising speed of the aircraft is $64\,m/s$. The maximum altitude is $4.267m$. The spatial range of the aircraft does not exceed $1.130km$ and the maximum payload is $416kg$. These data were collected under operation in International Standard Atmosphere (ISA). All presented information and more can be found in (Cessna, 2013).

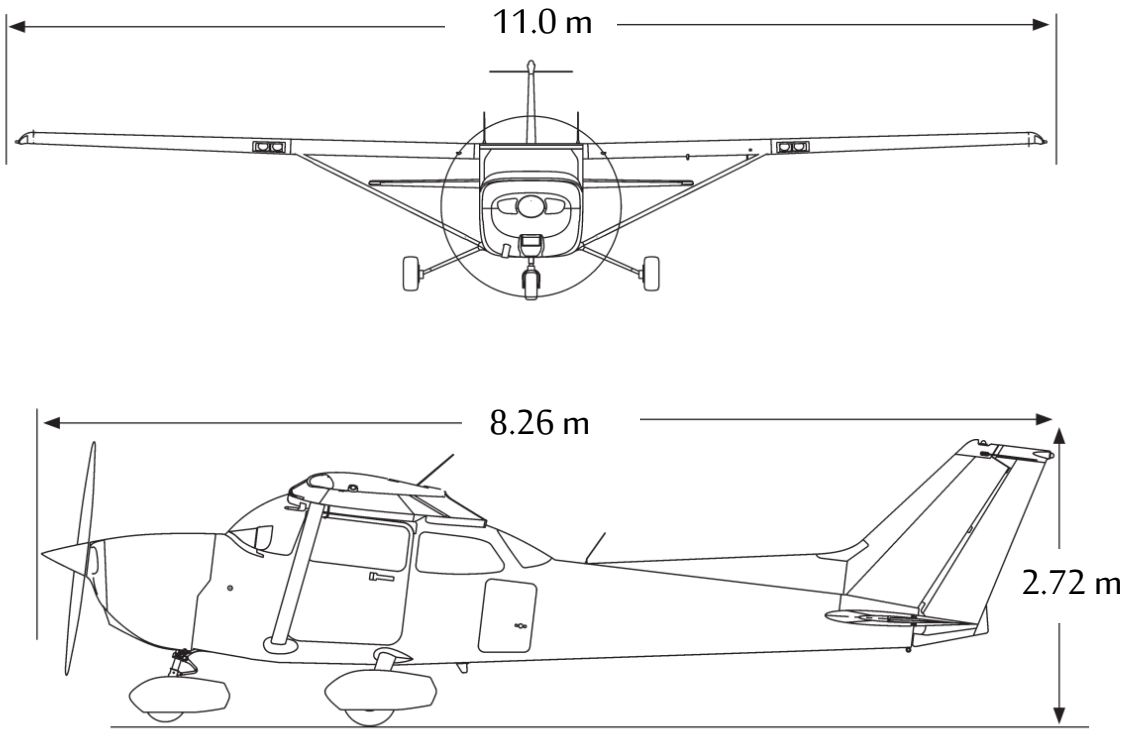


Figure 3.6: Cessna 172SP Skyhawk specifications. source: (Cessna, 2013)

4. UAV Motion Control

This chapter addresses the path following motion control problem. In Section 4.1 a brief overview of the topic is introduced. Then, Section 4.2 presents the first solution for the PF problem, namely carrot chasing algorithm. Section 4.3 addresses a nonlinear path following algorithm. The latter is employed in the coordination controller.

4.1 Introduction

Path following is a motion control problem concerned with driving the vehicle to a desired path parametrized in space, without temporal constraints. When contrasted to other motion control topics addressed in the literature, namely *trajectory tracking*[1], the path following shows some conveniences that makes it particularly interesting in the framework of this dissertation. The following advantages are pointed out by (Aguiar and Hespanha, 2007) and (Vanni, 2007).

- The vehicle is not imposed to move from its initial position to the starting point of the trajectory;
- Performance limitations due to unstable zero-dynamics can be avoided;
- Control signals are less likely to saturate.

The path following resolution typically lies in keeping a constant speed or follow a desired speed profile, while adjusting the orientation of the aircraft. The first is referred as dynamic assignment task and the second as geometric task.

In order to carry a performance comparison, two different solutions are addressed. Although not suitable for coordination control, the *carrot chasing*, presented in (Sujit et al., 2013), is a lightweight and robust to disturbances algorithm that solves the geometric task. It was chosen because its basic working principle does not require the introduction of further theoretical concepts. Afterwards, a nonlinear path following solution derived from (Aguiar and Hespanha, 2007) is presented. The latter solves both geometric and dynamic

[1]The trajectory tracking problem requires the vehicle to reach and follow a desired path associated with a timing law.

assignment task. Both strategies use a *virtual target point* (VTP) that moves along the path according to a specified law particular to each algorithm.

4.2 Carrot chasing control algorithm

Suppose the aircraft keeps a constant speed at a certain altitude. Its kinematic model is described as (3.6). Then, the path following problem may be formally stated as next.

Problem Statement 4.1. *Assume a desired spatial path* $\boldsymbol{p_d}(\gamma)$:$\mathbb{R} \to \mathbb{R}^2$, *where* $\gamma \in \mathbb{R}$ *is a parametric variable. Consider the following control input,* $u = r(t)$, *where* $r(t)$ *is the yaw rate. Design a feedback control law for* u *such that the position of the vehicle converges and follows the reference path, i.e.* $\|\boldsymbol{p} - \boldsymbol{p_d}\| \to \mathbf{0}$ *as* $t \to \infty$.

Since the vehicle cruises at a constant speed, this first approach is restricted to the geometric task. Applying trigonometric concepts, the carrot chasing solves the problem appropriately.

Consider the straight line path that joins an initial W_i and final waypoint W_{i+1}, as shown in Fig. 4.1. The main idea behind the carrot chasing algorithm is determining the point in the desired path that is nearest to the vehicle. This happens to be the orthogonal projection of the UAV on the path $\boldsymbol{p}^{\perp}$, which is also the desired position on the path $\boldsymbol{p_d}$. Once determined, the virtual target is placed at a distance δ from $\boldsymbol{p}^{\perp}$. The desired heading is computed based on the orientation of the vehicle in relation to the VTP. Then, a proportional controller derives the angular rate. The name of the algorithm comes from the fact the vehicle (the rabbit) is permanently chasing the VTP (the carrot). More details related to implementation are reported in Algorithm 1.

The distance from the vehicle to the nearest point in the desired path is called *cross-tracking error* (x-error). It is an important metric to evaluate guidance algorithms.

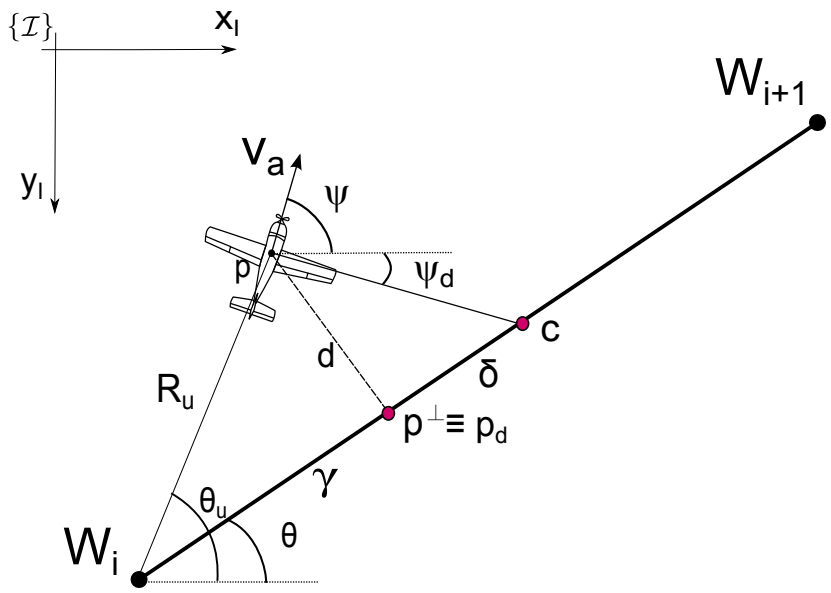


Figure 4.1: Carrot chasing frame

The parametric variable, consequently the desired position on the path, is determined by the position of the vehicle relatively to the trajectory. The position error $\|\boldsymbol{p} - \boldsymbol{p_d}\|$ converges to zero as $\|\boldsymbol{p} - \boldsymbol{c}\| \to \delta$. If wind is taken into account, then course angle and desired course angle must be considered instead of heading angle and desired heading angle, respectively.

Algorithm 1 Carrot Chasing

Initialize: $WP_i = (x_i, y_i)$, $WP_{i+1} = (x_{i+1}, y_{i+1})$, θ

1: $R_u = \|WP_i - p\|, \quad \theta_u = atan2(y - y_i, x - x_i)$

2: $\beta = normalizeAngle^{a}(\theta - \theta_u)$

3: $\gamma = \sqrt{R_u^2 - (R_u \sin\beta)^2}$

4: $[c_d, c_d] = [(\gamma + \delta)\cos\theta + x_i, (\gamma + \delta)\sin\theta + y_i]$

5: $\psi_d = k * atan2(c_d - y, c_d - x)$

6: $\dot{\psi} = w_1 = normalizeAngle(\psi_d - \psi)$

[a] Given an input angle, the function normalizeAngle returns an equivalent angle in the domain [-π,π]

4.3 Nonlinear Lyapunov based control algorithm

In this section a nonlinear Lyapunov based control law is developed. The derived solution considers both dynamic assignment and geometric tasks. In Section 4.3.1 the control for the motion in the xy plane is derived. Then, in Section 4.3.2, some additional notes for a vehicle that moves in the 3D space are reported.

4.3.1 Horizontal plane

Consider the same kinematic model described by (3.6). However, this time let the input control signal be

$$\boldsymbol{u} = [v_a(t)\ r(t)]$$

The path following dynamic assignment and geometric tasks are formally introduced.

Problem Statement 4.2. *Assume a pre-specified desired spatial path* $\boldsymbol{p_d}(\gamma) : \mathbb{R} \to \mathbb{R}^2$ *parametrized by* $\gamma \in \mathbb{R}$ *and* $v_d(\gamma) \in \mathbb{R}$*, a desired speed assignment. Suppose also that* $\boldsymbol{p_d}(\gamma)$ *is sufficiently smooth with respect to* γ *and its derivatives are bounded. Design a feedback control law for* $\boldsymbol{u}$ *and* $\ddot{\gamma}$ *such that i) the position of the vehicle converges and remains inside a tube centred around the desired path, i.e.* $\|\boldsymbol{p} - \boldsymbol{p_d}\| \to \|\boldsymbol{\epsilon}\|$ *where* $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]^T \in \mathbb{R}^2$ *is a nonzero constant vector that can be made arbitrary small and ii) the vehicle satisfies the desired speed assignment, i.e.* $\|\dot{\gamma} - v_d(\gamma)\| \to 0$.

Let Fig. 4.2 represent the path following problem in the horizontal plane. From Problem 4.2, $\boldsymbol{e}$, the error associated with the position of the vehicle and z, the error for the evolution of the parametric variable, can be defined according to

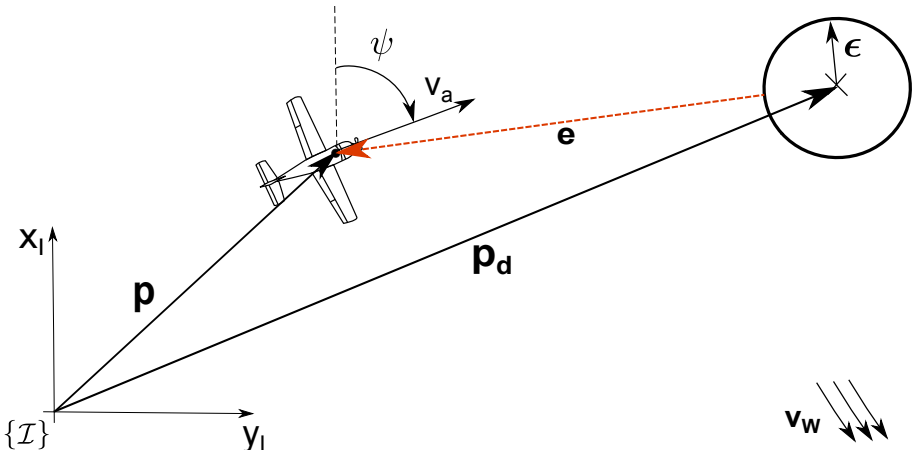


Figure 4.2: Path following frame on the xy plane

$$\begin{aligned} \boldsymbol{e} &= R^T(\psi)(\boldsymbol{p} - \boldsymbol{p_d}(\gamma)) - \boldsymbol{\epsilon} \\ z &= \dot{\gamma} - v_d(\gamma) \end{aligned} \tag{4.1}$$

The task consists in assuring that the error is ultimately bounded and, after a transient time, it converges to a region close to the origin. Define the composite error vector $\boldsymbol{e_c} = [\boldsymbol{e}, z]^T$. Derivating the position error with respect to obtain

$$\dot{e} = \dot{R}^T(\psi)(\boldsymbol{p} - \boldsymbol{p_d}(\gamma)) - R^T(\psi)(\dot{\boldsymbol{p}} - \dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma}) \tag{4.2}$$

Let the time derivative of the rotation matrix be

$$\dot{R}(\psi) = R(\psi)S(r) \tag{4.3}$$

where $S(r)$ is the skew-symmetric matrix $S(r) = \begin{bmatrix} 0 & -r \\ r & 0 \end{bmatrix}$. Substituting (3.6) and (4.1) in (4.2) yields

$$\dot{\boldsymbol{e}} = (R(\psi)S(r))^T(\boldsymbol{p} - \boldsymbol{p_d}(\gamma)) + R^T(\psi)(R(\psi)\boldsymbol{v} + \boldsymbol{v_w} - \dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma})$$

From (4.1), $(\boldsymbol{p} - \boldsymbol{p_d}(\gamma)) \rightarrow R(\psi)(\boldsymbol{e} + \boldsymbol{\epsilon})$

$$\begin{aligned} \dot{\boldsymbol{e}} &= S^T(r)R^T(\psi)R(\psi)(\boldsymbol{e} + \boldsymbol{\epsilon}) + R^T(\psi)R(\psi)\boldsymbol{v} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} \\ &= S^T(r)(\boldsymbol{e} + \boldsymbol{\epsilon}) + \boldsymbol{v} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} \\ &= S^T(r)\boldsymbol{e}^2 + S^T(r)\boldsymbol{\epsilon}^3 + \boldsymbol{v} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} \\ &= -S(r)\boldsymbol{e} + S(\boldsymbol{\epsilon})r + \boldsymbol{v} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{d\dot{\boldsymbol{p}}_{\boldsymbol{d}}(\gamma)}{\partial \gamma} \\ &= -S(r)\boldsymbol{e} + \begin{bmatrix} 1 & \epsilon_2 \\ 0 & -\epsilon_1 \end{bmatrix}\begin{bmatrix} v_a \\ r \end{bmatrix} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} \end{aligned}$$

Let $\Delta = \begin{bmatrix} 1 & \epsilon_2 \\ 0 & -\epsilon_1 \end{bmatrix}$. The position error dynamics is expressed by (4.4)

$$\dot{e} = -S(r)\boldsymbol{e} + \Delta\boldsymbol{u} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} \tag{4.4}$$

By now, it shall be clear to the reader why the vehicle was set to converge and remain inside a tube centered around $\boldsymbol{p_d}(\gamma)$, and not the desired position itself. If $\boldsymbol{\epsilon}$ had not been introduced, the control variable r (yaw rate) would not appear in (4.4). As a result, the error would not converge to zero.

Meanwhile the the dynamic of the parametric variable error is described by

$$\dot{z} = \ddot{\gamma} + \dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma}$$

In most mission scenarios, except for a transitional phase, the desired speed profile is set to be constant. In the scope of this text, it will be considered constant with no exception. As a consequence of that, we can simplify the notation to $v_d(\gamma) = v_d$ and the foregoing expression may be stated as

$$\dot{z} = \ddot{\gamma} \tag{4.5}$$

Proposition 4.1. *Consider the system described by (3.6) in a closed-loop with the control laws*

$$\begin{aligned} u &= \Delta^{-1}(R^T(\psi)v_d\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} - R^T(\psi)\boldsymbol{v_w} - K_p\boldsymbol{\epsilon}) \\ \ddot{\gamma} &= \boldsymbol{e}R^T(\psi)\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial \gamma} - k_\gamma z \end{aligned} \tag{4.6}$$

where $K_p = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}$ *is a diagonal matrix whose eigenvalues are positive and* k_γ *is a positive constant. The origin* $\boldsymbol{e_c} = \boldsymbol{0}$ *is a globally asymptotically stable equilibrium point for the system.*

Proof. Define a Lyapunov function V_e for the position error and a second Lyapunov function V_γ for the speed assignment error.

$$V_e = \frac{1}{2}\|\boldsymbol{e}\|^2 = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{4.7a}$$

$$V_z = \frac{1}{2}z^2 \tag{4.7b}$$

and define a composite Lyapunov function

$$\begin{aligned} V_c &= V_e + V_z \\ V_c &= \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} + \frac{1}{2}z^2 \end{aligned}$$

Then take the derivative of V_c with respect to time

$$\dot{V}_c = \dot{V}_e + \dot{V}_z = \boldsymbol{e}^T\dot{\boldsymbol{e}} + z\dot{z}$$

[2]From the definition of skew-symmetric matrix: $S^T(r)\boldsymbol{e} = -r \times \boldsymbol{e} = -S(r)e$

[3]From the definition of ske-symmetrix matrix and cross product: $S^T(r)\boldsymbol{\epsilon} = -r \times \boldsymbol{\epsilon} = \boldsymbol{\epsilon} \times r = S(\boldsymbol{\epsilon})r$

Substituting $\dot{\boldsymbol{e}}$ by (4.4) and $\dot{z}$ by (4.5) yields

$$\dot{V}_c = \boldsymbol{e}^T(-S(r)\boldsymbol{e} + \Delta\boldsymbol{u} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)\dot{\gamma}\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial\gamma}) + z\ddot{\gamma}$$

According to (4.1), $\dot{\gamma} = v_d + z$. Hence, the last equation becomes

$$\dot{V}_c = \boldsymbol{e}^T(-S(r)\boldsymbol{e} + \Delta\boldsymbol{u} + R^T(\psi)\boldsymbol{v_w} - R^T(\psi)v_d\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial\gamma}) + z(\ddot{\gamma} - \boldsymbol{e}^T R^T(\psi)\frac{\partial \boldsymbol{p_d}(\gamma)}{\partial\gamma})$$

Now let the control variables $\boldsymbol{u}$ and $\ddot{\gamma}$ be as expressed in (4.6)

$$\begin{aligned}
\dot{V}_c &= \boldsymbol{e}^T(-S(r)\boldsymbol{e} - K_p\boldsymbol{e}) + z(-k_\gamma z)\\
\dot{V}_c &= -\boldsymbol{e}^T S(r)\boldsymbol{e} - \boldsymbol{e}^T K_p\boldsymbol{e} + -k_\gamma z^2\\
\dot{V}_c &= -\boldsymbol{e}^T K_p\boldsymbol{e} - k_\gamma z^2 \leq -\boldsymbol{e_c}^T K_c\boldsymbol{e_c}
\end{aligned}$$

where $K_c = \begin{bmatrix} K_p & 0 \\ 0 & k_\gamma \end{bmatrix}$. The following inequalities hold true

$$\begin{aligned}
\frac{1}{2}\|\boldsymbol{e_c}\| \leq V_c \leq \frac{1}{2}\|\boldsymbol{e_c}\|\\
\frac{\partial V_c}{\partial t} + \frac{\partial V_c}{\partial \boldsymbol{e_c}} f(t, \boldsymbol{e_c}) \leq -\boldsymbol{e_c}^T K_c\boldsymbol{e_c} \leq 0
\end{aligned}$$

The function $\frac{1}{2}\|\boldsymbol{e_c}\|^2$ is radially unbounded and the conditions stated in Theorem 2.5 are satisfied. Therefore, the origin $\boldsymbol{e_c} = \boldsymbol{0}$ is a globally uniformly asymptotically stable equilibrium point. It is proved that the position error $\boldsymbol{e}$ converges to a neighborhood of $\boldsymbol{\varepsilon}$ and the speed assignment error z converges to zero.

□

The smaller the values of $\boldsymbol{\epsilon}$ are, the closest to the neighborhood of the desired path the vehicle converges. However, the input signal is pushed towards saturation. Analyzing the matrix Δ, the value ϵ_2 may be set to null, but ϵ_1 cannot be zero, or a singularity arises.

Fig. 4.3 represents the discussed path following controller. The parametric variable is an internal variable that determines the position of the VTP. The demonstrated control assures that the target moves slower if the vehicle is behind it. If the vehicle is ahead,

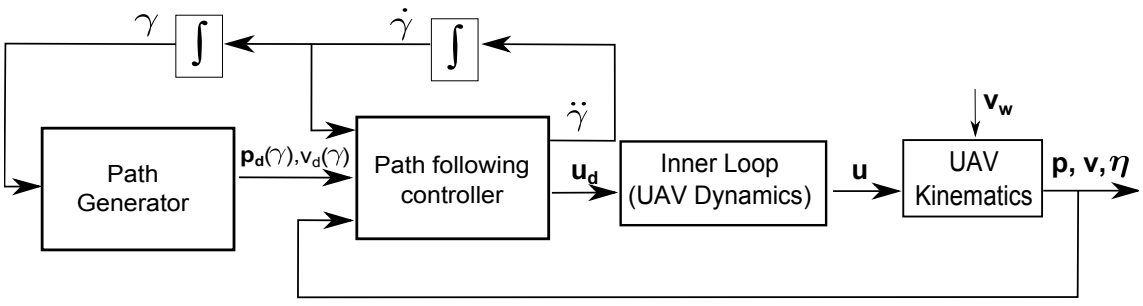


Figure 4.3: Nonlinear Lyapunov base path following controller

the target moves faster. Also, optimization techniques may be employed to determine the initial value $\gamma(0)$. A straightforward possibility is to set the initial value to the point in the path nearest to the vehicle. Controlling the parametric variable rate $\dot{\gamma}$ is essential for the coordination controller explored in Chapter 5.

4.3.2 3D space

Consider the kinematic model described by (3.7) for a vehicle that moves freely in the 3-dimensional space. Take the control input

$$\boldsymbol{u} = [v_a(t)\ \boldsymbol{w}(t)]$$

where $\boldsymbol{w}(t) = [p, q, r]$ is the body angular velocity vector. Let the variables and matrices in Section 4.3.1 defined in $\mathbb{R}^2$ and $\mathbb{R}^{2\times 2}$ be restated to its equivalents in $\mathbb{R}^3$ and $\mathbb{R}^{3\times 3}$, respectively. Also, assume

$$\Delta = \begin{bmatrix} 1 & 0 & -\epsilon_3 & \epsilon_2 \\ 0 & \epsilon_3 & 0 & -\epsilon_1 \\ 0 & -\epsilon_2 & \epsilon_1 & 0 \end{bmatrix} \tag{4.8}$$

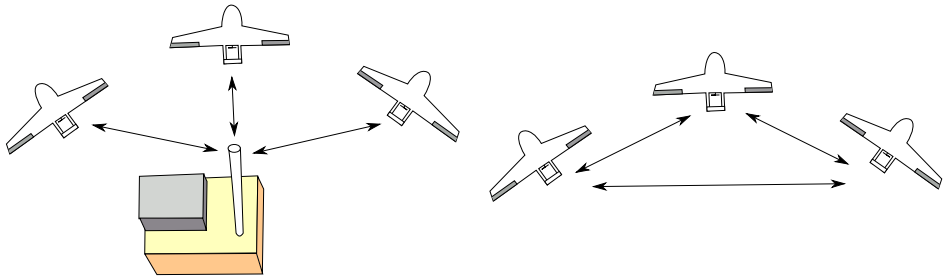
and $\bar{\Delta} = \Delta^T(\Delta\Delta^T)^{-1}$ its pseudo-inverse. Then, consider Problem 4.2, where the desired path is defined as $\boldsymbol{p_d}(\gamma) : \mathbb{R} \to \mathbb{R}^3$. The origin $\boldsymbol{e_c} = \boldsymbol{0}$ of the closed loop system with control law (4.6) is a globally stable equilibrium point. The proof is not addressed, since it is similar to the one in the previous section.

5. Cooperative Motion Control

In this chapter the cooperative control of a fleet of autonomous vehicles, which forms the core of this dissertation, is addressed. Section 5.1 clarifies the context in which the controller is inserted. Then, in Section 5.2 the coordination problem is formally stated. Next, the problem is tackled in the frame of two different topologies. Section 5.3 assumes a constant communication topology and Section 5.4 assumes a switched communication topology.

5.1 Introduction

According to the decision-making level, the coordination controller is distinguished as centralized or decentralized. Fig. 5.1a shows a centralized architecture with four nodes, namely three UAVs and a ground station, which is responsible for taking decisions. Centralized controllers are not robust, as a single failure point may lead to undesired behavior of the entire system. In Fig. 5.1b a decentralized architecture is illustrated. These controllers are not as vulnerable to failure points as the centralized model. In addition, the solution may be less complex, since each controller does not need to take directly into account all the vehicles in the fleet. This work focuses on decentralized strategies.



(a) Centralized Controller (b) Decentralized Controller

Figure 5.1: Different topology for coordination controllers

In Chapter 4, closed loop control laws for the motion control problem of a single UAV were discussed. In particular, in the solution addressed in Section 4.3, the vehicle approaches a virtual target that moves along the desired path, according to a reference speed profile $v_d(\gamma)$. In this chapter, the speed profile of the VTP is redefined, in order to

achieve cooperation among multiple vehicles.

5.2 Problem Statement

Consider a fleet of n vehicles denoted by the set $\mathcal{N} = \{1, \ldots, n\}$. The parametric variable γ_i defines the position of the VTP associated to the *ith* vehicle in a precise manner. Each vehicle converges to its respective virtual target point. Therefore, if the n virtual target points asymptotically synchronize, the vehicles asymptotically reach a desired formation. In other words, coordination is achieved if and only if $\gamma_{ij} = \gamma_i - \gamma_j = 0$, $\forall_{i,j} \in \mathcal{N}$. The parameter γ_i is said to be the coordination state.

The dynamic of each state is strictly dependent of $v_d(\gamma)$. Therefore the definition of desired speed profile is extended to

$$\boldsymbol{v_d} = v_L \mathbf{1} + \tilde{\boldsymbol{v}}_{\boldsymbol{r}} \tag{5.1}$$

where the elements of $\boldsymbol{v_d} = [v_{d1}, \ldots, v_{dn}]$ and $\boldsymbol{v_r} = [v_{r1}, \ldots, v_{rn}]$ correspond to desired and correction speeds of each vehicle $i \in \mathcal{N}$, respectively. The formation speed, denoted by $v_L(\gamma)$, is common to all vehicles in the flock.

Let $\mathcal{N}_i$ be the set of vehicles that the *ith* UAV is able to receive information from. Recall from the graph theory concepts introduced in Section 2.3 that is not necessarily true that $j \in \mathcal{N}_i \Rightarrow i \in \mathcal{N}_j$, as unidirectional communication is considered.

Problem Statement 5.1. *Assume that for each vehicle $i \in \mathcal{N}_i$, the variables γ_i and γ_j, $j \in \mathcal{N}_i$ are available. Derive a control law for $\tilde{v}_{ri}$, such that, for all $i, j \in \mathcal{N}$, $(\gamma_i - \gamma_j)$ and $(\dot{\gamma}_i - \dot{\gamma}_j)$ converges to zero as $t \to \infty$.*

5.3 Constant communication

Assume a quasi strongly connected communication topology, i.e. there is a transmitter vehicle from whom any other element in the fleet receives information. Suppose that this topology is constant over time and let L be the associated Laplacian matrix. The coordination error for each vehicle is defined as

$$\xi_i = \sum_{j \in \mathcal{N}_i} \gamma_i - \gamma_j$$

Consider the coordination error vector $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_n]$ and the coordinate state vector $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_n]$. Then,

$$\boldsymbol{\xi} = L\boldsymbol{\gamma} \tag{5.2}$$

The dynamic of the coordination error is given by

$$\dot{\boldsymbol{\xi}} = L\dot{\boldsymbol{\gamma}}$$

Substituting (5.1) in (4.1) and applying to the dynamic of the coordination error, yields

$$\dot{\boldsymbol{\xi}} = L(\boldsymbol{z} + v_L\mathbf{1} + \tilde{\boldsymbol{v}}_{\boldsymbol{r}})$$

with $\boldsymbol{z} = [z_1, \ldots, z_n]$. Recalling the fact that $L\mathbf{1} = \mathbf{0}$, we obtain

$$\dot{\boldsymbol{\xi}} = L\boldsymbol{z} - L\tilde{\boldsymbol{v}}_{\boldsymbol{r}} \tag{5.3}$$

Proposition 5.1. *Consider n vehicles driven by the control law specified in (4.6). Let the connected inter-vehicle communication topology be represented by the Laplacian matrix L. Assume that the vehicles exchange their coordination state γ_i continuously with the neighbouring nodes. Let the coordination control law be*

$$\tilde{\boldsymbol{v}}_{\boldsymbol{r}} = -K_\xi L\boldsymbol{\gamma} \tag{5.4}$$

where K_ξ is a diagonal matrix whose elements are positive. Then, the coordination error $\boldsymbol{\xi}$ and the speed profile converge to a neighbourhood of the origin and v_L, respectively, as $t \to \infty$.

Proof. Consider the candidate Lyapunov function

$$V_\xi = \frac{1}{2}\|\boldsymbol{\xi}\|^2$$

and its time derivative

$$\dot{V}_\xi = \boldsymbol{\xi}^T\dot{\boldsymbol{\xi}}$$

Substitute (5.2) in the control law (5.4) and the remainder result in the time derivative of the candidate Lyapunov function

$$\begin{aligned}\dot{V}_\xi &= \boldsymbol{\xi}^T(L\boldsymbol{z} - LK_\xi\boldsymbol{\xi}) \\ &= \boldsymbol{\xi}^T L\boldsymbol{z} - \boldsymbol{\xi}^T LK_\xi\boldsymbol{\xi} \\ &= \boldsymbol{\xi}^T L\boldsymbol{z} - (1+\theta-\theta)\boldsymbol{\xi}^T LK_\xi\boldsymbol{\xi} \\ &= -(1-\theta)\boldsymbol{\xi}^T LK_\xi\boldsymbol{\xi} + \boldsymbol{\xi}^T L\boldsymbol{z} - \theta\boldsymbol{\xi}^T LK_\xi\boldsymbol{\xi}\end{aligned}$$

where θ is any variable such that $0 \leq \theta \leq 1$. Assume that the control law expressed by (4.6) assures that the speed assignment error vector $\boldsymbol{z}$ remains bounded. It is then possible to see that for a $\boldsymbol{\xi}$ large enough such that

$$\|\boldsymbol{\xi}\| \geq \|\frac{1}{\theta}K_\xi^{-1}\boldsymbol{z}\|$$

we obtain

$$\dot{V}_\xi \leq -(1-\theta)\boldsymbol{\xi}^T LK_\xi\boldsymbol{\xi} \leq 0$$

The solutions for the equality are $\boldsymbol{\xi} = \mathbf{0}$ and $\boldsymbol{\xi} = \mathbf{1}$. The latter solution requires $\boldsymbol{\xi} = L\boldsymbol{\gamma} = \mathbf{1}$. However, the unit vector spans the kernel of L, i.e. $L\mathbf{1} = \mathbf{0}$. Therefore, as it cannot be a base of its image, the solution is impossible. Applying Theorem 2.7, we can conclude that $\boldsymbol{\xi}$ is ISS with respect to z. Further, it follows that

$$\|\boldsymbol{\xi}\|_a \leq \frac{1}{\theta}K_\xi^{-1}\|\boldsymbol{z}\|$$

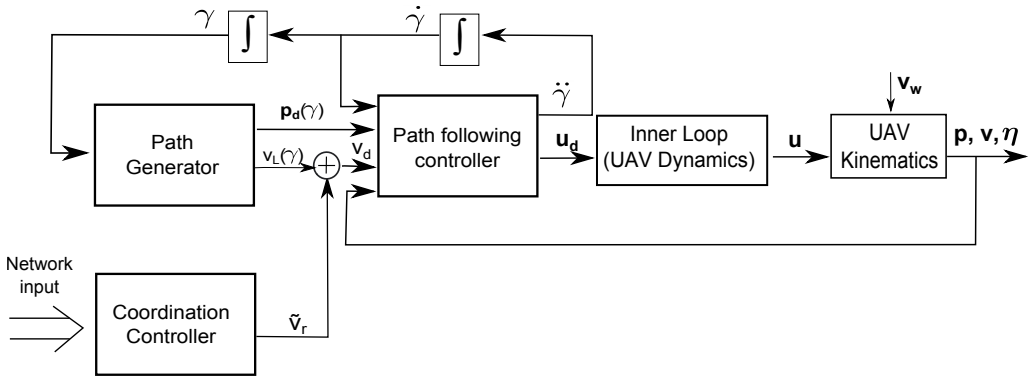


Figure 5.2: Cooperative path-following controller

The matrix K_ξ^{-1} is also a diagonal matrix. Its elements corresponds to the inverse eigenvalues of K_ξ. Therefore, higher eigenvalues reduce the neighborhood to which $\boldsymbol{\xi}$ converges. Substituting the fore inequality in (5.4) yields

$$\|\tilde{\boldsymbol{v}}_{\boldsymbol{r}}\|_a \leq \frac{1}{\theta}\|\boldsymbol{z}\|$$

The correction speed is also bounded. It is possible to conclude that the equilibrium point $\boldsymbol{\xi} = \mathbf{0}$ is uniformly stable. □

The cooperative path-following controller is shown in Fig. 5.2.

5.4 Switching communication topology

In most communication topologies, no vehicle can simultaneously communicate with the entire fleet. Also, it is not possible for a vehicle to continuously communicate with another, as intermittent failures are inherent to the nature of the communication system and/or a time division multiplexing strategy may be used to access the medium. Therefore, considering a time-varying communication network among the vehicles is a more reliable approach.

Recalling the ideas of switching topology presented in Section 2.3.4, consider a piece-wise constant switching signal $p(t)$, whose discontinuities are apart from each other by a minimum time span $\tau > 0$, called *dwell time*. The communication topology may fail to be connected at any time instant. However, over a defined period $T > 0$, the union graph $\mathcal{G}_{p(t)}$ is uniformly quasi strongly connected.

Assume that the dynamics of the parametric variable are given by:

$$\dot{\boldsymbol{\gamma}} = \boldsymbol{v_d} \tag{5.5}$$

The following *agreement* theorem is borrowed from (Lin et al., 2007)

Theorem 5.1. *Assuming that the union graph of the communication topology is UQSC, the closed loop control law*

$$\dot{\boldsymbol{\gamma}} = -KL_p\boldsymbol{\gamma} \tag{5.6}$$

assures that the coordination errors γ_i-γ_j, $\forall\, i, j \in \mathcal{N}$ converge exponentially fast to zero and $\dot{\boldsymbol{\gamma}} \to \mathbf{0}$ as $t \to \infty$.

In (Ghabcheloo et al., 2009), the authors generalize the theorem for $v_L \neq 0$. Assume (5.5) and

$$\dot{\boldsymbol{\gamma}} = v_L \mathbf{1} - K L_p \boldsymbol{\gamma} \tag{5.7}$$

and the change of variables

$$\tilde{\boldsymbol{\gamma}} = \boldsymbol{\gamma} - \mathbf{1} \int_1^t v_L(\tau) d\tau$$

Substituting (5.7) in the derivative of the fore mentioned equation, obtain

$$\begin{aligned} \dot{\tilde{\boldsymbol{\gamma}}} &= v_L \mathbf{1} - K L_p \boldsymbol{\gamma} - v_L \mathbf{1} \\ &= -K L_p \boldsymbol{\gamma} \end{aligned}$$

Applying Theorem 5.1, $\tilde{\gamma}_i$-$\tilde{\gamma}_j$ and $\dot{\tilde{\boldsymbol{\gamma}}}$ converges to zero as $t \to \infty$. Consequently, γ_i-γ_j and $\dot{\boldsymbol{\gamma}}$ converge exponentially fast to zero and v_L, respectively, as $t \to \infty$.

These results are also valid when the dynamics of the parametric variable are given by the control law (4.6). However, the coordinate vector $\dot{\boldsymbol{\gamma}}$ does not converge to $v_L \mathbf{1}$, but to a neighborhood close to that value.

6. Software Architecture

This chapter discusses the development of a software environment suitable for evaluating guidance algorithms based on the pre-existing plataform *Flight Variable Management System* (FVMS). In Section 6.1, different methodologies to validate models are confronted. The importance of a more accurate software platform relatively to traditional numerical simulators is explained. Next, Section 6.2 discusses the FVMS, where its essential components, Microsoft Flight Simulator and FSCUIP are also introduced. Section 6.3 presents the contribution of this work to the FVMS platform.

6.1 Introduction

Developing or validating existing algorithms requires simulation and analysis. There are well known numerical simulators that offer toolboxes and a range of facilities to implement and run algorithms. In the scope of this dissertation, *Matlab*, developed by *Mathworks*, was the chosen platform.

Under some abstractions and assumptions, the model may be simplified and the simulation runtime kept short. However, the complexity of an implementation in traditional numerical simulators increases drastically as more precise models, that take into account the cross nature of the movements, constraints imposed by dynamics, environment perturbation, etc., are demanded. In this case, the variables could become unmanageable, the simulation runtime intolerable or an implementation could be even impossible. As pointed out in (Demers et al., 2007), even if no simplification is assumed, few details may be lost during the modeling phase or coding in the simulation platform.

An ineffective validation may leads to drawbacks during field test with real UAVs. (Sampaio et al., 2013a) 1) The vehicle may misbehave, damaging itself or, more worrisome, people in the vicinity. 2) The battery imposes stringent time limits for a deep debugging process. 3) Time and cost expenses to move an engineering team to mission scenarios are also required.

The *software-in-the-loop*, shortened as SiL, is a tool that allows validating models in a precise fashion before field tests. It is characterized by the low cost of computer simulators allied with the reliability of hardware emulators. Moreover, it is a solution where the accuracy of the model and the simulation speed do not compete (Demers et al.,

2007), unlike traditional numerical simulators.

In an academic environment, as real UAVs are too expensive for a lecture budget, the SiL is a tool to teach and understand better the concepts behind real UAVs.

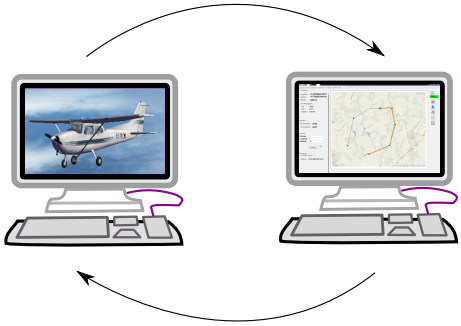


Figure 6.1: Software in the loop simulation

6.2 Flight Variable Management System

The FVMS is a recent but well-succeeded program to evaluate controllers for autonomous UAVs. The application was fully developed by ART Team at Engineering School of São Carlos (EESC-USP). It consists in a custom *Graphical User Interface* (GUI) that allows SiL simulation using Microsoft Flight Simulator 2004 (MSFS04) flying dynamics. The FVMS was originally written in C/C#. For the reasons enumerated in Sampaio et al. (2014a), the platform migrated to Borland Delphi 7.0.

It was first presented in *Aeroconf 2013*. This is well documented in (Sampaio et al., 2013a), where the architecture and potentialities are highlighted. The dynamic behavior of a quadcopter is fully modeled in the MSFS04. Sensors are emulated and the aircraft controlled in closed loop with different methods. In (Sampaio et al., 2013b) an optimal MIMO H_∞ robust controller to stabilize the attitude of a quadcopter is demonstrated. In (Sampaio et al., 2014a), the FVMS is configured in a *hardware in the loop* (HiL) fashion.

6.2.1 Microsoft Flight Simulator

The Microsoft Flight Simulator was launched in 1982 and, since then, the simulator has been continuously updated. It is a robust flight simulator provided with both game features and aerospace engineering. This powerful software is capable of mimicking with great accuracy the dynamics of an aircraft and perturbations from outer environment.

The weather can be configured, simulating rain, snow, clear skies, as well as gust, atmospheric instability, turbulence and more complex disturbances. It also includes flight analysis tools that record data such as altitude, heading, speed and more, like a real-world plane. Combined with outstanding graphical scenery database, the fidelity of MSFS makes of it an asset.

The choice for the specific version Microsoft Flight Simulator 2004 lies in the following reasons:

- This version requires less processing resources than newer version. Nonetheless the graphical response is satisfactory.
- In (Louali et al., 2011) the authors analyse the real-time profile of MFSF04 with a HiL architecture. The results show that MSFS04 behaves like a real-time system with sampling rate up to a 75Hz and standard deviation about 100ns.
- The number of variables available through the communication library is larger for the 2004 version.

The MSFS supports MDL extension. Therefore, it is relatively easy to add custom 3D models. Through an CFG file, aircraft parametric information like mass, inertia moment, wing span, etc. are set up. The first model successfully implemented by the ART Team was the german UAV AscTec Pelican.

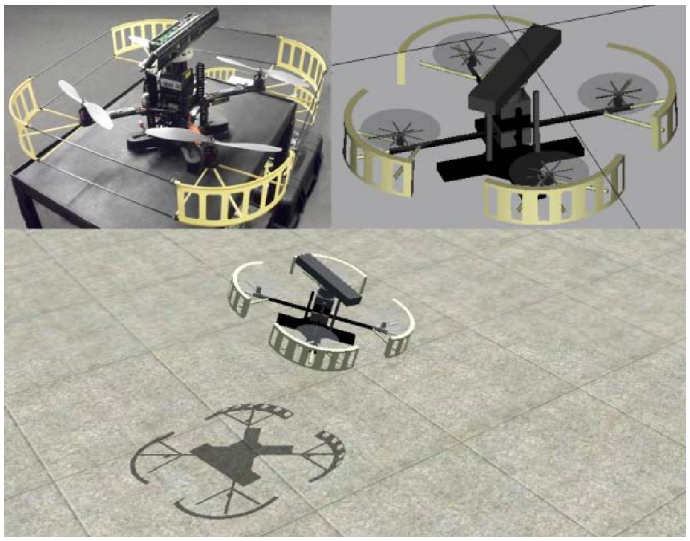
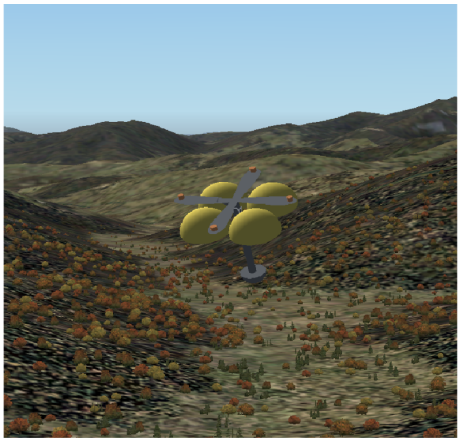


Figure 6.2: AscTec Pelican Quadcopter (top left), its 3D virtual representation (top right) and the vehicle on the MSFS04 environment. Source: (Sampaio et al., 2013a)

The possibility of modelling dynamic systems in MSFS04 is a crucial feature for the ART team, as great effort is pushed towards new UAV morphologies. Prior to field test, each new morphology parameters are estimated. Then, the vehicle is fully modelled in MSFS04 for validation in SiL. Fig. 6.3a shows the *SquidCop* in the MSFS04 environment. This vehicle, described in (Sampaio et al., 2014b), was primarily designed for in-flight launch by a second UAV or manned craft. The center of gravity lies at the lower end of the beam. Consequently, it dynamics behaves like a regular pendulum. This is a major improvement from the control point of view, as the system is globally asymptotically stable. Illustrated in Fig. 6.3b, the *Sharky* is a motor glider quadcopter well suited for in-flight or vertical take-off and landing (VTOL). It may also fly in a wing-fixed fashion. The reader is referred to (Sampaio et al., 2014c) for further details.

The last vehicle illustrated here is a native aircraft from MSFS04, the Cessna 172SP Skyhawk. The virtual aircraft inherited the popularity of the real one. The Skyhawk ranks among the most popular aircraft on the simulator.



(a) Squidcop quadcopter. Source: (Sampaio et al., 2014b)

(b) Sharky aerial vehicle. Source: (Sampaio et al., 2014c)

Figure 6.3: The models on MSFS of new UAV morphologies developed by ART Team.

Figure 6.4: Cessna 172SP Skyhawk in MSFS04

6.2.2 FSUIPC

The Flight Simulator Universal Inter-Process Communication (FSUIPC) is a Dynamic Link Library (DLL) written by Peter Dawson. It enables full duplex communication between MS Flight Simulator and FVMS. In Sampaio et al. (2013a), the authors demonstrate that FSUIPC provides reliable communication for frequencies up to 250Hz. More information about the DLL can be found in Downson (2012).

6.2.3 Conceptual representation

A conceptual diagram of the platform is depicted in Fig. 6.5. The UAV system is emulated by MSFS. It produces real-time data from a wide range of sensors and receives actuating signals. The FVMS is responsible for controlling, monitoring and logging simulation data. In between, there is the communication layer. The exchange of variables is available through (FSUIPC).

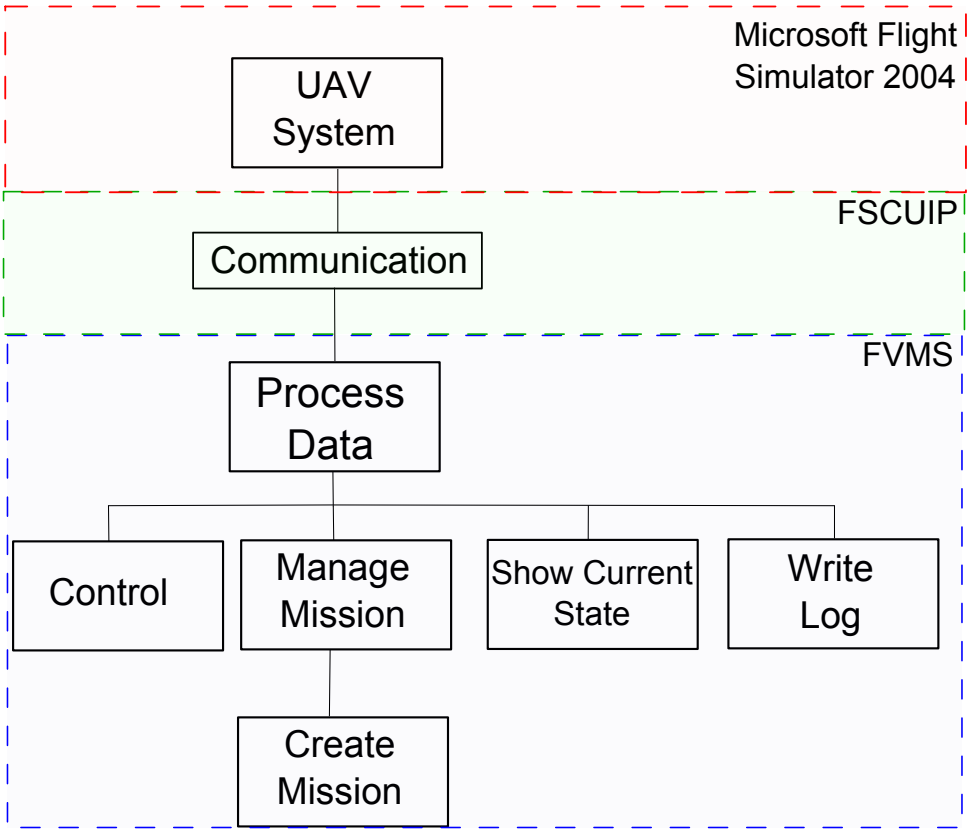


Figure 6.5: FVMS Conceptual Diagram

6.3 GNC Module

In this section the FVMS guidance, navigation and control module is introduced. The GNC problem is as shown in Fig. 6.6. The navigation estimates the vehicle position and attitude in a specified coordinate frame. Commanding the velocity and attitude, the guidance steers the vehicle from its current position to a desired reference path with a

given speed profile. The control computes the actuating inputs based on the reference determined by the guidance system.

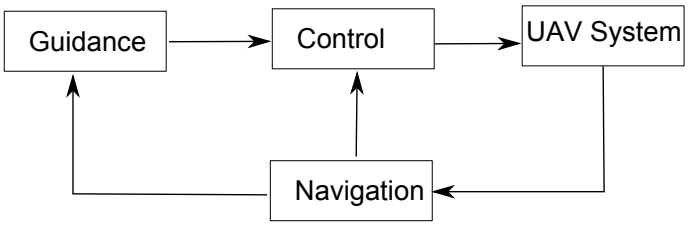


Figure 6.6: Guidance, navigation and control

The FVMS architecture is efficient in respect to time response (Sampaio et al., 2013a). However, to fully explore the potential of the platform, that offers more than 2000 control variables, a more transparent and modular organization is required. In accordance with ART Team objectives, the first step into a slightly different architecture was taken in this dissertation. The focus was the guidance system, although some control and navigation tools were also developed.

6.3.1 Architecture

The GNC architecture proposed is as shown in Fig. 6.7. The variables produced by the Flight Simulators are stored in a particular memory address of the PC running the application (step 1). Through FSUIPC, FVMS is able to intercept the memory and read the desired variables (step 2). Information like GPS position and orientation are available in the *Navigation library* (step 3). This library is responsible for processing, converting and making them available in the FVMS environment (step 4). The guidance algorithm requests the vehicle status and computes the desired output according to a specified control law. In order to send the actuation vector, the guidance task uses services offered by the *Control library* (step 5). This library processes the output signal and sends it to the simulator through the communication layer (step 6). The FSUIPC writes the data in the correct memory address (step 7). The MSFS captures the updated signals, which are the input for the dynamics of the simulated UAV (step 8). The graphic interface is refreshed with the new pose of the vehicle (step 9).

There are different libraries yet to be coded. The objective is to increase overall communication transparency, so designers can focus exclusively on control and high-level aspects.

6.3.2 GNC Components

The GNC module empowers the designer to implement and evaluate custom GNC algorithms. The module also provides high level functions for mission management and user interface, which enable interaction, experiment replication and on-line data analysis. Fig. 6.8 illustrates the information flow among the tasks in the module. It follows a brief overview of the different compenents. For more information about the functions and types, the reader is reffered to Appendix A

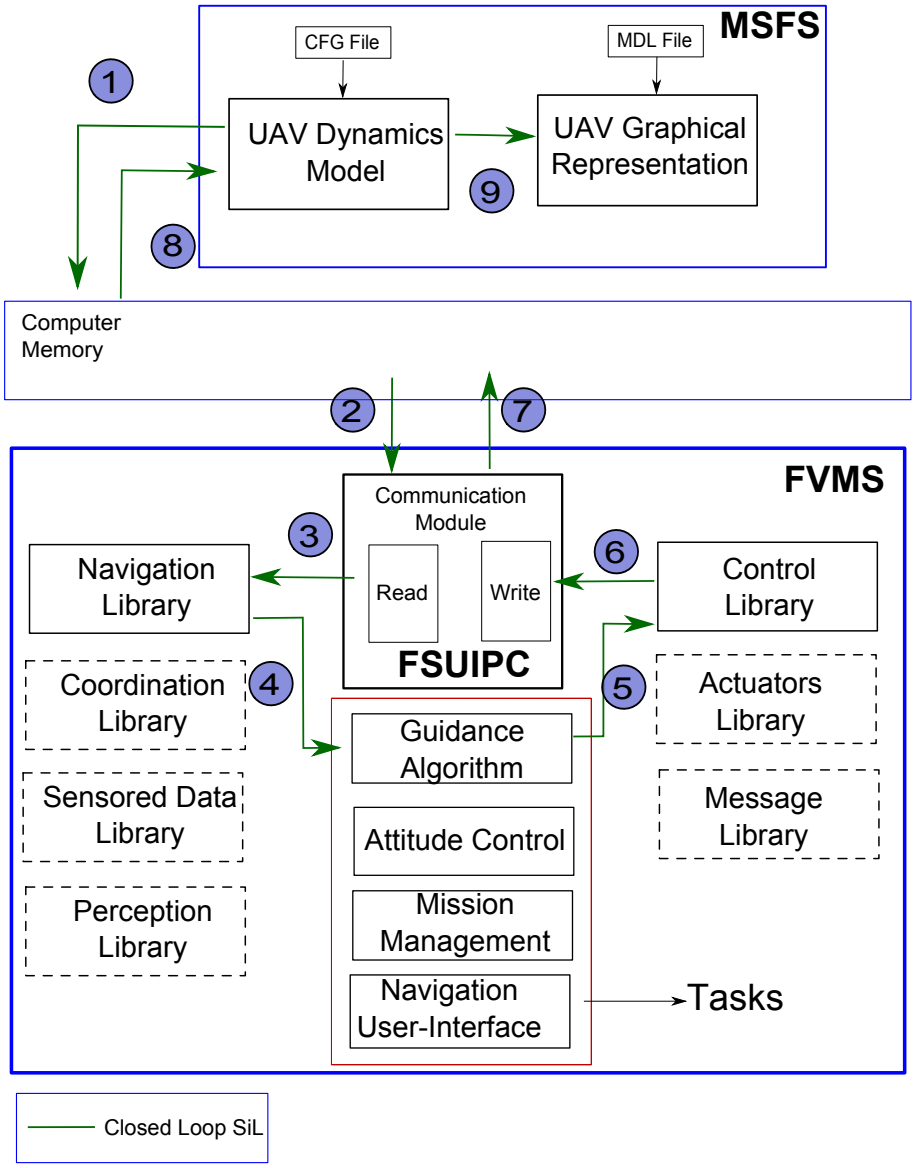


Figure 6.7: FVMS architecture

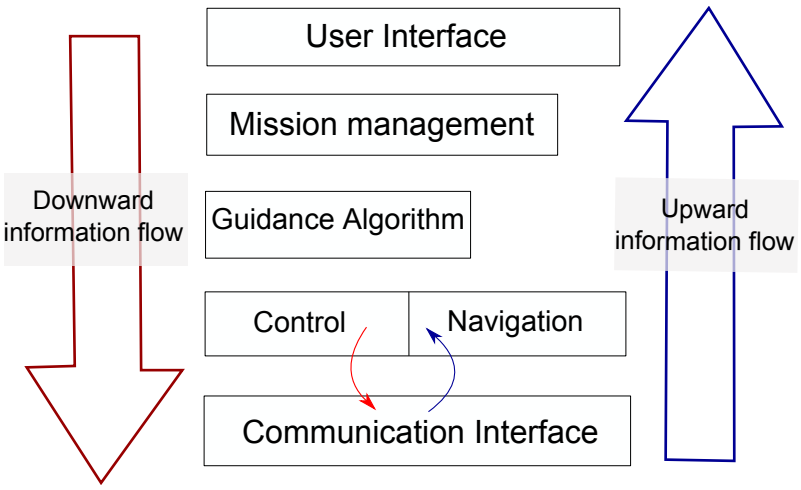


Figure 6.8: GNC module information flow

User Interface

The user interface was developed with the support of Google Maps API. Based on this javascript interface, the user selects the desired waypoints by clicking on the map. The WGS84 latitude, longitude and altitude of the waypoint may be manually inserted if replication is required. It is possible to choose whether the waypoint is a straight line or a loiter (circle pattern) with a given radius. The path is automatically generated and the flight plan saved on a specific structure.

The FVMS GNC user interface is depicted in Fig. 6.9. On the left panel, information about attitude, position and speed is displayed. The map is exposed on the center. It contains an animation of the vehicle and footprint, VTP and additional features as displayed. On the left most panel it is possible to verify the connection with MFSF04. Buttons allow executing, stopping and cancelling a mission. More functionalities are indicated in the referred figure.

Mission Management

When the user runs a mission the mission management task is activated. It receives the flight plan and manages its execution. The initial and final waypoint, cross-tracking error, azimuth angle, distance to target, path pattern and radius are stored in a structure. The fore mentioned structure is sent to the desired guidance control algorithm. Currently, FVMS supports two guidance control algorithms. By default, a nonlinear Lyapunov Based Control is responsible for the guidance system. If the distance to the desired waypoint reaches a minimum value (default 100m), the target waypoint changes to the next in the list. If the cross-tracking error grows too large (default 1000m), the mission manager automatically changes to default guidance control into loiter mode.

Guidance Algorithm

The guidance algorithm is responsible for executing a path. The controller designer is free to develop new guidance algorithms. The variables related to the path are sent by the mission manager. The navigation library provides the vehicle current status. Once the control law is determined the control library offers services to send desired angular position and airspeed. Therefore, the communication layer is transparent to designers, who can focus on the kinematic problem.

Navigation

The navigation library offers a set of data from MFSF04. It is an extension of class SENSOR from previous versions, emulating GPS, IMU, Gyro, altimeters and other sensoring devices present in real UAVs. As discussed in Sampaio et al. (2013a), MSFS reproduces real sensors affected by noisy influence of disturbing environmental conditions.

The navigation library provides for FVMS tasks updated information about vehicle status received from MSFS. In order to use the library, the *FVMSNavigation* must be declared in the use section.

Control

The control library is responsible for generating the actuation signals for MSFS. It is possible to send roll, pitch, yaw and airspeed to the simulator. By the writing of this dissertation, the FVMS control library supports two vehicles. An H_∞ attitude controller for quadcopters is available. (for more information consult Sampaio et al. (2013b)). In addition, PID controllers were developed and validated for attitude and speed control of a Cesnne Skyhawk wing-fixed aircraft. These features are also available for any FVMS task. To use the library, the *FVMSNavigation* must be declared in the use section.

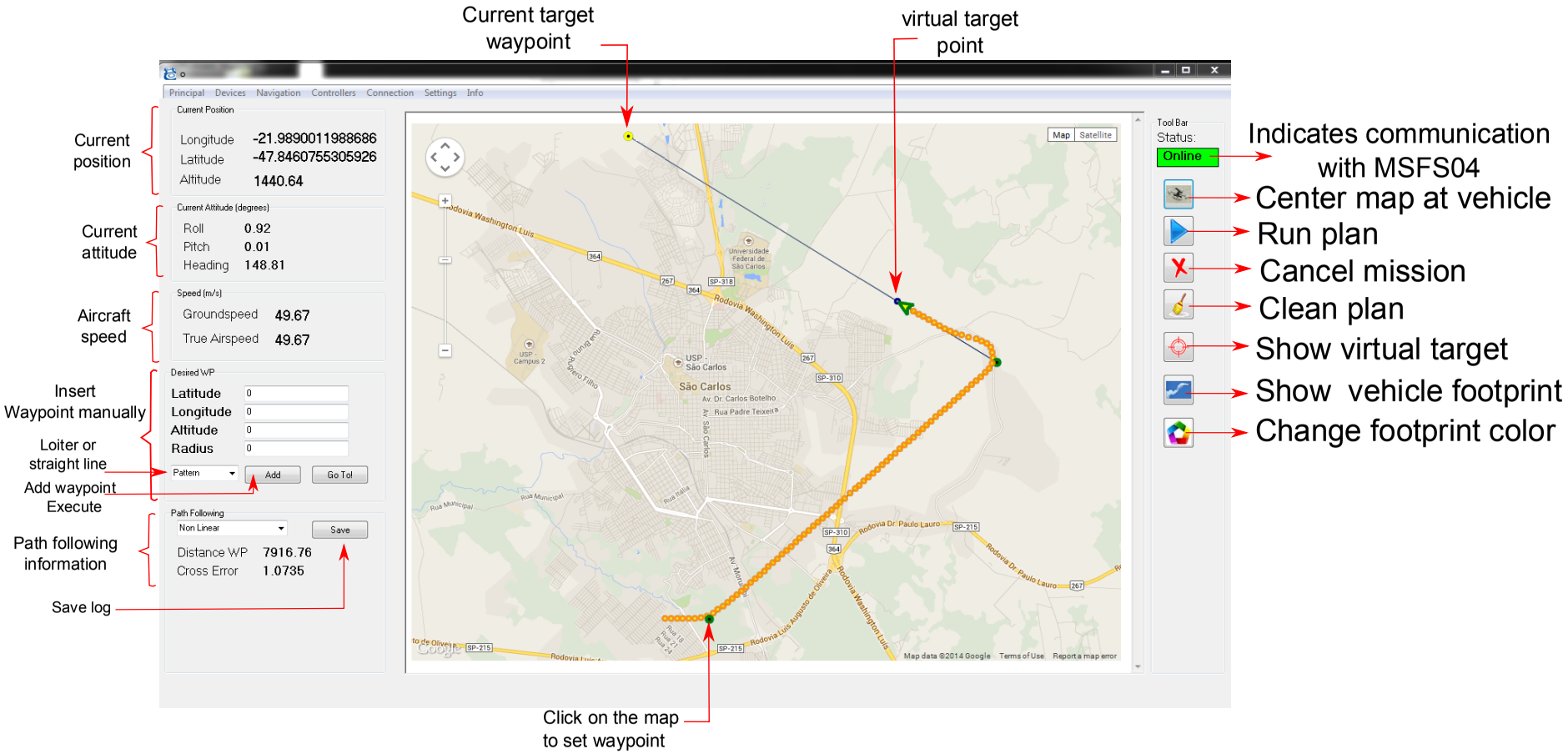


Figure 6.9: FVMS GNC user interface

7. Simulation Results

This chapter presents the results obtained in Matlab® and FVMS for the control laws addressed in the previous chapters. In Section 7.1, by means of numerical simulations, the convergence of the methods are assessed. The discussion continues in Section 7.2 with data collected from FVMS. Section 7.3 summarizes the main results. For the sake of simplicity the nonlinear Lyapunov based path following is referred as *nonlinear* along the text.

7.1 Numerical simulator

In this section, the discussed results do not take the dynamics of the aircraft in consideration. However, some constraints are imposed for the actuators. The yaw rate and pitch rate saturate, respectively, at 0.3 rad/s and 0.15 rad/s, the maximum airspeed magnitude is 55 m/s and the minimum 45 m/s.

In Subsection 7.1.1 the results from the carrot chasing algorithm, explained in Section 4.2, and the nonlinear Lyapunov based control algorithm, discussed in Section 4.3, are compared. Also, the performance of the latter for the 3D motion control is reported. In Subsection 7.1.2, the coordination controller (Section 5) for different topologies is evaluated.

7.1.1 Path Following

2D Path-following

Fig. 7.1 allows to compare the carrot chasing and nonlinear Lyapunov based algorithms. On the left column of the figure, the path, position error and yaw rate input for the carrot chasing algorithm are shown. The respective information for the nonlinear are on the right column of the same figure. Only the geometric task is addressed, as carrot chasing typically does not stipulate a control law for the speed assignment.

The navigation maps (top row) show the difference of VTP (blue dot) evolution strategy between the two methods. While in carrot chasing the target point is always ahead of the vehicle, in nonlinear the vehicle eventually reaches the VTP and follows it. The position errors (middle row) show that both algorithms converge to their desired path in

a quite similar fashion. However, the difference in the yaw rate input (bottom row) is noticeable.

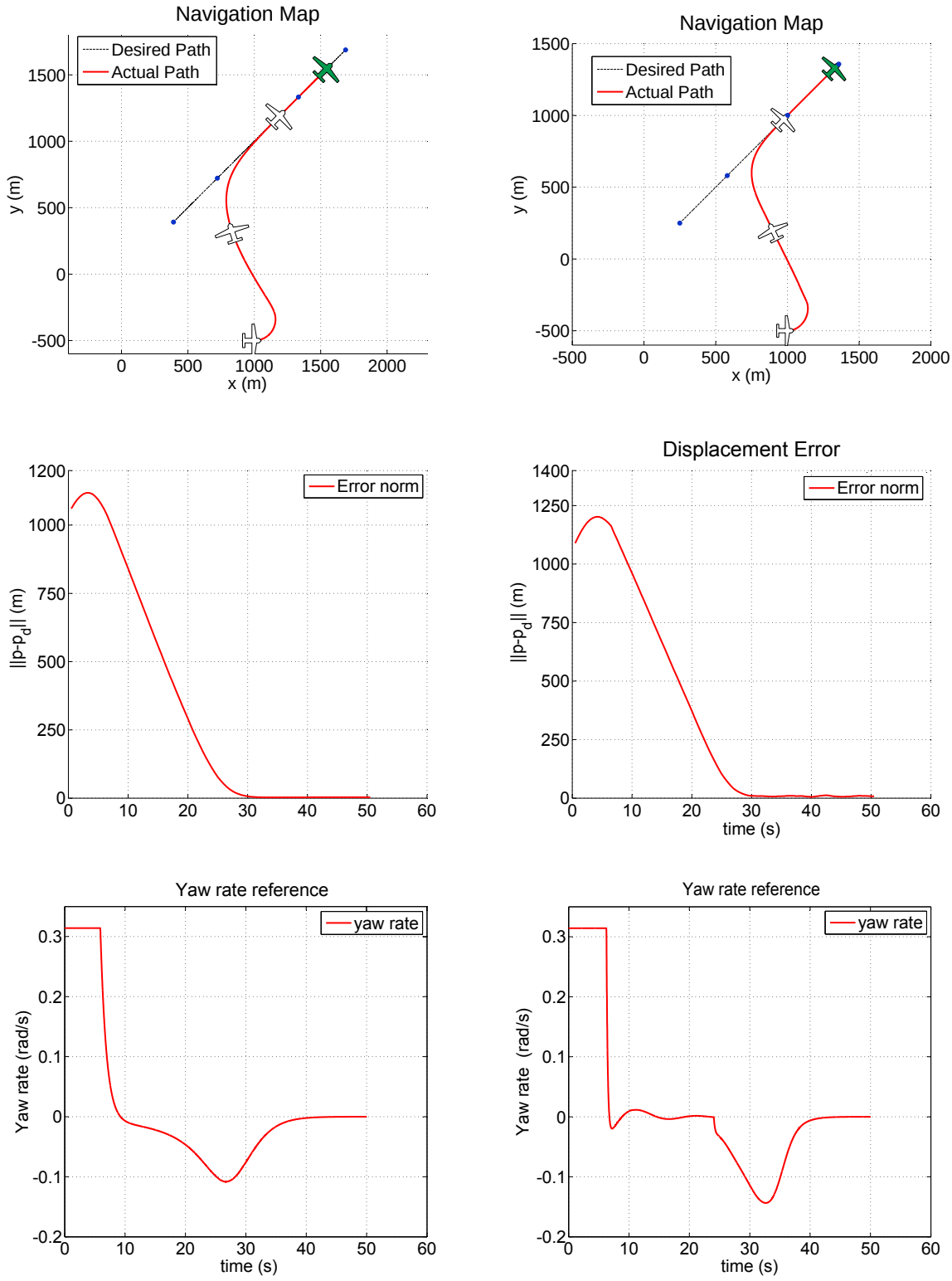


Figure 7.1: Horizontal plane path following comparison. Carrot chasing results on the left column and nonlinear Lyapunov based on the right column

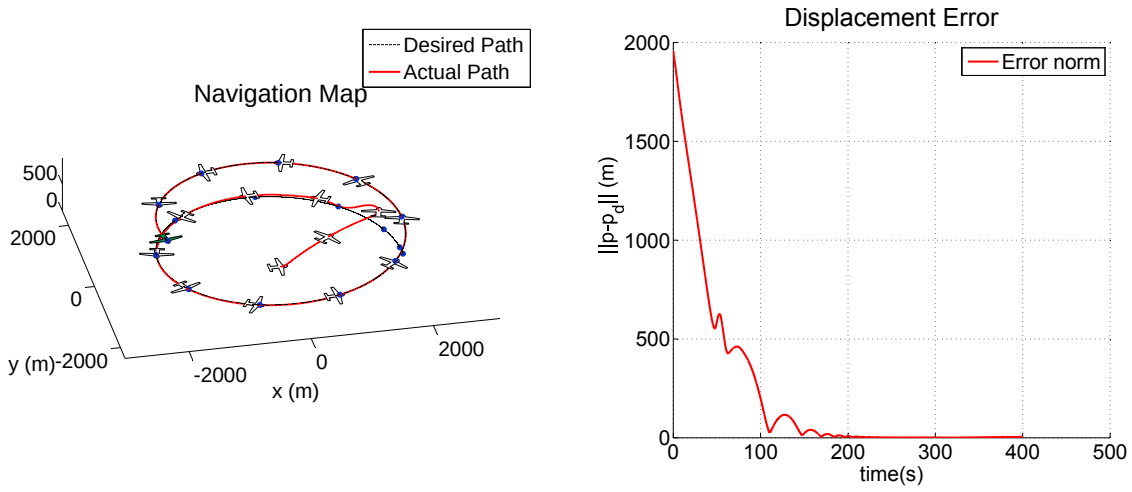
In overall, the nonlinear Lyapunov based algorithm has a good performance. It is

comparable to carrot chasing, whose performance has been confronted with a number of solutions in (Sujit et al., 2013)

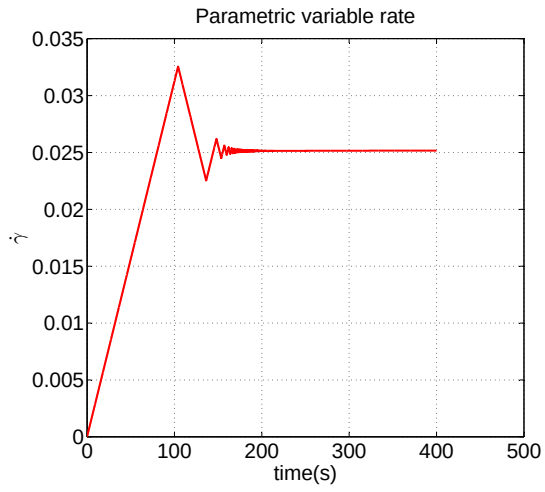
3D Path-following

In order to validate the nonlinear 3D path-following, an ascending loiter path, as shown in Fig. 7.2a, is employed. The radius of the loiter is $2000m$ and the speed assignment is set to $v_d = \frac{50}{2000} = 0.025$. The latter is directly related to the angular speed of the vehicle, in this specific test.

The convergence of the errors to of the equilibrium points $\boldsymbol{e} = \mathbf{0}$ and $z = 0$ is illustrated in Fig. 7.2b and Fig. 7.2c, respectively.



(a) (b)



(c)

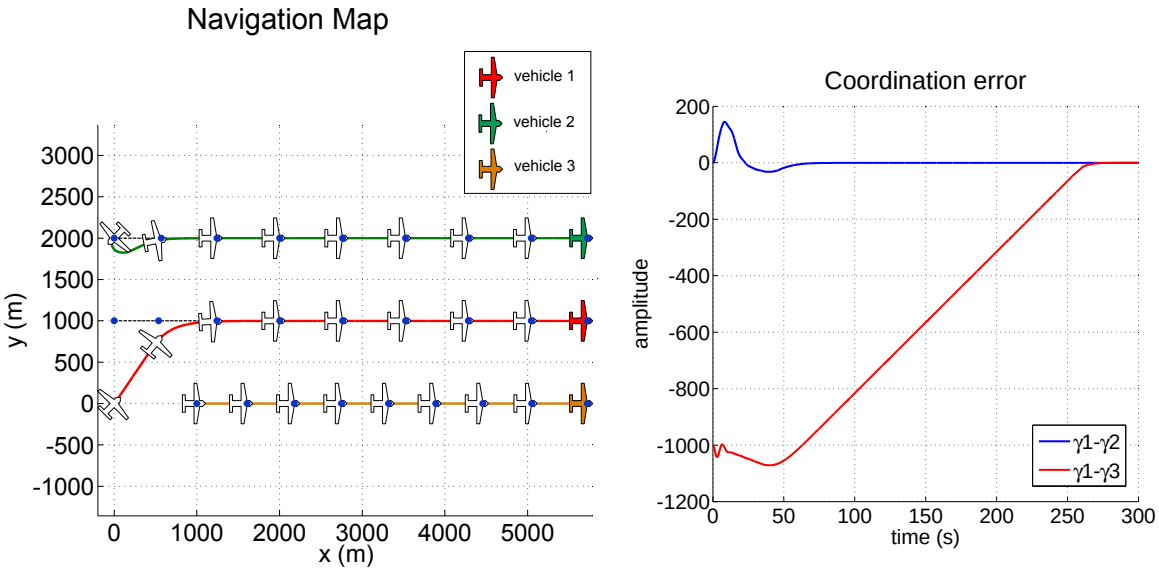
Figure 7.2: 3D path following. Path visualization (a), distance from desired point (b) and parametric variable rate (c)

7.1.2 Coordination control

Consider a fleet of three vehicles. Each element runs the nonlinear path following algorithm previously evaluated. In this context, the coordination controllers for different topologies are analyzed.

Constant topology

Consider a quasi strongly connected inter-vehicle communication topology. *Vehicle 1* communicates its coordination state to the other vehicles. *Vehicle 2* and *Vehicle 3* do not communicate each other. Fig. 7.3a shows how the flock evolves in time. This figure is better interpreted if visualized from right to left. It is possible to see that the fleet achieved coordination near the footprint before the last. Moreover, Fig. 7.3b clearly shows that *vehicle 1* and *vehicle 2* are in coordination after a short time interval. *Vehicle 3* takes longer to be in coordination, but eventually synchronization is accomplished.



(a) (b)

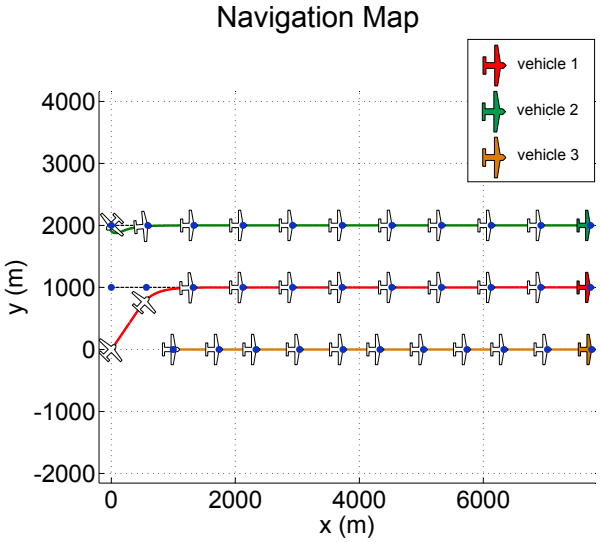
Figure 7.3: Constant topology. Illustration of the fleet (a) and coordination error (b)

Switching topology

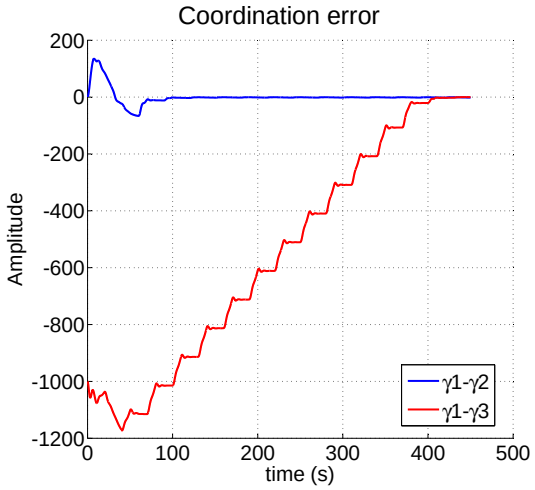
Consider once again a fleet of three vehicles. The initial configuration and paths are as in the constant communication example. However, this time, let the communication be uniformly quasi strongly connected. The switching signal $p_i(t) = [v_2v_1, v_3v_1, 0, \cdots, 0]$ is a periodic signal with period $T = 30s$, described as next:

$$p_i(\tau) = \begin{cases} [1,0,0,\cdots,0]^T, & \text{for } 0 < \tau \leq \mathrm{T}/3 \\ [0,1,0,\cdots,0]^T, & \text{for } \mathrm{T}/3 < \tau \leq 2\mathrm{T}/3 \\ [0,0,0,\cdots,0]^T, & \text{for } 2\mathrm{T}/3 < \tau \leq \mathrm{T} \end{cases}$$

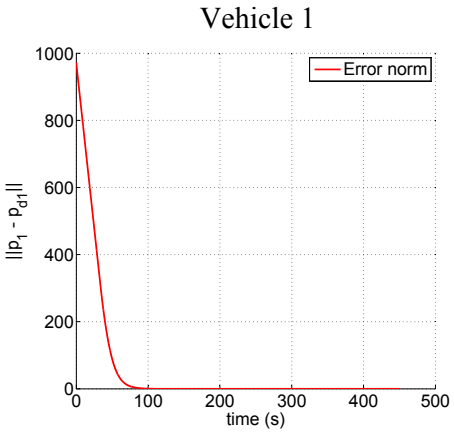
In other words, in the first third *vehicle1* transmits its coordination state to *vehicle2* (link v_2v_1 active). In the second third, *vehicle3* is able to receive the state of *vehicle1*



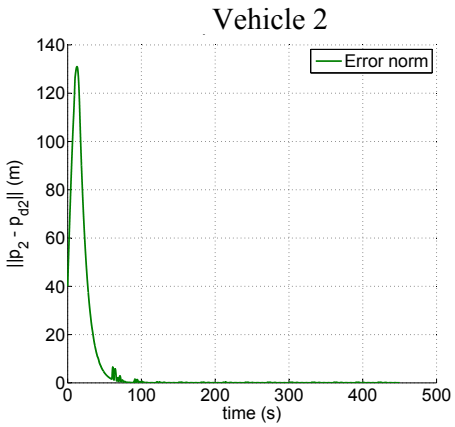
(a)



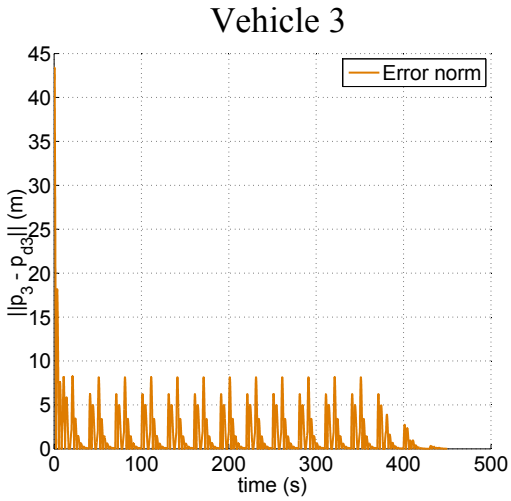
(b)



(c)



(d)



(e)

Figure 7.4: Switching topology. Fleet evolution (a) and coordination error (b) over time.

(link v_3v_1 active). In the last third, no communication is available between the elements in the fleet. (see Fig. 2.5)

The coordination among the vehicles is illustrated in Fig. 7.4a. In the switching topology it takes longer to synchronize the vehicles, namely $400s$, in contrast to $250s$ for the constant communication topology. This is expected since the correction speed is null when the vehicles are not communicating. Fig. 7.4c, 7.4d and 7.4e show that each vehicle independently converges to its desired path. The inertia of the vehicle corresponds to poles at relatively low frequencies. Therefore, if the dynamic of the vehicle was considered, the noisy signal presented in Fig. 7.4e would have been filtered.

7.2 SiL Simulation

The FVMS SiL simulations are performed with a fixed wing aircraft such as the Cessna C17SP. Fig. 7.5 shows the simulation environment.

It is not feasible to control the yaw rate of such aircraft. Therefore, the coordinated turn assumption (3.12), that relates yaw rate and bank angle, is considered. A similar relation is obtained for pitch rate and pitch angle.

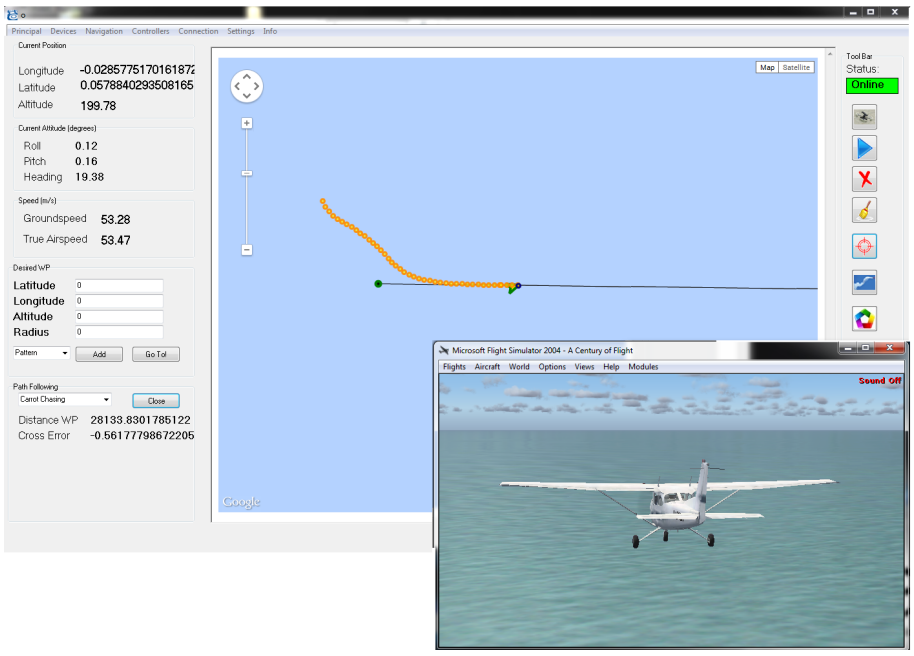


Figure 7.5: FVMS and MSFS SiL environment.

7.2.1 Path following

Carrot chasing vs nonlinear Lyapunov based

In order to compare nonlinear Lyapunov based and carrot chasing algorithms in the FVMS platform, the vehicle is initially set backwards to its desired path. Fig. 7.6a, which is a snapshot from FVMS map interface, shows the path trailed by the vehicle for carrot

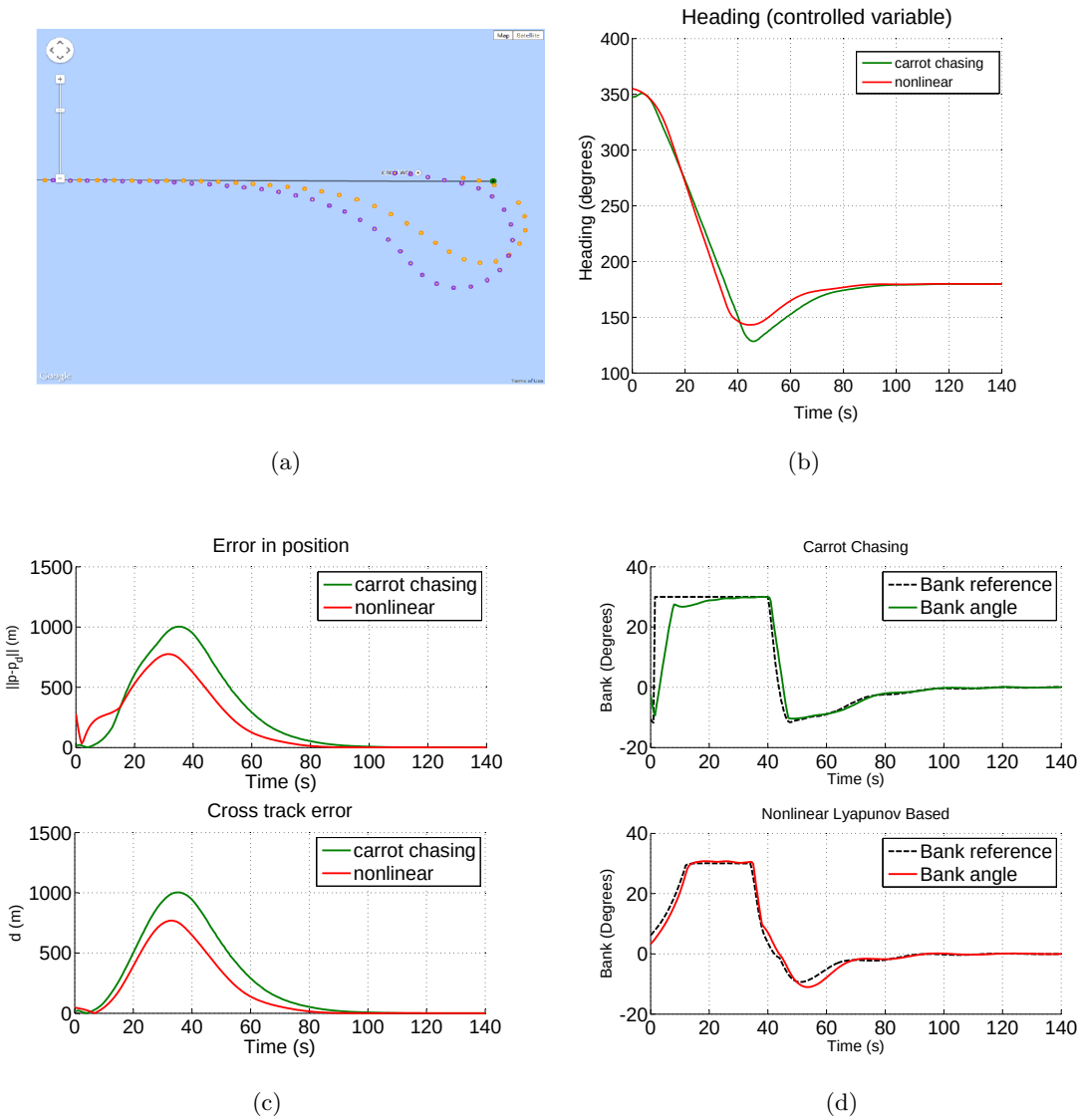


Figure 7.6: Comparision between the carrot chasing and the nonlinear Lyapunov based algorithms. Path traveled (a), heading (b), position error (c) and actuating input (d).

chasing (orange) and nonlinear (purple). The latter performs the dynamic assignment task. In Fig. 7.6b the initial and final orientation is illustrated.

The nonlinear PF shows an apparent better performance. Analyzing the error from the desired path and the cross-tracking error, it keeps both values tighter than the carrot chasing (Fig. 7.6c). Fig. 7.6d shows the delay in the bank control input. It is possible to observe a stronger control input from the carrot chasing followed by a slower convergence. Although both missions are set up to have a constant forward speed $v_{gs} = 50m/s$ on steady-state, only nonlinear specify a control law for the airspeed. By keeping the speed low while turning, nonlinear achieves a faster convergence and a smaller cross tracking error.

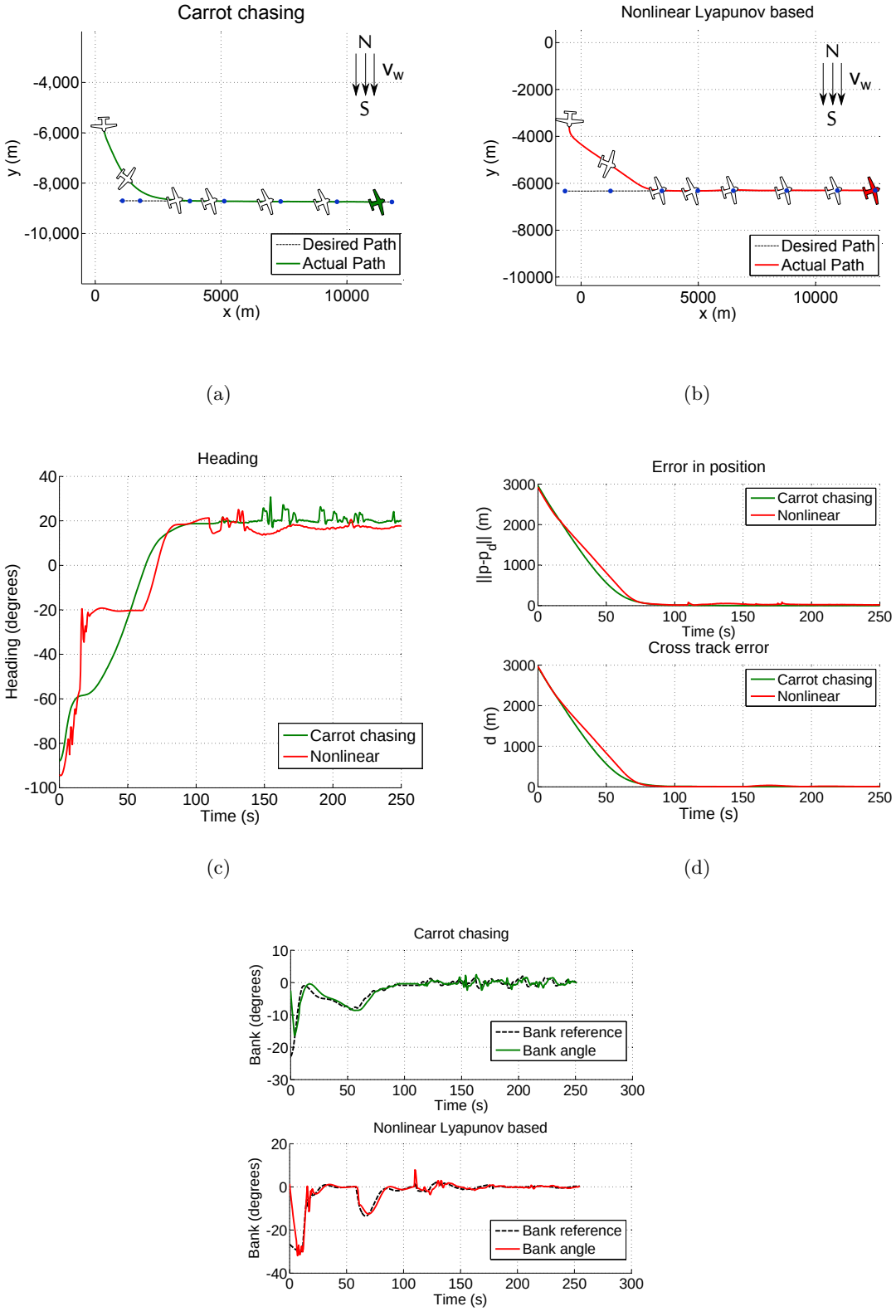


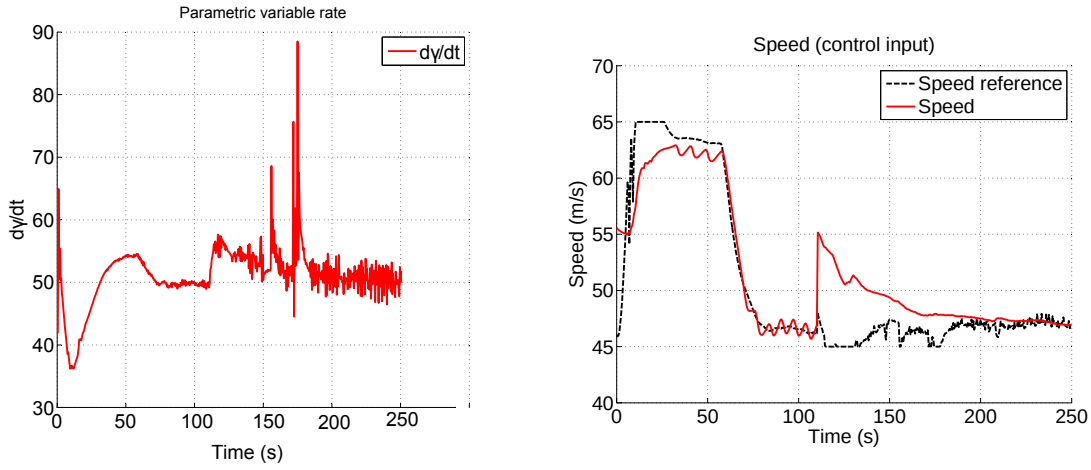
Figure 7.7: robustness test. Wind directed from north to south, wind speed = 18 m/s, gust = 28m/s around 125s. Path travelled for carrot chasing (a) and nonlinear PF (b), heading angle (c), error (d), actuating signal (e).

Robust control

The robustness of both carrot chasing and nonlinear path following is tested in an adverse scenario. The wind is set to 18 m/s, directed from north to south. The flight is disturbed with gust of winds (28 m/s) and moderate turbulence at approximately $t = 125s$. The desired path is orthogonal to the wind direction. The speed assignment is adjusted to $v_d = 50$. In this specific case, it corresponds to the desired ground speed.

In order to fly in the right direction, the vehicle slightly faces the wind (Fig. 7.7a and Fig. 7.7b). While the desired path orientation is around 0^o, the vehicle is oriented around 20^o, as shown in Fig. 7.7c. Analyzing the heading, it is possible to visualize when each simulation is disturbed by gusts and turbulence. However, the controllers' behavior assure that the error, depicted inf Fig. 7.7d, remain bounded. The bank control input is illustrated in Fig. 7.7e.

Specifically, for the nonlinear PF, Fig. 7.8 illustrates the parametric variable rate evolution and the speed control input. The disturbance directly affects the position of the vehicle. Consequently, the parametric variable rate is also disturbed, as its control law has a term that depends on the vehicle position.

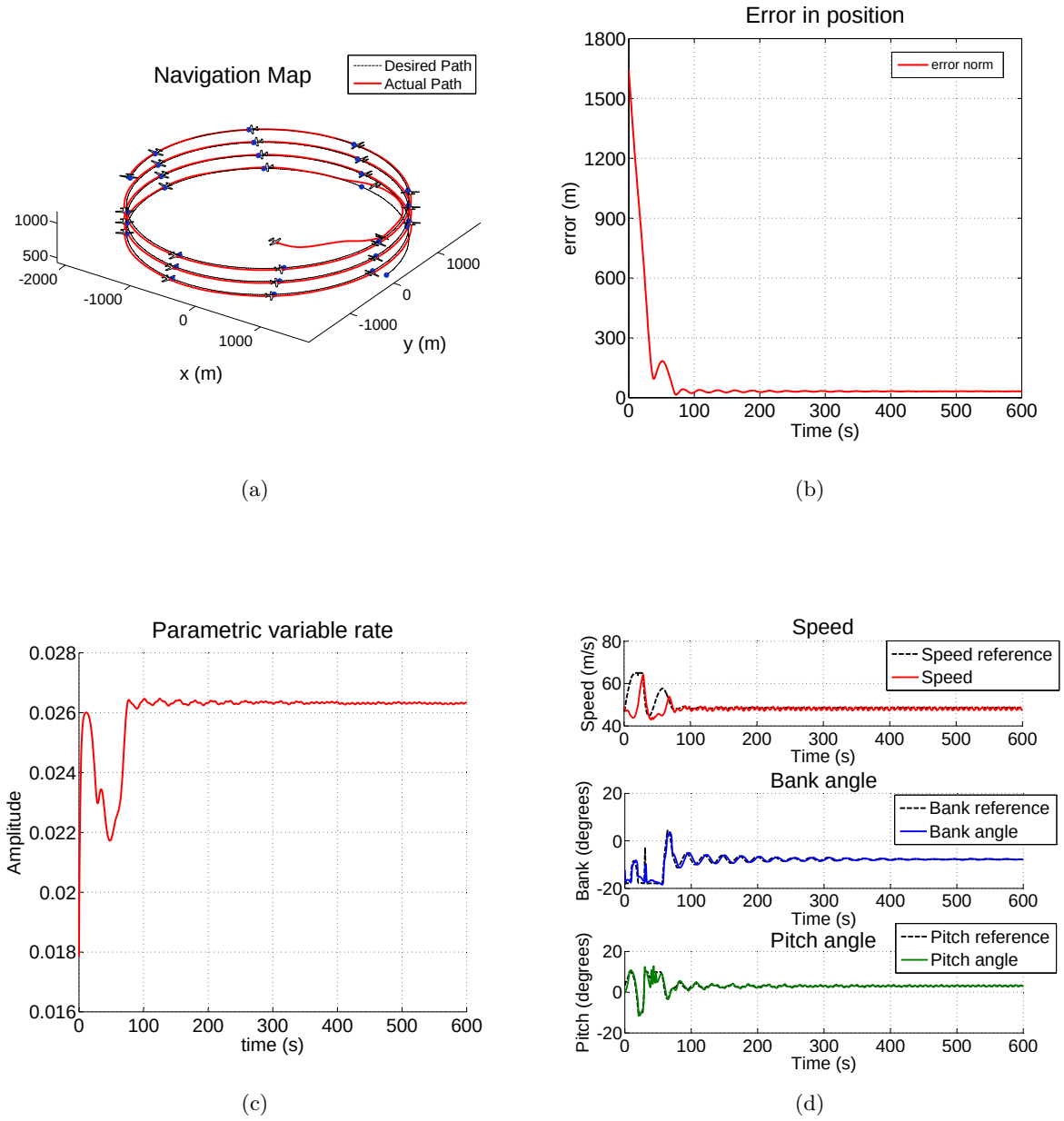


(a) (b)

Figure 7.8: Nonlinear PF, parametric variable rate (a) and speed control input (b).

3D path following

An ascending loiter is parametrized to evaluate the 3D nonlinear path following. The radius is set to $1800m$ and $v_d = \frac{48}{1800} = 0.0267$. The evolution of the vehicle along the path is shown in Fig. 7.9a. As depicted in Fig. 7.9b and Fig. 7.9c, the error stabilizes at its equilibrium point. When analyzing the control input (Fig. 7.9d), the importance of delay filters is noticeable. FVMS provides filters for both roll and pitch references, but it does not implement a filter for airspeed assignment. Once the filter is implemented, a smoother convergence may be achieved.



(a) (b)

(c) (d)

Figure 7.9: SiL: 3D path folllowing. Path travelled (a), position error (b), parametric variable rate (c) and input (d).

7.3 Summary

In this chapter it is shown that the performance of the nonlinear Lyapunov based algorithm is as good as for the carrot chasing. The coordination control is achieved in both constant and switching topologies. The synchronization in a switching topology takes longer, as expected, but the communication effort is much smaller. In addition, simulations carried in the FVMS SiL platform confirm some of the previous results. Also, the controller robustness under weather disturbances is evaluated. The results show that the nonlinear algorithm is able to keep the vehicle on track despite outer disturbances.

8. Conclusion

8.1 Summary

This dissertation focused on cooperative control of a formation of UAVs. In order to tackle the problem in a clear and transparent manner, the cooperative motion control problem was divided in path following and coordinate control. The former assures that the vehicle follows a virtual target, whose evolution in time is adjusted by the latter. Nonlinear Lyapunov stability theorems and graph theory support the control laws herein addressed.

The results from SiL simulations show that the performance of nonlinear path following is comparable to the one of other techniques in the literature. The obtained results are encouraging for both 2D and 3D scenarios. The dynamics of the vehicle are not explicitly addressed. This allows the kinematic controller to be extended to other vehicles, equipped with a dynamic controller. The coordination results confirm that it is possible to achieve coordination in a switching inter-vehicle communication topology. Moreover, the results report that in a uniformly quasi strongly connected topology, even under intermittent link failures, coordination is accomplished.

Equipping the FVMS with a GNC module opens up new possibilities for the platform. The software was upgraded with a more friendly user-interface. Guidance algorithms may be evaluated in a reliable fashion. In addition, the tool shows great potential in the academic scope, assisting students in the learning process.

8.2 Future work

Cooperation in SiL

The MS Flight Simulator does not allow more than one aircraft per simulation. Therefore, achieving communication between FVMS running in different machines is a point which deserves further attention. Running MSFS and FVMS in different machines would allow evaluating cooperative controllers with SiL simulations. The communication may be implemented in a TCP/IP network.

Communication topology

In this work the existing delays in communication links are not addressed. However, according to the dynamic of the system, the delay may play an important role in vehicle synchronization. Studying the delay impact is left as a future topic to be explored.

Obstacle avoidance

Obstacle avoidance is a topic that naturally emerges when discussing autonomous robots. This topic requires special attention when dealing with multiple autonomous vehicles, as the systems have to interact with each other to avoid collisions. The problem requires generating a collision-free trajectory and online information to evasion maneuvers, if necessary. The SiL platform discussed, FVMS, allows to use optical flow techniques to map the ambient and sense possible obstacles.

Bibliography

Outback joe. Internet, September 2012. URL http://southburnett.com.au/news2/2012/09/can-anyone-find-outback-joe/.

Antonio Pedro Aguiar and Joao Pedro Hespanha. Position tracking of underactuated vehicles. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 1988–1993 vol.3, June 2003.

A.P. Aguiar and J.P. Hespanha. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *Automatic Control, IEEE Transactions on*, 52(8):1362–1379, Aug 2007.

A.P. Aguiar and A.M. Pascoal. Coordinated path-following control for nonlinear systems with logic-based communication. In *Decision and Control, 2007 46th IEEE Conference on*, pages 1473–1479, Dec 2007.

Cessna. Skyhawk - the best-selling and most-flow aircraft ever built, 2013. URL http://www.cessna.com/single-engine/skyhawk.

Peng Cheng and Vijay Kumar. An almost communication-less approach to task allocation for multiple unmmaned aeria vehicles. In *IEEE International Conference on Robotics and Automation*, pages 19–23, Pasadena, CA, USA, May 2008.

S. Demers, P. Gopalakrishnan, and L. Kant. A generic solution to software-in-the-loop. In *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1–6, Oct 2007.

Peter Downson. Fsuipc: Application interfacing module for microsoft flight simulator, August 2012. URL http://www.schiratti.com/dowson.

M. Egerstedt and Xiaoming Hu. Formation constrained multi-agent control. *Robotics and Automation, IEEE Transactions on*, 17(6):947–951, December 2001.

P. Encarnacao and A. Pascoal. 3d path following for autonomous underwater vehicle. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2977–2982 vol.3, 2000.

P. Encarnacao and A. Pascoal. Combined trajectory tracking and path following: an application to the coordinated control of autonomous marine craft. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 964–969 vol.1, 2001.

William Etter, Paul Martin, and Raul Mangharam. Cooperative flight guidance of autonomous unmanned aerial vehicles, 2011.

J. Alexander Fax and Richard M. Murray. Graph laplacians and stabilization of vehicle formations. In *Proceedings 2002 IFAC World Congress*, Barcelona, Spain, 2002.

J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *Automatic Control, IEEE Transactions on*, 49(9):1465–1476, September 2004.

R. Ghabcheloo, A. Pascoal, C. Silvestre, and I. Kaminer. Nonlinear coordinated path following control of multiple wheeled robots with bidirectional communication constraints. *International Journal of Adaptive Control and Signal Processing*, 21(2-3):133–157, 2007.

R. Ghabcheloo, A. 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*, 48(1):234–265, 2009.

B. Grocholsky, J. Keller, V. Kumar, and G. Pappas. Cooperative air and ground surveillance. *Robotics Automation Magazine, IEEE*, 13(3):16–25, September 2006.

Stephen Philip Jackson. *Controlling Small Fixed Wing UAVs to Optimize Image Quality from On-Board Cameras.* PhD thesis, Universiry of California, Berkley, U.S.A, 2011.

Hassan K. Khalil. *Nonlinear System.* Prentice Hall, 2002.

L. Lapierre, D. Soetanto, and A. Pascoal. Nonlinear path following with applications to the control of autonomous underwater vehicles. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 2, pages 1256–1261 Vol.2, Dec 2003.

Zhiyun Lin, Bruce Francis, and Manfredi Maggiore. State agreement for continuous-time coupled nonlinear systems. *SIAM J. Control Optim.*, 46(1):288–307, March 2007.

Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

R. Louali, A. Belloula, M.S. Djouadi, and S. Bouaziz. Real-time characterization of microsoft flight simulator 2004 for integration into hardware in the loop architecture. In *Control Automation (MED), 2011 19th Mediterranean Conference on*, pages 1241–1246, June 2011.

M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks.* Princeton University Press, 2010.

T. Oliveira, P. Encarnacao, and A.P. Aguiar. Moving path following for autonomous robotic vehicles. In *Control Conference (ECC), 2013 European*, pages 3320–3325, July 2013.

S. Park, J. Deyst, and J. P. How. Performance and lyapunov stability of a nonlinear path-following guidance method. *AIAA Journal on Guidance, Control, and Dynamics*, 30(6):1718–1728, November-December 2007.

J. Roberts and R. Walker. Flying robots to the rescue [competitions]. *Robotics Automation Magazine, IEEE*, 17(1):8–10, March 2010.

Jan Roskam. *Airplane Flight Dynamics and Automatic Flight Controls, Part I*. DARcoporation, Lawrence, KS, U.S.A., 2001.

UAV Roundup. Uav roundup 2013. *Aerospace America*, pages 26–36, July-August 2013.

Erol Sahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20. Springer Berlin Heidelberg, 2005.

Rafael C.B. Sampaio, Marcelo Becker, Adriano A.G. Siqueira, Leonardo W. Freschi, and Marcelo P. Montanher. Fvms: A novel sil approach on the evaluation of controllers for autonomous mav. In *Aerospace Conference, 2013 IEEE*, pages 1–8, March 2013a.

Rafael C.B. Sampaio, Marcelo Becker, Adriano A.G. Siqueira, Leonardo W. Freschi, and Marcelo P. Montanher. Novel sil evaluation of an optimal h_∞ controller on the stability of a mav in flight simulator. In *Aerospace Conference, 2013 IEEE*, pages 1–8, March 2013b.

Rafael C.B. Sampaio, Andre C. Hernandes, Marcelo Becker, , Eduardo Cazarini, Daniel V. Magalhaes, and Adriano A.G. Siqueira. New hil evaluation of an h_∞ controller on the stabilization of a mav in flight simulation. In *Aerospace Conference, 2014 IEEE*, pages 1–7, March 2014a.

Rafael C.B. Sampaio, Andre C. Hernandes, Marcelo Becker, , Fabio M. Zanin, Joao E. M. Nobrega, Caio A. C. Martins, and Fabio M. Catalano. Squidcop design and evaluation of a novel quadrotor mav for in-flight launching airground missions. In *Aerospace Conference, 2014 IEEE*, pages 1–7, March 2014b.

Rafael C.B. Sampaio, Andre C. Hernandes, Marcelo Becker, , Fabio M. Zanin, Joao E. M. Nobrega, Caio A. C. Martins, and Fabio M. Catalano. A new hybrid motor glider-quadrotor mav for v-stol launching. In *Aerospace Conference, 2014 IEEE*, pages 1–7, March 2014c.

Claude Samson. Path-following and time-varying feedback stabilization of a wheeled mobile robot. In *Proceedings of the ICARCV*, Singapore, 1992.

Robert J. Shaw. Dynamics of flight, 2014. URL http://www.grc.nasa.gov/WWW/k-12/UEET/StudentSite/dynamicsofflight.html.

Charley Shimanski. Mountain rescue association reaffirms its position opposing charging subjects for the costs of their rescues. Charge Position, August 2009.

P.B. Sujit, S. Saripalli, and J.B. Sousa. An evaluation of uav path following algorithms. In *Control Conference (ECC), 2013 European*, pages 3332–3337, July 2013.

M. Valenti, D. Dale, J. How, and J. Vian. Mission health management for 24/7 persistent surveillance operations. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Myrtle Beach, SC, August 2007.

Francesco Vanni. Coordination motion control of multiple autonomous underwater vehicles. Master's thesis, Department of Mechanical Engineering, Insituto Superior Técnico, Universidade Técnica de Lisboa, 2007.

Chris Veness. Calculate distance, bearing and more between latitude/longitude points, January 2010. URL http://www.movable-type.co.uk/scripts/latlong.html.

Jan Willmann, Federico Augugliaro, Thomas Cadalbert, Raffaello D'Andrea, Fabio Gramazio, and Matthias Kohler. Aerial robotic construction: Towards a new field of architectural research. *International Journal of Architectural Computing*, 10-3, 2012.

Appendix - FVMS GNC libraries and functions

A.1 Types and conversions

New types were created to support motion control algorithms. They are available in the *FVMSTypes* library. In order to use any of the types developed, the user needs to include the library in proper section (use section). The following types are highlighted:

TCartesian: Cartesian coordinates (x,y,z).

Name	Type	Unit	Description
x	double	meter	position of the vehicle in inertial x-axis
y	double	meter	position of the vehicle in inertial y-axis
z	double	meter	position of the vehicle in inertial z-axis

TWGS84: Global coordinate specified in WGS84 normalization.

Name	Type	Unit	Description
Latitude	double	degree	Latitude
Longitude	double	degree	Longitude
Altitude	double	meter	Altitude

TAttitude: Attitude of the aircraft in respect to $mathcal I$, expressed in $mathcal B$.

Name	Type	Unit	Description
roll	double	radian	roll angle
pitch	double	radian	pitch angle
yaw	double	radian	yaw angle

TVehicleState: Stores current state of the vehicle.

Name	Type	Unit	Description
pos	TCartesian	-	position of the vehicle
attitude	TAttitude	-	attitude of the aircraft
tas	double	meter/second	true air speed norm
gs	double	meter/second	ground speed norm
vx	double	meter/second	forward ground speed value
vy	double	meter/second	lateral ground speed value
vz	double	meter/second	vertical ground speed value

TPath: Stores current state of the vehicle.

Name	Type	Unit	Description
WPi	TCartesian	-	coordinates of initial waypoint
WPf	TAttitude	-	coordinates of final waypoint
losAngle	double	radians	LoS Angle
crossTrError	double	meter	cross tracking error
loiter	boolean	-	0-straight line, 1-loiter
radius	double	meter	loiter radius in meters. 1-straight line
active	boolean	-	0-path is not active, 1-path active

As the information needs to be processed or show in different formats, a conversion library was developed. The user can use the following functions through *FVMSConversion* library.

function WGS84ToCartesian(WGS84_ coord: TWGS84): TCartesian;

notes: Converts coordinates specified in WGS84 into cartesian coordinates

function cartesianToWGS84(cartesian_ coord: TCartesian): TWGS84;

notes: Converts cartesian coordinates into WGS84 standard

There are several ways to transform WGS84 coordinates into cartesian. The complexity increases as more precision is required. It was not in the scope of this project to deep investigate this matter. Therefore a simple transformation was employed. As the earth radius is assumed constant, high values of latitude are distorted during conversion. For more information the reader is referred to (Veness, 2010)

function degToCoord(deg: Double): TCoordinate;

notes: Converts a coordinate specified in degree to (degree, minutes and seconds)

A.2 Navigation library

The navigation library is able to read the computer memory by means of FSCUIP DLL. It focuses on reading data from the vehicle, like IMU and GPS sensors. As long as a connection exist between FVMS and simulator. By default, the state of the vehicle is refreshed at a rate of 2Hz. This rate may be easily modified. This library offers reading function to a range of variables that are employed in navigation algorithms, as described next.

procedure UpdateVehicleState;

notes: Responsible to read the memory and store in a TVehicle type

The following data are referred to the last updated values.

function getVehicleStatus: TVehicleState;

notes: Returns the state of the vehicle. This function shall be used with cautions as may generate excessive overhead.

function getPosition: TCartesian;

notes: Returns the current cartesian position (x,y,z) of the vehicle

function getAttitude: TAttitude;

notes: Returns the attitude of the aircraft

function getBank: double;

notes: Returns the bank angle in radians

function getPitch: double;

notes: Returns the pitch angle in radians

function getHeading: double;

notes: Returns the yaw angle (heading) in radians

function getTrueAirSpeed: double;

notes: Returns the module of true airspeed of the vehicle in m/s

function getGroundSpeed: double;

notes: Returns the module of groundspeed of the vehicle in m/s

function getVx: double;

notes: Returns the groundspeed forward component (x-axis) in m/s

function getVy: double;

notes: Returns the groundspeed lateral component (y-axis) in m/s

function getVz: double;

notes: Returns the groundspeed vertical component (z-axis) in m/s

A.3 Control library

The control library is responsible to send actuating messages related to thrust and control surfaces mentioned in Section 3.3 aileron, elevator and rudder. Once more, the computer memory is written via FSCUIP DLL. The library accessible through *FVMSControl*.

function sendRoll(roll_d: double): Integer;

notes: Compute and send the aileron deflection to simulator. Return the current roll angle

function sendPitch(pitch_d: double): Integer;

notes: Compute and send the elevator deflection to simulator. Return the current pitch angle

function sendHeading(heading_d: double);

notes: Compute and send the rudder deflection to simulator. Return the current heading angle

function sendSpeed(speed_d: double): double;

notes: Set the thrust and return the current airspeed

procedure sendThrust(thrust: Smallint);

notes: Send the thrust to simulator

A.4 Controllers library

The controllers library was created to aid designer using basic controller. The current version support PID described by the following equation:

$$u = K_p e + K_i \sum_i e_i dt + K_d \frac{e - e_{ant}}{dt} \tag{A.1}$$

There are a set of adjustable parameter. The library is available at *FVMSControllers*

TPID: PID controller structure.

Name	Type	Unit	Description
Kp	double	-	proportional gain
Ki	double	-	integral gain
Kd	double	-	derivative gain
e_sum	double	inherited	error sum
e_ant	double	inherited	previous error
antiWindUp	double	-	Anti wind up value
saturation	double	-	Output saturation value
dt	double	seconds	sample time

constructor Create; overload;

notes: Default constructor

constructor Create(Kp, Ki, Kd: double); overload;

notes: Customized constructor

function controlLoop(error: double): Double;

notes: Calculate and return the PID output for a single iteration

procedure setParameters(Kp, Ki, Kd: double);

notes: Set proportion, integral and derivative gains

procedure setAntiWindUp(antiWindUp: Double);

notes: Modify anti wind up value

procedure setSampleTime(dt: double);

notes: Modify sample time

procedure cleanSum;

notes: Clean the integral sum