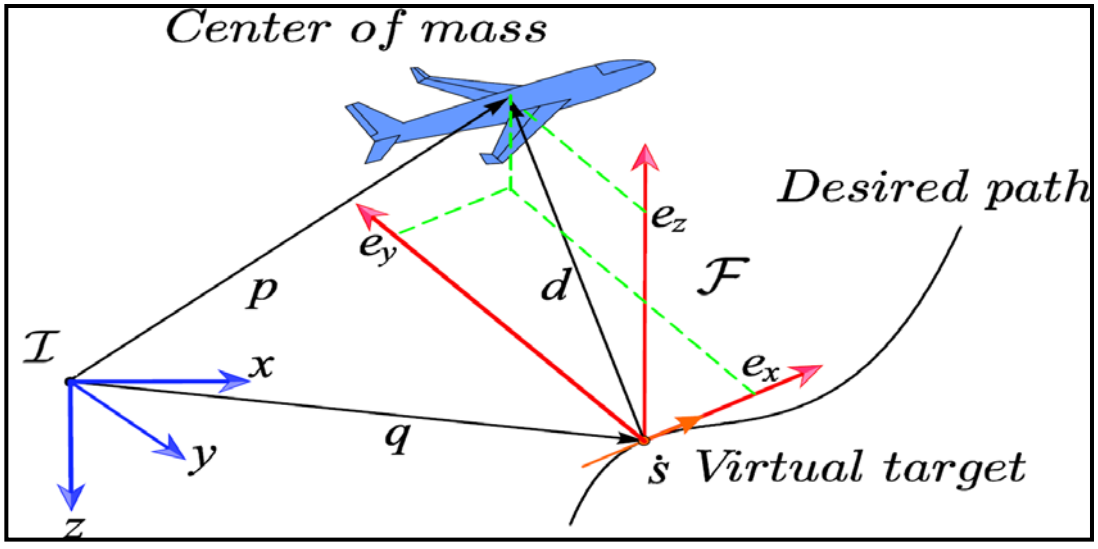
Autonomous take-off and landing for a fixed wing UAV

Israel Lugo Cárdenas

Par **Israel LUGO CARDENAS**

Autonomous take-off and landing for a fixed wing UAV

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 6 juin 2017
Spécialité : Automatique : Unité de recherche Heudyasic (UMR-7253)

D2364

Autonomous take-off and landing for a fixed wing UAV

by

Israel Lugo Cárdenas

Submitted to the Departement Technologies de l'Information et des Systèmes in partial fulfillment of the requirements for the degree of

Spécialité : Automatique

UNIVERSITY OF TECHNOLOGIE OF COMPIEGNE

6 June 2017

Author . . . . . .
Departement Technologies de l'Information et des Systèmes

Certified by .....
Rogelio LOZANO
DR CNRS, Heudiasyc, UTC
Thesis Supervisor

Certified by .....
Sergio SALAZAR
UMI LAFMIA, CINVESTAV
Thesis Supervisor

Autonomous take-off and landing for a fixed wing UAV

by

Israel Lugo Cárdenas

Submitted to the Departement Technologies de l'Information et des Systèmes on
6 June 2017, in partial fulfillment of the requirements for the degree of Doctor

Spécialité : Automatique

Abstract

This work studies some of the most relevant problems in the direction of navigation and control presented in a particular class of mini-aircraft. One of the main objectives is to build a lightweight and easy to deploy vehicle in a short period of time, an unmanned aerial vehicle capable of following a complete mission from take-o/ to the following waypoints and complete the mission with an autonomous landing within a delimitated area using a graphical interface in a computer.

The Trajectory Generation It is the part that tells the drone where it must travel and are generated by an algorithm built into the drone. The classic result of Dubins is used as a basis for the trajectory generation in 2D and we have extended it to the 3D trajectory generation.

A path following strategy developed using the Lyapunov approach is presented to pilot a

xed wing drone across the desired path. The key concept behind the tracking controller is the reduction of the distance between the center of mass of the aircraft p and the point q on the path to zero, as well as the angle between the velocity vector and the vector tangent to the path.

In order to test the techniques developed during the thesis a customized C # .Net application was developed called MAV3DSim (Multi-Aerial Vehicle 3D Simulator). The MAV3DSim allows a read / write operation from / to the simulation engine from which we could receive all emulated sensor information and sent to the simulator. The MAV3DSim consists of three main elements, the simulation engine, the computation of the control law and the visualization interface. The simulation engine is in charge of the numeric integration of the dynamic equations of the vehicle, we can choose between a quadrotor and a xed wing drone for use in simulation. The visualization interface resembles a ground station type of application, where all variables of the vehicle s state vector can be represented on the same screen.

The experimental platform functions as a test bed for the control law prototyping. The platform consists of a xed wing aircraft with a PX4 which has the autopilot function as well as a Raspberry PI mini-computer which to the implementation of the generation and trajectory tracking.

The complete system is capable of performing an autonomous take-o/and landing, through waypoints. This is accomplished by using each of the strategies developed during the thesis. We have a strategy for take-o/ and landing, which is generated by the navigation part that is the trajectory generator. Once we have generated the path, it is used by the trajectory tracking strategy and with that we have landing and take-o/ autonomously.

Resume

Ce travail étudie certains des problèmes les plus pertinents dans le sens de la navigation et contrôle présentés dans une classe particulière de mini-véhicules aériens. L' un des principaux objectifs c' est à réaliser un véhicule léger et facile à déployer dans un court laps de temps, un véhicule sans pilote drone capable de suivre une mission complète, du décollage aux points de cheminement suivants et de terminer la mission avec un atterrissage autonomes à l' intérieur d' unezone délimitée en utilisant une interface graphique dans un ordinateur ou une tablette.

La génération de trajectoire Il est la partie qui dit le drone où il doit voyager et sont génères par un algorithme intégré sur le drone. Le résultat classique de Dubins est utilisé comme base pour la génération de trajectoire en 2D et nous avons étendu à la génération de trajectoire 3D.

Une stratégie de suivi de trajectoire développée en utilisant làpproche de Lyapunov, est présenté pour piloter un drone à voilure fixe à travers tout le chemin désiré. Le concept clé derrière le contrôleur de suivi de trajectoire s'appuie sur la réduction de la distance entre le centre de masse de l'avion p et le point sur la trajectoire q à zéro, ainsi que l'angle entre le vecteur vitesse et la tangente à la trajectoire.

Afin de tester les techniques mises au point au cours de la thèse une application C# .Net personnalisée a été développé nommé MAV3DSim (Multi-Aerial Vehicle 3D Simulator). Le MAV3DSim permet une opération de lecture/écriture de/vers le moteur de simulation à partir de laquelle nous pourrions recevoir toutes les informations de capteurs émulés et envoyés par le simulateur. Le MAV3DSim est constitué de trois éléments principaux, le moteur de simulation, le calcul de la loi de commande et l'interface de visualisation. Le moteur de simulation est en charge de l'intégration numérique des équations dynamique du drone, ici nous pouvons choisir entre un quadrirotor ou un drone d'aile fixe pour l'utiliser dans la simulation. L'interface de visualisation ressemble à un type d'application de la station au sol, où toutes les variables du vecteur d'état du drone peut être représenté sur le même écran.

La plate-forme expérimentale qui fonctionne comme un banc d'essai pour la loi de commande prototypage. Le plate-forme est constitué d'un avion de aile fixe avec un PX4 qui a la fonctionne d'autopilote ainsi qu'un mini-ordinateur Raspberry PI qui à la implémentation de la génération et suivi de trajectoire.

Le système complet est capable d'effectuer un décollage et d'atterrissage autonome, à travers des points de suivi. Ceci est accompli en utilisant chacune des stratégies développées au cours de la thèse. Nous avons une stratégie pour le décollage et l'atterrissage, ce qui est généré par la partie de navigation qui est le générateur de trajectoire. Une fois que nous avons généré le chemin, il est utilisé par la stratégie de suivi de trajectoire et avec ce que nous avons l'atterrissage et le décollage autonome.

Thesis Supervisor: Rogelio LOZANO
Title: DR CNRS, Heudiasyc, UTC

Thesis Supervisor: Sergio SALAZAR
Title: UMI LAFMIA, CINVESTAV

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 UAV Overview

Much effort is currently spent on the research and production of unmanned vehicles, particularly those related to Unmanned Aerial Vehicles (UAV) as they have certain advantages over piloted or remotely controlled vehicles and they are preferred over piloted aircrafts due to low cost of the UAV, therefore the UAV can be expandable. Also they have the ability to accomplish dangerous missions in hazardous environments that can not be done by piloted aircraft. This fact keeps the pilot and crew out of harm's way during potentially dangerous missions while also allowing the aircraft to be made smaller and avoid all the hardware necessary to sustain on-board life support.

Nowadays there are several companies that produce ready-made UAV systems with well developed ground station software that are made for commercial use. One of the companies that has this profile is The UAS Europe [72] that provide UAV for agricultural, research and surveillance purposes. UAS Europe provides a wide range of professional Ground Control Station (GCS) which allow the operator to handle all the tasks related to flying the UAV. They have their own flight control system which allows fully autonomous flights, from take-off to mission landing. The drawback of this solution is the cost. Usually a large fund is required in order to acquire a complete system , such as the UAS Europe system.

However with the recent increase in development of UAV development, there are now powerful enough devices yet at an affordable cost. Drone Deploy [18] offers an easy to use software

to handle flight planning and communications with the UAV. The complete software solution handles flight planning, manual download of the data and post-processing. Although they are focused on a solution for 3D mapping and the use of quadrotors to take the image since they have better stationary stabilization than the fixed wing aircraft. In addition, another company is Botlink [71]. Botlink provides a cloud-based platform which features a fully automated UAV with manual flying from any smart phone or tablet. Similar platforms can be found offering a variety of solutions with different features.

Another area that increase its interest over the last few years is the control methods used for UAVs. There is a vast research conducted around UAVs, the primary focus relies on the theoretical background that is required to set up and fly a UAV. The control methods and planning techniques are the most common subjects of interest.

In the literature there are several control methods developed for the stabilization and maneuvering of the aircraft. Among the most famous methods are the Linear Quadratic Control (LQR), which is applied to longitudinal dynamics of the UAV in [7] and [47], the Model Predictive Control (MPC) implemented in [26] as an an iterative scheme to solve the nonlinear optimization problem, also a nonlinear model predictive control is used to design a high-level controller for a fixed wing UAV in [40] and in [38].the adaptive control methods based on the approximation of the dynamic inversion. Despite the superior performance of these methods, not many are implemented in real applications. One of the reasons is the computational cost of the implemented methods. Hence the commonly used and most implemented methods is Proportional-Integral-Derivative (PID) controller, due to its low complexity and computational cost and the adequate performance as in [3].

Path planning is wideley documented in ground robotics and manipulators systems. However the field of robotics has extended to the airborne, with the Unmanned Aerial Vehicles. There are numerous references to UAV guidance laws reported in the literature, an integrated approach is described in [39], a navigation system designed to track straight lines between waypoints is described in [62]. A recent innovation is the use of vector fields, where a velocity vector field is specified over space and the vehicles are commanded to follow these velocity vectors [48]. Stability for tracking straight lines, circular arcs, and circular paths is shown in [29].

1.2 Motivation

The main motivation of this work is to reduce the risk for humans in dangerous environments, which is commonly encountered in missions performed by the piloted aircrafts. The pilot will be secure far from the danger and the UAV will provide with the information needed to carry on with the assigned mission.

In the case of emergency situations such as natural disasters, finding potential survivors requiring medical attention is of major importance. Such missions require high navigation precision and long operation times -this is tedious for human pilots. UAV systems can be planned to autonomously execute complete missions from takeoff to landing. In this way, video footage of every square meter of a devastated area can be collected or even medicines, food, water, etc. can be delivered to a temporarily non-accessible person.

The use of UAVs and specifically the fixed wing UAV could help to solve this kind of problems. However time is crucial, so there is a need for a fast deploying UAV and also with a fully autonomous mission the operator will be able to monitor the information provided by the aircraft instead of control it manually or semi-autonomously.

One of the advantages of the autonomous take-off and landing systems is the elimination of the human error in the equation. Human error is responsible for roughly 60% of the UAV accidents during operation, and surprisedly 50% of the incidents are during the take-off and landing procedure [2]. By eliminating the operator from manually controlling the aircraft during take-off and landing procedures and replace them with an autonomous system will greatly increase safety during operations.

1.2.1 Scope of Work

This thesis presents work for the creation and implementation of a low cost, autonomous aircraft. The aircraft is capable of performing a fully autonomous mission, from the take-off to landing, passing through designated waypoints. This is done with the use of different control techniques and path generation algorithms. These algorithms were tested on an experimental platform. To test the functionality of the developed algorithms a simulation environment is created to test the performance of the controllers before its implementation in the experimental platform.

1.3 Thesis Overview

Each of the following chapters provides a unique contribution to the overall goal of the autonomous take-off and landing system, which are summarized below.

Chapter 2 A full nonlinear dynamic model is derived for the general case of the fixed wing UAV. The chapter provides several reference systems and rotation with respect to the different axes present in the dynamic model of the fixed wing aircraft.

Chapter 3 In order to test the controllers developed and the path generation algorithms, a simulation platform is needed, which is the first test of the method developed. After that an experimental platform is needed. This chapter provides a detailed description of the simulation and experimental platforms developed during the PhD.

Chapter 4 This chapter contains the preliminary results dealing with the path generation and control in two dimensions. The path generation and the nonlinear lyapunov-based controller developed is fully tested in the simulation and experimental platforms described in Chapter 3.

Chapter 5 In this chapter the extended work to the three dimensional space is presented. The 2D path generation and control are extended to handle the third dimension, again both are tested on the simulation platform.

Chapter 6 The final contribution of this thesis, the autonomous take-off and landing algorithm, which uses the previous 3D path generation and the 3D path-following controller.

Chapter 7 This chapter presents some concluding remarks of the technique developed in this thesis, Future work is also addressed in this chapter.

1.4 Publications

1.4.1 Journal

[J1] E. S. Espinoza, O. Garcia, I. Lugo-Cárdenas, R. Lozano , "Modeling and Sliding Mode Control of a Micro Helicopter-Airplane System" in *Journal of Intelligent and Robotic Systems.* vol 73, no. 4, pp.469-486. January 2014

1.4.2 Conferences

[C1] I. Lugo-Cárdenas, S. Salazar, R. Lozano,"Lyapunov Based 3D Path Following Kinematic Controller for a Fixed Wing UAV" in *Proceedings of the 20th World Congress The International Federation of Automatic Control (IFAC2017)* At Toulouse, France July 2017

[C2] I. Lugo-Cárdenas, G. Flores, R. Lozano,"The MAV3DSim Hardware in the Loop Simulation Platform for Research and Validation of UAV Controllers" in *International Conference on Unmanned Aircraft Systems (ICUAS2016)*

[C2] G. Flores · I. Lugo-Cárdenas · R. Lozano , "6-DOF Hovering Controller Design of the Quad Tiltrotor Aircraft: Simulations and Experiments" in *Proceedings of the IEEE Conference on Decision and Control 2015(CDC2015)*, December 2014

[C3] I. Lugo-Cardenas · Gerardo F. · S. Salazar, "R. Lozano Dubins path generation for a fixed wing UAV" in *Conference: 2014 International Conference on Unmanned Aircraft Systems (ICUAS2014)*

[C4] I. Lugo-Cárdenas · G. Flores · R. Lozano , "The MAV3DSim: A Simulation Platform for Research, Education and Validation of UAV Controllers" in *Proceedings of the 19th World Congress The International Federation of Automatic Control (IFAC2014)*, At Cape Town, South Africa August 2014

[C5] A. Malo Tamayo · I. Lugo-Cárdenas · E.S. Espinoza Quesada , "Simulation Environment for the Development of Control Algorithms for Air or Water Vehicles", in *2nd IFAC Workshop on Research, Education and Development of Unmanned Aerial Systems(REDUAS2013)*, At University of Technology of Compiegne, Compiegne, France

Chapter 2

Mathematical Model

The basis of the analysis, simulation and control of an aircraft relies on the mathematical model of the vehicle. The movement of an aircraft can be seen as a rigid body and is described by six nonlinear second order differential equations and while numerous reference systems are used in aerospace applications, we limited to four reference systems: Inertial Geodesic, Earth-Centered Earth-Fixed, North-East-Down and Body systems.

2.1 Frame Reference Systems and Rotations

This section describes the various reference systems in which a vehicle can be represented in space. Also in this section the rotations that can be applied to a vehicle are presented, they are also introduced the Euler angles and the rotation matrix. The coordinate systems are a representation in space and help us to know the position of an object, depending on the selected coordinate system can be a position vector, said vector position is defined as the vector whose origin point O as end point P, see Figure 2-1, i.e., the vector applied from the origin O having as components the Cartesian coordinates $x,\ y,\ z$, the point P.

2.1.1 Coordinates Systems

In navigation and control of an aircraft there are several frameworks or coordinate systems used during the analysis and design of control systems[68]. In the navigation of an air vehicle at least two coordinate systems are needed. One to represent the orientation of the body and the

other one for the representation of the position of the vehicle.

2.1.2 Inertial Reference Frame

In an inertial frame the Newton's laws of motion are applied. Then any coordinated frame fixed to the Earth's surface is an inertial reference frame. We define the reference frame as $F_E\left(O_E, X_E, Y_E, Z_E\right)$. The origin O_E of the reference frame can be placed arbitrarily to suit the particularly needs of what we need to do. The frame axes can point in any direction in a perpendicular way to follow the rule of the right hand frame [23].

2.1.3 Geodesic Reference Frame

The geodesic reference frame is widely used in GPS-based navigation systems. In Figure 2-1 we can observe that the reference frame place a point P_g close to the Earth's surface in terms of longitude, latitude and height or altitude, which are indicated by (λ, τ, h). The longitude λ measures the rotational angle between the prime meridian and the measured point P_g and it has a range from $-180°$ to $180°$. The latitude measures the angle between the equatorial plane and the normal of the reference ellipsoid that passes through the measured point. The height is the local vertical distance between the measured point and the reference ellipsoid. The coordinate vectors that are expressed in the geodetic reference frame are expressed with a g subscript

2.1.4 Earth-Centered Earth-Fixed Frame

The ECEF frame rotates with the earth around its spin axis and the fixed point on the earth has a fixed set of coordinates. The origin O_e of the ECEF frame is located at the center of the earth, the z-axis Z_e is along the spin axis of the earth and points towards the north pole, the x-axis X_e intersects the sphere of the earth at $0°$ latitude and $0°$ longitude, finally the y-axis Y_e is orthogonal to the z- and x-axes following the right-hand rule.

Any coordinate vector expressed in the ECEF frame are denoted with a subscript$s$ e as follows:

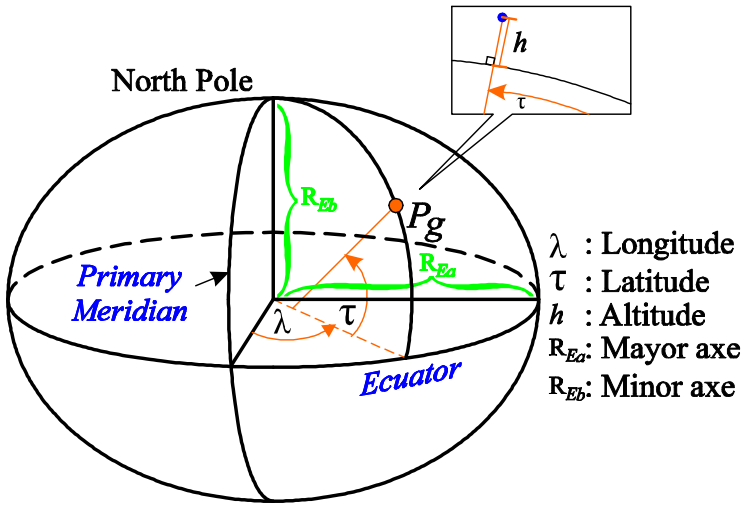


Figure 2-1: Geodesic reference frame

$$P_e = \left(\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} \right) \tag{2.1}$$

2.1.5 North-East-Down Coordinate System

The north east down (NED) frame is also known as a local tangent plane (LTP).It is a coordinate frame fixed on any arbitrary point on earth's surface. Based on the WGS84 episode model[59], the origin and axis are defined as follow:

- The origin O_n is arbitrarily fixed to a point on the earth's surface.
- The x-axis X_n points toward the ellipsoid north.
- The y-axis Y_n points toward the ellipsoid east.
- The z-axis Z_n points downward along the ellipsoid normal in order to comply with the right-hand rule.

Coordinate vectors expressed in the NED system are denoted with a subscript n. The position and velocity vectors are defined as follows

$$\mathbf{P}_n = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix}$$

$$\mathbf{V}_n = \begin{bmatrix} u_n \\ v_n \\ w_n \end{bmatrix}$$

During the flight test we normally select the origin of this coordinate system as the aircraft's takeoff point. We use $h = -z$ to denote the actual height of the unmanned system.

The vehicle-carried vertical axis system [19] has its origin at the center of gravity of the vehicle. The X_v axis points toward the ellipsoid north, the Y_v axis ward the ellipsoid east, and the Z_v axis points downward . This axis system is obtained by a translation of the (NED) coordinate system to the vehicle center of gravity. The attitude of the aircraft (heading, pitch, and bank angles) is described in terms of the orientation of the aircraft body axes with respect to the vehicle-carried vertical axes.

2.1.6 Body Frame

The body frame is not an inertial system and it is fixed to the moving vehicle. The orientation of the body coordinate axes is defined as follow

- The origin O_b is located an the center of gravity (CG) of the flying vehicle.
- The x-axis X_b points forward through the nose of the aircraft lying in the symmetric plane of the vehicle
- The y-axis Y_b points to the right of the x-axis, on the right side of the vehicle.
- The z-axis Z_b points downward to comply with right-hand rule.

Coordinate vectors expressed in the body frame are appended with a subscript b. We can define the NED velocity vector as

$$\mathbf{V}_b = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.2}$$

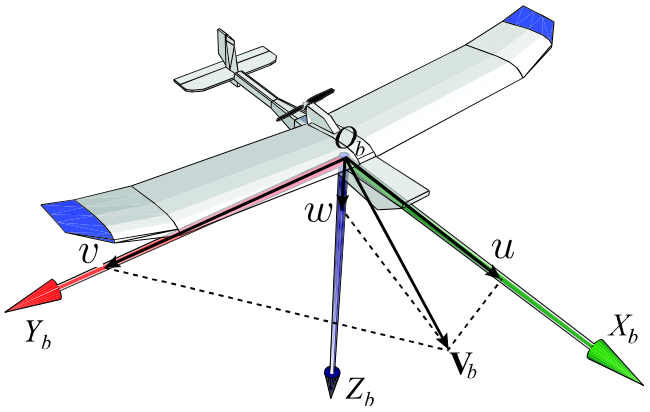


Figure 2-2: Body coordinate system

2.1.7 Wind and Stability Axes

In the case that the X_B axis of the body frame pitch with angle of attack α, then the body axis frame is referred to as *Stability Frame* $F_S(O_S, X_S, Y_S, Z_S)$ as shown in Figure 2-3. Stability frame generates the aerodynamic forces, and thus reducing the aerodynamic model to its simplest form. Whenever the Body Frame faces a wind, it is going to yaw into the wind with Angle of Sideslip β. Then the body axis frame is referred to as Wind Frame $F_W(O_W, X_W, Y_W, Z_W)$. The wind frame axes definition is illustrated in Figure 2-3. The angle of side-slip 13 and angle of attack a define the orientation of the wind axes with respect to the body axes.

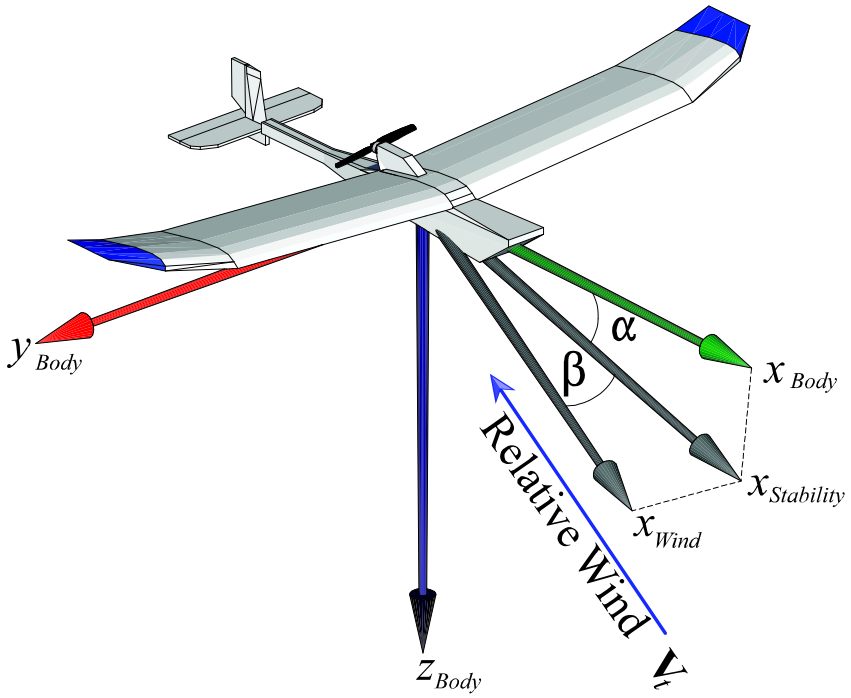


Figure 2-3: Wind and stability axes

The definition of the velocities of the body axis are

$$u = V\cos\alpha\cos\beta \tag{2.3}$$

$$v = V\sin\beta \tag{2.4}$$

$$w = V\sin\alpha\cos\beta \tag{2.5}$$

The velocity V, angle of attack α and the angle of side slip β can be expressed in terms of the velocities of the body axis

$$V = |\mathbf{V}_b| = \left(u^2 + v^2 + w^2\right)^{\frac{1}{2}} \tag{2.6}$$

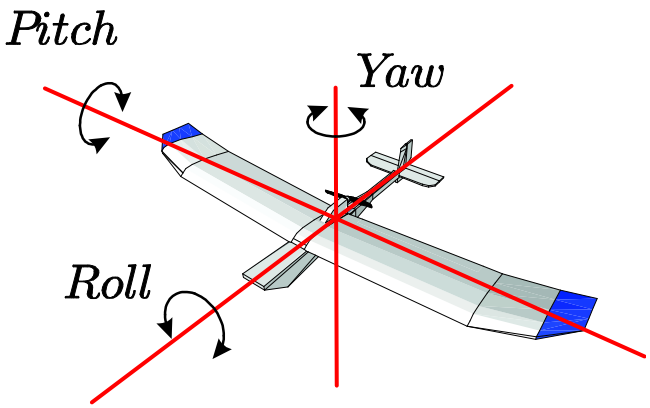
$$\alpha = \tan^{-1}\frac{w}{u} \tag{2.7}$$

$$\beta = \sin^{-1}\frac{v}{V} \tag{2.8}$$

2.1.8 Euler Rotations

The Euler angles[20] are used to describe the orientation of a rigid body in a 3-dimensional Euclidean space, Figure.2.1.8 . There are several conventions for Euler angles, depending on the axes about which the rotations are carried out, but the most common definitions is given by Euler angles (ϕ, θ, ψ) as follow

- Yaw angle, denoted by ψ, is the angle from the vehicle-carried NED X-axis to the projected vector of the body X-axis on the $X - Y$ plane of the vehicle-carried NED frame. The right-handed rotation is about the vehicle-carried NED Z-axis.
- Pitch angle, denoted by θ , is the angle from the X-axis of the once-rotated intermediate frame to the body frame X-axis. The right-handed rotation is about the Y-axis of the once-rotated intermediate frame
- Roll angle, denoted by ϕ, is the angle from the Y-axis (or Z-axis) of the twice rotated intermediate frame to that of the body frame.



The three relative rotation matrices are respectively given by

$$R_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

$$R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.10}$$

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2.11}$$

The kinematic relationships between the NED and body frames are used in the flight dynamics modeling and automatic control. For translational kinematics, we have

$$\mathbf{V}_b = R_n^b \mathbf{V}_n$$

where R_n^b is the rotation matrix from the NED frame to the body frame and is given by

$$R_n^b = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{2.12}$$

In rotational kinematics we focus on the velocity vector $\omega_{b/n}^b$, which describes the rotation of the vehicle NED frame with respect to the body frame. Following the sequence of the Euler

angles, the velocity vector can be expressed as

$$\begin{aligned}\omega_{b/n}^{b} &= \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_{\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ r \end{bmatrix} + R_{\theta} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\ &= S \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \end{aligned} \tag{2.13}$$

where p, q, r are the standard symbols for the components of $\omega_{b/n}^{b}$. S was derived by [50] from the total angular velocity and is given by

$$S = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\theta\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{2.14}$$

2.2 Mathematical Model

The nonlinear state equations for the aircraft problem can be seen as a state vector $\mathbf{x}$ composed of four 3×1 sub-vectors which represents the aircraft rotational velocity, translational velocity, the vehicle attitude and vehicle position:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \mathbf{x}_3^T & \mathbf{x}_4^T \end{bmatrix} \tag{2.15}$$

where

$$\begin{aligned} \mathbf{x}_1 &= \begin{bmatrix} p & q & r \end{bmatrix}^T \\ \mathbf{x}_2 &= \begin{bmatrix} V & \alpha & \beta \end{bmatrix}^T \\ \mathbf{x}_3 &= \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \\ \mathbf{x}_4 &= \begin{bmatrix} x & y & h \end{bmatrix}^T \end{aligned}$$

with $\mathbf{x}_1$, is the rotational velocity, $\mathbf{x}_2$, is the translational velocity, $\mathbf{x}_3$ the vehicle attitude and $\mathbf{x}_4$

the vehicle position. The vehicle rotational and translational velocity are defined within the aircraft body and -fixed axis systems.

The rotational acceleration terms in the $\mathbf{x}$ vector are derived from the moment equation

$$\mathbf{M} = \frac{d}{dt}\mathbf{H} \tag{2.16}$$

where M is the total moment on the vehicle and H is the total angular momentum of the vehicle. The total angular momentum can be replaced with the product of the inertia tensor I and the rotational velocity vector Ω, thus for the angular momentum we obtain

$$\mathbf{H} = I\mathbf{\Omega} \tag{2.17}$$

The inertia tensor is assumed to be constant with time. Equation 2.16 can be expanded to

$$\mathbf{M} = \frac{\delta}{\delta t}\left(I\mathbf{\Omega}\right) + \mathbf{\Omega} \times \left(I\mathbf{\Omega}\right) \tag{2.18}$$

where $\frac{\delta}{\delta t}$ is the time derivative operator in a moving reference frame.

The definition of the total moment of the vehicle in 2.16 follow:

$$\mathbf{M} = \begin{bmatrix} \sum L \\ \sum M \\ \sum N \end{bmatrix} = \begin{bmatrix} \overline{L} + L_T \\ M + M_T \\ N + N_T \end{bmatrix} \tag{2.19}$$

with L, M and M being the aerodynamic total moments about the x, y and z body axes, and L_T, M_T and N_T the sums of all actuator-induced moments.

$$\mathbf{I} = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix} \tag{2.20}$$

where I_x, I_y and I_z are the moments of inertia about the X_b, Y_b and Z_b body axes and I_{xy}, I_{xz}

and I_{yz} are the products of inertia in the $x-y,\ x-z$ and $y-z$, respectively; and

$$\mathbf{\Omega} = \mathbf{x}_1 = \begin{bmatrix} p & q & r \end{bmatrix}^T \tag{2.21}$$

where p, q, and r are the rotational rates about the X_b, Y_b and Z_b. We can rewrite equation 2.18 as

$$\frac{\delta}{\delta t}\mathbf{\Omega} = \mathbf{I}^{-1}\left(\mathbf{M} - \mathbf{\Omega} \times \mathbf{I\Omega}\right) \tag{2.22}$$

This is the vector sub-function for the rotational acceleration. Using 2.22 and 2.21 we know that

$$\frac{\delta}{\delta t}\mathbf{\Omega} = \begin{bmatrix} \dot{p} & \dot{q} & \dot{r} \end{bmatrix}^T$$

When the symmetry plane is the $x-z$ plane we have

$$I_{xy} = I_{yz} = 0$$

and the only term outside the main diagonal left is I_{zx}. Then the inertia tensor I is given by

$$I = \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix}$$

Expanding equation 2.18 to a set of scalar equations

$$L = \dot{p}I_{xx} + qr\left(I_{zz} - I_{yy}\right) - \left(\dot{r} + pq\right)I_{xz} \tag{2.23}$$

$$M = \dot{q}I_{yy} + pr\left(I_{xx} - I_{zz}\right) - \left(r^2 + p^2\right)I_{xz} \tag{2.24}$$

$$N = \dot{r}I_{zz} + pq\left(I_{yy} - I_{xx}\right) - \left(\dot{p} - qr\right)I_{xz} \tag{2.25}$$

From equation 2.22 the inverse of the inertia tensor I^{-1} is given by

$$\mathbf{I}^{-1} = \frac{1}{I_{xx}I_{zz} - I_{xz}^2}\begin{bmatrix} I_{zz} & 0 & I_{xz} \\ 0 & \left(I_{xx}I_{zz} - I_{xz}^2\right)I_{yy}^{-1} & 0 \\ I_{xz} & 0 & I_{xx} \end{bmatrix} \tag{2.26}$$

Rearranging terms the equation 2.24 can be written as a vector function

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (C_1 r + C_2 p)\, q \\ C_7 rp + C_6 \left(r^2 - p^2\right) \\ (C_8 p + C_9 r)\, q \end{bmatrix} + \begin{bmatrix} C_3 & 0 & C_4 \\ 0 & C_5 & 0 \\ C_4 & 0 & C_{10} \end{bmatrix} \begin{bmatrix} \sum L \\ \sum M \\ \sum N \end{bmatrix} \tag{2.27}$$

where

$$\begin{aligned} C_0 &= \left(I_{xx} I_{zz} - I_{xz}^2\right)^{-1} \\ C_1 &= C_0 \left(\left(I_{yy} - I_{zz}\right) I_{zz} - I_{xz}^2\right) \\ C_2 &= C_0 I_{xz} \left(I_{xx} - I_{yy} + I_{zz}\right) \\ C_3 &= C_0 I_{zz} \\ C_4 &= C_0 I_{xz} \\ C_5 &= I_{yy}^{-1} \\ C_6 &= C_5 I_{xz} \\ C_7 &= C_5 \left(I_{zz} - I_{xx}\right) \\ C_8 &= C_0 \left(\left(I_{xx} - I_{yy}\right) I_{xx} + I_{xz}^2\right) \\ C_9 &= C_0 I_{xz} \left(I_{yy} - I_{zy} - I_{xx}\right) \\ C_{10} &= C_0 I_{xx} \end{aligned}$$

and finally we can see 2.27 in scalar form as follows

$$\dot{p} = (C_1 r + C_2 p)\, q + C_3 \sum L + C_4 \sum N \tag{2.28}$$

$$\dot{q} = C_7 rp + C_6 \left(r^2 - p^2\right) + C_5 \sum M \tag{2.29}$$

$$\dot{r} = (C_8 p + C_9 r)\, q + C_4 \sum L + C_{10} \sum N \tag{2.30}$$

The inertia moments are usually considered as constant for simulation purposes and the variations of mass and center of gravity are not considered.

2.2.1 Translational acceleration

Derivation of the translational acceleration is based on the force equation

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{V}) \tag{2.31}$$

where $\mathbf{F}$ is the total force acting on the vehicle and m is the vehicle mass, which is considered as constant. Equation 2.31 can be expanded as

$$\mathbf{F} = m\left(\frac{d}{dt}V + \omega \times \mathbf{V}\right) \tag{2.32}$$

and taking the following definition of $\mathbf{F}$ and $\mathbf{V}$

$$\mathbf{F} = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T \tag{2.33}$$

where F_x, F_y, and F_z are the sum of aerodynamic, gravitational and forces ejerced by the motor engine in the body axis X_b, Y_b and Z_b, and using 2.2 we can rearrange the terms in equation 2.32 we obtain the expression for the translational acceleration

$$\frac{d}{dt}\mathbf{V}_b = \frac{1}{m}\mathbf{F} - \boldsymbol{\omega} \times \mathbf{V}_b \tag{2.34}$$

$$\dot{\mathbf{V}}_b = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{F_x}{m} + rv - qw \\ \frac{F_y}{m} + pw - ru \\ \frac{F_z}{m} + qu - pv \end{bmatrix} \tag{2.35}$$

where

$$F_x = X_T + X_a + X_g \tag{2.36}$$

$$F_y = Y_T + Y_a + Y_g \tag{2.37}$$

$$F_z = Z_T + Z_a + Z_g \tag{2.38}$$

where the subindex T means the force generated by the propulsion system, subindex a means that they are aerodynamic forces and subindex g means they are forces produced due to the

earth gravity

The forces produced by the propulsion system is generated by the motor engine and is defined as

$$\begin{aligned} X_T &= T_t \\ Y_T &= 0 \\ Z_T &= 0 \end{aligned}$$

The aerodynamic forces of the body axis can be written in terms of the lift L, drag D and side force Y forces as follows

$$F_a = S_\alpha^T F_w = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} -D \\ Y \\ -L \end{bmatrix}$$

$$X_a = -D\cos\alpha + L\sin\alpha \tag{2.39}$$

$$Y_a = Y \tag{2.40}$$

$$Z_a = -D\sin\alpha - L\cos\alpha \tag{2.41}$$

The gravitational forces $X_g,\ Y_g,\ Z_g$ can be expressed as

$$X_g = -mg\sin\theta \tag{2.42}$$

$$Y_g = mg\sin\phi\cos\theta \tag{2.43}$$

$$Z_g = mg\cos\phi\cos\theta \tag{2.44}$$

2.2.2 Attitude Rates

This transformation from earth-fixed to body axes can be expressed by the equation

$$\Omega = S\left(\frac{d}{dt}\mathbf{E}\right) \tag{2.45}$$

where $\mathbf{E}$ is the attitude vector whose components are the Euler angles defined in 2.1.8

$$\mathbf{E} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T \tag{2.46}$$

premultyplying both sides of the equation (2.45) by S^{-1} and rearranging the terms the terms of the equation we get the equation for the attitude rates

$$\frac{d}{dt}\mathbf{E} = S^{-1}\Omega$$

where

$$S^{-1} = \frac{1}{\cos\theta}\begin{bmatrix} \cos\theta & \sin\theta\sin\phi & \sin\theta\cos\phi \\ 0 & \cos\theta\cos\phi & -\cos\theta\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.47}$$

which can be expanded into the scalar equations

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{2.48}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{2.49}$$

$$\dot{\psi} = q\sin\phi\sec\theta + r\cos\phi\sec\theta \tag{2.50}$$

2.2.3 Earth-relative velocity

The matrix R_n^b that transforms earth axis system vectors into the body axis system is defined by equation (2.12) as

$$R_n^b = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{2.51}$$

The relationship between earth-relative velocities and body axis velocities is expressed by

$$\mathbf{V} = R_n^b\left(\frac{d}{dt}\mathbf{R}\right)$$

where $\mathbf{R}$ is the earth axis system vector defining the location of the vehicle

$$\mathbf{R} = \begin{bmatrix} x & y & z \end{bmatrix}^T$$

with $z = -h$

The equation for the earth-relative velocity can be formulated as

$$\frac{d}{dt}\mathbf{R} = R_{nb}^{-1}\mathbf{V} \tag{2.52}$$

these velocities are expressed in terms of body axis velocities. Using equation (2.52) and the definitions for the body axis velocities in equations (2.3) to (2.5) allow us to define the earth-relative velocities to be expressed in terms of V, α, β :

$$\dot{h} = V\left(\cos\alpha\cos\beta\sin\theta - \sin\beta\sin\phi\cos\theta - \sin\alpha\cos\beta\cos\phi\cos\theta\right) \tag{2.53}$$

$$\dot{x} = V\left(\begin{array}{c} \cos\alpha\cos\beta\cos\theta\cos\psi + \sin\beta\left(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi\right) \\ + \sin\alpha\cos\beta\left(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi\right) \end{array}\right) \tag{2.54}$$

$$\dot{y} = V\left(\begin{array}{c} \cos\alpha\cos\beta\cos\theta\sin\psi + \sin\beta\left(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi\right) \\ + \sin\alpha\cos\beta\left(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi\right) \end{array}\right) \tag{2.55}$$

2.2.4 Force Coefficients and Aerodynamic Moments

The aerodynamic forces and moments that act on the aircraft are the result of multiple factors and the impact of the same varies depending on the flight conditions as well as change from one vehicle to another. In general, these forces and moments are nonlinear functions dependent mainly on Mach number, angle of attack α, lateral slip angle β, altitude, rotational velocities and deflections of control surfaces.

The forces and moments are related to the force and coefficients and the dimensionless moments by means of the equations of forces

$$D = \bar{q}SC_D$$

$$L = \bar{q}SC_L$$

$$Y = \bar{q}SC_Y$$

and the moment equations

$$\begin{aligned} \bar{L} &= \bar{q}SbC_l \\ M &= \bar{q}S\bar{c}C_m \\ N &= \bar{q}SbC_n \end{aligned}$$

where

$$\begin{aligned} \bar{q} &= \text{dynamic pressure} \\ S &= \text{wing reference area} \\ b &= \text{wing span} \\ \bar{c} &= \text{geometric mean wing length} \end{aligned}$$

and the dynamic pressure $\bar{q}$ is modeled as follows

$$\bar{q} = \frac{1}{2}\rho V_t^2$$

and the aerodynamic coefficients are

$$\begin{aligned} C_D &= C_D(C_L) + \Delta C_D(\delta_e) + \Delta C_D(\beta) + \Delta C_D(M) \\ C_L &= C_L(\alpha, T_c) + \Delta C_L(\delta_e) + \Delta C_L(M) \\ C_Y &= C_Y(\beta) + \Delta C_Y(\delta_r) \\ C_l &= C_l(\beta) + \Delta C_l(\delta_a) + \Delta C_l(\delta_r) + \frac{b}{2V_T}\left[C_{l_p}p + C_{l_r}r\right] \\ C_m &= C_m(C_L, T_c) + \Delta C_m(\delta_e) + \Delta C_m(M) + \frac{\bar{c}}{2V_T}\left[C_{m_q}q + C_{m_\alpha}\dot{\alpha}\right] \\ C_n &= C_n(\beta) + \Delta C_n(\delta_r) + \Delta C_n(\delta_a) + \frac{b}{2V_T}\left[C_{n_p}p + C_{n_r}r\right] \end{aligned}$$

While the various dimensionless coefficients C_D, C_Y, C_L C_l, C_m, C_n depend mainly on the aerodynamics of the angles α, β and depend less on others variables. It can be seen the dependence on the change of velocity of the aerodynamic angles velocities and the dependence of the components p, q and r of the angular velocity of the aircraft's center of gravity. The

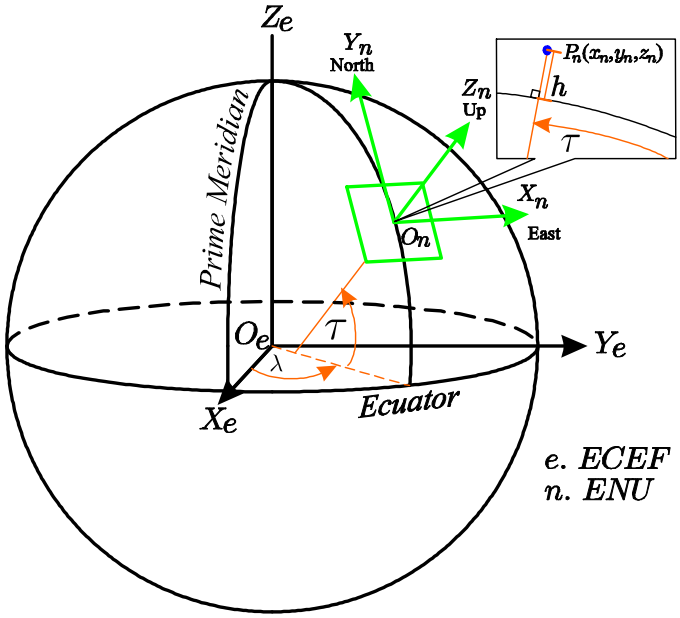


Figure 2-4: Geodetic, ECEF and ENU coordinates frames

coefficients are also dependent on the control surfaces, otherwise the vehicle could not be controlled. Aerodynamic coefficients are dependent on other factors such as thrust level and effects due to proximity to the ground.

2.2.5 Coordinates Transformation

The geodetic coordinate system is used in many fields, such as: navigation, surveying and cartography, in order to define the position of an object on the Earth's surface we use a set of three values called *geodetic coordinates* [22]. However, the geodetic coordinates lack of an intuitive understanding of distance, unlike other coordinate systems as the local East, North, Up (ENU) Cartesian coordinate system. The local ENU coordinates are formed from a plane tangent to the Earth's surface fixed to a specific location and it is known as a Local Tangent Plane (LTP). By convention the east axis is labeled x, the north y and the up z. The three different coordinate systems are represented in the Figure 2-4.

Geodetic to ECEF coordinates

Here we introduce the equations to convert geodetic coordinates measurements to Local Tangent Plane coordinates. The method used passes through the Earth-Centered, Earth-Fixed (ECEF) rectangular coordinate system on the way to the Local Tangent Plane.

Geodetic coordinates (latitude τ , longitude λ , height h) can be converted into ECEF coordinates using the following relationships:

$$\begin{aligned} X &= (N(\tau) + h)\cos\tau\cos\lambda \\ Y &= (N(\tau) + h)\cos\tau\sin\lambda \\ Z &= \left(N(\tau)\left(1 - e^2\right) + h\right)\sin\tau \end{aligned} \tag{2.56}$$

where

$$N(\tau) = \frac{a}{\sqrt{1 - e^2\sin^2\tau}}$$

The semi-major axis and the first numerical eccentricity of the ellipsoid are represented by a and e, respectively, the numeric value of this constants can be found in the definition of the World Geodetic System 1984 [59]. $N(\tau)$ is the distance from the surface to the to the Z-axis along the ellipsoid normal.

ECEF to Local Tangent coordinates

A local reference point is needed to perform a coordinate transformation from ECEF to the local ENU coordinates. The launching site position will serve as the local reference point. If the launching site is at (λ_0, τ_0, h_0) in geodetic coordinates, then using the previous coordinate transformation we obtain (X_0, Y_0, Z_0), the launching site expressed in ECEF coordinates. The aircraft location is defined as (λ, τ, h); we use the same coordinate transformation to obtain (X, Y, Z), the aircraft position expressed in ECEF coordinates. The vector pointing from the launching site to the aircraft in the ENU coordinate system is computed as follows

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R} \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix} \tag{2.57}$$

where

$$\mathbf{R} = \begin{bmatrix} -\sin\lambda_0 & \cos\lambda_0 & 0 \\ -\sin\tau_0\cos\lambda_0 & -\sin\tau_0\sin\lambda_0 & \cos\tau_0 \\ \cos\tau_0\cos\lambda_0 & \cos\tau_0\sin\lambda_0 & \sin\tau_0 \end{bmatrix}$$

The World Geodetic System of 1984 (WGS84)[59] comprises a standard coordinate system and is one of the most used coordinate system used on GPS devices and we will use the coordinate transformations defined in this section to express the position of the airplane in the local ENU tangent plane which is suitable for the mathematical model and control purposes.

Chapter 3

Simulation and Experimental Platform

There is an enormous variety of UAV applications and the greate interest around them has produced a new industry in the productions of drones. There are different groups involved in the development of UAV products, from actuators, sensors to complete autopilots especially developed for the UAV, this includes the software for the autopilot, and we can find the solutions as open or closed source.

Few work has been done on development of complete model-based UAV simulators. For example, a real-time simulation of a quadrotor is presented in [31], where the real-time simulation was performed in MATLAB/Simulink by means of the xPC Target, in which a pair of host PC and two PC targets were used. In [66] a commercial flight simulator has been used as the simulation engine for the quadrotor Pelican from Asc. Technologies; this represents a disadvantage due to the fact that the source code is not available for review and/or modification.

For several decades, simulation and implementation has been bridged through the use of Hardware In the Loop Simulation (HIL). HIL simulation combines a simulated system with physical hardware. For example, a software simulation of the system plant is augmented with actuators and sensors from the designed system. HILS systems have facilitated the development in numerous fields, including automotive engineering [28], [32], aerospace [36], power systems [49] and robotics [13].

In this chapter we present the simulation and experimental platform for UAVs. The simulation platform named MAV3DSim (Multi-Aerial Vehicle 3D Simulator), which is capable of simulating realistic scenarios by using elaborated versions of UAV mathematical models. The MAV3DSim simulator allows the user to test controllers before being implemented on the UAV platform; in this manner, the control engineer can design controllers by taking simplified mathematical models and then test such controllers on the complete model provided by the simulator. On the other hand, the MAV3DSim simulator has several characteristics which improves its efficiency, such as the ability of tuning gains online and the visualization of any variable involved in the system, also it has the possibility to export all the acquired data to a MATLAB compatible format for plotting and further analysis.

The use of small UAV's are specially appealing because of the variety of inexpensive components for building and repairing the UAV. This chapter describes the equipment used in the development of the experimental platform.

3.1 MAV3DSim Simulator

The top-level requirements to support a flight test of the low-cost UAV were identified as:

1. Test all custom developed software extensively : guidance, navigation and control algorithms.

2. Test onboard computer hardware, operating system implementation, and software execution in real-time.

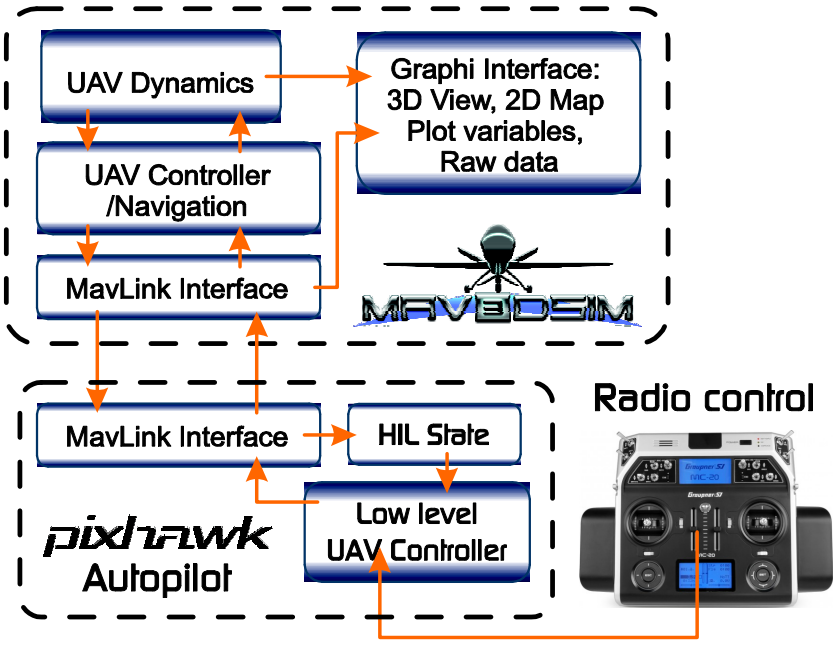
3. Rehearse all procedures and flight test plans.

4. Rehearse control failure and implement mechanisms to gain manual control of the aircraft in the case of control failure.

5. Use it in flight test location as a ground control station(GCS) with the experimental platform.

In order to satisfy this requirements we developed a custom simulation platform that can be used for development and test of new control and guidance algorithms. The MAV3DSim

simulator is a custom application developed in the Microsoft's C# .Net programming language. The MAV3DSim can collect all the simulated sensor data from the simulation engine and use this as input for the controllers. The sensor data is in a standard protocol named MAVLink[70], this protocol sends sensor data as well as control commands, sensor data such as the inertial accelerations, rotational velocities, GPS position, airspeed and attitude of the simulated vehicle and control commands like the angular deflection of the control surface of an airplane and the speed of the main rotor. The MAV3DSim could work as a pure simulation platform, as presented in [33], but we now have extended its functionality to perform Hardware In the Loop (HIL) simulations; using the very same protocol we can communicate to the embedded hardware: the Pixhawk autopilot [14]. Once the simulator collects the data from the simulation engine it will send it to the Pixhawk autopilot and it will process it as if it is the data collected from the physical sensors (inertial measurement unit, GPS, airspeed, barometer, etc.), then the autopilot will compute the control and send it back to the simulation engine as shown in Figure.3.1.



MAV3DSim Hardware in the loop block diagram.

The MAV3DSim hardware in the loop simulator consists of three main components, the simulator engine, the Pixhawk autopilot and the data visualization interface. The simulation

engine is in charge of the numeric integration of the dynamic equations of the UAV, here we can choose between a fixed wing UAV and a quadrotor for use it in the simulation. The input of the simulation engine is the Pixhawk autopilot's output and using this information it computes the new vehicle state and sends it back to the autopilot as sensor data. The autopilot has implemented a variety of controllers, in our case it receives the current state of the vehicle as an input, then calculates the controller output and sends it to the simulation engine as deflection command for the control surfaces. The data visualization interface looks like a GCS type of application, where all the variables of the state vector of the UAV can be represented in different ways, and with the addition of a 3D visualization of the attitude and position of the UAV in a 3D scenario.

It is worth mentioning that the MAV3DSim has the facility to implement and test new controllers, as showed in our previous work[33], were two different types of controllers were successfully implemented on the simulator using two different types of UAVs, a fixed wing and a quadcopter. These controllers should be programmed directly in the source code of the simulator, there are already implemented mechanisms to execute different controllers and the addition of new ones should not be any problem.

Mathematical Model

In this section we present the mathematical model used by the simulation engine for the airplane and the quadrotor.

These equations are derived and fully described in [53], this reference was found inside the source code of the CRRCSim simulator and we have validated its correct implementation by comparing the programmed source code with the equations described in the NASA report [53].

For any aircraft in the simulation engine, the state vector $\mathbf{x}$ is a 13×1 vector representing the vehicle location, the inertial velocity, the vehicle attitude and the vehicle rotational velocity. The rotational and inertial velocities are referenced in the body frame while the attitude and vehicle location are referenced to an inertial frame.

Translational Equations

$$\begin{aligned} \dot{\lambda} &= \frac{V_N}{R} \\ \dot{\tau} &= \frac{V_E}{R\cos\lambda} \\ \dot{R} &= -V_D \end{aligned} \tag{3.1}$$

where λ is the latitude and τ is the longitude. R is the distance from center of the earth to the vehicle. The time differential inertial velocity vector $[V_N, V_E, V_D]$ is computed using the following equations

$$\begin{aligned} \dot{V}_N &= \frac{F_N}{m} + \frac{V_N V_D - V_E^2 \tan\lambda}{R} \\ \dot{V}_E &= \frac{F_E}{m} + \frac{V_E V_D + V_N V_E \tan\lambda}{R} \\ \dot{V}_D &= \frac{F_D + F_G}{m} - \frac{V_N^2 + V_E^2}{R} \end{aligned} \tag{3.2}$$

where F_N, F_E and F_D are the components of the applied force vector on the vehicles center of gravity and F_G is the force of gravity.

Attitude equations in quaternions

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{3.3}$$

These equations represent the time derivative of the rotation expressed in quaternions, and to obtain an equivalent representation of the angle from the quaternion expressed in the Euler angles(ϕ, θ, ψ) we have the following relations

$$\begin{aligned}
\tan\phi &= \frac{2(q_2q_3+q_0q_1)}{q_0^2-q_1^2-q_2^2+q_3^2} \\
\sin\theta &= -2(q_1q_3-q_0q_2) \\
\tan\psi &= \frac{2(q_1q_2+q_0q_3)}{q_0^2+q_1^2-q_2^2-q_3^2}
\end{aligned} \tag{3.4}$$

The rotational velocity dynamic are presented in the following equations

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (C_1r + C_2p)\,q \\ C_7rp + C_6\left(r^2-p^2\right) \\ (C_8p + C_9r)\,q \end{bmatrix} + \begin{bmatrix} C_3 & 0 & C_4 \\ 0 & C_5 & 0 \\ C_4 & 0 & C_{10} \end{bmatrix} \begin{bmatrix} L \\ M \\ N \end{bmatrix} \tag{3.5}$$

where

$$\begin{aligned}
C_0 &= \left(I_{xx}I_{zz}-I_{xz}^2\right)^{-1} \\
C_1 &= C_0\left(\left(I_{yy}-I_{zz}\right)I_{zz}-I_{xz}^2\right) \\
C_2 &= C_0I_{xz}\left(I_{xx}-I_{yy}+I_{zz}\right) \\
C_3 &= C_0I_{zz} \\
C_4 &= C_0I_{xz} \\
C_5 &= I_{yy}^{-1} \\
C_6 &= C_5I_{xz} \\
C_7 &= C_5\left(I_{zz}-I_{xx}\right) \\
C_8 &= C_0\left(\left(I_{xx}-I_{yy}\right)I_{xx}+I_{xz}^2\right) \\
C_9 &= C_0I_{xz}\left(I_{yy}-I_{zy}-I_{xx}\right) \\
C_{10} &= C_0I_{xx}
\end{aligned}$$

and I_{xx}, I_{yy}, and I_{zz} are the moments of inertia about the $x,\ y$, and z body axes, respectively, and I_{xy} , I_xz, and I_{yz} are the products of inertia in the $x-y$, $x-z$, and $y-z$ body axis planes, respectively. L, M, and N being the aerodynamic total moments about the x, y, and z body

axes.

The fixed-wing and the quadrotor use the same mathematical model, the only difference is in the computation of the forces $[F_N, F_E, F_D]$ and moments $[L, M, N]$, the fixed-wing moments and forces mainly depends on the aerodynamic coefficients and the deflection of the control surfaces and the quadrotor dynamics depends on the moments and forces generated by the rotors.

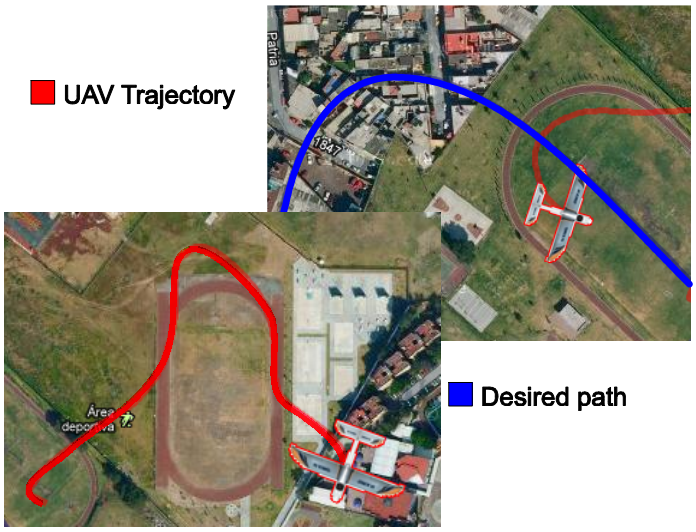
The input of the simulator is the deflection of the ailerons, in the case of the fixed-wing aircraft, and it affects directly the forces $[F_N, F_E, F_D]$ and moments $[L, M, N]$ applied in the simulation.

3.1.1 Graphic User Interface

A very important feature is the display of the state vector $\mathbf{x}$ delivered by the simulator. We will explain next the different visualization options that the MAV3DSim includes.

Map

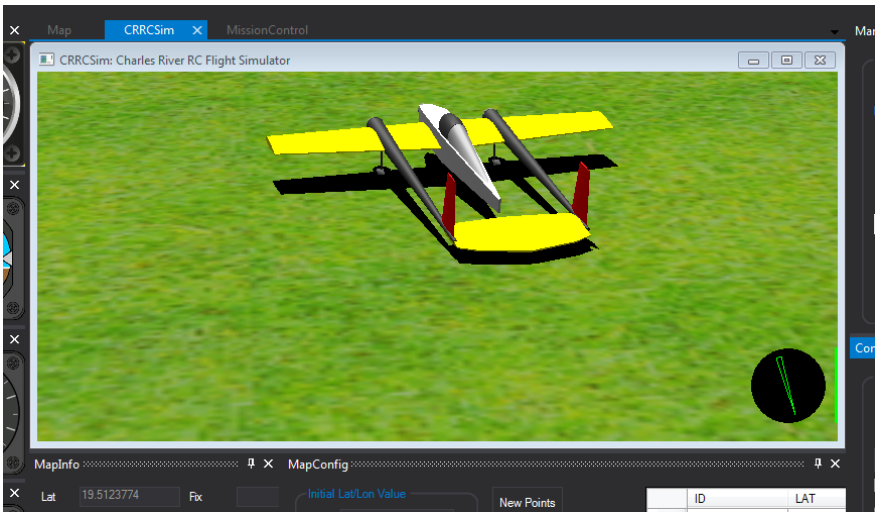
One of the main visualization is the map, which can locate the aircraft in some point on the Earth and it is provided by Google Maps, using the displayed map we can visualize the position of the aircraft and also the path generated by the aircraft. Another feature of the map visualization is that we can add a desired or reference path along with waypoints for trajectory tracking as seen in Figure. 3.1.1.



Map visualization of the UAV desired and actual trayectory.

3D View

A 3D view is also available in the simulator, and it is useful for the 3D representation of the simulated aircraft attitude, it can be seen as a vintage point where a pilot at the ground could be standing or also with a close-up for a better visualization of the aircraft's attitude. The 3D view is provided by the CRRCSim simulator engine, Figure 3.1.1.



3D view is provided by the CRRCSim simulator engine

Avionics instruments

Avionics instruments like those used in commercial aircraft are used to display some of the state variables of the aircraft:

- Altimeter: Indicates the altitude relative to a reference level at which the aircraft is flying.
- Attitude indicator: Shows the position of the longitudinal and transversal aircraft axes with respect to the natural horizon, this is obtained by reading the roll and pitch angles.
- Heading indicator: Displays the aircraft heading with respect to magnetic north.
- Airspeed indicator: Gives the aircraft speed relative to the surrounding air.
- Vertical speed indicator: Displays the vertical speed of the airplane, going down towards the center of the Earth is negative velocity and going up is positive.

Those avionics instruments are depicted in Figure. 3-1

Figure 3-1: Avionic instruments in the MAV3DSim.

Plain data visualization

There are several ways to visualize plain data obtained from the simulator and the controls obtained from the Pixhawk, a simple way is a raw data panel which shows the instant data from either sources. A more useful visualization is provided by the plot of the data with respect to time, which shows the changes of the any variable w.r.t. time. An alternative way is to export this data as a Comma Separated Value (CSV) format for further analysis.

Gain tuning

Gain tuning is a time consuming task and to reduce the time of the gain tuning it has the possibility to change the gains online and to see the effect of the new gains in the simulation. Each slide can be set to a specific gain and the range can be set as needed, depicted in Figure 3.1.1.

Integrated tool to set the gains of the developed controllers.

3.2 Airframe

We decide to use commercially available airframes as the cost in time and effort of designing a fixed wing airplane is too high. Therefore we use airframes already tested and which are guaranteed to fly. We test a total of 3 airframes and they will be described below.

3.2.1 Bixler

The first airframe was the HobbyKing^{TM} Bixler^{TM}, an in-expensive platform which is made of Expanded PolyOlefin (EPO) foam and is very easy to repair after an inevitable crash. It is a small platform with a wingspan of 1.4m and a length of 0.925 and a total wing area of 26dm^2. The motor used in this airframe is a 2620-1900kv Brushless Outrunner drived by an electrical speed controller of 20 amps and it uses 4 9g standard servos to move the control surfaces aileron, elevator and rudder. As it is a small airframe it cant carry a lot of payload and a small battery of 2200mAh and 3s Li-Po battery was used.

3.2.2 FPV-Raptor

The FPV-Raptor from Lanyu Hobby frame has a blow moulded fuselage which uses Nylon and is virtually indestructible, it can handle very well hard landings and crashes. The flying surfaces are all made from tough EPO foam which are durable and easy to repair. It has a 1.6 m. wingspan and a total length from nose to tail of 1.044m, for the propulsion systems it has a 2812 1400Kv brushless motor and the actuators for the control surfaces are 4 9g standard servos. The total weight is 950g and the battery used with this airframe is a 4000mAh Li-Po Battery. Under the canopy it has a more than enough space for mounting batteries and all the external electronics, such as autopilot and on-board computer.

3.2.3 Penguin

The final airframe used as a development platform is a Finwing Penguin. The Penguin is a flying aircraft specially designed for First Person View(FPV) flights, this means it can carry the necessary payload for the autopilot and external sensors needed. It is made of EPO with wood reinforcement in the fuselage and carbon fiber tube reinforcement in the wings. The model has a wingspan of 1.72m and measuring 1.230m from nose to tail, the total wing area is $36dm^2$. This model is powered by a brushless motor M2815 driving a 15" $\times$ 8" propeller drived by an electronic speed controller (ESC) of 60 Amps. The control surfaces aileron and elevator are drived by a 17g servo and the rudder used a 9g servo. The battery used with this airframe is a Li-Po 14.8V with 4 cells and 5000mAh. The total weight of the airframe is 0.98kg including motors and servos but not including battery and extra equipment. This airframe is capable of carrying a heavy payload of max. 1000g, the payload has to include the battery and external electronics, such as the autopilot and navigation hardware.

3.3 Flight Controller Unit

There are a variety of flight controller units available which offer the ability to connect multiple sensors and process the information to control the aircraft. Among these there is a smaller selection that would be suitable for small-sized UAVs. In order to properly choose the adequate solution, existing units had to be evaluated. Existing units examined are separated into commercially made closed-source and open-source products. It should be noted that this is not a comprehensive study of autopilots systems but rather provides a general survey of what is available.

From the whole range of commercially available autopilot solutions, we choose to examine two types: closed-source and open-source. Closed-source commercial autopilots comparison include the Cloud Cap Piccolo II[10], MicroPilot MP2128g[56] and Lockheed Martin Kestrel Flight Systems Autopilot v2.4[52]; these closed-source autopilots have been in the market for many years and they also appear referenced in literature: Cloud Cap Piccolo II [73][45][17], MicroPilot MP2128g [11][51] and Kestrel [46]. The specifications for these autopilots are given in Table 3.1. The open-source autopilots examined includes the Paparazzi Lisa, 3D Robotics APM 2.6, and Pixhawk Autopilot; we can also find these open-source autopilots in literature: Paparazzi Lisa, 3D Robotics APM 2.6 and Pixhawk The specifications for these units are given in Table 3.2.

After the comparison of the autopilots available we choose for the main controller of the UAV the Pixhawk autopilot, which is a high-performance autopilot-on-module system. This autopilot-on-module offers a complete open source flight stack, which can be fully reviewed and modified to fulfil our needs. It is suitable for fixed wing, multi rotors, helicopters and any other robotic platform. It has a wide range target which goes from the high-end researcher to the amateur enthusiast. In the following subsections we will review in detail the hardware and firmware

3.3.1 Hardware

The heart of the autopilot board is the 32bit STM32F427 Cortex M4 core which runs at 168Mhz with 256 KB in RAM and 2MB of flash memory with the addition of a STM32F103 for a fail-safe

Autopilot	Cloud Cap Piccolo II	MicroPilot MP2128g	Kestrel Autopilot v2.4
Sensor			
Inertial sensor	3-axis, $\pm$10 g accelerometer 3-axis $\pm$300 deg/s gyroscope	3-axis, $\pm$5 g accelerometer 3-axis gyroscope	3-axis, $\pm$10 g accelerometer 3-axis, $\pm$300 deg/s gyroscope
Magnetometer	Add-on supported	Add-on supported	2-axis and 3-axis
Altimeter (barometric)	30 cm resolution	30 cm resolution	25 cm resolution
Airspeed (pitot tube)	up to 290 km/h	up to 480 km/h	0-210 km/h
GPS	4 Hz	4 Hz	4 Hz
Digital I/O	16	8	12
Analog inputs	4x 10 Bit	32x 24 bit at 5Hz	3x 12 bit
Other inputs	CAN bus	-	4-8 PWM signals, 4 Serial port (STD, SPI, I2C)
Data handling			
Sampling rate	20 Hz	5-30 Hz	100 Hz
Local output	LPT	Serial	Serial
Storage	-	1.5 MB on-board	512 kb on-board
RF link	40 km	4.8 km	25 km
Estimated cost	$20,000+	$6,000+	$2,500+

Table 3.1: Closed-source commercially available autopilot comparison

Autopilot	Cloud Cap Piccolo II	MicroPilot MP2128g	Kestrel Autopilot v2.4
Sensor			
Inertial sensor	3-axis, $\pm$10 g accelerometer 3-axis $\pm$300 deg/s gyroscope	3-axis, $\pm$5 g accelerometer 3-axis gyroscope	3-axis, $\pm$10 g accelerometer 3-axis, $\pm$300 deg/s gyroscope
Magnetometer	Add-on supported	Add-on supported	2-axis and 3-axis
Altimeter (barometric)	30 cm resolution	30 cm resolution	25 cm resolution
Airspeed (pitot tube)	up to 290 km/h	up to 480 km/h	0-210 km/h
GPS	4 Hz	4 Hz	4 Hz
Digital I/O	16	8	12
Analog inputs	4x 10 Bit	32x 24 bit at 5Hz	3x 12 bit
Other inputs	CAN bus	-	4-8 PWM signals, 4 Serial port (STD, SPI, I2C)
Data handling			
Sampling rate	20 Hz	5-30 Hz	100 Hz
Local output	LPT	Serial	Serial
Storage	-	1.5 MB on-board	512 kb on-board
RF link	40 km	4.8 km	25 km
Estimated cost	$20,000+	$6,000+	$2,500+

Table 3.2: Closed-source commercially available autopilot comparison

system. The Pixhawk autopilot posses a variety of interfaces which make the system expandable for additional sensors.

Some of the communication interfaces are listed below:

- 5x UART (serial ports), one high-power capable, 2x with HW flow control.
- 2x CAN (one with internal 3.3V transceiver, one on expansion connector).
- PPM sum signal input.
- I2C.
- SPI.
- 3.3 and 6.6V ADC inputs.
- Internal microUSB port and external microUSB port extension.

As for the integrated sensors we have a ST Micro L3GD20H 16 bit gyroscope, a ST Micro LSM303D 14 bit accelerometer/magnetometer, a Invensense MPU 6000 3-axis accelerometer/gyroscope and a MEAS MS5611 barometer. Using the software provided by the PX4 Flight Stack we can use an implementation of an External Kalman Filter (EKF) to have an estimation of the orientation and angular velocity of the aircraft. There are other sensors that can be added to extend the capabilities of the Pixhawk autopilot such as optical sensor[14] to calculate the optical flow at 400 HZ, a 3DR uBlox LEA-6H GPS Receiver with a 5 Hz update rate, a LIDAR-lite Rangefinder for precision altitude up to 40-meter range with 1cm resolution, to mention a few.

3.3.2 Firmware

The firmware for the Pixhawk autopilot modules runs on top of the very efficient small operating system called NuttX, which provides a POSIX-style environment for c++ programming (i.e. printf(), pthreads, /dev/ttyS1, open(), write(), poll(), ioctl(), etc).

The PX4 middleware runs on top of the operating system and provides device drivers and a micro object request broker (uORB) for asynchronous communication between the individual

tasks running on the autopilot. The PX4 flight control stack is a custom, BSD licensed flight control stack, providing fully autonomous waypoint flight for multicopter and fixed wing aircraft. It uses a common codebase and common flight management code. It follows a very flexible and structured approach, which allows to run plane and multicopter controllers with the same waypoint and safety state machine handling.

The PX4 Flight Stack has implemented flight modes for different levels of autonomy in the aircraft and are listed below:

- MANUAL: The pilot's control inputs are passed directly to the output mixer.
- ALTCTL: When the roll, pitch and yaw inputs (RPY) are all centered (less than some specified deadband range) the aircraft will return to straight and level flight and keep its current altitude. It will drift with the wind.
- POSCTL: Neutral inputs give level, flight and it will crab against the wind if needed to maintain a straight line.
- AUTO_LOITER: The aircraft loiters around the current position at the current altitude (or possibly slightly above the current altitude).
- AUTO_MISSION: The aircraft obeys the programmed mission sent by the ground control station (GCS). If no mission received, aircraft will LOITER at current position instead.
- OFFBOARD: In this mode the position, velocity or attitude reference / target / setpoint are provided by a companion computer connected via serial cable and MAVLink. The offboard setpoint can be provided by APIs like MAVROS.

The experimental platform including the Pixhawk autopilot and the external sensors can be depicted in Figure 3.3.2.

Experimental platform with the Pixhawk autopilot and used electronics.

3.4 Examples Usage of the MAV3DSim

The literature presents many examples where a simplified model of the system to investigate a control algorithm[37][34][30]. This is a common practice that simplifies the process of control design. However, in a lot of cases the controller must be validated on a real platform, which does not necessarily match the model. In few cases, the designer tests the controller on the complete system model, this is due to the difficulty to represent the behavior in any simulation software such as MATLAB SIMULINK.

Therefore, a simulating tool which can represent in an accurate manner the real system behavior is needed. In order to validate the simulation platform we are using the stable version of the PX4 flight stack firmware[15], which is the firmware installed in the Pixhawk autopilot. To this end, the operation of the MNAV3DSim simulator is showed by two examples: We use the implementation of the waypoint following of the PX4 stack firmware. We use the MAV3DSim as a ground station when performing the same waypoint following on the experimental platform.

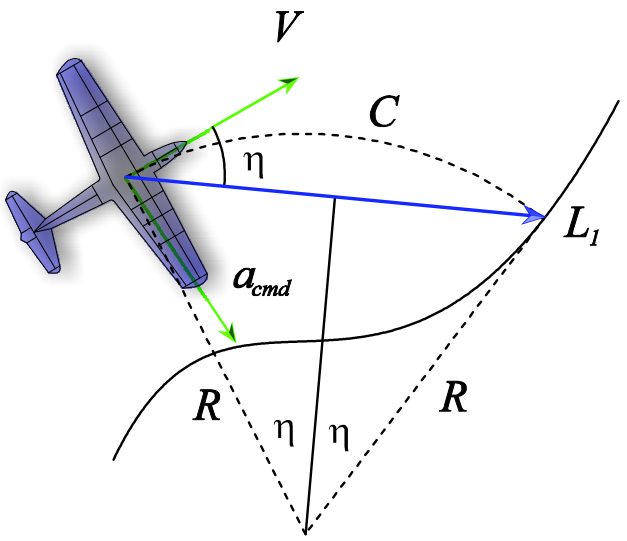


Figure 3-2: L_1 guidance geometry.

3.4.1 L_1 Controller for Waypoint Following

This section describes the guidance law implemented on the PX4 Firmware for waypoint following, this guidance is fully presented and extensively analyzed in [65][63]. We will briefly describe the guidance law and then we will continue with the implementation of the waypoints following in the PX4 Flight Stack.

The key idea behind this guidance law is that it generates a lateral acceleration commands to steer the velocity vector towards a desired point, chosen to be a specified look-ahead distance along the desired path in front of the vehicle.

Analyzing the geometry shown on Figure 3-2, V is the UAV's horizontal velocity vector w.r.t the ground and C is the circular arc of radius R that lies tangent to the velocity vector and connects with the intersection with the desired L_1 vector's distance and the path, and the L_1 being a constant look-ahead distance vector from the UAV position to the path in the desired direction of travel. This L_1 vector is then divided by two into equal segments by the line bisecting the chord of the arc C. Then by trigonometry we know that:

$$\frac{|L_1|}{2} = R \sin \eta \tag{3.6}$$

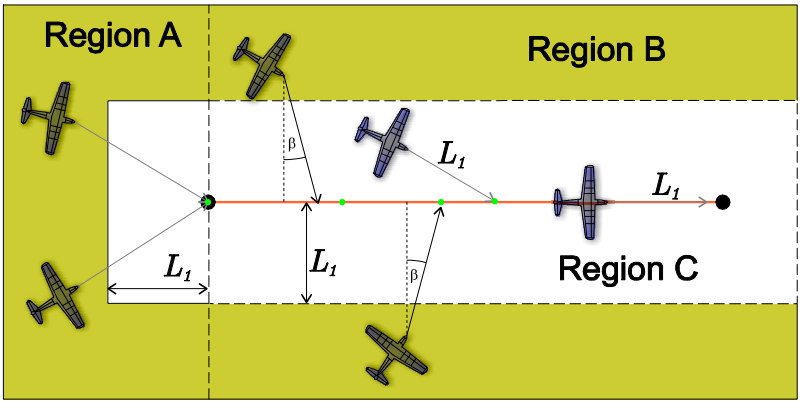


Figure 3-3: L_1 guidance regions accordingly to the position of the aircraft.

So the lateral acceleration a_c required to follow the circular path C is given by:

$$a_c = \frac{|V|^2}{R} \tag{3.7}$$

Therefore the lateral acceleration command is determined by:

$$a_c = 2\frac{|V|^2}{|L_1|}\sin\eta \tag{3.8}$$

It is clear that the implementation of this control law only requires the selection of the $|L_1|$ distance and to determine $\sin\eta$, where η is the angle from the velocity vector V to L_1 also referred as the line of sight angle.

For the UAV to actually track the desired trajectory, the lateral acceleration command a_c computed in 3.8 must be converted to a bank angle command ϕ_{cmd} using the following turn equation:

$$\phi_{cmd} = \tan^{-1}\left(\frac{a_c}{g}\right) \tag{3.9}$$

Now we discuss the implementation of the PX4 Stack Flight of the waypoint following strategy. This strategy consists in two parts, 1) the computation of a reference point L_{1ref} and 2) the computation of the acceleration command from eq. 3.8.

The computation of L_{1ref} depends primarily of the current location of the aircraft w.r.t. the

path to follow. The path to follow is defined by a n number of waypoints that are sufficiently away from each other to ensure the aircraft is able to make a successful turn. If two waypoints are too close from each other they will be treated as a single waypoint and after reaching it, the aircraft will continue with the next waypoint. There are 3 different sections in which the aircraft could be located (Figure 3-3):

- Region A: Is situated behind the waypoint, this usually occurs at the beginning of the mission and the reference point L_{1ref} will be the very first waypoint.
- Region B: Here the aircraft is in between of the two waypoints so the reference point L_{1ref} is placed over the path with an β angle, so it does not enter the path perpendicularly.
- Region C: In this region we can directly apply the computation of the L_{1ref} on the path at a L_1 distance ahead of the aircraft.

Using the map interface provided by the MAV3DSim we can choose the waypoints, select the proper altitude for the waypoints and finally save them into the Pixhawk's flash memory.

The experiment begins first with a manual control of the aircraft to perform a take-off and gain enough altitude to initiate the waypoint following logic, we can change at a semi-autonomous or assisted control using the radio control switches, after the aircraft gained gain enough altitude we activate the L_1 waypoint following with the radio control and let the aircraft follows each waypoint, we can at all time recover the manual control of the autopilot with the radio control. In this case the experiment is finished with a manual landing. A video of the working simulator can be seen in `https://youtu.be/swr097xFV2w`

3.4.2 MAV3DSim as a Ground Control Station

When performing tests on the flying field, it is important to know the current state of the vehicle. To that end, we could use the MAV3DSim as a Ground Control Station (GCS), as it communicates with the Pixhawk autopilot via the MAVLink protocol using a wireless link and it has all the avionics instruments and the 2D map display to know the position, orientation and velocity of the vehicle at all times.

We perform the very same experiment described in the previous section but in the experimental platform described in Section 3.2, using the very same waypoints and the same location

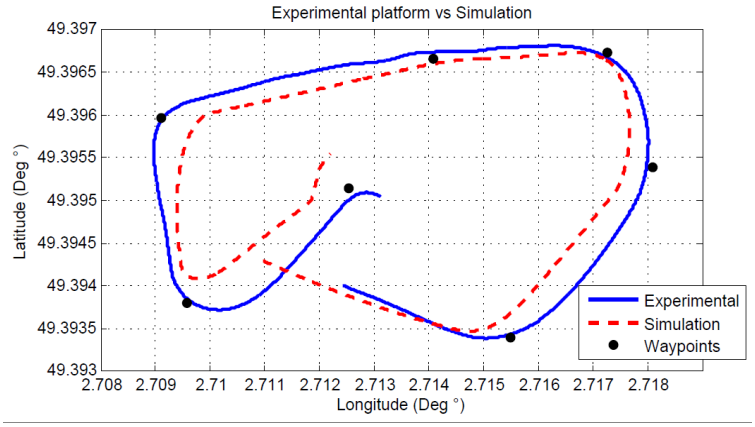


Figure 3-4: Experimental platform vs simulation following the same waypoints.

and as in the simulation case we start with the manual take-off and continue with the waypoint following to finish the mission with a manual landing. A video of the experiment on the field could be seen in `https://youtu.be/gxhkDxKoG0U`.

Figure 3-4 shows the path followed in the simulation and the experimental platform. In both cases we use the same waypoints and the same parameters and gains in the Pixhawk autopilot. In Figure 3-5 the altitude of the simulator and the experimental platform are compared with the setpoint designated for the different altitudes of the waypoints.

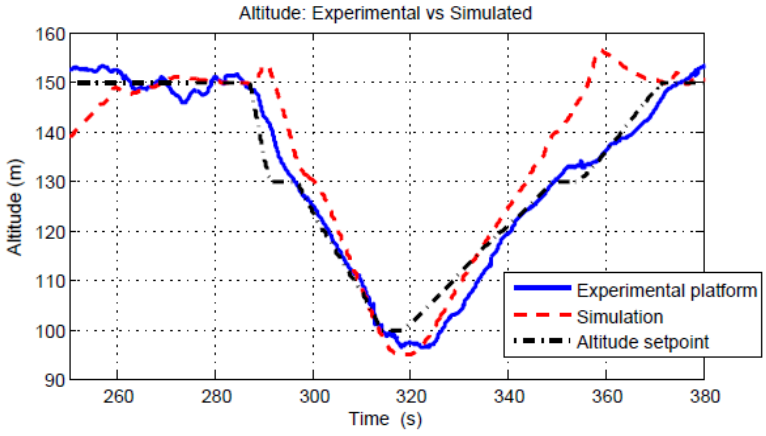


Figure 3-5: Altitude on the simulation platform and the experimental platform.

Chapter 4

Path Generation and Control in 2D

In the case of path following control scenario, we must certainly decide the path to be followed before the guidance logic uses it to generate the error and perform the corresponding control logics. Usually the first step is to introduce several fixed points in space, i.e. waypoints, and define the desired path as the sum of straight lines that connect this waypoints. This approach, simple as it is, sometimes will not fulfill the demand of accuracy of the application because the resulting path is no smooth enough. On the literature we can find a great variety of examples on this issue, as well as other important factors that arise, regarding the performance in each case. For some missions it is crucial to pass through the way point, for other mission it is important that the vehicle converges and stays on the path, others are more concerned on finding the minimum path, and so on.

Dubins[44] showed that a car-like robot with initial prescribed heading can arrive to its final position and heading, with exactly three paths segments which are either arcs of circles with a minimal radius or straight lines segments. Reeds and Sheep [35] solve a similar problem in which the vehicle can move forward as well as backward. Kavaraki and Svestka[41] use the Probabilistic Road Map (PRM) method which explore all the possible paths within the space surrounding the vehicle and finally select the lowest cost route. Other planning techniques used by Kuwata and Richards[42] are based on optimizations methods, such as Mixed Integer Linear Programming or Model Predictive Control techniques. Mehta and Egerstedt[54] used optimal control for constructing control programs from a given collection of motion primitives.

In this section we present a path generator for a fixed-wing UAV using a reduced kin-

ematic version of the lateral dynamics of an airplane, with constant altitude and velocity. This path generator uses the Dubins paths to generate the new path from the current position and direction of the plane to the desired position and direction.

4.1 Mathematical model

The Dubins aircraft model is described by the subsequent equations:

$$\dot{x} = V_t \cos\psi \tag{4.1}$$

$$\dot{y} = V_t \sin\psi \tag{4.2}$$

$$\dot{\psi} = \omega$$

in which x and y denotes the inertial position of the aircraft, ψ is the heading angle, ω is the heading rate, ϕ is the roll angle, V_t is the airspeed, i.e. the speed of an aircraft relative to the surrounding air. The velocity of the aircraft is held constant by a velocity-hold system. Since the simplified model is a function of x, y, ϕ also an altitude-hold controller is needed for the simulation performed on the simulation platform MAV3DSim described in the previous chapter. Also, we assume no sideslip at a banked-turn maneuver.

The heading rate ω is induced by the roll angle of the airplane as

$$\omega = \frac{g}{V_t} \tan\phi \tag{4.3}$$

where g is the gravity acceleration. The roll angle is considered bounded under the following condition

$$|\phi| \leq \phi_{\max} \tag{4.4}$$

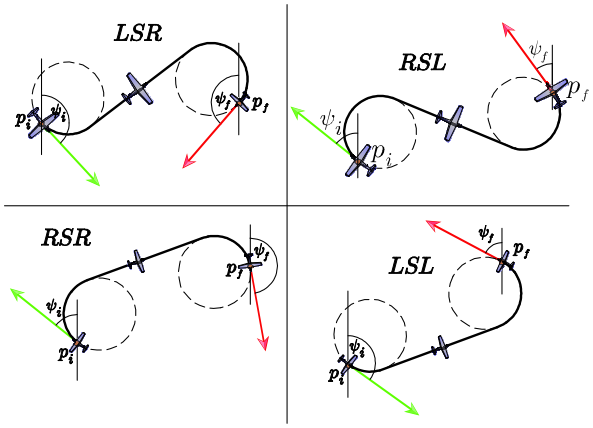
Assuming a coordinated turn, and given the boundedness of the roll angle ϕ the minimum turn radius ρ that the aircraft can fly is given by

$$\rho = \frac{V_t^2}{g \tan(\phi_{\max})} \tag{4.5}$$

In this kinematic model the position of the airplane can be represented by $p(x, y, \psi)$ with ψ

measured from the y axis and (x, y) measured in the local ENU reference frame.

4.2 Path Generation in 2D



In this section, the classical result of Dubins[44] is used as a basis for path generation. Dubins showed that the shortest path consist of exactly three path segments which are either a) arcs of a minimal radius or b) straight lines. The four different configurations for the Dubins paths which are composed by two curved segments and a straight line segment are arranged as shown in Figure 4.2. The four cases of Dubins paths are LSL, LSR, RSR, RSL; in which L stands for Left, R stands for Right and S for Straight.

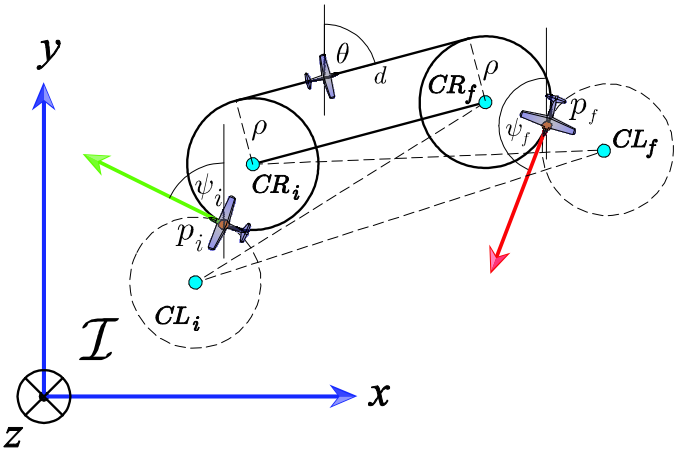
The first step in determining the Dubins paths is to choose what type of path must be used. We have the initial and final configuration of the airplane, this is the initial position p_i, the initial heading ψ_i the final position p_f and the final heading ψ_f and with every initial-final configuration we can generate the 4 types of Dubins paths,i.e. from the starting point it can turn to the right or the left and arrive to the final point from the right or the left. We choose the shortest path by comparing the distance between the center of the circles, see Figure 4.2. The smallest distance between the center of the circles gives us the shortest Dubin path according to the Table 4.1.

Based on the initial and final configuration $(p_i, \psi_i)$and(p_f, ψ_f), respectively, and the minimal turn radius ρ from (4.5), the center of each circle is computed as follows

Table 4.1: Dubins path selection.

Shortest distance	**Dubins Path**
CR_iCR_f	RSR
CR_iCL_f	RSL
CL_iCL_f	LSL
CL_iCR_f	LSR

$$
\begin{aligned}
CR_i &= (x_{Ri}, y_{Ri}) = (x_i + \rho\cos\psi_i, y_i - \rho\sin\psi_i) \\
CL_i &= (x_{Li}, y_{Li}) = (x_i - \rho\cos\psi_i, y_i + \rho\sin\psi_i) \\
CR_f &= (x_{Rf}, y_{Rf}) = \big(x_f + \rho\cos\psi_f, y_f - \rho\sin\psi_f\big) \\
CL_f &= (x_{Lf}, y_{Lf}) = \big(x_f - \rho\cos\psi_f, y_f + \rho\sin\psi_f\big)
\end{aligned}
$$

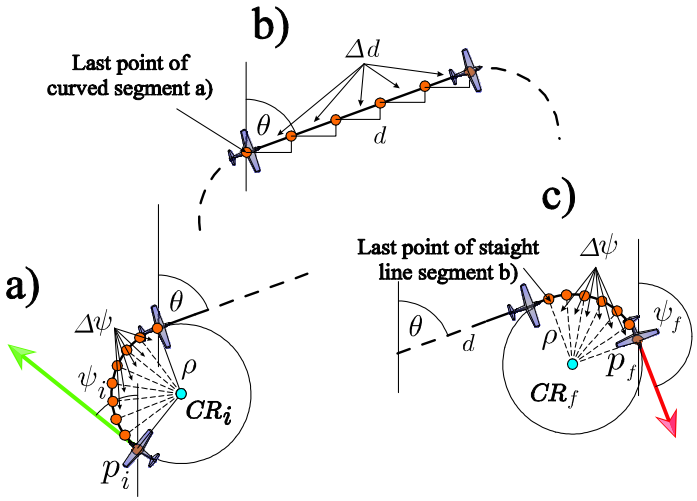


The Dubins paths are chosen by comparing the distance between the center of the circles segments.

4.2.1 Dubins path RSR

The initial and final configuration (p_i, ψ_i) and (p_f, ψ_f),respectively, are given w.r.t. an inertial frame (Local ENU frame). The RSR is generated by a clockwise rotation from the initial position describing an arc of radius ρ and center CR_i with coordinates (x_{Ri}, y_{Ri}) until the

aircraft heading achieves an angle of θ degrees. Then it follows a straight line segment d, finally it continues with a turn to the right describing an arc of radius ρ and center in CR_f with coordinates (x_{Rf}, y_{Rf}) until the plane arrives to the final heading ψ_f as seen in Figue 4.2.



The path generator algorithm produce an array of points p_n

The angle θ is the angle of the straight line segment d which is measured from the vertical y axis and computed as follows

$$\theta = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Rf} - y_{Ri}}{x_{Rf} - x_{Ri}}\right) \tag{4.6}$$

The length $\bar{d}$ of the straight line segment d equals the distance $\overline{CR_iCR_f}$ between the center of the circles CR_i and CR_f and is computed as

$$d = \sqrt{(x_{Rf} - x_{Ri})^2 + (y_{Rf} - y_{Ri})^2} \tag{4.7}$$

The path generator algorithm produce an array of n points p_n which starts in $p_0 = p_i$ and ends in $p_n = p_f$.

The coordinates of the $n - th$ point p_n of the arc segments are obtained by rotating the initial point p_i clockwise around CR_i as a center

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Ri} + \rho \sin(\psi_n) \\ y_{Ri} + \rho \cos(\psi_n) \end{bmatrix} \tag{4.8}$$

where ψ_n starts at ψ_i and is incremented by given $\Delta\psi$ each time. These procedure is repeated until $\psi_n = \theta$, see Figure 4.2.1a.

Each point in the straight line segment is computed by incrementing the previous point p_{n-1} in a given Δd in direction of the angle θ as follows

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{n-1} + \Delta d \sin(\theta) \\ y_{n-1} + \Delta d \cos(\theta) \end{bmatrix} \tag{4.9}$$

The elements p_n of the final segment are computed by rotating the final point of the straight line clockwise around CR_f as a center; see Figure 4.2.1c.

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Rf} + \rho \sin(\psi_n) \\ y_{Rf} + \rho \cos(\psi_n) \end{bmatrix} \tag{4.10}$$

where ψ_n starts in θ and each time is incremented by $\Delta\psi$. This procedure is repeated until $\psi_n = \psi_f$; see Figure 4.2.1c.

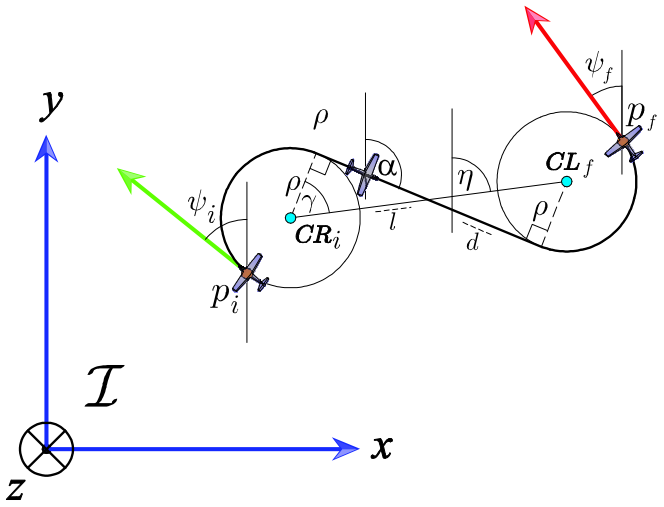
The complete path generation is summarized in algorithm 1

Algorithm 1 Generate Dubin path RSR

$n = 1$; $p_0 = p_i$
$\psi_n = 0$
$\theta = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Rf} - y_{Ri}}{x_{Rf} - x_{Ri}}\right)$
while $\psi_n \leq \theta$ **do**
  $p_n.x = x_{Ri} + \rho \sin(\psi_n)$; $p_n.y = y_{Ri} + \rho \cos(\psi_n)$
  $\psi_n = \psi_n + \Delta\psi$; $n = n + 1$
end while
$d_{sum} = 0$
while $d_{sum} \leq \bar{d}$ **do**
  $p_n.x = p_{n-1}.x + \Delta d \sin\theta$; $p_n.y = p_{n-1}.y + \Delta d \cos(\theta)$
  $d_{sum} = d_{sum} + \Delta d$; $n = n + 1$
end while
while $\psi_n \leq \psi_f$ **do**
  $p_n.x = x_{Ri} + \rho \sin(\psi_n)$; $p_n.y = y_{Ri} + \rho \cos(\psi_n)$
  $\psi_n = \psi_n + \Delta\psi$; $n = n + 1$
end while

4.2.2 Dubins path RSL

This is the case where the closest circles are CR_i and CL_f, see Table 4.1. From the initial and final configuration, (p_i, ψ_i) (p_f, ψ_f) the RSL path is generated with a clockwise rotation from the initial position p_i describing an arc of circle of radius ρ with center CR_i with coordinates (x_{Ri}, y_{Ri}) until the heading aircraft achieves the angle θ. Then it follows a straight line segment d, finally it will turn to the left describing an arc of radius ρ and center in CL_f with coordinates (x_{Lf}, y_{Lf}) until the aircraft reaches the final heading. See Figure 4.2.2.



Right-Straight-Left (RSL) Dubins path.

In this case the angle θ is computed is computed aided by the triangle formed by the center of the circle CR_i the midpoint of the segment d and the point of the circle tangent to the straight line d using the following formula

$$\theta = \eta - \gamma + \frac{\pi}{2} \tag{4.11}$$

where η is the angle of the segment $\overline{CR_iCL_i}$ measured from the y axis as in Figure 4.2.3 and is computed as follows

$$\eta = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Lf} - y_{Ri}}{x_{Lf} - x_{Ri}}\right) \tag{4.12}$$

γ is the angle between the segment $\overline{CR_iCL_f}$ and the normal to the tangent point of circle CR_i and the segment d. γ is computed as follows

$$\gamma = \tan^{-1}\left(\frac{2\rho}{d}\right) \tag{4.13}$$

The length of the straight line segment d is computed with the distance l from the segment $\overline{CR_iCL_f}$ and the radius ρ as

$$d = \sqrt{l^2 - 4\rho^2} \tag{4.14}$$

The coordinates of the $n-th$ point p_n of the arc segments are obtained by rotating the initial point p_i clockwise around CR_i as a center using (4.8), see Figure 4.2.1a.

Each point in the straight line segment is computed by incrementing the previous point p_{n-1} in a given Δd in direction of the angle θ as in (4.9)

Algorithm 2 Generate Dubin path RSL

$n = 1$; $p_0 = p_i$
$\psi_n = 0$
$\eta = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Lf}-y_{Ri}}{x_{Lf}-x_{Ri}}\right)$
$\gamma = \tan^{-1}\left(\frac{2\rho}{d}\right)$
$\theta = \eta - \gamma + \frac{\pi}{2}$
while $\psi_n \leq \theta$ **do**
  $p_n.x = x_{Ri} + \rho\sin(\psi_n)$; $p_n.y = y_{Li} + \rho\cos(\psi_n)$
  $\psi_n = \psi_n + \Delta\psi$; $n = n + 1$
end while
$d_{sum} = 0$
while $d_{sum} \leq \bar{d}$ **do**
  $p_n.x = p_{n-1}.x + \Delta d\sin\theta$; $p_n.y = p_{n-1}.y + \Delta d\cos(\theta)$
  $d_{sum} = d_{sum} + \Delta d$; $n = n + 1$
end while
while $\psi_n \leq \psi_f$ **do**
  $p_n.x = x_{Li} + \rho\sin(\psi_n)$; $p_n.y = y_{Li} + \rho\cos(\psi_n)$
  $\psi_n = \psi_n + \Delta\psi$; $n = n + 1$
end while

The elements p_n of the final segment are computed by rotating the final point of the straight line clockwise around CL_f as a center; see Figure 4.2.1c.

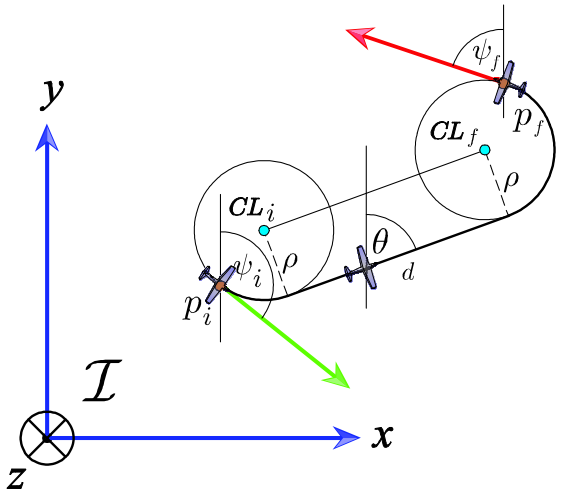
$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Lf} + \rho\sin(\psi_n) \\ y_{Lf} + \rho\cos(\psi_n) \end{bmatrix}$$

where ψ_n starts in θ and each time is incremented by $\Delta\psi$. This procedure is repeated until $\psi_n = \psi_f$; see Figure 4.2.1c.

The complete path generation is summarized in algorithm 3

4.2.3 Dubins path LSL

The LSL case is very similar to the RSR but with the turns to the left instead of right and it occurs when the smallest distance between the circles (see Figure 4.2) is $\overline{CLiCLf}$. The LSL path is generated with a counterclockwise rotation from the initial position p_i describing an arc of a circle of radius ρ and center in CL_i with coordinates (x_{Li}, y_{Li}) until the aircraft heading achieves an angle of θ degrees. Then it follows a straight line segment d and finally it continues with the a turn to the left describing an arc of radius ρ and center in CL_f with coordinates (x_{Lf}, y_{Lf}) until the airplane achieves the final heading ψ_f, as depicted in Figure 4.2.3.



Left-Straight-Left (LSL) Dubins path.

The angle θ measured from the vertical y axis is

$$\theta = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Lf} - y_{Li}}{x_{Lf} - x_{Li}}\right) \tag{4.15}$$

The length of the segment d equals the distance $\overline{CL_iCL_f}$ and it is computed as

$$d = \sqrt{(x_{Lf} - x_{Li})^2 + (y_{Lf} - y_{Li})^2} \tag{4.16}$$

The coordinates of the $n-th$ point p_n of the arc segments are obtained by rotating the initial point p_i counterclockwise around the CL_i as a center, as follows

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Li} + \rho \sin(\psi_n) \\ y_{Li} + \rho \cos(\psi_n) \end{bmatrix} \tag{4.17}$$

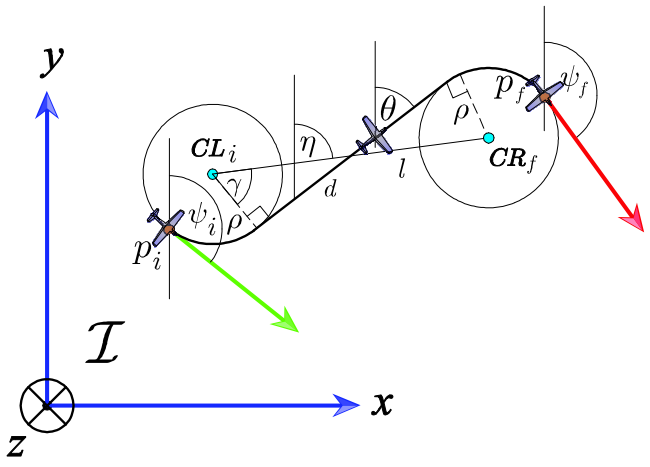
where ψ_n starts at zero and is incremented each time by $\Delta\psi$ until it reach the angle θ.

Each point in the straight line segment is computed by incrementing the previous point p_{n-1} in Δd in the same direction as θ using equation (4.9). The last curved segment is a turn to the left and the segment coordinates are computed as follows

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Lf} + \rho \sin(\psi_n) \\ y_{Lf} + \rho \cos(\psi_n) \end{bmatrix} \tag{4.18}$$

4.2.4 Dubins path LSR

According to Table 4.1 the Dubins path LSR is when the shortest distance is the one between the circles CL_i and CR_f. The first segment of this path is a left turn which generated with a counter-clockwise rotation from the initial position p_i describing an arc of radius ρ with center in $CL_i = (x_{Li}, y_{Li})$ until the airplane reach the heading θ, then it follows a straight line segment of length d and it finish with a right turn described by the arc of the circle of radius ρ with center in $CR_f = (x_{Rf}, y_{Rf})$ and it will turn until it achieve the angle ψ_f as depicted in Figure 4.2.4.



Left-Straight-Left (LSR) Dubins path.

The computation of the angle θ is carried out by the triangle formed by the center of the circle CL_i the midpoint of the segment d and the point of the circle tangent to the segment d using the following equation

$$\theta = \eta + \gamma - \frac{\pi}{2} \tag{4.19}$$

where

$$\eta = \frac{\pi}{2} + \tan^{-1}\left(\frac{y_{Rf} - y_{Li}}{x_{Rf} - x_{Li}}\right)$$

and

$$\gamma = \cos^{-1}\left(\frac{2\rho}{d}\right)$$

The length of the straight line segment d is computed with the following equation

$$d = \sqrt{l^2 - 4\rho^2} \tag{4.20}$$

The coordinates of the $n - th$ point p_n of the arc segments are obtained by rotating the initial point p_i counterclockwise around the CL_i as a center, as follows

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Li} + \rho \sin(\psi_n) \\ y_{Li} + \rho \cos(\psi_n) \end{bmatrix} \tag{4.21}$$

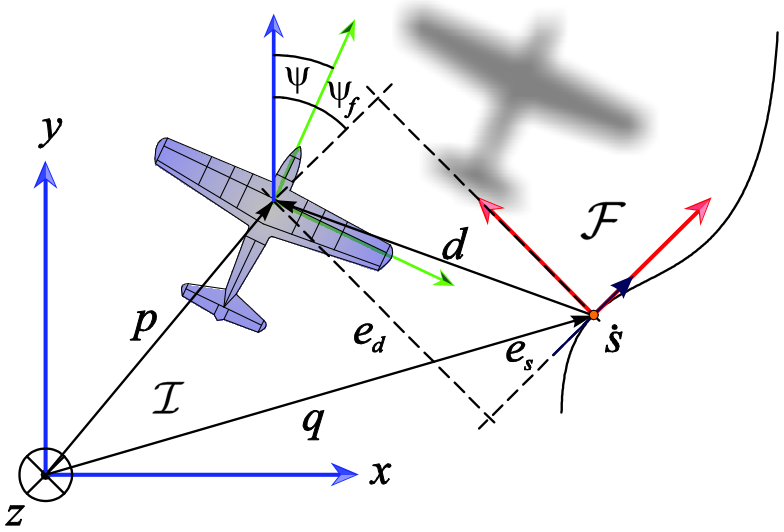
where ψ_n starts at zero and is incremented each time by $\Delta\psi$ until it reach the angle θ.

Each point in the straight line segment is computed by incrementing the previous point p_{n-1} in Δd in the same direction as θ using equation (4.9). The last curved segment is a turn to the right and the segment coordinates are computed as follows

$$p_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_{Rf} + \rho \sin(\psi_n) \\ y_{Rf} + \rho \cos(\psi_n) \end{bmatrix} \tag{4.22}$$

4.3 Problem Statement

In this section, the problem statement is introduced and a dynamic system suitable for control purposes is formulated.



Path following control problem schema.

Considering Figure 4.3, the key idea behind the path-following controller relies on reducing two expressions to zero: the first one is the distance between the aircraft's center of mass p and the the point q on the path, the second one is the angle between the airspeed vector and the tangent to the path at q.

To accomplish these objectives, we introduce a virtual particle moving along the geometric path at a velocity $\dot{s}$. Consider a frame attached to such particle, this frame plays the role of a body axis of the virtual particle, and is the so called Serret-Frenet frame denoted by $\mathcal{F}$ [55]. It is worth noting that the particle velocity evolves according to a conveniently defined control law $\dot{s}$, yielding an extra controller design parameter.

With this set-up in mind, the aforementioned angle and distance will become the coordinates of the error space, where the control problem is stated and solved.

4.3.1 Error dynamics for the path-following controller

Consider that the 2-D geometric path is represented by smooth functions parameterized by t, i.e. $x_s(t)$ and $y_s(t)$. Thus, $(x_s(t), y_s(t))$ represent the virtual particle coordinates.

The inertial position of the aircraft is defined by $p = [x \;\; y]^T$ in the inertial reference frame $\mathcal{I}$. For the purpose of following the given path, we define the inertial vector error $d^I = p - q(s)$ expressed in $\mathcal{F}$, which will be minimized in order to track the path. Such error vector d^I has been decomposed into its components e_s and e_d, corresponding to the error in the x-axis of the frame $\mathcal{F}$ and the error in the y-axis of the frame $\mathcal{F}$, respectively as it is shown in Figure 4.3.

From the Figure 4.3, we can see that the tangent vector to the path at $q(s)$ is parallel to x-axis of the frame $\mathcal{F}$. The angle ψ_f is measured from the inertial frame to the tangent vector of $q(s)$.

Considering an arbitrary point q on the path, and let

$$R = \begin{pmatrix} \cos(\psi_f) & -\sin(\psi_f) \\ \sin(\psi_f) & \cos(\psi_f) \end{pmatrix} \tag{4.23}$$

the rotation matrix from $\mathcal{F}$ to $\mathcal{I}$, parameterized locally by ψ_f. Thus, the error d^I expressed in the Serret-Frenet frame is given by

$$d^{\mathcal{F}} = \begin{bmatrix} es \\ ed \end{bmatrix} = R^T d^{\mathcal{I}} = R^T (p - q(s)) \tag{4.24}$$

Furthermore, we define the yaw angle error as

$$\tilde{\psi} = \psi - \psi_f \tag{4.25}$$

The angle ψ_f can be computed by using the information provided by the geometric path and its first derivative with respect to the parameter t, as follows

$$\psi_f = \arctan \frac{y'_s}{x'_s} \tag{4.26}$$

where $x'_s = \frac{dx_s}{dt}$, $y'_s = \frac{dy_s}{dt}$.

To obtain the error state dynamic equations suitable for control purposes, we must compute the time derivative of (4.24) and (4.25). By differentiating (4.24), it follows that

$$\begin{aligned} \dot{d}^{SF} &= R^T\left(\dot{p} - \dot{q}\left(s\right)\right) + \dot{R}^T\left(p - q\left(s\right)\right) \\ &= R^T\left(\dot{p} - \dot{q}\left(s\right)\right) + S\left(\dot{\psi}_f\right) R^T\left(p - q\left(s\right)\right) \end{aligned} \tag{4.27}$$

where $S(\dot{\psi})$ is given by

$$S(\psi) = \begin{pmatrix} 0 & -\dot{\psi}_f \\ \dot{\psi}_f & 0 \end{pmatrix} \tag{4.28}$$

From (4.1), the time derivative of p and $q(s)$ can be represented as follows

$$\dot{p} = R(\psi) \begin{pmatrix} V \\ 0 \end{pmatrix} \tag{4.29}$$

$$\dot{q} = R \begin{pmatrix} \dot{s} \\ 0 \end{pmatrix} \tag{4.30}$$

The time derivative of (4.25) results in

$$\dot{\tilde{\psi}} = \omega - \dot{\psi}_f \tag{4.31}$$

with

$$\dot{\psi}_f = k(s)\dot{s} \tag{4.32}$$

where $\frac{d\psi_f}{dt} = k(s)$ is the path curvature. The path curvature is expressed as a function of the path coordinates $(x_s(t), y_s(t))$ and its first and second derivatives with respect to the parameter t, i.e. $x'_s = \frac{dx_s}{dt}$, $y'_s = \frac{dy_s}{dt}$. Thus, the path curvature $\frac{d\psi_f}{dt} = k(s)$ is given by

$$k = \frac{|y''_s x'_s - y'_s x''_s|}{({x'_s}^2 + {y'_s}^2)^{3/2}} \tag{4.33}$$

Finally, by substituting (4.29) and (4.30) in (4.27) and using (4.31) we obtain the error kinematic model suitable for the control purposes as

$$\begin{aligned} \dot{e}_s &= V_t \cos\tilde{\psi} - (1 - k(s)\, e_d)\, \dot{s} \\ e_d &= V_t \sin\tilde{\psi} - k(s)\, e_s \dot{s} \\ \psi &= \omega - k(s)\, \dot{s} \end{aligned} \tag{4.34}$$

4.4 Path following controller

In this section we present a nonlinear path following control strategy. Such control strategy is done in two steps. The first step yields a kinematic controller by adopting the yaw rate ω from 4.1 as a virtual control input. The second step addresses the vehicle dynamics in order to obtain the control law for the input variable ϕ. Such control law relies on the kinematic controller previously derived.

4.4.1 Kinematic Controller Design

Following a similar approach as in [43], we introduce a desired approach angle parameterized by $k_\delta > 0$ as

$$\delta(e_d) = -\psi_a \frac{e^{2k_\delta e_d} - 1}{e^{2k_\delta e_d} + 1} \tag{4.35}$$

where $0 < \psi_a < \pi/2$. The sigmoid function (4.35) is bounded and differentiable with respect to the error e_d. It provides the desired relative course transition of the fixed-wing MAV to the path as a function of e_d. Moreover, (4.35) satisfies the condition $e_d\delta(e_d) \leq 0\ \forall e_d$. Such condition guides the MAV to the correct direction, i.e., turn left when the MAV is on the right side of the path, and turn right in the opposite situation.

In order to study the control law for the system (4.1), we propose a Lyapunov function candidate given by

$$V(e_d, e_s, \tilde{\psi}) = \frac{1}{2}e_d^2 + \frac{1}{2}\left(\tilde{\psi} - \delta(e_d)\right)^2 + \frac{1}{2}e_s^2 \tag{4.36}$$

The time derivative of (4.36) along the trajectory of (4.1) is computed as follows

$$\dot{V}(e_d, e_s, \tilde{\psi}) = \left(\tilde{\psi} - \delta(e_d)\right)(\omega + \beta) + (e_d)\left(V_t \sin\left(\delta(e_d)\right)\right) + (e_s)\left(V_t \cos\tilde{\psi} - \dot{s}\right) \tag{4.37}$$

where

$$\beta = -\mathcal{C}_{\mathcal{C}}(s)\,\dot{s} - \dot{\delta}(e_d)\left(V_t \sin\tilde{\psi} - \mathcal{C}_{\mathcal{C}}(s)\,e_s\dot{s}\right) + (V_t e_d)\left(\frac{\sin\psi - \sin\left(\delta\left(e_d\right)\right)}{\tilde{\psi} - \delta\left(e_d\right)}\right)$$

where the derivative with respect to $e_d$of (4.35) is

$$\dot{\delta}(e_d) = -\frac{4\psi_a k_\delta e^{2k_\delta e_d}}{(e^{2k_\delta e_d} + 1)^2} \tag{4.38}$$

Substituting the following kinematic control law

$$\begin{aligned} \dot{s} &= V_t \cos\tilde{\psi} + k_s e_s \\ \omega &= -\beta - k_{\omega 1}\left(\tilde{\psi} - \delta\left(e_d\right)\right) \end{aligned} \tag{4.39}$$

where k_s, k_{ω_1} are positive real numbers, in (4.37), yields

$$\dot{V}\left(e_d, e_s, \tilde{\psi}\right) = -k_s e_s^2 - k_{\omega_1}\left(\tilde{\psi} - \delta\left(e_d\right)\right)^2 + V_t e_d\left(\sin\left(\delta\left(e_d\right)\right)\right) \leq 0$$

To conclude convergence of the states $(e_s, e_d, \tilde{\psi})$to zero, we state de LaSalle's Invariance Principle

Theorem 1 *LaSalle's* Theorem

Let $\mathcal{O}$ be a positively invariant set of system (16). Let $\Omega \subset \mathcal{O}$ a set in which every solution starting in $\mathcal{O}$ converges to Ω. Furthermore, let $\mathcal{M}$ be the largest invariant set contained in Ω. Then, as $t \to \infty$, every bounded solution starting in $\mathcal{O}$ converges to $\mathcal{M}$.

Proof. *Convergence of the states $(e_s, e_d, \tilde{\psi})$ to zero.*

The proof relies on Theorem 1. Consider the system (16) and the radially unbounded Lyapunov function candidate (4.36). Let us define the compact set $\mathcal{O}$ as $\mathcal{O} = \{V(e_d, e_s, \tilde{\psi}) \leq a\}$,

where $a \in \Re^+$. Define the set Ω as

$$\Omega = \{[e_d \;\; e_s \;\; \tilde{\psi}]^T \in \mathcal{O} \;\; : \;\; \dot{V}(e_d, e_s, \tilde{\psi}) = 0\} \tag{4.40}$$

Equivalently, the expression $\dot{V}(e_d, e_s, \tilde{\psi}) = 0$ means that $e_s = e_d = 0$ and $\tilde{\psi} = \delta$. Since δ is a function of the error e_d, it is easy to verify that any point starting from Ω is an invariant set. Hence, by LaSalle Theorem, every trajectory starting in $\mathcal{O}$ converges to 0 as $t \to \infty$, i.e. $\lim_{t\to\infty} e_s = 0$, $\lim_{t\to\infty} e_d = 0$ and therefore $\lim_{t\to\infty} \tilde{\psi} = \delta(e_d) = 0$. ■

4.5 Simulation Example

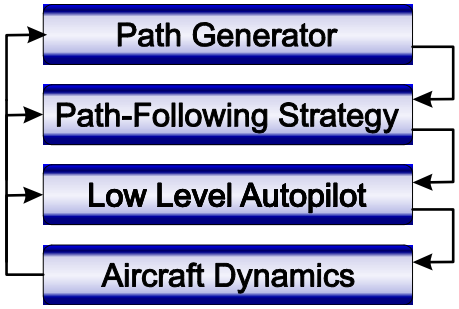
Simulations were done on a complete simulation platform, the MAV3DSim(Multi-Aerial Vehicle 3D Simulator) provides a complete 6 degrees of freedom (DoF) computer model of fixed wing aircraft. The MAV3DSim software layers are described briefly in this section. The application scenario is in the use of the path generation and path-following algorithms to command a desired path to the fixed wing UAV. The results from the simulation are presented at the end of this section.

4.5.1 MAV3DSim Simulation Platform

The MAV3DSim is a custom C# .Net based application and implements a complete 6DoF nonlinear model. It has a 3D representation to visualize the position and orientation of the plane, also, it has the capability to load maps directly from Google Maps servers and set the launching site on any location on Earth. The trajectory generated by the plane can be seen on the map, this map is the tangential plane to the Earth.

The data generated by the simulator is coded in the same manner as the common sensors,i.e. it send data emulating an inertial measurement unit(IMU) sending inertial gyroscope, accelerometer and magnetometer, a GPS radio in the latitude/longitude format, altitude and airspeed. It can receive commands to move the control surfaces aileron elevators, rudder, and the thrust of the fixed-wing UAV. The position provided by the simulator is in a standard geodetic WGS84 Latitude(λ), Longitude(τ) and Height(h), and we will use the transformation to the local tangent ENU described in Section 2.2.5.

The software layers, depicted in the Figure 4.5.1, are briefly described as follows



Communication scheme between the MAV3DSim and the CRRCSim.

Path Generator

This layer is in charge of the generation of paths using the Dubins path generation described in Section 4.2. It can generate new paths and maintain the old ones for later use. It is possible to interact online with the path generation and change the course of action of the aircraft in any time either by an autonomous action or by a human interaction. Once the path is fully generated, it is transmitted to the path-following strategy.

Path-Following Strategy

The path-following control described in section 4.4 is implemented in this layer. The path is stored in an array of n points of the form (x_m, y_m) starting with $m = 0$ then the path following strategy computes the errors $es, ed, \tilde{\psi}$ from (4.24) and (4.25), with this information it computes control input ω and $\dot{s}$ from (4.39). The control ω is a desired heading rate and is induced into the aircraft dynamics through the roll angle using (4.3). The computed roll angle ϕ will be used by the low level autopilot

Low Level Autopilot

The role of low-level autopilot is to stabilize the aircraft in roll and pitch angles, maintain a constant altitude and airspeed by implementing a PD controller for each dynamic (roll, pitch, altitude and airspeed). The altitude and airspeed setpoints are manually introduced by a graphic user interface, the altitude controller outputs the pitch setpoint and the roll setpoint is obtained from the path following controller.

Aircraft Dynamics

This layer integrates the set of differential equations representing the aircraft dynamics. The input of this layer are the inputs of the low level autopilot layer and the outputs are the data from the simulated sensors: GPS position, aircraft attitude, airspeed. The aircraft dynamics layer sends the outputs to the upper layers.

4.5.2 Simulation Scenario

The path generated to weep the search area.

We use the MAV3DSim simulation platform along with the Dubins path generator and the path-following strategy previously described to present a simulation scenario. The description of the scenario is as follows: A person is missing and is located somewhere in a known area. The main task of the UAV is to find this person, so it will sweep this area in order to find the missing person. First we need to define the search area as a rectangle with the aid of a user interface, then using the proposed path generator algorithm define the a path for sweeping the rectangle area. The starting point of the path will be one of the corners of the rectangle and it selects the closest to the current position of the UAV as depicted in Figure 4.5.2. The UAV will travel along the path until it is sufficiently close to the lost person (red dot in Figure 4-1). When the missing person is found a circular path is generated to surround the missing person. The simulation can be seen in `https://www.youtube.com/watch?v=_AUW8_g-jb0`

Figure 4-1: Circular path generated to surround the missing person.

4.6 Experimental flight

During search and surveillance mission it is sometimes required that the aircraft fly through a $n\,(n \geq 3)$ number of waypoints in a given order. We will be using the path generation described in 4.2 and the kinematic controller described in 4.4, and we have the following assumptions for the experimental flight

- The airspeed V_t of the vehicle is assumed to be constant through the mission. We use a well tuned PID controller to maintain the airspeed constant.
- The aircraft has a bound on its maximum turn rate and minimum turn radius.
- Due to the nonholonomic nature of the vehicle it is not possible to pass through all waypoints and at same time fly over all portions of the straight lines between the waypoints, thus, our main priority is to pass through the exact point of the waypoints.
- The waypoints have a separation not greater than 50m, this is due to the battery limitation of the aircraft, which give us a flight time of 30 minutes.

First consider n waypoints denoted by $w_1, w_2, ..., w_n$. The assumption is that initially the aircraft is in an arbitrary position, using the path generation algorithm it creates a path from

the current position and orientation to the first waypoint with final angle equal to the angle formed by the line between the first two waypoints. As the experimental aircraft position is obtained via GPS all the computation should be using the $(latitud, longitude)$ for the position of the aircraft, so the waypoints have the following structure

$$w_n = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{4.41}$$

The computation of the bearing angle θ_b between two waypoints (w_{n-1}, w_n) its done as follows

$$\begin{aligned} y &= \sin(y_n - y_{n-1})\cos(x_n) \\ x &= \cos(x_{n-1})\cos(x_n) - \cos(y_n - y_{n-1}) \\ \theta_b &= \tan^{-1}\left(\frac{y}{x}\right) \end{aligned} \tag{4.42}$$

Using the following algorithm we generate the path from the current position of the aircraft

Algorithm 3 Generate waypoints path

n =number of waypoints; $i = 0$ // Current waypoint
θ_b =GetBearing(waypoints[0],(waypoints[1]) ; Angle between waypoints
Points = GeneratePath(CurrentPosition, CurrentHeading, waypoints[0], θ_b) ; Generate initial path to arrive the first waypoint
while $i \leq n - 2$ **do**
  θ_{b1} =GetBearing(waypoints[i],(waypoints[i+1]) ; Angle between waypoints
  θ_{b2} =GetBearing(waypoints[i+1],(waypoints[i+2]) ; Angle between waypoints
  Points.Add(GeneratePath(waypoints[i],θ_{b2},waypoints[i+1],θ_{b2} =))
end while
θ_b =GetBearing(waypoints[n-1],(waypoints[n]) ; Angle between waypoints
Points.Add(GeneratePath(waypoints[n-1],θ_b,waypoints[n],θ_b =)); Last waypoint

The experimental platform is equipped with an Pixhawk autopilot and a Raspberry Pi companion computer. The autopilot is in charge of the stabilization of the aircraft at the received command from the companion computer. The companion computer is in charge of the navigation/Path generation and the nonlinear control described in the previous section.

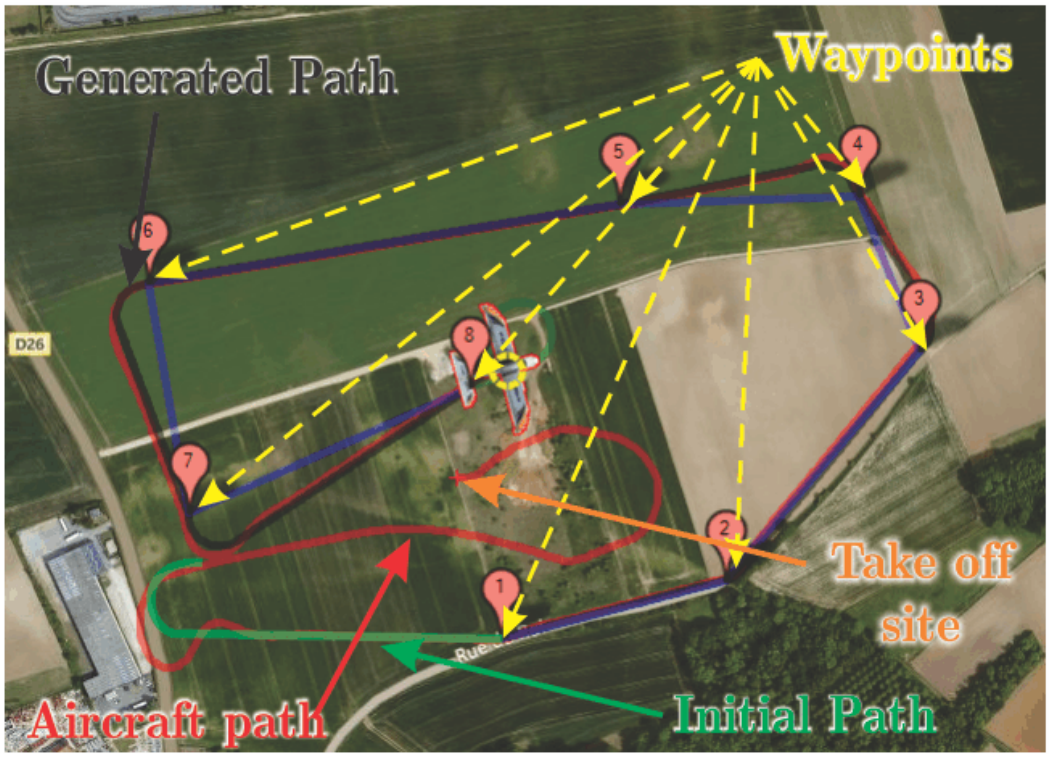


Figure 4-2: Generated path (in black) to pass through all the waypoints

The experiment is depicted in Figure 4-2. We start the experiment with a manual take-off from the take off site to some point near the first waypoint and until we reach the reference altitude of $100m$. After arriving to the desired altitude we initiate the path generation strategy. The path generator takes as an input the current position and heading of the aircraft and generates an initial path to the first way point, the green line in the picture, then it will continue generating the path for all the remaining waypoints, in the picture it is the black line.

From the Figure 4-3 we can observe the performance of the controller, as there is still a small error in the following of the path, we consider the performance of the system is acceptable. At the beginning of the experiment we enter the first curve with more velocity that the calculated for the minimum turn radius and therefore the aircraft step out the path for a moment, once the airspeed is regulated to the correct value the aircraft follow the path without issues. Figure 4-4 show the computed controllers for the roll θ, pitch ϕ and the virtual particle velocity $\dot{s}$, the red lines is the moment of transition from one waypoint to the other. In the Figure 4-5 we can

verify that the errors tend asymptotically to zero, each time the aircraft arrives toward a new waypoint the error is increased but rapidly compensated. Finally in figures 4-6 and 4-7 we can see the altitude and airspeed vs the setpoint to verify that it remains regulated by the PID controller implemented.

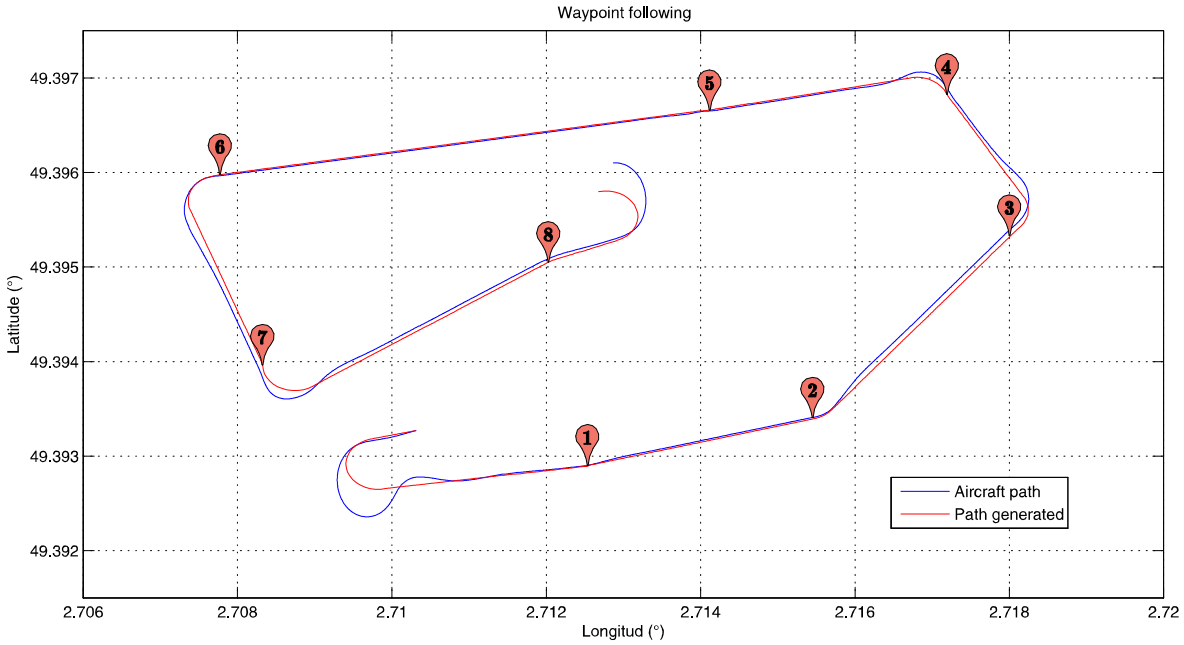


Figure 4-3: Geneerated path to pass through all the waypoints.

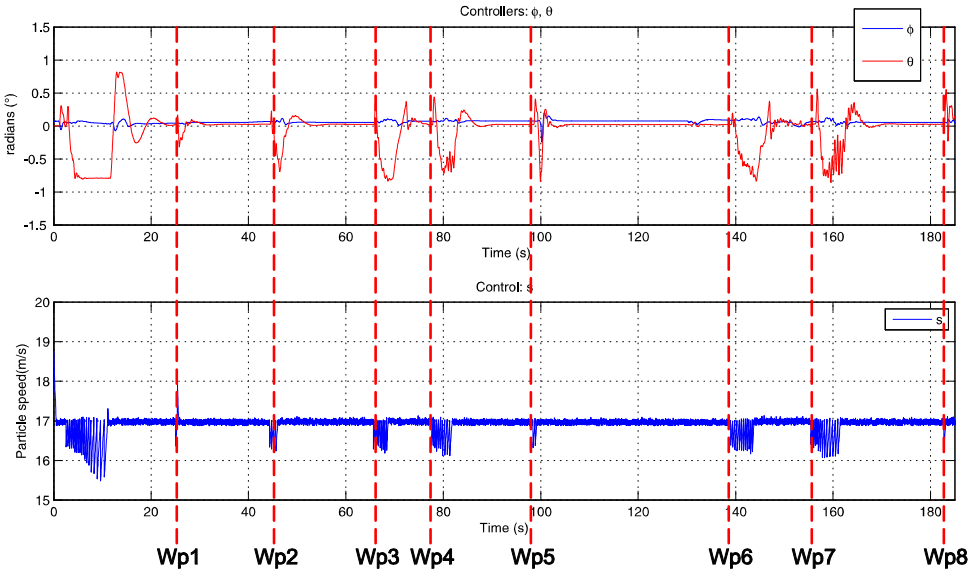


Figure 4-4: Control signals

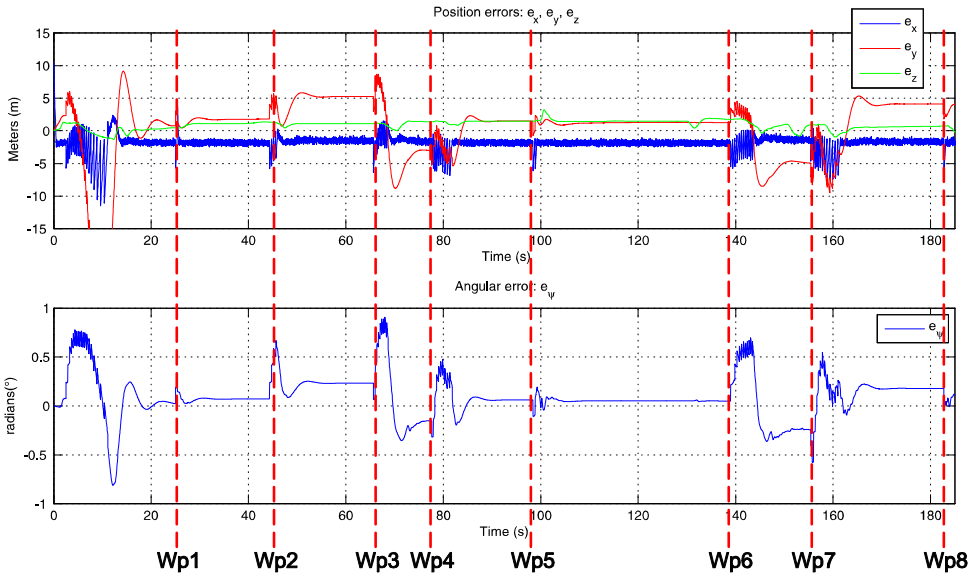


Figure 4-5: Position errors and angular error.

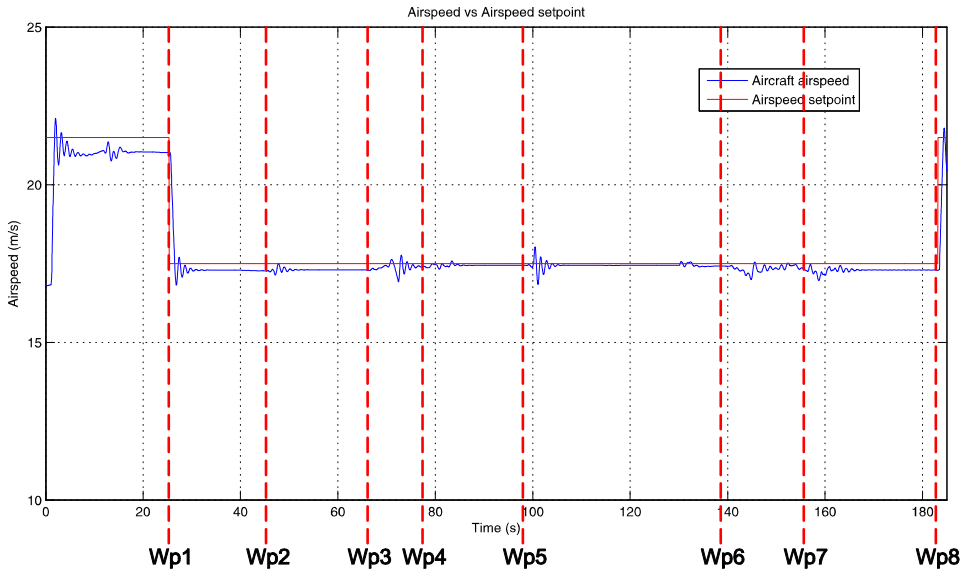


Figure 4-6: Aircraft airspeed in blue vs airspeed setpoint.

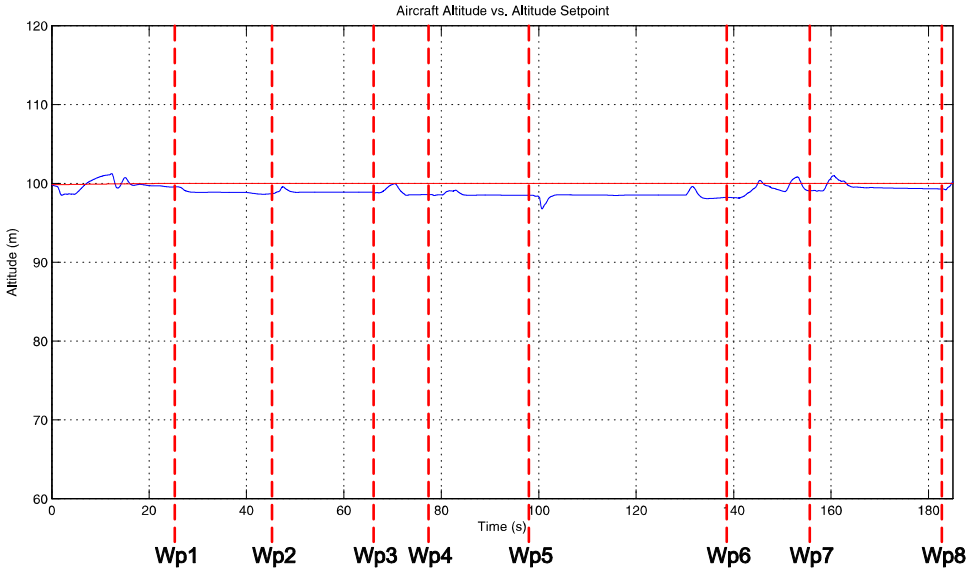


Figure 4-7: Aircraft altitude vs altitude setpoint

Chapter 5

Path Following in 3D

Real-time operation of a UAV involve movement in a three dimensional space. Therefore it is of very importance to handle the problem of generation of three dimensional trajectories and to be able to follow them with the UAV. In this chapter we consider that the UAV is flying in a three dimensional space. The path following is of great importance for the autonomous operations of the UAVs. In recent years the missions involving autonomous flights of the UAV have become more and more complicated, therefore a precise 3D path-following strategy is required. In the literature there are several linear and nonlinear guidance methods. Most of them have been developed for two dimensional path-following, which can be classified into three approaches; the error kinematics based approach, the vector field based approach, and the virtual target following approach.

In the error kinematics approach, several nonlinear control techniques have been applied for the regulation of the error state variables, where the error variables can be defined in many ways, including cross-track error[5] [69], along-track error[77] [40] and vehicle heading error[3]. Once the dynamic model of the state variables is derived, a nonlinear control design method is applied to regulate the errors [9] [12].

In the vector field approach, a vector field is designed so that the vehicle converges to the desired path along the vector field [60] [25]. However the vector field can only de designed for some types of planar curves, such as a straight line or a circle[24], a sinusoidal path [27], etc. Therefore the vector-field-based approach is not applicable to the general case of trajectories in a 3D space.

The virtual tarjet following, also known as look-ahead approach, the guidance control is designed to track a virtual particle moving along the desired path, which is ahead of the vehicle. Different concepts are taking in the design process of these type of methods, such as the pure pursuit guidance [58], line-of-sight guidance [1], proportional navigation guidance [74],trajectory shaping guidance [64].

Finally in recent years a new method has been developed using the virtual target following approach, the nonlinear path following guidance law [16]. The strengths of this method are the simplicity of the guidance command and that it enables tight tracking of curved paths by anticipating the upcoming desired path and wind effect compensation.

5.1 3D Dubins path

One of the classical paths for aircraft maneuvers is the circular helix, whose projection on the $x - y$ plane is a circle. The path can also be seen as a path on the surface of a cylinder. An important property of this trajectory is that the ration of curvature and torsion remains constant. The cylinder is used as the most logical extension of the circle to the 3D space, as depicted in the Figure 5-1, in which the same rotation is executed and at the same time a change in the altitude is performed.

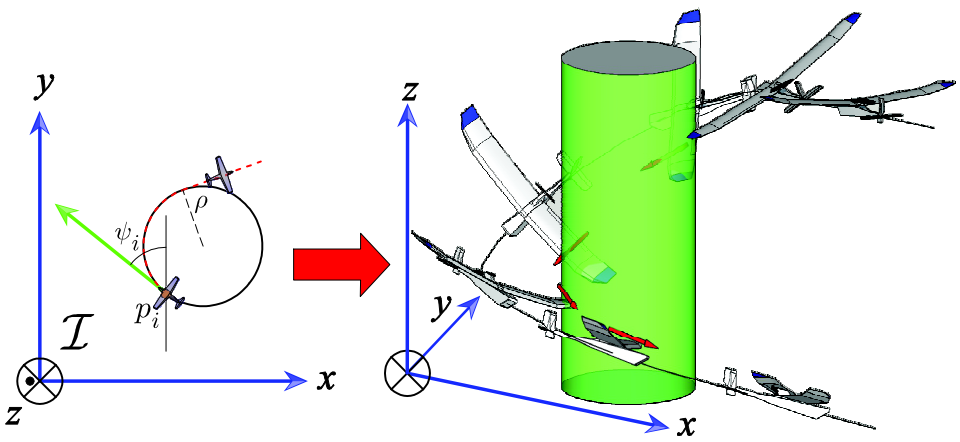


Figure 5-1: The cylinder is considered as the 3D extension of the circle.

The previous condition of constant altitude is relaxed so that the initial and final position

does not lie on the same plane. The controller also needs to take this into consideration and to introduce the altitude as a new state in the model used to develop the previous controller.

The design of the Dubins path is shown on Section 4.2, the Dubins path has two circular segments and a straight line segment, and all the three segments are in the same $x - y$ plane, thus it is easy to find the common tangent to the initial and final position. A similar approach but extended to the 3D case is developed to compute the 3D Dubins path. In view of the change in altitude, there are eight possible cases for the Dubins path in 3D, $LSLU$, $LSRU$, $RSRU$, $RSRLU$, $LSLD$, $LSRD$, $RSRD$, $RSRLD$ which L stands for Left, R stands for Right, S for Straight, U for upwards and D for Downwards. The downwards trajectories are depicted in Figure 5-2 and the Figure 5-3 shows the upwards movements.

As in the 2D case the first step is to choose the type of Dubins path. We have the initial and final configuration defined as follows

$$\mathbf{c}_i \left(p_i, \psi_i\right) \tag{5.1}$$

$$\mathbf{c}_f \left(p_f, \psi_f\right) \tag{5.2}$$

where $p_i = [x_i, y_i, z_i]$ is the initial position of the path and $p_f \left(x_f, y_f, z_f\right)$ is the final position of the path, both are expressed in an inertial frame, the initial heading angle ψ_i and the final heading is ψ_f.There are 8 possible selections for the 3D Dubins paths, from the difference in altitude we can choose if it is a downwards or upwards maneuver. We choose the shorteest path by comparing the proyection of the circles of the top an bottom of the cylinders on the same $X - Y$ plane, as in Figure 5-4. The smallest distance between the center of the circles gives us the shortest Dubin path according to Table 5.1.

Based on the initial and final configuration $\mathbf{c}_i$ and $\mathbf{c}_f$ and the minimal radius ρ from (4.5), the center of the top and bottom circles of the cylinder are computed as follows

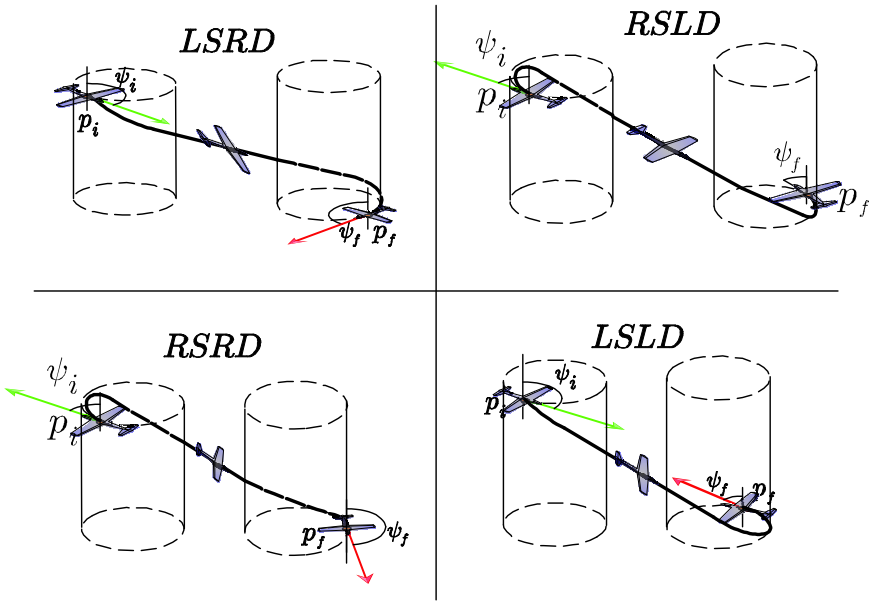


Figure 5-2: 3D Dubins paths, downwards trayectory

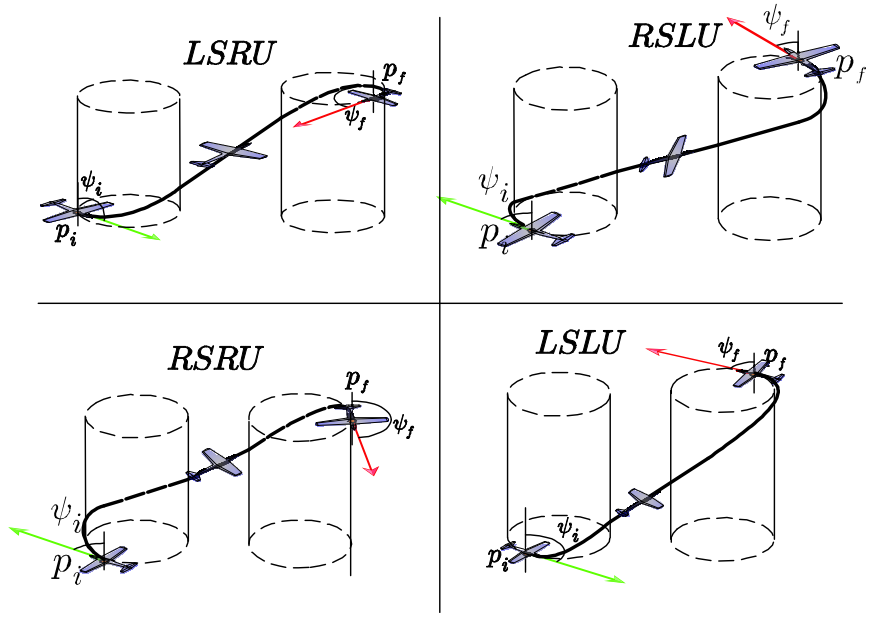


Figure 5-3: 3D Dubins paths, upwards trayectory

$$
\begin{aligned}
CR_i &= (x_{Ri}, y_{Ri}, z_{Ri}) = (x_i + \rho\cos\psi_i, y_i - \rho\sin\psi_i, z_i) \\
CL_i &= (x_{Li}, y_{Li}, z_{Ri}) = (x_i - \rho\cos\psi_i, y_i + \rho\sin\psi_i, z_i) \\
CR_f &= (x_{Rf}, y_{Rf}, z_{Ri}) = \left(x_f + \rho\cos\psi_f, y_f - \rho\sin\psi_f, z_f\right) \\
CL_f &= (x_{Lf}, y_{Lf}, z_{Ri}) = \left(x_f - \rho\cos\psi_f, y_f + \rho\sin\psi_f, z_f\right)
\end{aligned}
$$

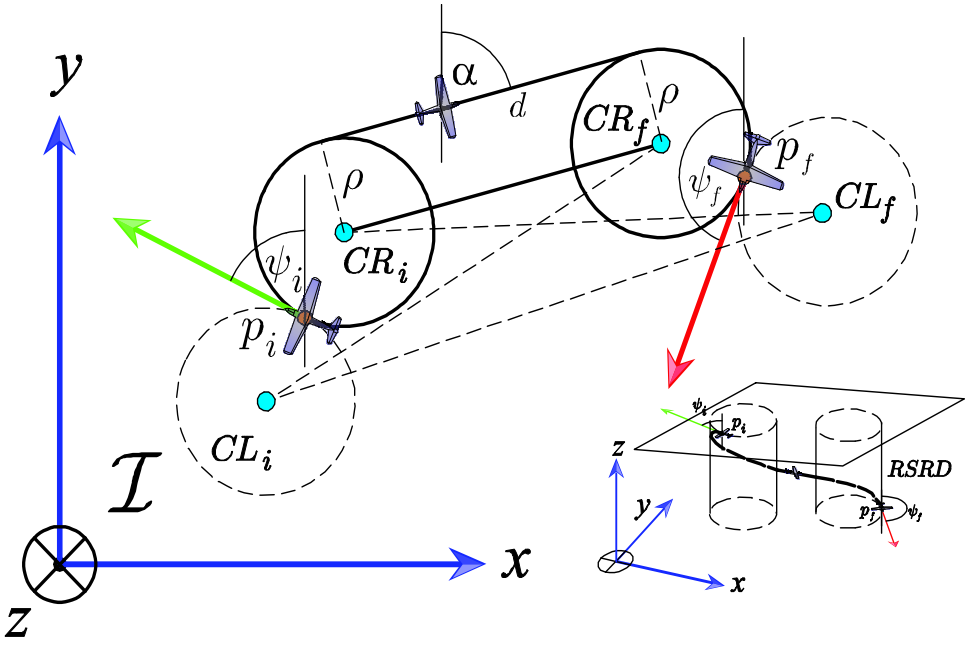


Figure 5-4: Minimun distance on the proyection of the circles of the cylinder on the same plane.

5.2 3D Dubins Path Generation

In this section the classical result of Dubins is extended to the 3D case. The initial and final configuration (p_i, ψ_i) and $\left(p_f, \psi_f\right)$,respectively, are given w.r.t. an inertial frame (Local ENU frame). The general procedure for the 3D Dubins paths is to divide the path Γ in three segments as follows

Table 5.1: 3D Dubins path selection.

Shortest distance	**Dubins Path**
CR_iCR_f	$\begin{cases} if\ z_i > z_f \rightarrow RSRD \\ if\ z_i < z_f \rightarrow RSRU \end{cases}$
CR_iCL_f	$\begin{cases} if\ z_i > z_f \rightarrow RSLD \\ if\ z_i < z_f \rightarrow RSLU \end{cases}$
CL_iCL_f	$\begin{cases} if\ z_i > z_f \rightarrow LSLD \\ if\ z_i < z_f \rightarrow LSLU \end{cases}$
CL_iCR_f	$\begin{cases} if\ z_i > z_f \rightarrow LSRD \\ if\ z_i < z_f \rightarrow LSRU \end{cases}$

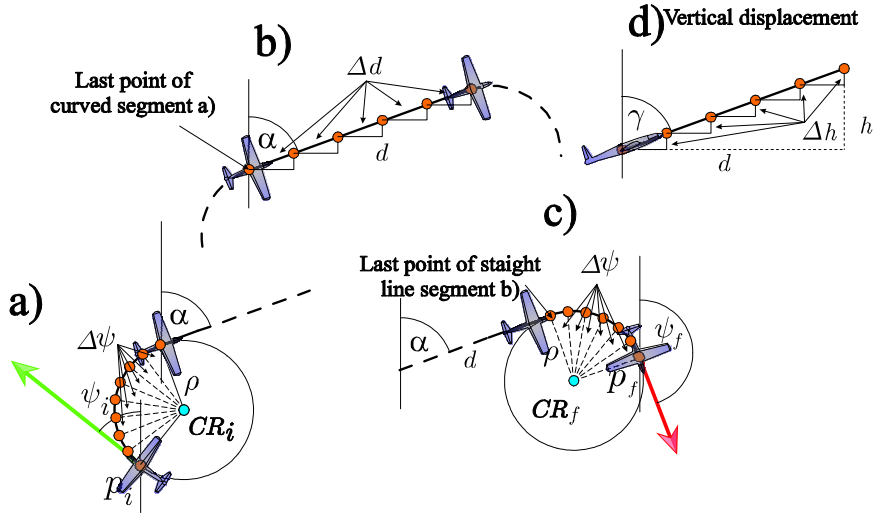


Figure 5-5: The 3D Dubins paths generation.

1. The first segment Γ_1 starts from the initial configuration $\mathbf{c}_i\,(p_i, \psi_i)$ and begin a turn clockwise or counter.clockwise describing an arc of radious ρ and center in CR_i or CL_i until the heading angle ψ is equal to the angle α, this will turn the aircraft into the correct direction to follow a straight line to the tangent of the second circunference, as depicted in Figure 5-5.a. Also the elevation angle γ of the path is selcted such as the overall vertical displacement results in the change of altitude between the initial and final position in altitude z_i and z_f, as seen in FIgure 5-5.d

2. The second segment Γ_2 consists of a straight line path. The heading and elevation angle of the path remains constant. This path will continue until it travels a distance d between

the common tangents of the two circles, see Figure 5-5.b.

3. The third segment Γ_3 is again an arc of radious ρ and center in CR_f or CL_f. The initial heading angle of the path is α, as it remains constant in the previous path, then it will change from α to the final heading angle ψ_f. The elevation path γ will remain constant as in the prevous segment, in order to arrive at the desired altitude, see Figure 5-5.c.

3D Dubins $RSRU/P$ and $LSLU/P$

In th cases of 3D Dubins path $RSRU$, $RSRD$, $LSLU$ and $LSLD$ the angle α and γ and distance d needed to generate the path are computed in a similar manner. The angle α is the angle of the straight line segment d which is measured from the vertical $y-$axis. Depending on the turn to the left or a turn to the right, we use the angle $\alpha = \alpha_R$ when it is a turn to the right and the angle $\alpha = \alpha_L$ when it is a turn to the left. The visual representation of these angles can be seen in Figure 5-6 and they are computed as follows

$$\alpha_R = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Rf} - y_{Ri}}{x_{Rf} - x_{Ri}}\right) \tag{5.3}$$

$$\alpha_L = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Lf} - y_{Li}}{x_{Lf} - x_{Li}}\right) \tag{5.4}$$

The length d, the straight line segment Γ_2, is equal to the distance between the center of the circles CR_i and CR_f, or CL_i and CL_f, depending on whether is a turn to the left or a turn to the right. In this case we also have a difference if it is a turn to the left or a turn to the right, we use the distance $d = d_R$ if the path is a $RSRU$ or $RSRD$ type, and $d = d_L$ if the path is a $LSLU$ or $LSLD$ type, these distance are computed as follows

$$d_R = \sqrt{(x_{Rf} - x_{Ri})^2 + (y_{Rf} - y_{Ri})^2 + (z_{Rf} - z_{Ri})^2} \tag{5.5}$$

$$d_L = \sqrt{(x_{Lf} - x_{Li})^2 + (y_{Lf} - y_{Li})^2 + (z_{Lf} - z_{Li})^2} \tag{5.6}$$

The previously defined distance d is needed to compute the elevation path γ, we also need to compute the traveled distance in the Γ_1 and Γ_3 segments, defined as d_1 and d_2. The total distance is the sum of the three distace d, d_1, d_2 . The diference in altitude is also needed for

the computation of the angle γ. All the needed values are computed as follow

$$d_1 = |\psi_i - \alpha| * 2\rho \tag{5.7}$$

$$d_2 = |\alpha - \psi_f| * 2\rho \tag{5.8}$$

$$d_T = d + d_1 + d_2 \tag{5.9}$$

$$h = |z_i - z_f| \tag{5.10}$$

Finally the γ angle is computd as follows

$$\gamma = \tan^{-1}\left(\frac{h}{d_T}\right) \tag{5.11}$$

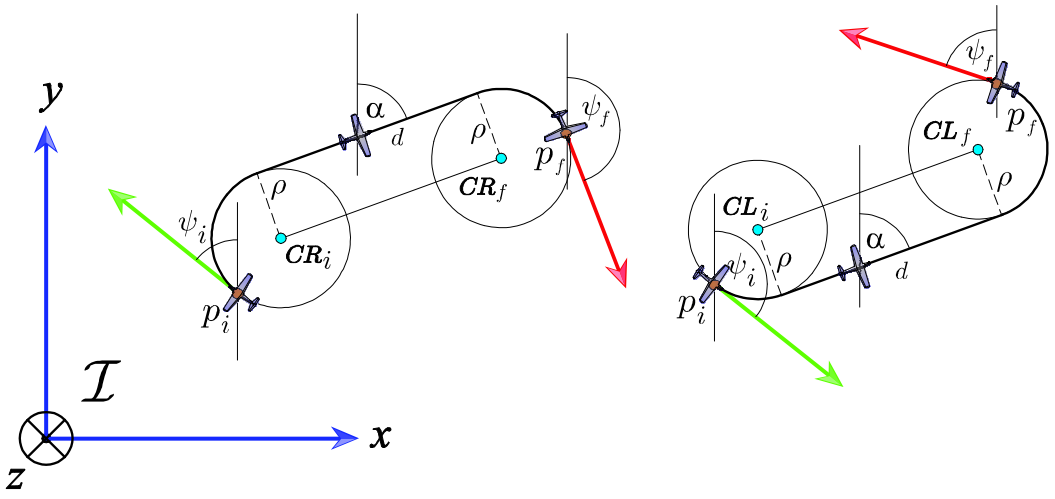


Figure 5-6: 3D Path generation, turn to the left and turn to the right.

5.2.1 3D Dubins *RSLU/P* and *LSRU/P*

In the cases of 3D Dubins path *RSLU*, *RSLP*, *LSRU* and *LSRP* again we have the need of computing the angle α and γ and distance d. The angle γ is computed as in the previous section, but that is not the case for the computation of the angle α and the distance d. As depicted in Figure 5-7 we have introduce two new angles and an extra measurment of distance. The angle η is the angle between the segment joining the center of the circles CR_i and CL_f or

CL_i and CR_f , depending on the path selection. The μ angle is the angle between the segment $\overline{CR_iCL_f}$ or $\overline{CL_iCR_f}$, depending on the path selection, and the point of the tangent to the circle and the segment d. The distance l is the distance between the two circles forming the path. We start with the computation of the angle α as follows

$$\alpha = \eta + \mu - \frac{\pi}{2} \tag{5.12}$$

The length of the straight line segment d is computed with the following equation, we use the distance $d = d_{RL}$ if the path is a $RSLU$ or $RSLD$ type, and $d = d_{LR}$ if the path is a $LSRU$ or $LSRD$ type, the these distance are computed as follows

$$d_{RL} = \sqrt{l^2 + 4\rho + (z_{Lf} - z_{Ri})^2} \tag{5.13}$$

$$d_{LR} = \sqrt{l^2 + 4\rho + (z_{Rf} - z_{Li})^2} \tag{5.14}$$

We use the angle $\eta = \eta_{RL}$ if the path is a $RSLU$ or $RSLD$ type, and $\eta = \eta_{LR}$ if the path is a $LSRU$ or $LSRD$ type. We compute these angles as follows

$$\eta_{RL} = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Lf} - y_{Ri}}{x_{Lf} - x_{Ri}}\right) \tag{5.15}$$

$$\eta_{LR} = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Rf} - y_{Li}}{x_{Rf} - x_{Li}}\right) \tag{5.16}$$

The μ angle is computed as follows

$$\mu = \tan^{-1}\left(\frac{2\rho}{d}\right) \tag{5.17}$$

Finally we use the distance $l = l_{RL}$ if the path is a $RSLU$ or $RSLD$ and $l = l_{LR}$ if the path is a $LSRU$ or $LSRD$ type. is computed as follow

$$l_{RL} = \sqrt{(x_{Rf} - x_{Li})^2 + (y_{Rf} - y_{Li})^2 + (z_{Rf} - z_{Li})^2} \tag{5.18}$$

$$l_{LR} = \sqrt{(x_{Lf} - x_{Ri})^2 + (y_{Lf} - y_{Ri})^2 + (z_{Lf} - z_{Ri})^2} \tag{5.19}$$

The 3D path generator algorithm produces an array of n points p_n which starts at $p_0 = p_i$

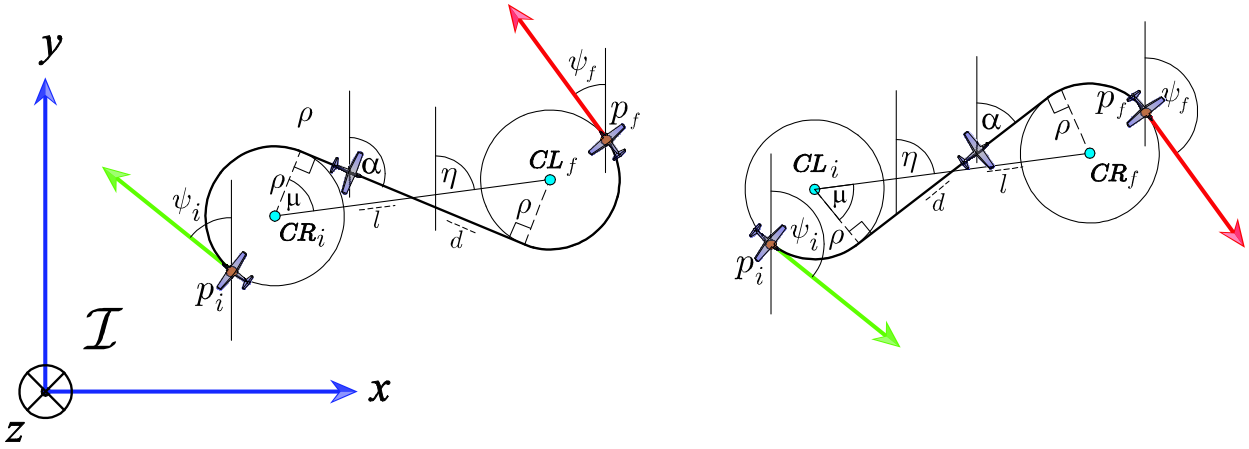


Figure 5-7: 3D Dubins path RSL and LSR

and ends in $p_n = p_f$. The Γ path is parameterized by its arc length s, starting in $s = 0$ and finalizing in $s = d_T$. The coordinates of the $n - th$ point p_n of the Γ_1 and Γ_3 arc segments are obtained with the following equation

$$p(s) = \mathbf{c} + \begin{bmatrix} \rho\cos(\lambda s + \psi) \\ \rho\sin(\lambda s + \psi) \\ s\rho\tan(\gamma) \end{bmatrix} \tag{5.20}$$

where $\mathbf{c} = \mathbf{c}_i$ if it is the path Γ_1 and $\mathbf{c} = \mathbf{c}_i$ if it is the path Γ_3. λ is the direction of rotation, $\lambda = 1$ for a turn to the left and $\lambda = -1$ for a turn to the right. The angle $\psi = \psi_i$ when it is the path Γ_1 and $\psi = \psi_f$ when the path is Γ_3.

Now the path Γ_2, the straight line path, is generated using the following equation

$$p(s) = p + \begin{bmatrix} s\cos(\psi) \\ s\sin(\psi) \\ s\tan(\gamma) \end{bmatrix} \tag{5.21}$$

where p is the last point of the Γ_1 segment. This path will continue where the straight line segment to conect the two cylinders anf complete the Γ path.

The output of the 3D path generation algorithm can be seen in Figure 5-8

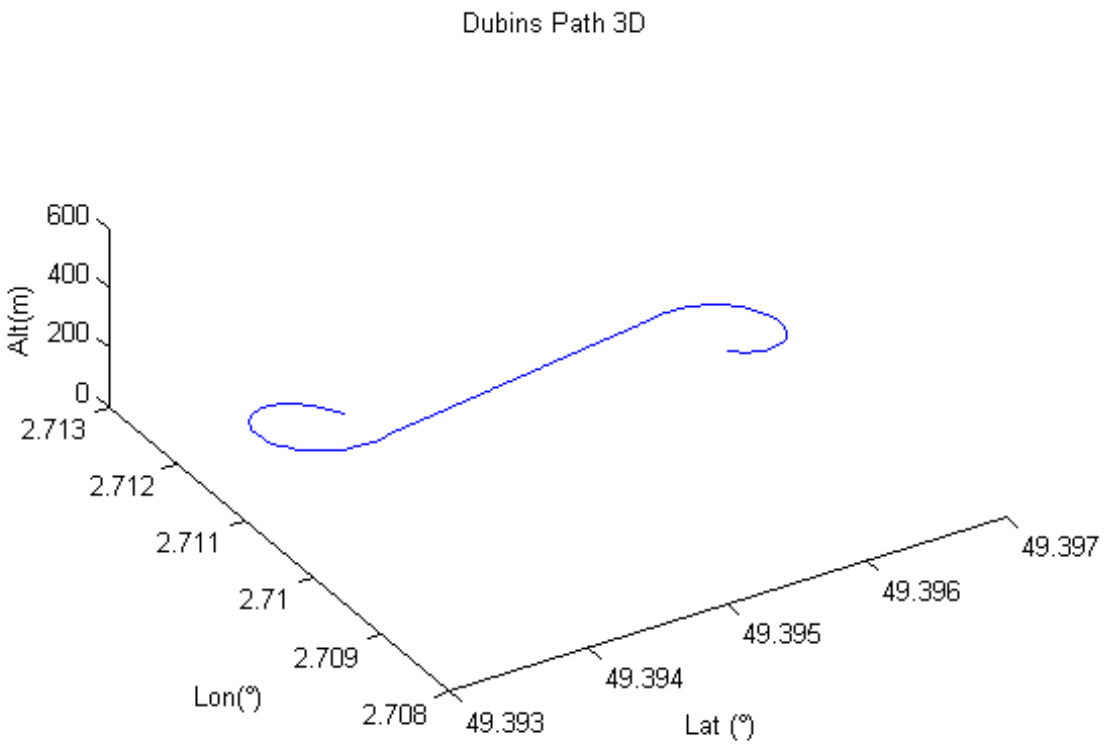


Figure 5-8: Output of the 3D path generation.

5.3 Control Strategy

This section provides a rigorous kinematic formulation for the problem of steering the aircraft along a desired path.

5.3.1 Aircraft Kinematic Model

The Dubins aircraft was first introduced in [8]. We have built upon this model to increase accuracy in modeling the aircraft kinematics and to be more consistent with the commonly used aircraft models. Our model works under the assumption that the autopilot is well tuned, this means that the airspeed, flight-path angle and bank angles states converge with desired response to their commanded values. The following kinematic model describes the motion of the UAV:

$$\begin{aligned} \dot{x} &= V\cos\psi\cos\theta \\ \dot{y} &= V\sin\psi\cos\theta \\ \dot{z} &= -V\sin\theta \\ \dot{\psi} &= \omega \end{aligned} \tag{5.22}$$

where x, y and z denote the inertial position of the aircraft in a 3D inertial frame. ψ is the heading angle, ω is the heading angular rate, θ is the pitch angle, V is the airspeed of the aircraft which remains constant.

If we consider coordinated turn conditions, then we have the sideslip angle β equal to zero, the model for the heading angle, as in [21], is given by,

$$\dot{\psi} = \frac{g}{V}\tan\phi \tag{5.23}$$

where g is the magnitude of gravity at sea level, ϕ is the roll angle, and $\dot{\psi}$ is the heading angular rate.

The state and control vectors of the above model are described respectively as:

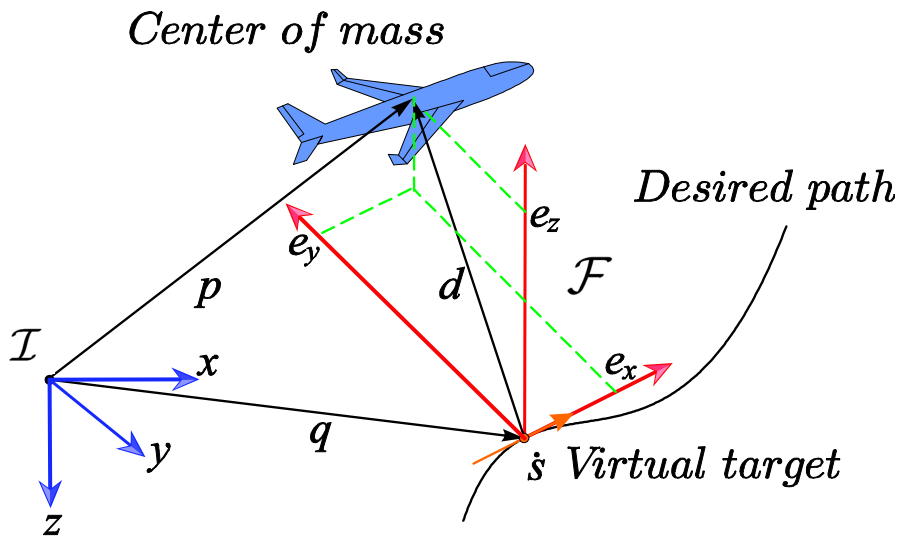


Figure 5-9: 3D path following problem.

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} x & y & z & \psi \end{bmatrix} \\ \mathbf{u} &= \begin{bmatrix} \theta_c & \phi_c \end{bmatrix} \end{aligned}$$

5.3.2 Problem Statement

The key idea behind the path-following controller relies on reducing two expressions to zero: The distance d between the aircrafts center of mass p and the point q on the path and the angle between the airspeed vector and the vector tangent to the trajectory.

As depicted in Figure. 5-9, using a moving particle s along the trajectory at a velocity $\dot{s}$. Considering a frame attached to the virtual particle, the so called Serret Frame denoted by $\mathcal{S}$. We use $\dot{s}$ as a control input.

The angle and distance will become in the coordinates of the error space.

5.3.3 Error dynamics

Consider the 3D curve represented by smooth functions parameterized by t. Thus $(x_S(t), y_S(t), z_S(t))$ represent the virtual particle coordinates The inertial position of the aircraft is defined by

$p = \begin{bmatrix} x & y & z \end{bmatrix}^T$ in the inertial reference frame $\mathcal{I}$. To follow the given trajectory we define the inertial vector error $d^{\mathcal{I}} = p - q(s)$ expressed in $\mathcal{I}$, which will be minimized in order to track the desired trajectory. Note that the tangent vector coincides with the x-axis of the Serret frame $\mathcal{S}$, now consider the rotation of the vector tangent to the path in the z and y axis w.r.t the inertial frame $\mathcal{I}$ by the angles ψ_S and θ_S, respectively. The angles ψ_S and θ_S can be obtained, as in [61], from the parameterized curve as

$$\psi_S = \tan^{-1} \frac{y'_S(t)}{x'_S(t)}; \tag{5.24}$$

$$\theta_S = \frac{z'_S(t)}{\sqrt{x'_S(t)^2 + y'_S(t)^2}} \tag{5.25}$$

where $x'_S = \frac{dx_S(t)}{dt}$, $y'_S = \frac{dx_{yS}(t)}{dt}$, $z'_S = \frac{dz_S(t)}{dt}$

Consider the rotation matrix R_S from $\mathcal{I}$ to $\mathcal{S}$ and the rotation matrix R_B from $\mathcal{I}$ to $\mathcal{B}$:

$$R_S = \begin{bmatrix} \cos\theta_S \cos\psi_S & \cos\theta_S \sin\psi_S & -\sin\theta_S \\ -\sin\psi_S & \cos\psi_S & 0 \\ \sin\theta_S \cos\psi_S & \sin\theta_S \sin\psi_S & \cos\theta_S \end{bmatrix} \tag{5.26}$$

$$R_B = \begin{bmatrix} \cos\theta \cos\psi & \cos\theta \sin\psi & -\sin\theta \\ -\sin\psi & \cos\psi & 0 \\ \cos\psi \sin\theta & \sin\theta \sin\psi & \cos\theta \end{bmatrix} \tag{5.27}$$

The error d^I expressed in the Serret frame is given by

$$d^S = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = R_S d^I = R_S (p - q) \tag{5.28}$$

Furthermore we define the yaw angle error as

$$e_\psi = \psi - \psi_S \tag{5.29}$$

To obtain the error state dynamic equations, we must compute the time derivative of (5.28)

and (5.29). By differentiating (5.28) we obtain:

$$\begin{aligned} \dot{d}_S &= \dot{R}_S (p - q) + R_S (\dot{p} - \dot{q}) \\ &= \omega_S \times R_S (p - q) + R_S (\dot{p} - \dot{q}) \end{aligned} \tag{5.30}$$

where

$$\begin{aligned} \omega_S &= \begin{bmatrix} 0 \\ \dot{\theta}_S \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta_S & 0 & -\sin\theta_S \\ 0 & 1 & 0 \\ \sin\theta_S & 0 & \cos\theta_S \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}_S \end{bmatrix} \\ &= \begin{bmatrix} -\dot{\psi}_S \sin\theta_S \\ \dot{\theta}_S \\ \dot{\psi}_S \cos\theta_S \end{bmatrix} \end{aligned} \tag{5.31}$$

From (5.22), the time derivative of p and q can be expressed as

$$\dot{p} = R_B^T \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix} \tag{5.32}$$

$$\dot{q} = R_S^T \begin{bmatrix} \dot{s} \\ 0 \\ 0 \end{bmatrix} \tag{5.33}$$

The time derivative of (5.29) and using (5.23) results in

$$\begin{aligned} \dot{e}_\psi &= \dot{\psi} - \dot{\psi}_S \\ &= \frac{g}{V} \tan\phi - \dot{\psi}_S \end{aligned}$$

with

$$\begin{aligned}\dot{\psi}_S &= \kappa \dot{s} \\ \dot{\theta}_S &= \tau \dot{s}\end{aligned}$$

where $\frac{d\psi_s}{dt} = \kappa$ is the path curvature. The path curvature is expressed as a function of the trajectory coordinates $(x(t), y(t))$ and its first and second derivatives w.r.t. the parameter t, i.e $x'_S = \frac{dx_S}{dt},\ y'_S = \frac{dy_S}{dt}$. Thus the path curvature $\frac{d\psi_S}{dt} = \kappa$ is given by

$$\kappa = \frac{|y''_S x'_S - y'_S x''_S|}{\left(x'^2_S + y'^2_S\right)^{3/2}} \tag{5.34}$$

Finally using (5.30) and substituting (5.31), (5.32) and (5.33) we obtain the error kinematics model suitable for control purposes

$$\begin{aligned}\dot{e}_x &= \tau \dot{s} e_z - \kappa \dot{s} \cos\theta_S e_y + V \sin\theta \sin\theta_S + V \cos\theta \cos\theta_S \cos e_\psi - \dot{s} \\ \dot{e}_y &= \kappa \dot{s} e_x \cos\theta_S + \kappa \dot{s} e_z \sin\theta_S + V \cos\theta \sin e_\psi \\ \dot{e}_z &= -\kappa \dot{s} e_y \sin\theta_S - \tau \dot{s} e_x + V \cos\theta \sin\theta_S \cos e_\psi - V \sin\theta \cos\theta_S \\ \dot{e}_\psi &= \frac{g}{V} \tan\phi - \kappa \dot{s}\end{aligned} \tag{5.35}$$

Using the previous error kinematics model we design a feedback control law for θ_c, ϕ_c and $\dot{s}$ such that all closed-loop signals are bounded and all the errors converge to zero.

5.3.4 Controller Design

In this section we present a nonlinear guidance controller to follow a desired path. We design the controller for the guidance using the error kinematic model (5.35). As we are considering that the aircraft is equipped with an autopilot unit and it is well tuned we use directly the desired pitch angle θ_c and the desired roll angle ϕ_c. We also introduce a virtual controller in the form of the velocity $\dot{s}$ of the virtual particle which moves along the desired trajectory to stabilize all the error signals $e_x,\ e_y,\ e_z,\ e_\psi$ to zero. The control objective can be archived using

a Lyapunov function candidate given by

$$V = \frac{1}{2}e_x^2 + \frac{1}{2}e_y^2 + \frac{1}{2}e_z^2 + \frac{1}{2}\left(e_\psi - \delta\left(e_y\right)\right)^2 \tag{5.36}$$

where $\delta\left(e_y\right)$ is a sigmoid function, as in our previous controller in Section 4.4, that introduce a desired approach angle as

$$\delta\left(e_y\right) = -\psi_a \frac{e^{2k_\delta e_y} - 1}{e^{2k_\delta e_y} + 1} \tag{5.37}$$

where $k_\delta > 0$ and $\psi_a \in \left[0, \frac{\pi}{2}\right]$. The sigmoid function (5.37) is bounded and differentiable w.r.t. the error e_y. It provides the desired relative course transition of the aircraft to the path as a function of e_y., and satisfies the following condition

$$e_y \sin\left(\delta\left(e_y\right)\right) \leq 0$$

which steer the aircraft in the correct direction, i.e., turn left when the aircraft is on the right side of the path, and turn right in the opposite situation. We note that the approach rate can be controlled by the adjustment of the parameter k_δ.

The time derivative of (5.36) along the trajectory of (5.35)

$$\begin{aligned}
\dot{V} \;=\; & e_x\left(V\sin\theta\sin\theta_S + V\cos\theta\cos\theta_S\cos e_\psi - \dot{s}\right) + \\
& e_y V\cos\theta\sin e_\psi + \\
& e_z\left(V\cos\theta\sin\theta_S\cos e_\psi - V\cos\theta_S\sin\theta\right) + \\
& \left(e_\psi - \delta\left(e_y\right)\right)\left(\frac{g}{V}\tan\phi - \kappa\dot{s} - \dot{\delta}\left(e_y\right)\left(\kappa\dot{s}e_x\cos\theta_S + \kappa\dot{s}e_z\sin\theta_S + V\cos\theta\sin e_\psi\right)\right)
\end{aligned} \tag{5.38}$$

Rearranging the terms of (5.38) we arrived to the expression

$$\dot{V} = e_x\left(\alpha - \theta\dot{s}\right) - e_z V\sin\theta\cos\theta_S + V e_y\sin\left(\delta\left(e_y\right)\right) + \left(e_\psi - \delta\left(e_y\right)\right)\left(\frac{g}{V}\tan\phi + \beta\right) \tag{5.39}$$

where

$$\alpha = V \sin\theta \sin\theta_S + V \cos\theta \cos\theta_S \cos e_\psi \tag{5.40}$$

$$\begin{aligned} \beta = & -\kappa\dot{s} - \dot{\delta}(e_y)\left(\kappa\dot{s}e_x \cos\theta_S + \kappa\dot{s}e_z \sin\theta_S + V\cos\theta \sin e_\psi\right) \\ & + V\frac{e_y \cos\theta \sin e_\psi + e_y \sin\delta(e_y) + e_z \cos\theta \sin\theta_S \cos e_\psi}{e_\psi - \delta(e_y)} \end{aligned} \tag{5.41}$$

and substituting the following kinematic control law

$$\dot{s} = (\alpha + k_x e_x) \tag{5.42}$$

$$\theta = \sin^{-1}\left(\frac{k_z e_z}{V\cos\theta_S}\right) \tag{5.43}$$

$$\phi = \tan^{-1}\left(\frac{V}{g}\left(-\beta - k_y\left(e_\psi - \delta(e_y)\right)\right)\right) \tag{5.44}$$

where k_x, k_y, k_z are positive real numbers, in (5.39), yields

$$\dot{V} = -k_x e_x^2 - k_z e_z^2 - k_y\left(e_\psi - \delta(e_y)\right)^2 + V e_y \sin\left(\delta(e_y)\right) \leq 0 \tag{5.45}$$

To conclude convergence of the states e_x, e_y, e_z, e_ψ to zero we use the LaSalle's invariance principle.

Theorem 2 LaSalle's Theorem

Let $\mathcal{O}$ be a positively invariant set of system (5.35). Let $\Omega \subset \mathcal{O}$ a set in which every solution which starts in Ω remains in Ω. Furthermore, let $\mathcal{M}$ be the largest invariant set contained in Ω. Then, as $t \to \infty$, every bounded solution starting in $\mathcal{O}$ converges to $\mathcal{M}$.

Proof: Convergence of the states (e_x, e_x, e_z, e_ψ) to zero.

The proof relies on Theorem 2. Consider the system (5.35) and the radially unbounded Lyapunov function candidate (5.36). Let us define the compact set $\mathcal{O}$ as $\mathcal{O} = \{V(e_x, e_y, e_z, e_\psi) \leq a\}$, where $a \in \Re^+$. Define the set Ω as

$$\Omega = \{[e_x \;\; e_y \;\; e_z \;\; e_\psi]^T \in \mathcal{O} \;\; : \;\; \dot{V}(e_x, e_y, e_z, e_\psi) = 0\} \tag{5.46}$$

Equivalently, the expression $\dot{V}(e_x, e_y, e_z, e_\psi) = 0$ means that $e_x = e_y = e_z = 0$ and $e_\psi = \delta(e_y)$.

Since δ is a function of the error e_d, it is easy to verify that any point starting from Ω is an invariant set. Hence, by LaSalle Theorem, every trajectory starting in $\mathcal{O}$ converges to 0 as $t \to \infty$, and the following limits are true

- $\lim_{t\to\infty} e_x = 0$
- $\lim_{t\to\infty} e_y = 0$
- $\lim_{t\to\infty} e_z = 0$
- $\lim_{t\to\infty} e_\psi = 0$

5.4 Simulation

Simulations were done on the complete simulation platform, the MAV3DSim, fully described in chapter 3.1. Figure 5-10 shows the MAV3DSim graphic user interface.

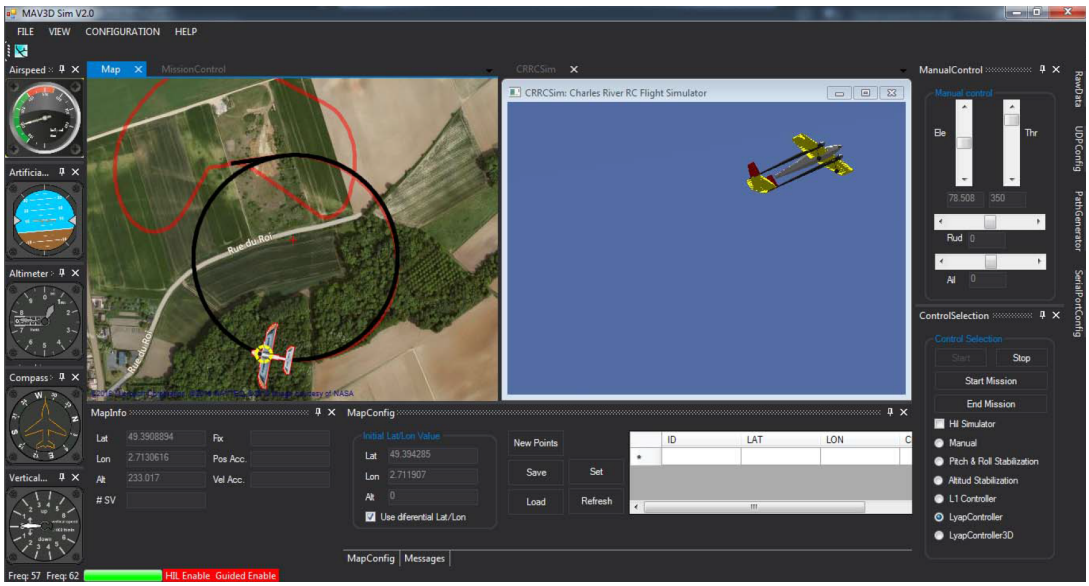


Figure 5-10: MAV3DSim simulation platform.

In order to show the controller performance, we have chosen the following scenario: The 3D reference path has been chosen as

$$\begin{aligned} q_x &= R\cos(s/R) \\ q_y &= R\sin(s/R) \\ q_h &= bs/R + 200 \end{aligned} \tag{5.47}$$

where $R = 200m$ is the rotation radius, s is the archlenght of the curve, the reference path has an initial altitude of $200m$. Using (5.47) we can now compute the curvature of the path using (5.34) as

$$\kappa = \frac{|y''_S x'_S - y'_S x''_S|}{\left(x'^2_S + y'^2_S\right)^{3/2}} = \frac{1}{R} \tag{5.48}$$

The constant velocity of the UAV was set to $10m/s$. The MAV3DSim simulator provides a full set of utilities for online gain tuning, with which we arrive to the following set of parameters for the guidance controller: $k_x = 0.1, k_y = 0.05, k_z = 0.05, \psi_a = \frac{\pi}{4}, k_\delta = 0.1$. The MAV3DSim is a complete simulation environment and the simulation starts with the vehicle in land, we start the simulation and manually take-off the aircraft to reach some altitude and then activate the path-following controller to follow the reference path.

Figure 5-11 shows the evolution of the reference in red line and the actual flight path with the blue line. It can be seen that the initial position of the aircraft is different from the initial position of the path and the control laws can eliminate this initial offset and steer the UAV along the path with a smooth movement. Figure 5-12 shows the position and attitude errors which converges to zero in 10s approximately. The output controllers are shown in Figure.5-13, here we can notice that the virtual target converge to the constant velocity of the aircraft.

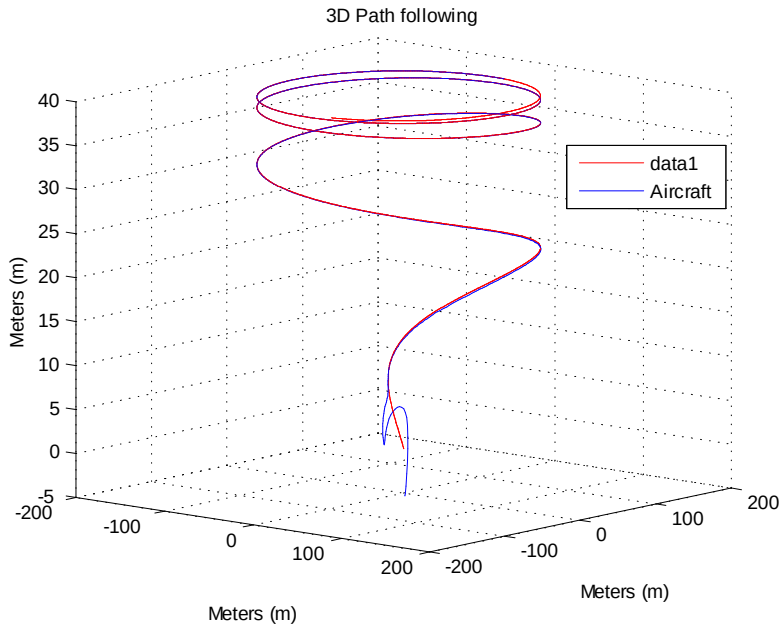


Figure 5-11: Reference and actual 3D path of the UAV.

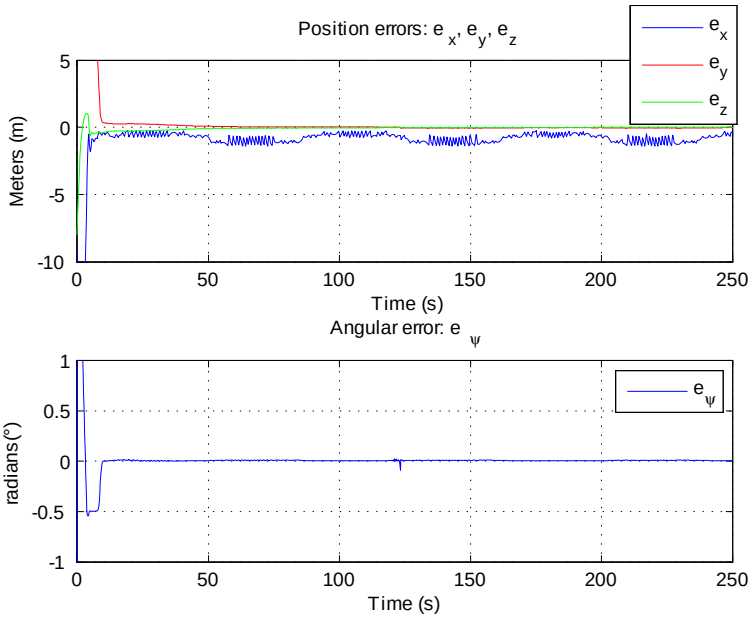


Figure 5-12: Position errors and yaw error.

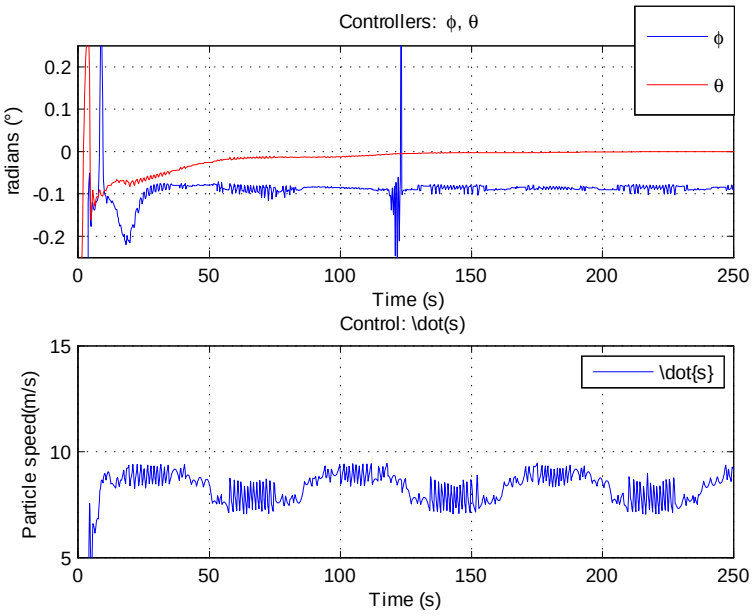


Figure 5-13: Commanded roll θ and pitch ϕ angles.

Chapter 6

Autonomous Take-off and Landing

A large number of UAVs do not have the capability of performing autonomous take-off and landings and usually a human pilot is in charge of these phases of the mission. In the landing phase there are some other technics used to recover the UAV, besides manual landing, there is also the use of a parachute[76], which is particularly suited to recover UAVs in unprepared terrain. A net-catcher is also used to recover the UAV[75], which are specially used where there is no access to conventional runways such as from the back of ships, within urban areas or in the field. There exists also a study on how to catch the fixed wing UAV with a suspended net between a group of multirotors [57]. Whenever a runway is available it is prefered to use an autonomous take-off and landing system, to minimize the human error and save time and effort in maneuvering the aircraft to a manual landing.

This chapter describes the proposed approach for solving the autonomous take-off and landing problem. Starting with a background review of the traditional take-off and landing procedure. In order to perform a take-off and landing we need to be able to control the airspeed of the vehicle. The 3D path generation described in Section 5.2 is used but extended to add a desired velocity term in the generated path. The same 3D controller is used altogether with the PID velocity controller used to regulate the velocity of the aircraft.

6.1 Traditional Takeoff and Landing

In order to solve the problem of autonomous take-off and landing, it becomes a necesity to understand the standar procedure of manual take-off and landing on a large scale aircraft, usually the procedure used by piloted aircrafs taking-off and landing in civilian airports. Typically the entire "flight" process begins with the aircraft accelerating to the runway, followed by the take-off procedure, cruising to its objective or waypoints, and finished by the landing at the destination site [67], sa depicted in Figure 6-1.

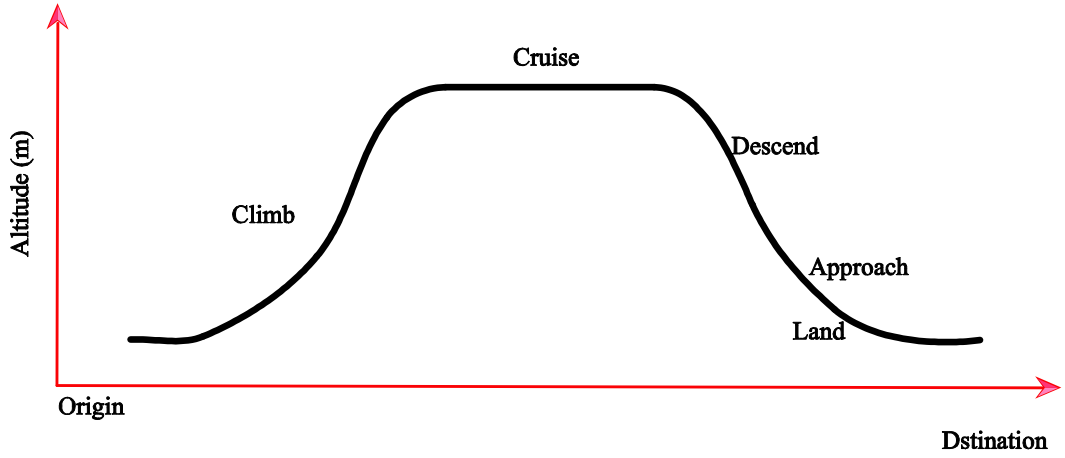


Figure 6-1: Traditional takeoff and landing

6.1.1 Traditional Aircraft Take-off and Climb Phases

The take-off phase of flight consists in the aircraft become airborne. This is done by setting the motors to full throttle to achieve take-off speed, which varies depending of air density, aircraft weight and airframe. The take-off speed is relative to the motion of the air, a head wind will reduce the ground speed needed to take-off, as there is more air flowing through the wings generating more lift for the aircraft.

After the aircraft becomes airborne, it has to climb to a certain altitude before it can cruise at this altitude in a safe and economic way. A climb is carried out by increasing the lift of wings supporting the aircraft by increasing the angle of attack of the wings, by increasing the thrust of the engines to increase speed, by increasing the surface area or shape of the wing to

produce greater lift, or by some combination of these techniques.

6.1.2 Traditional Aircraft Descent and Landing

The descent phase is when the aircraft decreases altitude. Descents are an essential component of an approach to landing. Other partial descents might be to avoid traffic, poor flight conditions , to enter warmer air, or to take advantage of wind direction of a different altitude.

Landing is the las part of a flight. The aircraft lands at an airport on a firm runway reducing its speed and descent rate. This speed reduction is accomplished by reducing thrust and generating more drag using flaps, landing gear or air brakes. Nowadays most commercial airports are equipped with an Instrument Landing System (ILS). The airport is equipped with several radio beacons placed on the runway for vertical and lateral guidance, Figure 6-2 shows the ILS indicator for vertical and lateral guidance.

Although the ILS are a good solution for the landing process, they are only found in airports and they are not available for operations wth UAVs. This chapter will focus on other alternatives for the autonomous take-off and landing process.

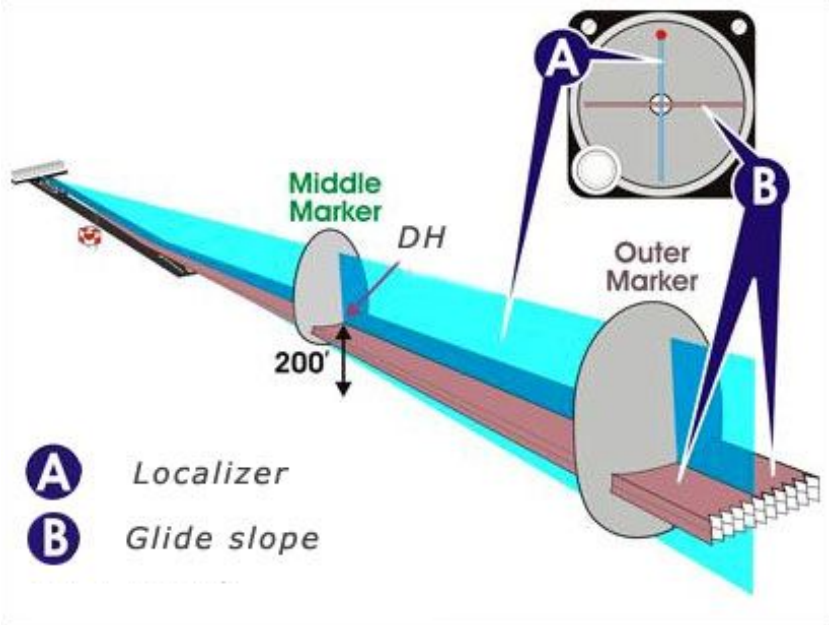


Figure 6-2: ILS guidance for landing.

6.2 4D Path Generation

The 4D path generation has been studied in [4] and [6], in both cases they generate a 4 state trajectory, a 3D point and the arrival time for that point, which gives the fourth dimension. In out problem we are no concern about the arrival time but rather the velocity of the aircraft in that specific point.

Following thee same approach as in the previous path generators, let us consider the initial and final configuration to be described as a five dimensional state vector as follows

$$\mathbf{c}_i \left(p_i, \psi_i, v_i\right) \tag{6.1}$$

$$\mathbf{c}_f \left(p_f, \psi_f, v_f\right) \tag{6.2}$$

where $p_i = [x_i, y_i, z_i]$ is the initial position of the path and $p_f \left(x_f, y_f, z_f\right)$ is the final position of the path, both are expressed in an inertial frame, the initial heading angle ψ_i and the final heading is ψ_f and v_i and v_f are the initial and final desired velocity. As in the 3D version we still have 8 possible selection of the 3D Dubins paths and thee selection is made according to table 5.1.

The angles and distances needed to compute the 4D path generation are computed as in sections 5.2.1 and 5.2. The computed variables are summarized in Table 6.1, where $x_{RR} = x_{Rf} - x_{Ri}$, $y_{RR} = y_{Rf} - y_{Ri}$, $z_{RR} = z_{Rf} - z_{Ri}$, $x_{LL} = x_{Lf} - x_{Li}$, $y_{LL} = y_{Lf} - y_{Li}$, $z_{LL} = z_{Lf} - z_{Li}$, $x_{RL} = x_{Lf} - x_{Ri}$, $y_{RL} = y_{Lf} - y_{Ri}$, $z_{RL} = z_{Lf} - z_{Ri}$, $x_{LR} = x_{Rf} - x_{Li}$, $y_{LR} = y_{Rf} - y_{Li}$, $z_{LR} = z_{Rf} - z_{Li}$.

The path Γ is parameterized by its arc length s; starting in $s = 0$ and finalizing in $s = d_T$. The coordinates of the $n - th$ point p_n of the Γ_1 and Γ_3 arc segments are obtained with the following equation

$$\begin{bmatrix} p\left(s\right) \\ v\left(s\right) \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ 0 \end{bmatrix} + \begin{bmatrix} \rho \cos\left(\lambda s + \psi\right) \\ \rho \sin\left(\lambda s + \psi\right) \\ s\rho \tan\left(\gamma\right) \\ v_i + \dfrac{d_T}{s}\left(v_f - v_i\right) \end{bmatrix} \tag{6.3}$$

where $\mathbf{c} = \mathbf{c}_i$ if it is the path Γ_1 and $\mathbf{c} = \mathbf{c}_f$ if it is the path Γ_3. λ is the direction of rotation, $\lambda = 1$ for a turn to the left and $\lambda = -1$ for a turn to the right. The angle $\psi = \psi_i$ when it is

Table 6.1: 3D Dubins path generation variables.

	$RSRU/D$	$LSLU/D$	$RSLU/SD$	$LSRU/D$
α	$\frac{\pi}{2} - \tan^{-1}\left(\frac{y_{RR}}{x_{RR}}\right)$	$\frac{\pi}{2} - \tan^{-1}\left(\frac{y_{LL}}{x_{LL}}\right)$	$\eta + \mu - \frac{\pi}{2}$	$\eta + \mu - \frac{\pi}{2}$
d	$\sqrt{x_{RR}^2 + y_{RR}^2 + z_{RR}^2}$	$\sqrt{x_{LL}^2 + y_{LL}^2 + z_{LL}^2}$	$\sqrt{l^2 + 4\rho + z_{LR}^2}$	$\sqrt{l^2 + 4\rho + z_{RL}^2}$
γ	$\tan^{-1}\left(\frac{h}{d_T}\right)$	$\tan^{-1}\left(\frac{h}{d_T}\right)$	$\tan^{-1}\left(\frac{h}{d_T}\right)$	$\tan^{-1}\left(\frac{h}{d_T}\right)$
d_1	$\lvert\psi_i - \alpha\rvert * 2\rho$	$\lvert\psi_i - \alpha\rvert * 2\rho$	$\lvert\psi_i - \alpha\rvert * 2\rho$	$\lvert\psi_i - \alpha\rvert * 2\rho$
d_2	$\lvert\alpha - \psi_f\rvert * 2\rho$	$\lvert\alpha - \psi_f\rvert * 2\rho$	$\lvert\alpha - \psi_f\rvert * 2\rho$	$\lvert\alpha - \psi_f\rvert * 2\rho$
d_T	$d + d_1 + d_2$	$d + d_1 + d_2$	$d + d_1 + d_2$	$d + d_1 + d_2$
h	$\lvert z_i - z_f\rvert$	$\lvert z_i - z_f\rvert$	$\lvert z_i - z_f\rvert$	$\lvert z_i - z_f\rvert$
η	—	—	$\frac{\pi}{2} - \tan^{-1}\left(\frac{y_{LR}}{x_{LR}}\right)$	$\frac{\pi}{2} - \tan^{-1}\left(\frac{y_{RL}}{x_{RL}}\right)$
μ	—	—	$\tan^{-1}\left(\frac{2\rho}{d}\right)$	$\tan^{-1}\left(\frac{2\rho}{d}\right)$
l	—	—	$\sqrt{x_{LR}^2 + y_{LR}^2 + z_{LR}^2}$	$\sqrt{x_{RL}^2 + y_{RL}^2 + z_{RL}^2}$

the path Γ_1 and $\psi = \psi_f$ when the path is Γ_3. v_i and v_f are the initial and final velocity.

Now the path Γ_2, the straight line path, is generated using the following equation

$$\begin{bmatrix} p(s) \\ v(s) \end{bmatrix} = \begin{bmatrix} p(s-1) \\ 0 \end{bmatrix} + \begin{bmatrix} s\cos(\psi) \\ s\sin(\psi) \\ s\tan(\gamma) \\ v_i + \frac{d_T}{s}(v_f - v_i) \end{bmatrix} \tag{6.4}$$

where $p(s-1)$ is the last point of the Γ_1 segment. This path will continue where the straight line segment to connect the two cylinders and complete the Γ path.

6.3 Take-off and Landing Trajectory

There are several phases in the design of the take-off and landing trajectory. The main idea is to design the trajectory for the full mission from take-off passing through the following waypoints and finishing in the landing. Using the 4D path generator described in the previous section, a trajectory trajectory suitable for take-off and landing purposes can be designed, and using the nonlinear 3D path following to complete the entire mission of the aircraft. In Figure 6-3 a general description of the several phases of the complete mission is depicted. There are five

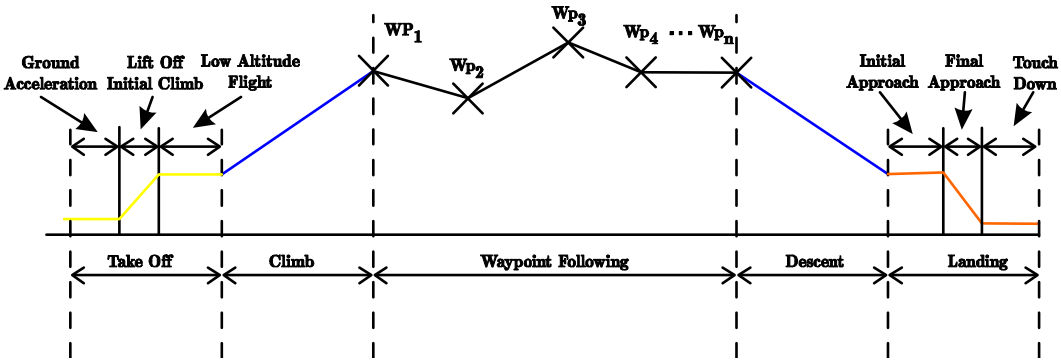


Figure 6-3: Autonomous take-off and landing phases.

main phases:

Take-off The take-off sequence begins with the aircraft landed and with zero velocity, followed by the initial climb of the aircraft and maintaining a low altitude flight.

Climb In this phase the aircraft is set a desired climb rate until it arrives to a desired altitude

Waypoint following The waypoint following is the mission pre-programmed on the aircraft and it will pass through all the waypoints before the descent and landing phase.

Descent In the descent phase the aircraft will reduce its altitude and it will line up with the landing runway.

Landing In the landing phase the aircraft is already line up with the wunway and it will reduce its ground velocity in order to have a smooth touchdown to finish the mission.

6.3.1 Take-off

A conventional runway can be divided into three main phases. The ground acceleration phase, the lift off phase and the low altitude flight phase. During the ground acceleration phase the aircraft is aligned with the centerline of the runway and accelerated until its velocity generates enough lift to guarantee a safe lift-off. Once the predefined lift-off velocity is reached the aircraft enters the initial climb phase, in which the angle of attack is increased in order to change the

climb rate and gain some altitude. The aircraft remains in the initial climb phase until a safe altitude is reached. At this point the take-off is considered complete and the normal flight of the mission can continue.

The design of the take-off trajectory consist in four 4D waypoints to use them as an input in the 4D path generator. The first waypoint is the initial position of the aircraft, which is aligned with the runway. The initial velocity is sup to 5 m/s in order to get the aircraft to move through the runway. The second waypoint is placed forward, giving sufficient space to the aircraft to gain the lift-off speed. The third waypoint is set up above the centerline of the runway with the desired altitude and maintaining the same airspeed as the previous waypoint. The final waypoint allow the aircraft to stabilize in a low altitude flight before continue with the waypoint following of the mission.

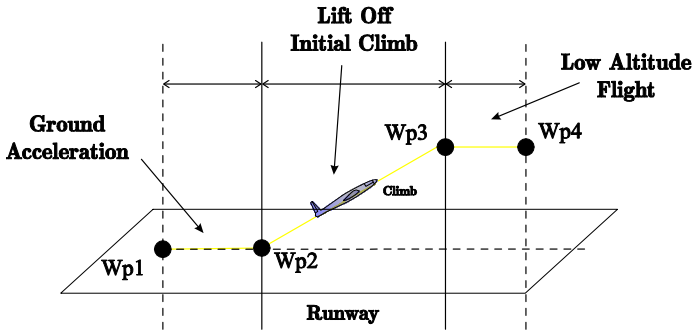


Figure 6-4: Thtake-off phase.

6.3.2 Landing

Conventional landing can be divided into three main phases, which are the initial approach, final approach and touch down. During the initial approach the aircraft is aligned with the runway and an initial descent is conducted to decrease gradually the altitude. The next phase is the final approach phase in which the aircraft performs a final descent and also the airspeed is reduced to enter in a flare descent. Finally the aircraft enter the touch down phase in which the aircraft touches the ground and ceases all motion in the main motor.

In order to design the landing trajectory suited to use as input in the 4D path generation a total of seven 4D waypoints are used. The first waypoint is for the aircraft to approach from

its current position to the runway. The second waypoint along with the first waypoint align the aircraft with the runway, a decrease in the airspeed is also conducted in this phase. The third waypoint performs a decrease in altitude and finish the initial approach phase with the fourth waypoint which generates a stabilization in altitude for the aircraft. The final approach is with the waypoint 5, which defines a slow descent rate, which is called flare, and gently lands the aircraft by maintaining the airspeed of the aircraft just above the stall speed. Once the aircraft is landed in the touch down phase it will stop the engine to finish all motion of the aircraft.

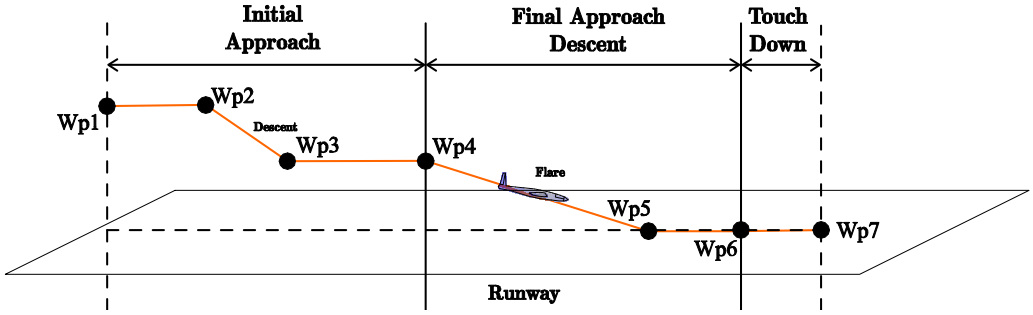


Figure 6-5: The landing trajectory.

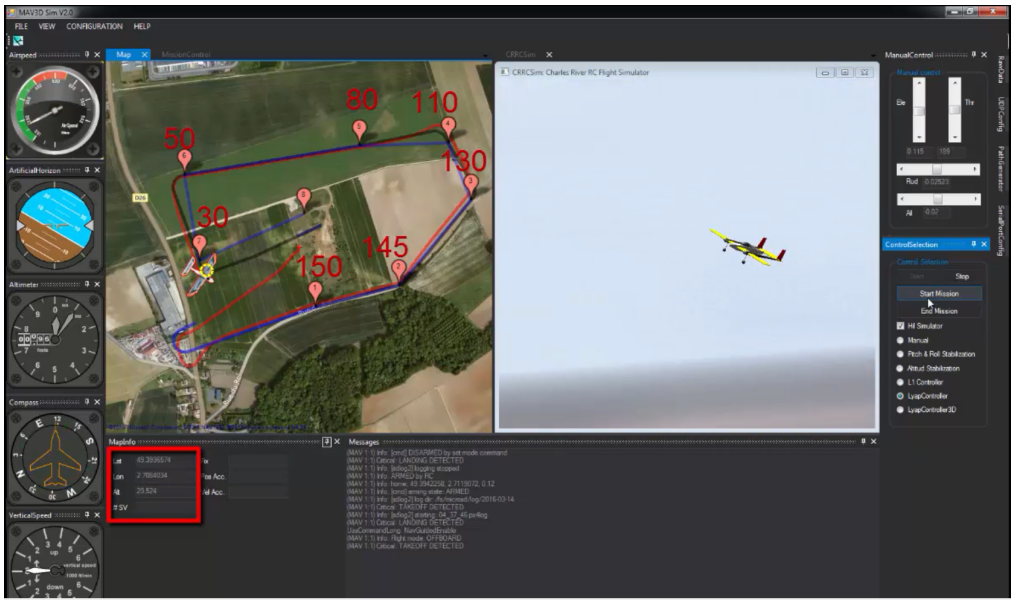
6.4 Simulation Results

Simulations were done on the complete simulation platform, the MAV3DSim, fully described in chapter 3.1. Figure 5-10 shows the MAV3DSim graphic user interface. The companion computer is pre-loaded with the desired 3D waypoints and the desired velocity at the velocity.

The simulation starts with the aircraft in the ground, aiming to the take-off runway, from this initial point and hading the take-off sequence is designed by the 4D path generation described in 6.2. All the waypoints are generated with the assumption that the aircraft is carefully placed and aiming the runway. Once the take-off sequence is completed it will continue with the path following, the same 4D path generation is used to generate the 4D path to pass through all the desired waypoints at the desired speed. The last waypoint is took as a landing waypoint. The landing sequence designed in previous section is generated with the last waypoint to be the touch-down waypoint. Figure 6-6 shows th trajectory generated by the 4D path generator

in red and the aircraft trajectory is in blue. The commanded altitude in red and the current aircraft's altitude in blue are shown in Figure 6-7. In Figure 6-8 the commanded airspeed in red is compared with the aircraft airspeed in blue. Finally the errors in longitude/latitude/altitude format and airspeed are depicted in Figure 6-9

The overall performance of the 4D path generation and the 3D path following strategy augmented by the airspeed PID controller is good enough to solve the take-off and landing problem. A complete video of the simulation can be found in https://youtu.be/oAkRx2iSe6I



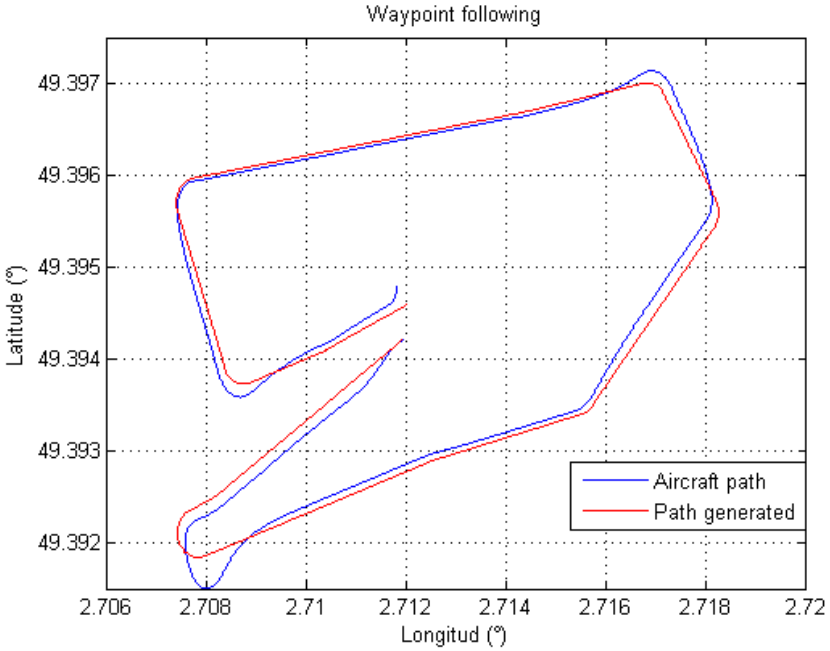


Figure 6-6: The trayectory generated by the 4D path generator and the aircraft.

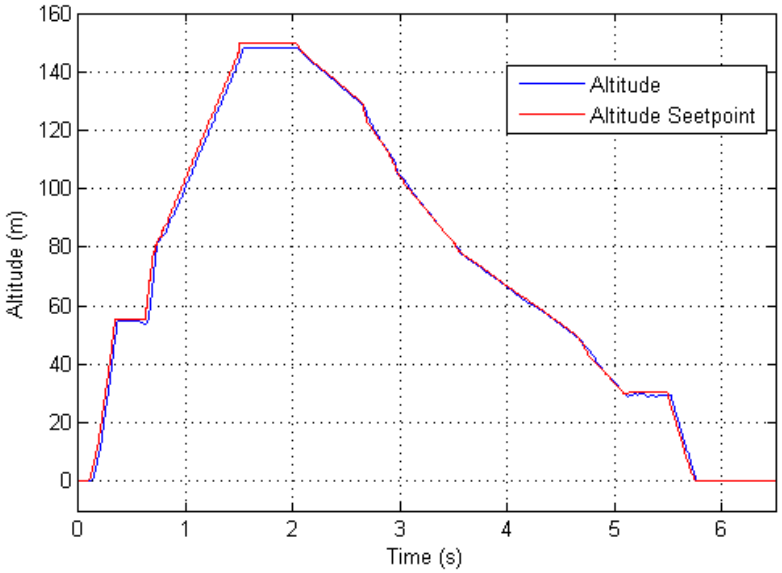


Figure 6-7: The altitude setpoint vs currente aircraaft altitude.

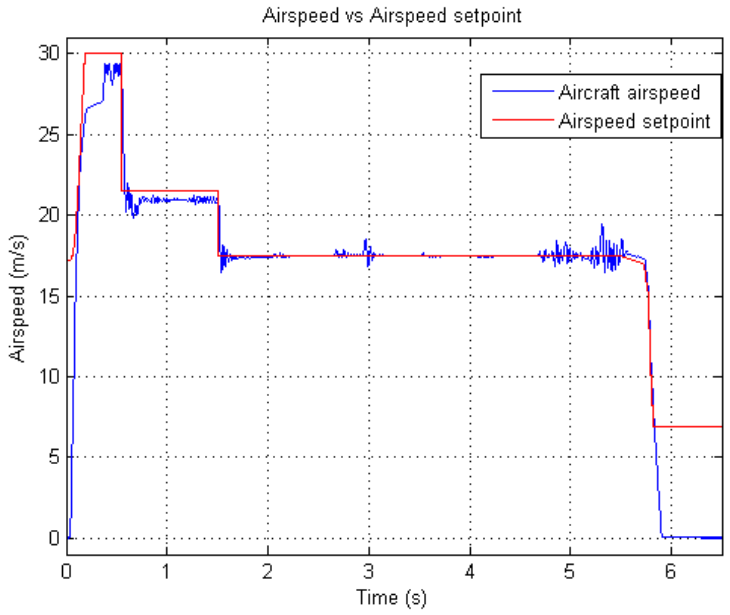


Figure 6-8: Commanded airspeed vs aircraft's airspeed

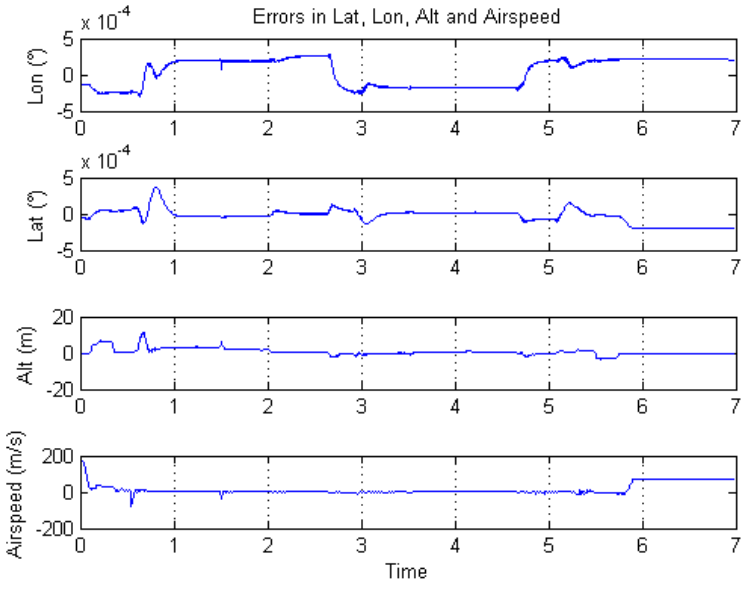


Figure 6-9: 3D position errors and airspeed error

Chapter 7

Conclusions and Future Work

In order to achieve that the UAV performs a complete mission, starting from the autonomous take-off and followed by the pass-through all the predefined waypoints and finishing with the autonomous landing, several research areas were studied, such as modeling, control, path following strategies, advanced simulation environments and embedded systems. In this chapter the concluding remarks and future development are discussed.

7.1 Simulation and Experimental Platform

A simulation platform is a powerful and necessary tool for the development and validation of different controllers. Here we presented a Hardware-in-the-Loop (HIL) simulator: The MAV3DSim that has proven to be a great candidate for the validation of different controllers on different UAV models Along with the MAV3DSim simulator, the experimental platform was presented for validating the hardware in the loop implementation by performing the same waypoint following experiment in both platforms using the same Pixhawk autopilot hardware.

In the future we will extend the MAV3DSim simulator for other type of aerial vehicles such as coaxial helicopters or hybrid configurations (airplane-quadrotor) and the MAV3DSim will serve as a test bed for validation of new controllers. It is important to mention that this software is not intended as an end user application but it could be used for professors as a test bed for students to try new controllers, also as the MAV3DSim has been developed using software from the open-source community, the source code of the simulator will be available in the Github

account of the project (https://github.com/mav3dsim) once it reaches a stable version.

7.2 Path Generation

The presented study propose a framework for the path generation needed in order to complete a predefined task without human interaction. Dubins paths have been utilized as a tool to estimate the shortest path from the current aircraft position and orientation to a| given point provided by the user in the form of waypoints. Starting from the 2D case in which only position in 2D and orientation is considered, using a similar approach it was extended to the 3D case, adding the altitude of the aircraft to the path generation. In order to achieve the autonomous take-off and landing the 4D path generation was needed. This case includes the velocity of the aircraft at the arrival point, not the arrival time which is commonly used in the 4D path generation.

The Dubins paths present a discontinuity in the thesis function of the curve. This discontinuity create the sense that the aircraft can change its turn rate instantaneously, which is obviously false. Further improvement involve the use of continuous curvature curves such as the clothoid path, which has the property that its curvature varies linearly over the path length. Another approach is to use the Pythagorean hodograph curve which are defined by polynomial curves which have hodographs that satisfy a Pythagorean condition which provides a continuous curvature.

7.3 Path following

In this thesis a nonlinear path-following kinematic controller for the fixed-wing aircraft based on a Lyapunov function candidate is presented. The controller performance was tested in the MAV3DSim simulation environment which has been proven to be an excellent test bed for UAV controllers development. The controller was designed using a kinematic model of the aircraft, but it was tested on a full 6DoF simulation environment with good performance. The error space dynamics presented in this paper can be used with other types of controller to follow the virtual target. The hardware in the loop simulation is an intermediate step between the pure simulation and the implementation on the experimental platform, and the next step is

to implement this controller on a fixed-wing UAV experimental platform. Using the 3D path following presented in this paper we can design another level of autonomy and create a path generator to design new trajectories, this can also be used to perform an automatic take-off and landing on the UAV.

Future work will addressed the inclusion of the aircrafts velocity in the kinematic model in order to obtain a complete 4D path following strategy designing all controllers at once. This will be tested on the simulation platform as well as with the experimental platform.

7.4 Autonomous Take-off and Landing

The scope of this theses was to present a navigation and flight control system designed to perform a fully autonomous operation of a small aircraft, including autonomous take-off and landing as well as passing through predefined waypoints. The proposed solution relies on a path-following and velocity tracking controller synthesized using a simplified aircraft kinematic model. The technique presented relies on a new error space that naturally describes the particular dynamic characteristics of the aircraft over a suitable flight path. The effectiveness of the new control law was fully tested in a simulation environment with the full nonlinear aircraft model, using the MAV3DSim. The quality of the results obtained clearly indicates that the methodology derived is suitable for the proposed application.

There is still room for improvement as it was mentioned in the previous sections, all of the components have future work in each of their areas. First of all it is necessary to take the step to test the strategy in the experimental platform, then develop the different improvements of the all the components that make possible the autonomous take-off and landing of the unmanned aircraft.

Bibliography

[1] G. Ambrosino, M. Ariola, U. Ciniglio, F. Corraro, E. D. Lellis, and A. Pironti, "Path generation and tracking in 3-d for uavs," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, p. 980–988, 2009.

[2] R. G. Arrabito, A. L. G. Ho, M. Rutley, J. Keillor, A. Chiu, H. Au, and M. hou, "Human factors issues for controlling uninhabited aerial vehicles," Canada,Toronto, August 2010.

[3] R. W. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed-wing uavs," *Journal of Aer Computing, Information, and Comunication*, vol. 2, no. 1, pp. 92–108, 2005.

[4] K. Bousson and P. Machado, "4d trajectory generation and tracking for waypoint-based aerial navigation," *Transactions on Systems and Control*, vol. 8, no. 3, 2013.

[5] A. Brezoescu, R.Lozano, and P.Castillo, "Lyapunov-based trajectory tracking controller for a fixed-wing unmanned aerial vehicle in the presence of wind," *International Journal of Adaptive Control and Signal Processing*, vol. 19, no. 3, pp. 372–384, 2015.

[6] M. Butt, K. Munawar, U. I. Bhatti, S. Iqbal, U. M. Al-Saggaf, and W. Ochieng, "4d trajectory generation for guidance module of a uav for a gate-to-gate flight in presence of turbulence," *International Journal of Advanced Robotic Systems*, vol. 13, no. 125, 2016.

[7] S. Y. V. Chingiz Hajiyev, "Lqr controller with kalman estimator applied to uav longitudinal dynamics," *Journal of Guidance, Control, and Dynamics*, vol. 4, pp. 36–41, 2013.

[8] H. Chitsaz and S. LaValle, "Time-optimal paths for a dubins airplane." in *Proc. IEEE Conference on Decision and Control (CDC'2007)*, New Orleans, LA, USA, Dec. 2007, pp. 2379–2384.

[9] V. Cichella, I. Kaminer, V. Dobrokhodov, E. Xargay, N. Hovakimyan, and A. Pascoal, "Geometric 3d path-following control for a fixed-wing uav on so(3)," in *AIAA Guidance, Navigation, and Control Conference*, Portland,Oregon, August 2011.

[10] Cloud Cap Technology, "Cloud cap technology, piccolo ii highly integrated uas autopilot," Jan. 2017. [Online]. Available: http://www.cloudcaptech.com/products/detail/piccolo-ii

[11] M. Czarnomski, R. Spitsberg, D. Dvora, R. Schultz, and W. Semke., "Benefits of autopilot integration for enhanced uas operations," in *In AIAA Infotech@Aerospace Conference*, Seattle, Washington, 2009.

[12] A. Das, F. Lewis, and K. Subbarao, "Backstepping approach for controlling a quadrotor using lagrange form dynamics," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1, pp. 127–151, 2009.

[13] P. J.-C. De Carufel J, Martin E, "Control strategies for hardware-in-theloop simulation of flexible space robots," *IEE Proc Control Theory Apply*, p. 569–579, 2000.

[14] P. Dev-Team, "Pixhawk," 2016. [Online]. Available: https://pixhawk.org/choice

[15] ——, "Px4 software stack," 2016. [Online]. Available: http://px4.io/technology/px4-software-stack/

[16] J. Deyst, J. How, and S. Park, "Lyapunov stability of a nonlinear guidance law for uavs," August 2005.

[17] V. Dobrokhodov, O. Yakimenko, K. Jones, I. Kaminer, E. Bourakov, and I. Kitsios, "New generation of rapid flight test prototyping system for small unmanned air vehicles," in *In Proceedings of the 25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, Portland, OR, Aug. 2006.

[18] DroneDeploy, "Dronedeploy, the complete mapping experience," Jan. 2017. [Online]. Available: https://www.dronedeploy.com/

[19] B. Etkin and L. D. Reid, *Dynamics of Flight: Stability and Control.* New York: John Wiley & Sons, 1998.

[20] L. Euler, *Novi Commentarii academiae scientiarum Petropolitanae 20.*

[21] F.A.A., *Pilot's handbook of aeronautical knowledge.* U.S.A.: United States Department of Transportation, 2016.

[22] J. Farrel and M. Barth, *The Global Positioning System and Inertial Navigation.* Hoboken, NJ, USA: John Wiley and Sons, 2007.

[23] J. Fleming, *Magnets and Electric Currents.* London: 2nd Edition. London: E.&F. N. Spon., 1902.

[24] E. W. Frew, D. A. Lawrence, C. Dixon, J. Elston, and W. J. Pisano, "Vector field path following for small unmanned air vehicles„" June 2006.

[25] ——, "Lyapunovguidancevector fields forunmannedaircraftapplications," July 2007.

[26] F. Gavilan, R. Vazquez, and E. CAMACHO, "An iterative model predictive control algorithm for uav guidance," *International Journal on Smart Sensing and Intelligent Systems*, vol. 51, no. 3, 2015.

[27] S. Griffiths, "Vector field approach for curved path following for miniature aerial vehicles," August 2006.

[28] H. H., "Hardware-in-the-loop simulation as a standard approach for the development, customization, and production test of ecu's," *Society of Automotive Engineers*, 1993.

[29] S. P. I. Rhee and C.-K. Ryoo, "A tight path following algorithm of an uas based on pid control," August 2010.

[30] J. W. A. C. I.D. Cowling, O.A. Yakimenko, "A prototype of an autonomous controller for a quadrotor uav," 2007.

[31] G. R. I.E. Putro, G.B. Kim and K. Yoon, "Real-time simulation of autonomous quadrotor," *proceedings of International Micro Air Vehicle Conference and Flight Competition, (IMAV2010)*, July 6-9 2010.

[32] S. S. Isermann R, Schaffnit J, "Hardware-in-the-loop simulation for the design and testing of engine-control systems," *Control Engineering Practice*, p. 643–653, May 1999.

[33] R. L. Israel Lugo-Cárdenas, Gerardo Flores, "The mav3dsim: A simulation platform for research, education and validation of uav controllers," *Proceedings of the 19th IFAC World Congress, IFAC2014*, pp. 713–717, 24-29 August 2014.

[34] S. S. J. M. Eklund, J. Sprinkle, "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit/evasion games on a fixed wing aircraft," 2005.

[35] J.A.Reeds and R.A.Sheep, "Optimal paths for a car that goes both fordward and backward," *Pacific Journal of Mathematics*, vol. 2, pp. 367–393, 1990.

[36] F. S. Johnson E, "Use of flight simulation to complement flight testing of low-cost uav's," 2001.

[37] R. T. J.W. Roberts, R. Cory, "On the controllability of fixed-wing perching," 2009.

[38] A. Kahn, "Adaptive control for small fixed-wing unmanned air vehicles," August 2010.

[39] I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre, "Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control," *Journal Guidance Contorl, and Dynamics*, vol. 21, no. 1, p. 29–38.

[40] Y. Kang and J. K. Hedrick, "Linear tracking for a fixed-wing uav using nonlinear model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1202 – 1210, 2009.

[41] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE T. Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[42] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Decentralized robust receding horizon control for multi-vehicle guidance," in *American Control Conference (ACC)*, Minneapolis, MN, June 2006, pp. 2047–2052.

[43] L. Lapierre and D. Soetanto, "Nonlinear path-following control of an auv," *Ocean Engineering*, vol. 11–12, no. 34, p. 1734–1744, 2007.

[44] L.E.Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, jul 1957.

[45] H. I. Leong, S. Keshmiri, and R. Jager, "Evaluation of a cots autopilot and avionics system for uavs," in *In AIAA Infotech@Aerospace Conference*, Seattle, Washington, 2009.

[46] N. Li, H. Liu, E. Earon, C. Fulford, R. Huq, and C.-A. Rabbath, "Multiple uavs autonomous mission implementation on cots autopilots and experimental results," in *In AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, Aug. 2009.

[47] Y. LI, C. Chen, and W. Chen, "Research on longitudinal control algorithm for flying wing uav based on lqr technology," *International Journal on Smart Sensing and Intelligent Systems*, vol. 6, no. 5, 2013.

[48] Y. Liang, Y. Jia, Z. Wang, and F. Matsuno, "Combined vector field approach for planar curved path following with fixed-wing uavs," July 2015.

[49] R. P. Liu Y, Steurer M, "A novel approach to power quality assessment: real time hardware-in-the-loop test bed," *IEEE Transactions on Power Delivery*, pp. 1200 – 1201, Apr. 2005.

[50] R. Maine and K. Iliff, "Application of parameter estimation to aircraft stability and control," NASA, Tech. Rep. 1168, 1986.

[51] D. Maroney, R. Bolling, R. Athale, and A. Christiansen, "Experimentally scoping the range of uas sense and avoid capability," in *In AIAA Infotech@Aerospace Conference*, Rohnert Park, California, May 2007.

[52] L. Martin, "Kestrel flight systems," Jan. 2017. [Online]. Available: http://www.lockheedmartin.com/us/products/procerus/kestrel-autopilot.html

[53] R. E. McFarland, "A standard kinematic model for flight simulation at nasa-ames," *National Aeronautics and Space Administration*, January 1975.

[54] T. Mehta and M. Egerstedt, "An optimal control approach to mode generation in hybrid systems." *Nonlinear Analysis: Theory, Methods and Applications*, vol. 65, no. 5, pp. 963–983, Sept 2006.

[55] A. Micaelli and C. Samson, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," INRIA, Tech. Rep. 2097, 1993.

[56] MicroPilot, "Micropilot - products - mp2128g," Jan. 2017. [Online]. Available: https://www.micropilot.com/products-mp2128g.htm

[57] J. B. Moe, *(Thesis)Autonomous landing of Fixed-Wing UAV in net suspended by Multirotor UAVs*. Norway: Norwegian University of Science and Technology, Department of Engineering Cybernetics, 2016.

[58] J. Morales, J. Martinez, M. Martinez, and A. Mandow, "Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2d laser scanner," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, no. 3, pp. 1–10, 2009.

[59] National Geospatial-Intelligence Agency, "World geodetic system 1984," Tech. Rep., 1984.

[60] D. R. Nelson, D. B. Barber, T. McLain, and R. Beard, "Vector field path-following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, p. 519–529, 2007.

[61] A. Neto and M. Campos, "A path planning algorithm for uavs with limited climb angle," in *2009 International Conference on Intelligent Robots and Systems (IROS)*.

[62] M. Niculescu, "Lateral track control law for aerosonde uav," January 2001.

[63] S. Park, J. Deysty, and J. Howz, "Performance and lyapunov stability of a nonlinear path-following guidance method," *Journal of Guidance, Control and Dynamics*, vol. 30, no. 6, p. 1718, 1957.

[64] A. Ratnoo, S. Hayoun, A. Granotz, and T. Shima, "Path following using trajectory shaping guidance," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 1, p. 106–116, 2015.

[65] J. H. S. Park, J. Deysty, "A new nonlinear guidance logic for trajectory tracking," 2004.

[66] R. Sampaio, M. Becker, and A. Siqueira, "Fvms: A novel sil approach on the evaluation of controllers for autonomous mav." *IEEEAC*, 2013.

[67] A. Sigworth, J. Davila, T. McNeil, J. Nicolalde, D. Scheve, and K. Funk, "Mission analysis, covell avionics," Jan. 2017. [Online]. Available: http://flightdeck.ie.orst.edu/ElectronicChecklist/HTML/mission.html

[68] B. Stevens and F. Lewis, *Aircraft Control and Simulation*. John Wiley and Sons Inc, 1992.

[69] D. Stojcsics, "Autonomous waypoint-based guidance methods for small size unmanned aerial vehicles," *Acta Polytechnica Hungarica*, vol. 11, no. 10, pp. 215–233, 2014.

[70] P. D. Team, "Mavlink micro air vehicle communication protocol," 2016. [Online]. Available: http://qgroundcontrol.org/mavlink/start

[71] The Botlink, "Botlink, a cloud-based drone," Jan. 2017. [Online]. Available: http://botlink.io/

[72] The UAS Europe, "The uas europe, agriculture, research and surveillance drones," Jan. 2017. [Online]. Available: http://www.uas-europe.se/

[73] D. Uhlig, K. Bhamidipati, and N. Neogi, "Safety and reliability within uav construction," in *In AIAA Modeling and Simulation Technologies Conference and Exhibit*, Hilton Head, South Carolina, Aug. 2007.

[74] J. S. Wit, "Vector pursuit path tracking for autonomous ground vehicles," Gainesville,Fl, August 2000.

[75] W.J.Crowther, "Perched landing and takeoff for fixed wing uavs," October 2000.

[76] T. Wyllie, *Parachute recovery for UAV systems*. New York: Aircraft Engineering and Aerospace Technology, 2001.

[77] S. Zhao, X. Wang, D. Zhang, and L. Shen, "Curved path following control for fixed-wing unmanned aerial vehicles with control constraint," *Journal of Intelligent & Robotic Systems*, no. 34, p. 1–13, 2017.