NTNU
Norwegian University of
Science and Technology

Autonomous Landing of Fixed-Wing UAV in Net suspended by Multirotor UAVs

A Fixed-Wing Landing System

Sigurd Olav Nevstad

Master of Science in Cybernetics and Robotics	
Submission date:	June 2016
Supervisor:	Tor Arne Johansen, ITK
Co-supervisor:	Kristoffer Gryte, ITK
	Kristian Klausen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Sigurd Olav Nevstad
Department:	Engineering Cybernetics
Thesis title:	Autonomous Landing of Fixed-Wing UAV in Net suspended by Multirotor UAVs - A Fixed-Wing Landing System

Thesis Description: The purpose of the thesis is to develop an autonomous landing system for a fixed-wing UAV in order to land in a net suspended by multirotor UAVs.

The following items should be considered:

1. Define the scope of the thesis and clarify what your contributions are.
2. Develop a guidance and control system with the Pixhawk autopilot software APM:Plane and DUNE for autonomous landing in a recovery net. Tune the control system for the X8 fixed-wing UAV.
3. Implement and design a system to coordinate a landing path between the fixed-wing UAV and the multirotors UAVs.
4. Test the system in software-in-the-loop (SITL) simulations.
5. Experimental testing of the system using the X8 fixed-wing UAV.
6. Conclude your results.

Start date:	2016-01-11
Due date:	2016-06-20

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Tor Arne Johansen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor:	Kristoffer Gryte, Dept. of Eng. Cybernetics, NTNU
	Kristian Klausen, Dept. of Eng. Cybernetics, NTNU

Abstract

This thesis presents an autonomous landing system designed for an fixed-wing unmanned aerial vehicle (UAV) to land in a net suspended by multirotor UAVs. Ordinary landing of an fixed-wing UAV usually involves landing on a long runway, but by landing in a net suspended below multirotor UAVs, operations with fixed-wing UAV can be performed from ships or other platforms with confined space where no runway is available. The focus in this thesis is placed on the fixed-wing UAV part of the landing, and involves design and implementation of a guidance and control system, together with a system for coordinating a landing path between the fixed-wing UAV and the multirotor UAVs.

The autonomous landing system uses the Pixhawk autopilot for low-level control, and a software toolchain from Underwater Systems and Technology Laboratory (LSTS), Porto, for high-level guidance and control, mission review and communication. The guidance and control system developed in this thesis is implemented in the LSTS toolchain, and consist of a decoupled lateral and longitudinal line-of-sight (LOS) guidance scheme for pitch and roll angle commands, and a speed controller for throttle command.

The system is tested both in software-in-the-loop (SITL) simulations and flight experiments at Agdenes airfield, where successful tests demonstrated both the feasibility and the good performance of the system using a Skywalker X8 fixed-wing UAV.

Sammendrag

(Norwegian Translation of the Abstract)

Denne avhandlingen presenterer et autonomt landingssystem utviklet for et ubemannet fly (drone) til bruk ved landing i et nett hengede under ubemannede multirotor droner. Ordinær landing av ubemannet fly innbærer ofte bruk av lange rullebaner, men ved å lande i ett nett som henger under ubemannede multirotor droner, kan operasjoner med ubemannede fly bli utført fra skip eller andre platformer med begrenset plass hvor man ikke har en rullebane tilgjenlig. Fokuset i denne avhandlingen er på det ubamennde flyet sin rolle i landingen, og innebærer design og implementering av gaidings- og kontrollsystem, sammen med et system som koordinerer landingsbanen mellom det ubemannede flyet og multirotor dronene.

Det autonome landingssystemet bruker Pixhawk autopilot for lav-nivå kontroll, og en programvarekjede fra Underwater Systems and Technology Laboratory (LSTS), Porto, for høy-nivå gaiding og kontroll, oppdragsanalyse og kommunikasjon. Gaidings og kontrollsystemet som er utviklet i denne avhandlingen er implemenert i LSTS programvarekjeden, og består av en dekoblet lateral og longitudinal siktelinjemetode for pitch og rull vinkel kommandoer, og en hastighetsregulator for for gasspådrag-kommando.

Systemet er testet både i programvare-i-sløyfen (SITL) simuleringer og fly eksperimenter ved Agdenes flyplass, hvor vellykkede tester demonstrerte både gjennomførbarheten og den gode ytelsen til systemet ved bruk av det ubemannede flyet Skywalker X8.

Preface

This thesis is submitted in partial fulfillment of the requirements for the Master of Science degree at the Norwegian University of Science and Technology.

I would like to thank my supervisor Tor Arne Johansen for giving me the opportunity to write my thesis on such an interesting subject as UAV guidance and control, and for supervising the thesis. A huge thanks goes to my co-supervisors, Kristoffer Gryte og Kristian Klausen, for outstanding support, assistance and guidance. I would also like to thank Jostein Borgen Moe and Kjetil Sørbø that also wrote their thesis on the subject of autonomous landing.

Finally, I would like to thank my family for their never-ending support.

Sigurd Olav Nevstad
Trondheim, June 2016

Table of Contents

List of Tables

List of Figures

Abbreviations

UAV	Unmanned aerial vehicle
GPS	Global Positioning System
SITL	Software-in-the-loop
FBWB	Fly-by-wire A
FBWA	Fly-by-wire B
NED	North East Down
GCS	Ground Control Station
DOF	Degrees-of-freedom
TECS	Total Energy Control System
PID	Proportional-Integral-Derivative
C2	Command and Control
IMC	Inter-Module Communication
DUNE	DUNE: Unified Navigation Environment
LOS	Line-of-Sight
IMC	Inter-Module Communication protocol
LQR	Linear Quadratic Controller

Nomenclature

Elevator	Control surface used primarily to control the pitch.
Aileron	Control surfaces used primarily to control the roll
Rudder	Control surface primarily used to control yaw
Elevons	Combined control surface for elevator and aileron
APM:Plane	APM autopilot software for fixed-wing UAVs
X8	A type of fixed-wing UAV developed by Skywalker technology.
Neptus	Ground control station software in the LSTS toolchain.
Loiter	Plane will circle around a defined point holding altitude
V_a	Airspeeed of the UAV
S	Surface area of the wing
b	Wingspan
C_{p_p}	Aerodynamic moment coefficient along the body frame
$C_{p_{\delta_a}}$	Aerodynamic moment coefficient along the body frame
C_{m_q}	Aerodynamic pitching moment coefficient
C_{m_α}	Aerodynamic pitching moment coefficient
$C_{m_{\delta_e}}$	Aerodynamic pitching moment coefficient
c	Mean aerodynamic chord of the wing
J_y	A element of the inertia matrix
α_0	Angle of attack at steady state.

Body frame variables	Forces and moments	Linear and angular velocities	Positions and Euler angles
Motions in the x direction	F_x	u	x
Motions in the y direction	F_y	v	y
Motions in the z direction	F_z	w	z
Rotation about the x axis(roll)	l	p	ϕ
Rotation about the y axis(pitch)	m	q	θ
Rotation about the z axis(yaw)	n	r	ψ

Part I

Introduction and Background Material

Chapter 1

Introduction

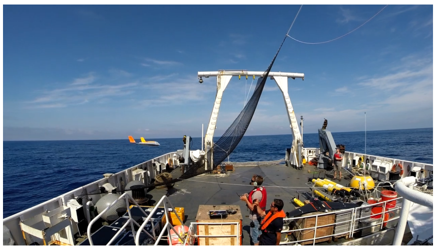
1.1 Background and Motivation

Unmanned aerial vehicles (UAVs) have in the past been used mainly for military purposes, but with the introduction of affordable equipment in the recent years, the number of civil and scientific applications is growing significantly each year. Applications such as aerial photography, powerline inspection and environmental monitoring are just a few examples of applications where UAVs have already been used with great results.

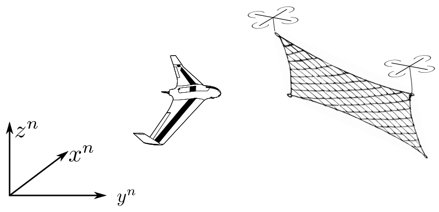
One of the most critical phases of an UAV operation, regardless of application, is the landing. One of the ultimate goals in UAV development is to make the autopilot system so autonomous that it replaces the human pilot, and where the UAV can perform both takeoff and landing entirely on its own.

An important field of research where the landing is crucial, is marine operations with fixed-wing UAVs, a field which enables applications such as sea-ice surveillance, monitoring of traffic and oil spill detection. These operations often involve take-off and landing from ships, using catapults for launching and large nets attached at the end of the ship for recovery. However, there are several challenges with this approach. In order to successfully land the UAV autonomously a high-precision guidance, navigation, and control (GNC) system is required. Furthermore, with the environmental forces like wind, waves and current affecting the ship motions, especially the wave-induced heave motion, creating robust UAV autopilots for automatic landing becomes very challenging.

At the UAV-lab at the Department of Engineering Cybernetics an alternative solution to landing on a ship is under development (Klausen et al., 2016). Here the landing-net is instead suspended below two coordinated multirotors. The landing can then be done at a safe distance away from the ship, and without the ship motions affecting the landing. The focus in this thesis is placed on the fixed-wing UAV part of this landing, developing an autonomous landing system with high precision control and coordination of the landing path between the fixed-wing UAV and the multirotors.



(a) Net recovery of fixed-wing UAV on ship.



(b) Net recovery of fixed-wing UAV in net suspended by multirotor UAVs.

Figure 1.1: Net recovery of fixed-wing UAV on ship and with multirotor UAVs.

1.2 Previous Work

In the literature both UAV control and approaches to automatic landing is well covered. Most of the textbooks that covers UAV control presents linear state-space models for lateral and longitudinal motion, obtained by linearizing the full UAV dynamics around trim conditions (Etkin and Reid, 1996; Stevens and Lewis, 2003).

The state-space models can then be used in flight control design using different state-space control design methods. One method that can be used is the linear quadratic regulator (LQR), which is one of the most widely used state-space control design method in aerospace (Lavretsky and Wise, 2013). The method have excellent performance, robustness and minimize the control usage. However, the main challenge with this method, and other state-space methods is determining the systems dynamics of the UAV with sufficient accuracy to ensure that the controllers are robust and work well practice. This often requires extensive modeling, wind tunnel testing and system identification (How et al., 2015). When the the system dynamics are either unknown or poorly modeled, the classical PID controller is considered a reasonable choice, and is therefore often used in low-level UAV control (Gautam et al., 2014).

In Beard and McLain (2012) the PID controller is also used for guidance. Here a simple autopilot is designed that uses the PID in both the guidance and control system. Another approach is to use line-of-sight (LOS) guidance laws, in You et al. (2012) the LOS guidance is used on both longitudinal and lateral plane for trajectory tracking.

Conventional landing of fixed-wing aircraft is normally done on runways. A common approach is to fly along a desired trajectory, until a predetermined touchdown point on the runway. The trajectory is typically divided into two segments, a glide-slope path with a constant flight path angle, and a flare path which provides a smooth transition from the glide-slope to the touchdown point. This approach is described in Nawrat (2014), Lavretsky and Wise (2013) and Stevens and Lewis (2003).

In Yoon et al. (2010) a spiral landing guidance is proposed for landing a small fixed-wing UAV in a recovery net. Here a similar approach to the glide-slope and flare path is used for the final approach to the recovery net, by using imaginary waypoints designed to smoothly guide the aircraft to the recovery net.

One of the most commonly used autopilots for fixed-wing UAVs is the APM:Plane open-source autopilot software. It has won several UAV competitions, and by being open-source its well suited for custom applications, education and research use. Two of the modes in APM:Plane are the Fly-by-wire A (FBWA) mode and the Fly-by-wire B (FBWB) mode. In FBWA the inputs to APM:Plane are desired roll, desired pitch and throttle. While in FBWB the inputs are desired roll, desired climb-rate and desired airspeed (APM:Plane Dev Team, 2016).

At the Underwater Systems and Technology Laboratory (LSTS), Porto, a toolchain is developed that can interact with all types of unmanned vehicles. Two of the key features of the toolchain is DUNE, a on-board software package used to write and run generic embedded software in C++ on-board a vehicle, and the Neptus ground control station software (LSTS, 2015a).

At NTNU Skulstad and Syversen (2014) developed a low-cost system for recovery of a fixed-wing UAV in a stationary net. The system was based on the open-source autopilot APM:Plane, and used two custom PID controllers for longitudinal guidance and a nonlinear controller for lateral guidance. Successful experiments was performed using the UAV Skywalker X8.

In 2015 Gryte (2015) developed a six degrees-of-freedom model of the Skywalker X8 using airfoil analysis, and later used the model to implement a software-in-the-loop (SITL) simulator using the JSBSim flight dynamics model. The model is still under development, and wind tunnel tests are being performed to improve the model. The same year Frølich (2015) designed an automatic ship landing system for the X8. Based on the APM:Plane mode fly-by-wire B (FBWB), a decoupled guidance system was developed that sent a desired climb rate, airspeed and bank angle to APM:Plane.

1.3 Contribution and Scope of This Thesis

The purpose of the thesis is to develop an autonomous landing system for a fixed-wing UAV in order to land in a net suspended by multirotor UAVs.

This involves:

- Developing a guidance and control system for autonomous landing in a recovery net suspended by multirotor UAVs.
- Tuning of the control system for the X8 fixed-wing UAV.
- Implementation and design of a system to coordinate a landing path between the fixed-wing UAV and the multirotors UAVs.

- Testing of the system in software-in-the-loop (SITL) simulations.
- Experimental testing of the system at Agdenes airfield.

Based on this the following contributions are made:

- A **longitudinal line-of-sight (LOS) guidance scheme** able to accurately track a desired height is designed and implemented in DUNE (Section 5.2.1).
- Design and implementation of a **coordinated landing path for the fixed-wing UAV**. Based on a virtual runway the landing path is coordinated between the fixed-wing UAV and the multirotor UAVs (Section 7.1.2).
- A **loiter plug-in for the ground control station software Neptus** is implemented, providing an easy-to-use synchronization mechanism for an operator to synchronize the landing between the X8 and the multirotors (Section 8.2).
- Combining linear control theory with the implementation of the low-level PID controllers in APM:Plane, **analytical expressions for how the controllers can be tuned** is found. The expressions are valid for all types of fixed-wing UAV airframes with a known model. The resulting tuning parameters for the X8-model are used in all simulations by the other master students using the X8 model with APM:Plane (Chapter 6).
- **Implemented support for the APM:Plane mode FBWA in DUNE** (Section 8.4).
- Design and implementation of a **speed controller for the X8** (Section 5.3).
- Both software-in-the-loop (SITL) simulations and flight experiments are performed in order to show the performance of the system (Chapter 9 and Chapter 10).

1.4 Structure of This Thesis

This thesis is divided into 11 chapters and 2 appendices. A short description of the chapters and appendices are given below.

Chapters

- Chapter 2 contains basic theory and background information. This includes a short description of coordinate frames, UAV kinematics and dynamics, transfer function models, landing path and guidance.
- The main software and hardware components used in this thesis is described in Chapter 3. The software section includes an introduction to APM:Plane and the LSTS toolchain, while the hardware section presents the main hardware components used by the landing system.
- Chapter 4 considers the most suited APM:Plane modes for net-landing and presents the mode used by the landing system.

- Chapter 5 presents first an overview of the guidance and control system that is designed and implemented, before each part of the system is presented in more detail.
- A method for how the low-level controllers in APM:Plane can be tuned when a model of the UAV is known is described in Chapter 6.
- The coordinated landing path between the X8 and the multirotors is presented in Chapter 7.
- Chapter 8 describes the implementation of the longitudinal LOS guidance, the coordinated landing path, the FBWA mode in DUNE and the loiter plugin developed for Neptus.
- The results from the software-in-the-loop (SITL) simulations and the field experiments are presented in chapters 9 and 10, respectively
- Chapter 11 finalize the thesis with a discussion and conclusion of the results, and recommendations for future work.

Appendices

- Appendix A contains information about a 3rd-order low-pass filtering reference model.
- Appendix B presents information about the parameters in the APM: Plane controllers.

Source Code

- The source code developed in this thesis can be found on the UAV-Lab git server `http://uavlab.itk.ntnu.no:88/` under the branch *uavlab*

Chapter 2

Background Theory

2.1 UAV Coordinate Frames

In order to describe the position and orientation of an UAV several different coordinate frames can be used. In this section two of the most common coordinate frames are explained. More details about coordinate frames can be found in Hofmann-Wellenhof et al. (2001)

2.1.1 NED

The North East Down (NED) frame, noted $\{n\}$, with origin o_n is defined relative to the Earth reference ellipsoid (WGS84). It is usually defined as the tangent plane on the surface of the Earth moving with the UAV. The y_n axis points towards East, the x_n axis points towards true North, while the z_n axis completes the right handed orthonormal by pointing down towards the ellipsoid.

2.1.2 BODY

The BODY-fixed frame , noted $\{b\}$, is moving and rotating with the UAV. The origin is fixed to the UAV, and usually placed at the center of mass of the UAV. The x_b axis points in the forward direction, and is aligned with the fuselag, the y_b axis goes through the right wing, while $z_b = x_b \times y_b$ completes the right handed orthonormal frame. The BODY-fixed frame is fixed to the rigid body and is related to the NED frame through the Euler angles roll, pitch and yaw. The frame is illustrated in Fig 2.1.

2.2 UAV Kinematic and Kinetic

The UAV equations of motion can be written in the vectorial representation used in Fossen (2011)

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \boldsymbol{M}_{rb}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{rb}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau}_{rb} \end{aligned} \tag{2.1}$$

where $\boldsymbol{\eta} \in \mathbb{R}^3$ is the position and orientation vector, $\boldsymbol{\nu} \in \mathbb{R}^6$ the linear and angular body-fixed velocity vector and $\boldsymbol{\tau} \in \mathbb{R}^6$ the generalized force vector. The matrix $\boldsymbol{M}_{rb}$ is the rigid-body mass matrix, $\boldsymbol{C}_{rb}$ is the coriolis-centripetal matrix and $\boldsymbol{J}_{\Theta}$ is a transformation matrix between NED and BODY.

The three vectors in Equation 2.1 is defined as:

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix}, \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix} \tag{2.2}$$

The vector components follow the SNAME (1950) notation and naming conventions, with some minor notation changes used in Beard and McLain (2012):

NED position	$\boldsymbol{p}^n_{b/n} = \begin{bmatrix} N \\ E \\ D \end{bmatrix} \in \mathbb{R}^3$	Attitude (Euler angles)	$\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^3$
Body-fixed linear velocity	$\boldsymbol{v}^b_{b/n} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3$	Body-fixed angular velocity	$\boldsymbol{\omega}^b_{b/n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3$
Body-fixed force	$\boldsymbol{f}^b_b = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \in \mathbb{R}^3$	Body-fixed moment	$\boldsymbol{m}^b_b = \begin{bmatrix} l \\ m \\ n \end{bmatrix} \in \mathbb{R}^3$

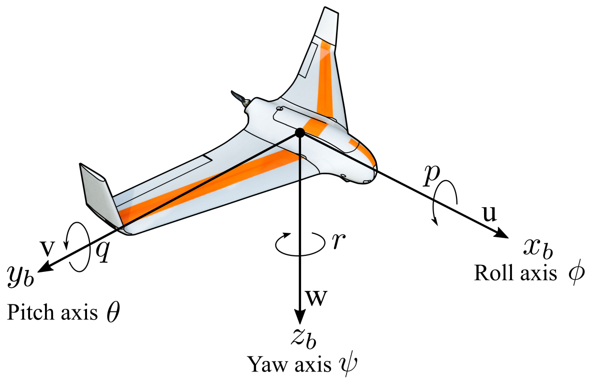


Figure 2.1: The velocities u, v, w, p, q and r in the body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$

The attitude of the UAV in the body frame can be represented by using the Euler angles, roll (ϕ), pitch (θ) and yaw (ψ). The angles are illustrated for the X8 in Figure 2.1, together with the body-fixed velocity vector $\boldsymbol{v}_{b/n}^{b}$ and the angular velocity vector $\boldsymbol{\omega}_{b/n}^{b}$.

As shown in Equation 2.1 the position and orientation vector $\boldsymbol{\eta}$ and the linear and angular body-fixed velocity vector $\boldsymbol{\nu}$ are related through a transformation matrix $\boldsymbol{J}_{\boldsymbol{\Theta}}(\eta)$. This transformation matrix is defined as

$$\boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \tag{2.3}$$

The body-fixed angular velocity vector $\boldsymbol{\omega}_{b/n}^{b}$ and the Euler rate vector $\dot{\boldsymbol{\Theta}}_{nb}$ is therefore related through the transformation matrix $\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}_{nb})$ according to

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{b/n}^{b} \tag{2.4}$$

Where the transformation matrix $\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}_{nb})$ is defined as

$$\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{2.5}$$

Writing the expression out, the transformation is therefore given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.6}$$

Furthermore, the body-fixed velocity vector $\boldsymbol{v}_{b/n}^{b} = [u, v, w]^T$ can be expressed in $\{n\}$, as

$$\dot{\boldsymbol{p}}_{b/n}^{n} = \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})\boldsymbol{v}_{b/n}^{b} \tag{2.7}$$

where $\dot{\boldsymbol{p}}_{b/n}^{n}$ is the NED velocity vector and $\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})$ is given as

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.8}$$

2.3 Guidance

The objective of guidance is to guide the vehicle to a desired point which in general can be moving. The guidance system does this by computing reference signals for the motion control system to follow. In Fossen (2011) different motion control scenarios are classified according to

- *Set-point regulation* is a special case where the desired position and attitude are chosen to be constant.

- *Trajectory tracking* where the objective is to force the system output $y(t) \in \mathbb{R}^m$ to track a desired output $y_d(t) \in \mathbb{R}^m$. The desired trajectory can be computed using reference models generated by low-pass filters, optimization methods or by simply simulating the vehicle motion using an adequate model of the vehicle. Feasible trajectories can be generated in the presence of both spatial and temporal constraints.

- *Path following* is following a predefined path independent of time. No restrictions are placed on the temporal propagation along the path. Spatial constraints can, however, be added to represent obstacles and other positional constraints if they are known in advance.

Trajectory tracking can therefore be considered as the problem of tracking a point that is moving as a function of time. Removing the time dependency can be done by introducing a path variable, which results in path following. The path following problem can be divided into straight-line paths and curved paths. The path-following controller for curved paths is a kinematic controller that generates desired states using a parametrization of the path. The drawback here is that the path must be parametrized and known in advance. In many cases this is not practical and a simpler path consisting of waypoints and straight-lines must be used.

The waypoints are generally constant coordinates defined in Euclidean space. As the vehicle comes within a circle of acceptance, a waypoint algorithm is used to switch to the next waypoint. In the sections below two of the most known guidance schemes used in path following are briefly outlined.

2.3.1 Pure-Pursuit

The pure pursuit is a two-point guidance scheme, where only the vehicle and target waypoint is considered in the geometry. The velocity of the vehicle is aligned with the vector pointing to the target waypoint. This scheme is often compared with a predator chasing a prey in the animal word, and often leads to a tail-chase behaviour. The pure pursuit guidance scheme is illustrated in Figure 2.2.

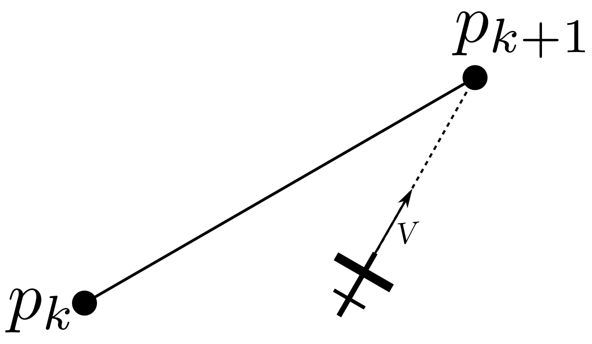


Figure 2.2: Pure pursuit guidance

2.3.2 Line-of-Sight Guidance

The line-of-sight (LOS) guidance law is a three-point guidance scheme that steers the vehicle towards the straight-line between the previous and the current waypoint. This is achieved by minimizing the cross-track error e and by constraining its motion along the LOS vector. An important factor in how the LOS guidance behave is the lookahead distance Δ. A small lookahead distance will induce more aggressive steering, while a large lookahead distance will result in a smoother steering. The LOS guidance scheme is illustrated in Figure 2.3.

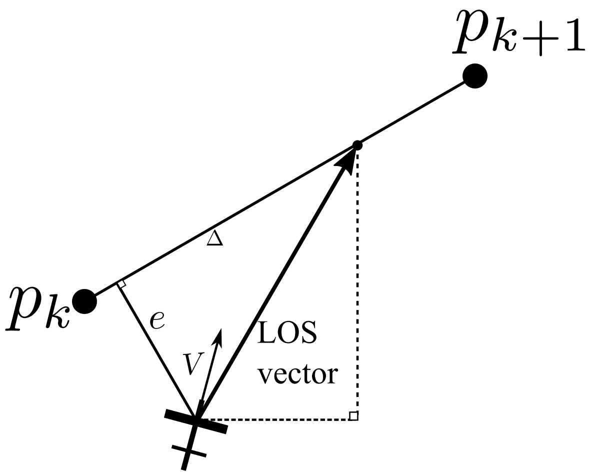


Figure 2.3: Line-of-sight guidance

2.4 Landing Path

In the literature the landing path is usually divided into a glide-slope path and a flare path. An example of a landing path is illustrated in Figure 2.4. In the glide-slope the main objective is to gradually lose height with a constant flight path angle γ, while the purpose of the flare path is to slow down the aircraft rate of decent, and provide a smooth transition from the glide-slope to a defined touchdown point.

In numerous of textbooks and papers Lavretsky and Wise (2013); Ju and Tsai (2008); González et al. (2013) the flare path is described by an exponential function defined by

$$h_{ref}(t) = h_0 e^{-t/\tau} \tag{2.9}$$

where h_0 is the flare initiation height, τ is a time constant and t is the current time throughout the flare maneuver. Furthermore, τ is assumed constant, and should be chosen based on the desired airspeed and the distance d to the touchdown point.

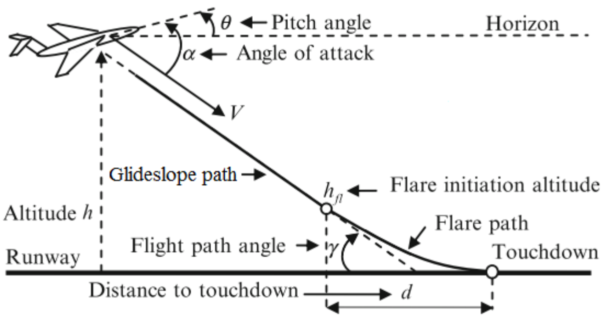

θ ← Pitch angle
Horizon
α ← Angle of attack
V
Glideslope path →
h_{fl} ← Flare initiation altitude
Altitude h
Flare path
Flight path angle → γ
Runway
Touchdown
Distance to touchdown → d

Figure 2.4: Aircraft on final approach to ladning.
Image courtesy of Lavretsky and Wise (2013)

2.5 UAV Transfer Functions

This section presents transfer function models for the roll and pitch dynamics of an UAV, i.e. the transfer function from the ailerons δ_a to the roll angle ϕ, and the transfer function from the elevator δ_e to the pitch angle θ. These models are derived in Beard and McLain (2012, Chapter 5.4), and a summary of the results are presented here. The transfer functions are later used to tune the roll and pitch controller in APM:Plane.

2.5.1 Transfer Function for Roll Dynamics

From Equation 2.6 we have:

$$\dot{\phi} = p + q\sin(\phi)\tan(\theta) + \mathrm{r}\cos(\phi)\tan(\theta) \tag{2.10}$$

Based on this equation, and using equations for aerodynamics and propulsion models, Beard and McLain (2012, p.69) find that the roll dynamics can be written in the Laplace domain as

$$\phi(s) = \left(\frac{a_{\phi 2}}{s(s+a_{\phi 1})}\right) + \left(\delta_a(s) + \frac{1}{a_{\phi 2}} d_{\phi 2}(s)\right) \tag{2.11}$$

where

$$a_{\phi 1} = -\frac{1}{2}\rho V_a^2 S b C_{p_p} \frac{b}{2V_a} \tag{2.12}$$

$$a_{\phi 2} = \frac{1}{2}\rho V_a^2 S b C_{p_{\delta_a}} \tag{2.13}$$

and $d_{\phi 2}$ is a nonlinear term, and is considered a disturbance on the system. The different variables and symbols are defined in the nomenclature.

A block diagram of Equation 2.11 is shown in Figure 2.5.

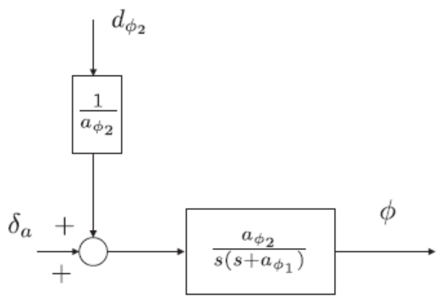


Figure 2.5: Block diagram for roll dynamics.
The inputs are the ailerons δ_a and the disturbance $d_{\phi 2}$

By assuming zero disturbance, $d_{\phi 2} = 0$, the transfer function can be written as

$$\frac{\phi(s)}{\delta_a(s)} = \frac{a_{\phi 2}}{s(s+a_{\phi 1})} \tag{2.14}$$

2.5.2 Transfer Function for Pitch Dynamics

From Equation 2.6 we have:

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{2.15}$$

Based on this equation, and using equations for aerodynamics and propulsion models, Beard and McLain (2012, p.73) find that the pitch dynamics can be written in the Laplace domain as

$$\theta(s) = \left(\frac{a_{\theta 3}}{s^2 + a_{\theta 1}s + a_{\theta 2}}\right) + \left(\delta_e(s) + \frac{1}{a_{\theta 3}}d_{\theta 2}(s)\right) \tag{2.16}$$

where

$$a_{\theta 1} = -\frac{\rho V_a^2 cS}{2J_y}C_{m_q}\frac{c}{2V_a} \tag{2.17}$$

$$a_{\theta 2} = -\frac{\rho V_a^2 cS}{2J_y}C_{m_\alpha} \tag{2.18}$$

$$a_{\theta 3} = \frac{\rho V_a^2 cS}{2J_y}C_{m_{\delta_e}} \tag{2.19}$$

and $d_{\theta 2}$ is a nonlinear term, and is considered a disturbance on the system. The different variables and symbols are defined in the nomenclature.

A block diagram of Equation 2.16 is shown in Figure 2.6.

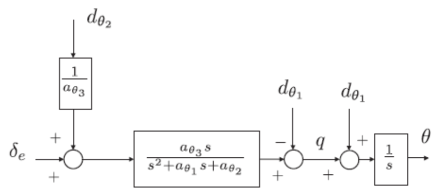


Figure 2.6: Block diagram for the transfer function from the elevator to the pitch angle. The inputs are the elevator δ_e and the disturbance $d_{\theta 2}$

$d_{\theta 1}$ is here defined as $d_{\theta 1} = q(cos\phi - 1) - rsin\phi$, and is considered a disturbance for small roll angles ϕ.

By assuming zero disturbance, $d_{\theta 2} = 0$ and $d_{\theta 1} = 0$, the transfer function can be written as

$$\frac{\theta(s)}{\delta_e(s)} = \frac{a_{\theta 3}}{s^2 + a_{\theta 1}s + a_{\theta 2}} \tag{2.20}$$

Chapter 3

System Components

This chapter provides a description of the main software and hardware components used by the autonomous landing system developed in this thesis.

3.1 Software

3.1.1 APM:Plane

APM:Plane is an open-source UAV software platform giving any fixed-wing aircraft full autonomous capability (APM:Plane Dev Team, 2016). It supports a variety of ground control station (GCS) software, which provides functionality such as mission planning/-operation, configuration and post-mission analysis.

APM:Plane is constantly under development, and its continuously being updated and improved by many dedicated volunteers in the open-source community. It has won several UAV competitions, and by being open-source its well suited for custom applications, education and research use.

Flight modes

APM:Plane has a wide range of different flight modes, so that the user can choose a mode that fits their flight needs and the flight behaviour they are are looking for. Depending on the mode and options the user chooses, the plane can be configured to be a simple flight stabilization system, a sophisticated autopilot or a training system. Some of the major flight modes are listed and briefly explained next.

Manual

In manual mode the pilot has full control over the UAV, using regular RC control, the servos are controlled directly with a radio controller.

Loiter

In loiter mode the UAV will fly in a circle around a point with a predetermined radius, holding a constant height.

Fly-by-wire A (FBWA)

FBWA is one of the most popular modes, where the UAV will hold roll and pitch specified by the pilot. The throttle is manually controlled.

Fly-by-wire B (FBWB)

In FBWB the roll control is the same as for the FBWA. But instead of controlling the pitch, the pilot now controls the climb-rate instead. The throttle is no longer controlled manually, but can instead be set to a desired airspeed. This mode uses a Total Energy Control System(TECS) which gives desired pitch and throttle based on the climb-rate and the desired airspeed.

Autotune

Autotune is a special mode similar to FBWA, but does automatic tuning of roll and pitch control gains.

Auto

In auto a list of waypoints specifies a mission for the UAV. It is an autonomous mode where the UAV follows the path defined by the wyapoints without the need for a pilot.

Guided

Guided mode is similar to the auto mode, but here only one waypoint is sent to the control system. The mode is described as a ”click to fly” mode, where the operator at the ground control station can order the UAV to fly to a specified point on the map, without setting up a mission.

Low-Level Roll and Pitch PID

Two modified proportional-integral-derivative (PID) controllers is used in APM:Plane to control respectively the roll and pitch of an fixed-wing UAV. A implementation of the controllers in Simulink is shown in Figure 3.1 and 3.2.

Compared to a standard PID controller several modifications are made. The first one is that the roll/pitch angle error is converted to angle rate error by multiplying the error with with a time constant $\tau = \frac{1}{\text{TCONST}}$ where `TCONST` is a design parameter representing the number of seconds from demanded to achieved angle. The standard value is 0.5, and is considered a reasonable value that works for nearly all types of airframes.

The second modification is that the controller gains are scaled with the airspeed of the UAV. This technique is called gain scheduling, and is applied to nonlinear systems in order to change the gains as the system dynamics changes. As shown in Section 2.5 the coefficients of the transfer functions describing the roll and pitch dynamics changes with with the airspeed V_a, and in order to compensate for this change gain scheduling is used. The gains are therefore scaled with a airspeed scaler defined as $scaler = \frac{15}{V_a}$ in a defined range. Shortly explained, the UAV will move the control surfaces more at low airspeed, and less at high airspeed.

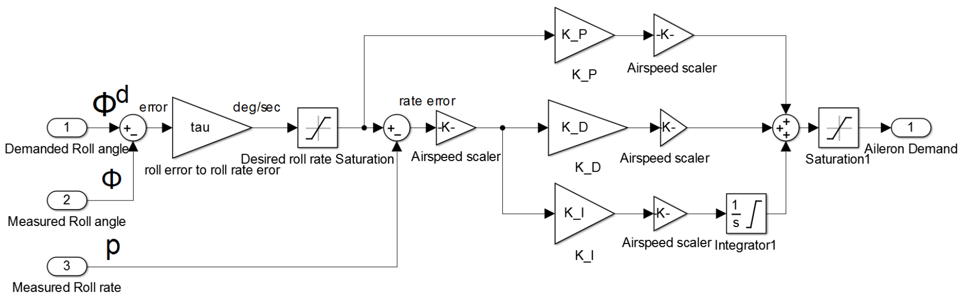


Figure 3.1: Roll controller

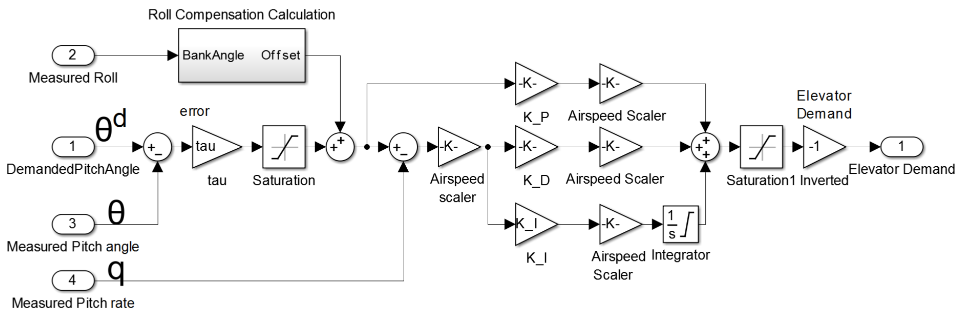


Figure 3.2: Pitch controller

3.1.2 LSTS Software Toolchain

The LSTS Toolchain is an open-source control architecture and software toolchain developed by the Underwater Systems and Technology Laboratory (Laboratório de Sistemas e Tecnologia Subaquática)(LSTS), which is an interdisciplinary research laboratory at the Faculty of Engineering, University of Porto (LSTS, 2015c).

The toolchain supports networked underwater, surface and aerial vehicles. It consists of the on-board software DUNE, the communication protocol IMC, the ground control station software Neptus, and the embedded operating system GLUED. A short introduction to each of these components are given below.

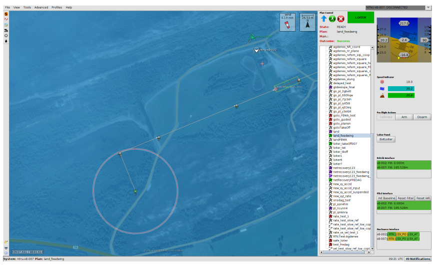
DUNE

DUNE: Unified Navigation Environment (DUNE) is the on-board software running on the vehicle. It is a runtime environment used to write and run generic embedded software in C++ on-board a vehicle. It is designed to consist of relatively small independent tasks, running on separate threads or processes, while communicating with each other using the IMC message bus. This leads to a high degree of modularity, where new tasks can easily be added, and old task can be enabled and disabled freely. Dune is used to interact with senors, payload and actuators, and is also responsible for communication, navigation, control, plan execution and vehicle supervision.

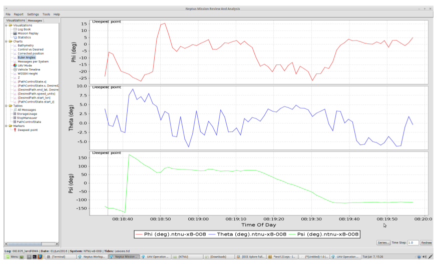
Neptus

Neptus is a graphical ground control station software used for mission planning, execution, review and analysis (LSTS, 2015b).

With Neptus the operator can graphically plan a mission for the UAV with the built-in map interface. When the plan is executed the operator can visually observe real-time data about the UAV, e.g. height, position, attitude and speed. After the mission Neptus can be used for analysis and playback of the log files. The Neptus interface is shown in Figure 3.3.



(a) Neptus map interface.



(b) Neptus review and analysis framework.

Figure 3.3: Neptus ground control station

Glued

Glued is a lightweight Linux distribution, targeted at embedded systems. It is platform independent and easily configurable for cross compilation. It is used as the OS on an embedded computer, where DUNE can be executed.

IMC

The Inter-Module Communication (IMC) protocol is a shared message-oriented communication protocol used by all the components in the LSTS toolchain. It is used internally between the different tasks in DUNE, as well as between the different components in the toolchain. It consists of a large set of shared message definitions that can be serialized and transferred over different means. Users can also easily add their own customized messages if needed. The IMC message bus is illustrated in Figure 3.4. More details about IMC can be found in Martins et al. (2009).

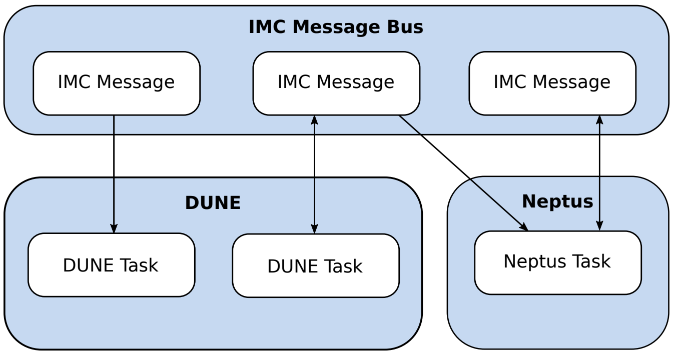


Figure 3.4: Illustration of the IMC message bus

3.2 Hardware

3.2.1 Fixed-Wing UAV

The fixed-wing UAV used in this thesis is the Skywalker X8. A flight picture of one of the X8 at the UAV-lab is shown in Figure 3.5. The X8 does not have a rudder, and uses elevons as control surfaces. The elevons combines the functions of the elevator (used for pitch control) and the aileron (used for roll control), and a mapping from aileron-elevator demand to elevon is therefore done in the low-level autopilot. The X8 is moulded out of expanded polyolefin (EPO) foam, making it highly durable, cheap and light weight. The X8 has therefore become a popular choice in both the model airplane and research community. Some key specifications and components used in the X8 at the UAV-lab is summarized in Table 3.1.

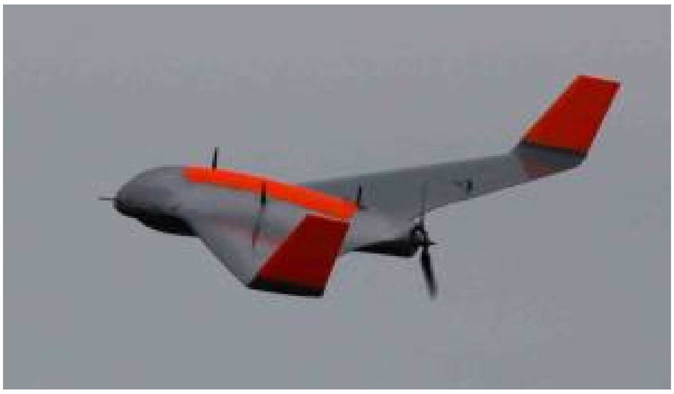


Figure 3.5: The Skywalker X8 in flight.

Airframe weight	3 kg
Maximum takeoff weight	4.2 kg
Wing span	2120 mm
Cruise speed	18 m/s
Flight time	45 - 60 min
Control surfaces	Elevons (combined aileron and elevator)
Low-level autopilot	Pixhawk w/APM:Plane
GPS	3DR uBlox GPS with Compass Kit
Servos	HiTec HS-5125MG
Motor	Hacker A40
Telemetry link	Ubiquiti Rocket M5

Table 3.1: Skywalker X8 components and specifications

3.2.2 Pixhawk Autopilot

Pixhawk is an advanced autopilot system designed by the PX4 open-hardware project and manufactured by 3D Robotics (3DRobotics, 2016). It was released in November 2013 and features the latest processor and sensor technology from ST Microelectronics®. Furthermore it supports a wide range of communication interfaces such as UART, CAN, I^{2}C and SPI.

The Pixhawk uses a real-time operating system called NuttX, which is designed from the ground up with emphasis on standards compliance and small footprint. The PX4 middleware running on NuttX supports 2 different flight control stacks, its own *PX4 Flight Stack* and *APM (ArduPilotMega)*. The flight stack used at the UAV-lab is the APM Flight Stack. APM offers firmware solutions for 3 kinds of vehicles. APM:Plane, APM:Copter and APM:Rover. The Skywalker X8 uses the AMP:Plane.

The Pixhawk is shown in Figure 3.6, information about processors and sensor specifications can be found at 3DRobotics (2016).

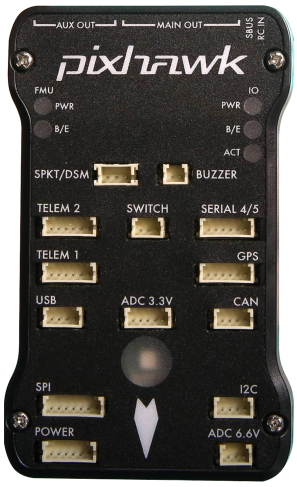
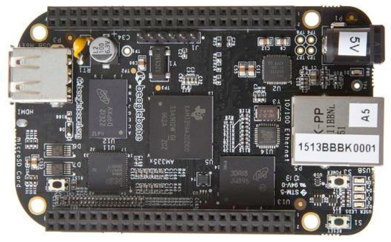


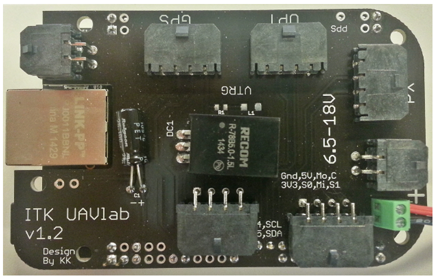
Figure 3.6: The Pixhawk autpilot. *Image courtesy of 3dr.com*

3.2.3 BeagleBone Black

The embedded computer used by the landing system is the BeagleBone Black (BBB). Using the Glued OS, it is able to run both DUNE and the Real Time Kinematic (RTK) positioning algorithms in real-time. BeagleBone Black is a low-cost, community-supported development platform built around the 1GHz Sitara AM335x ARM Cortex-A8 processor. The BBB is shown in Figure 3.7a. More information about the BBB can be found at BeagleBoard Foundation (2016).



(a) The BeagleBone Black. *Image courtesy of beagleboard.org*



(b) The BeagleBone Black with custom made add-on board attached.

Figure 3.7: The embedded payload computer Beaglebone Black. With and without a custom made add-on board, providing simpler connectivity to peripherals.

Part II

Methods

Chapter 4

Choosing APM:Plane Mode

APM:Plane modes have 13 built in flight modes, where several of them could in theory be used for landing in a net. This section considers 4 of the most suited, and present the mode used by the autonomous landing system.

4.1 Guided and Auto Mode

In both guided and auto mode the built-in guidance and controllers in APM:Plane is used, without any need for external guidance or controllers in DUNE.

The problem with using guided mode is that the list of waypoints that defines a mission need to be sent one at a time. First when the UAV has finished tracking to a target waypoint, the next waypoint can be sent. This again causes a problem, since the previous waypoint is not taken into account when the path to the next waypoint is generated. Instead of following the straight-line path defined between the previous waypoint and the next waypoint, it follows the straight-line path, defined from the location where the UAV switches waypoint to the location of the next waypoint. This is illustrated in Figure 4.1, where the circle around the waypoints is the circle of acceptance for when the next waypoint is selected, and the arrowed lines is one possible path that can be generated for the UAV.

For controlling the desired height two methods are used in guided and auto mode:

- Set-point: After reaching a waypoint, the desired height is set directly to the height of the next waypoint. The UAV will therefore try to reach the height of the next waypoint as quickly as possible.

- Glideslope: A gradually change in desired height is used. A parallel line between the start waypoint and end waypoint is used, and the desired height changes proportional to how long the UAV is along the line.

Unfortunately the APM:Plane guidance does not sufficiently handle the switching between these two methods. More specifically, when switching from glideslope handling to set-point handling a jump in the desired height occurs. This is shown in Figure 4.2. The figure also shows that the jump in desired height also causes a jump in desired pitch, witch causes the pitch to go from 4 degrees to -7 degrees, forcing the UAV into a steep dive. When doing a precision landing this steep dive behavior is not acceptable. Based on this both guided and auto mode is considered to not be a suited mode for landing.

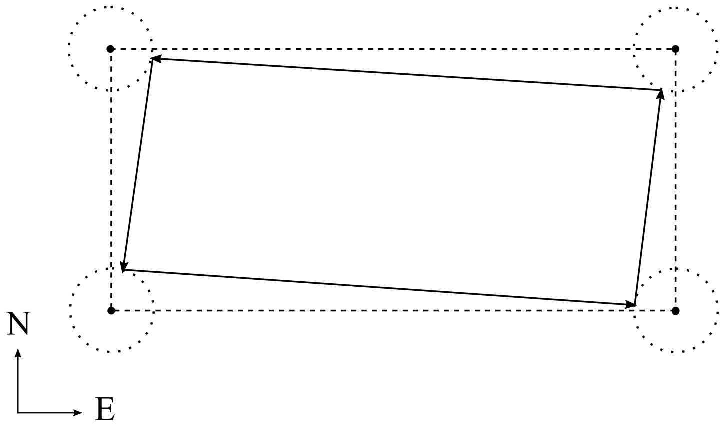


Figure 4.1: Path handling in guided mode.

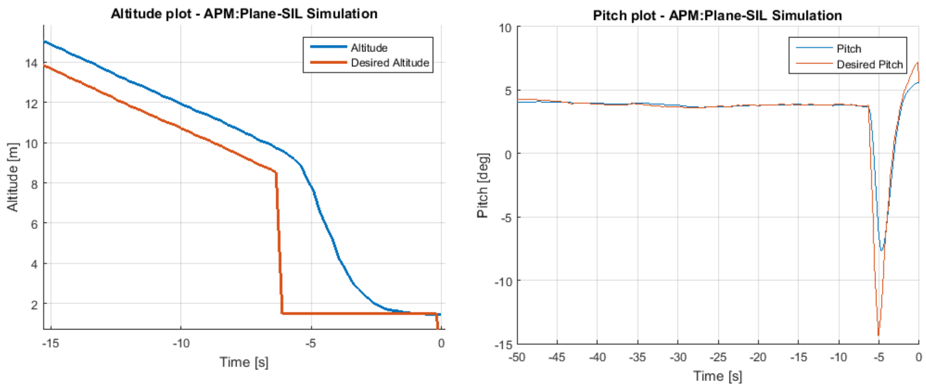


Figure 4.2: APM:Plane switching from glideslope handling to set-point handling.

4.2 Fly-By-Wire B Mode

In Fly-By-Wire B mode the APM:Plane takes desired roll, desired climb-rate, and desired airspeed as inputs. The desired roll is sent directly to the low-level PID, while the desired climb-rate and desired airspeed is sent to a Total Energy Control System(TECS), which calculates desired pitch and throttle.

The FBWB mode was used in the control system developed in Frølich (2015), where a PID controller for height that sent desired climb-rate was implemented in DUNE. The simulations results presented by Frølich showed an error between the height and the desired height at ± 1 meter when performing a landing. Since the simulation results often presents a best-case scenario, the error in height in a real test would probably be higher. An error at ± 1 meter is therefore not sufficient for a landing in a real test.

A code review on how the TECS controller in APM:Plane handles the desired climb-rate from DUNE was therefore done, and the result is outlined in Algorithm 1.

```
Loop
    $h_{change} = \dot{h}_{desired\_DUNE} \cdot \Delta_{second\_main\_loop}$
    $h_{desired} = h_{desired} + h_{change}$
    //Apply 2 point moving average to desired height
    $h_{desired} = \frac{1}{2}\left[h_{desired} + h_{desired\_last}\right]$
    //Apply first order lag to height demand
    $h_{desired\_new} = 0.05 \cdot h_{desired} + 0.95 \cdot h_{desired\_new\_last}$
    $\dot{h}_{desired} = h_{desired\_new} - h_{desired\_new\_last}$
EndLoop
```

Algorithm 1: APM:Plane handling of desired climb rate in FBWB-mode

The code review showed that the desired climb-rate from DUNE is not used directly in the TECS controller. The climb-rate from DUNE is instead converted to desired height by multiplying it with the number of seconds used in the last main loop cycle in APM:Plane. After that the desired height is passed through various filters, before a new climb-rate is calculated. In the end the desired climb-rate used by the TECS controller is different from the desired climb-rate sent by the control system in DUNE.

A test was performed measuring the difference between the climb-rate sent and the one used by TECS. The results is shown in Figure 4.3. In the figure we see a constant bias and lag between the signals. The lag comes from the various filters, and is measured to 0.5 seconds. Furthermore, TECS considers both $h_{desired}$ and $\dot{h}_{desired}$ when calculating the desired pitch and throttle.

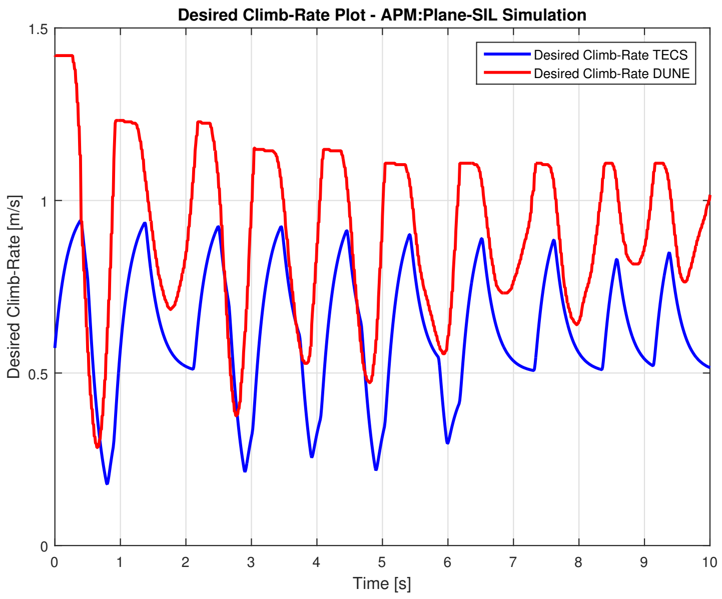


Figure 4.3: Comparison between desired climb-rate in TECS and the desired climb-rate from DUNE.

Based on the code review and the test results, the conclusion is that current implementation of FBWB-mode is not suited for the landing system.

4.3 Fly-By-Wire A Mode

In Fly-By-Wire A mode the APM:Plane takes desired roll, desired pitch and throttle as inputs, using only the low-level PID in APM:Plane. FBWA requires no changes in the APM:Plane software, and all high level control and guidance systems can be implemented in DUNE. The FBWA is therefore chosen as the APM:Plane mode used by the landing system, and support for FBWA in DUNE is therefore implemented as part of this thesis (Chapter 8.4).

Chapter 5

Guidance and Control System

This chapter presents the guidance and control system of the autonomous landing system. Section 5.1 describes briefly the overall system architecture. Section 5.2 describes the decoupled line-of-sight (LOS) guidance system, while section 5.3 describes the speed controller.

5.1 System Overview

The control and guidance system consists of a decoupled LOS guidance scheme for the lateral and longitudinal plane, a speed controller and the low-level APM:Plane PID controllers. The system is illustrated in figure 5.1.

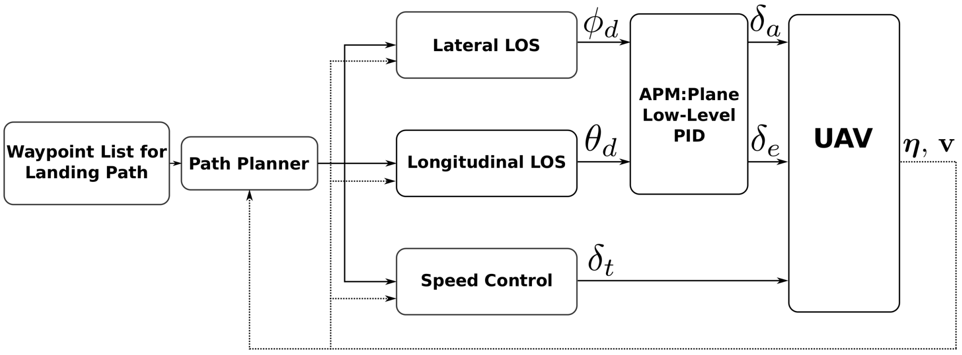


Figure 5.1: Simplified block diagram of the overall guidance and control system.

Based on the UAV position in relation to a target waypoint, the lateral and longitudinal LOS outputs desired roll and desired pitch respectively. These values are sent to the low-level APM:Plane PID which calculates the aileron and elevator deflections. The speed controller gives commanded throttle in order to follow a desired speed profile.

5.2 LOS Guidance for Decoupled Lateral and Longitudinal Planes

The guidance system is decoupled into two parts, namely, the longitudinal and lateral guidance. A simplified block diagram illustrating the overall LOS guidance system is shown in the figure below.

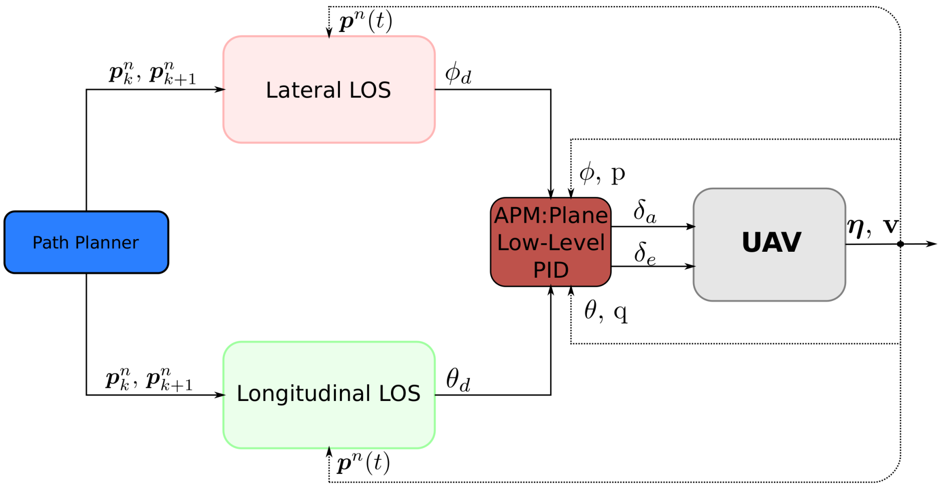


Figure 5.2: Block diagram of the decoupled longitudinal and lateral LOS guidance.

5.2.1 Longitudinal Line-of-Sight Guidance

The only available height controller in DUNE is called *Height*. It is a set-point controller that gives a desired climb-rate, in order to reach the height defined by the next waypoint as fast as possible.

In order to gain more control over the path between the previous and the next waypoint, a longitudinal LOS guidance scheme is developed to gradually change the height of the UAV when flying to the next waypoint. Furthermore, the objective of the longitudinal LOS is to provide a desired flight path angle $\gamma_{desired}$ used in order to follow a desired altitude reference, z_{ref}, calculated based on the UAV position along the two waypoints.

Consider the straight-line/glideslope path defined by two waypoints $\boldsymbol{p}_k^n = [x_k, y_k, z_k]^T \in \mathbb{R}^3$ and $\boldsymbol{p}_{k+1}^n = [x_{k+1}, y_{k+1}, z_{k+1}]^T \in \mathbb{R}^3$ as shown in Figure 5.3.

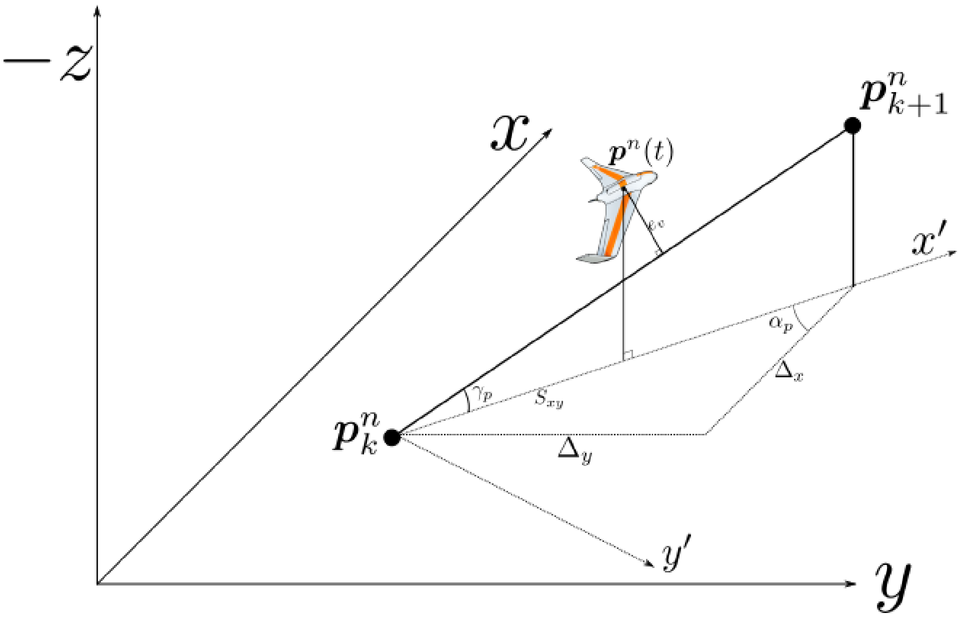


Figure 5.3: Path-fixed reference frame defined by the glideslope from $\boldsymbol{p}_k^n$ to $\boldsymbol{p}_{k+1}^n$

A path-fixed reference frame with origin in $\boldsymbol{p}_k^n$ can by described by two rotations. One rotation α_p around the z axis, and one rotation γ_p around the y-axis. The angles are defined as:

$$\alpha_p = atan2(y_{k+1} - y_k, x_{k+1} - x_k) \tag{5.1}$$

$$\gamma_p = atan2(-(z_{k+1} - z_k), L_{xy}) \tag{5.2}$$

Where L_{xy} is the length from p_k^n to p_{k+1}^n on the projection of the glideslope onto the xy-plane, and is given by

$$L_{xy} = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \tag{5.3}$$

and where $\mathrm{atan2(y, x)}$ is the four-quadrant version of $\mathrm{arctan(y/x)} \in [-\pi/2, \pi/2]$.

For an UAV located at $\boldsymbol{p}(t)^n = [x(t), y(t), z(t)]$ the along-track distance, cross-track error and vertical-track error with respect to the path-fixed reference frame is given by:

$$\boldsymbol{\varepsilon}(t) = \boldsymbol{R}_z^T(\alpha_p)\boldsymbol{R}_y^T(\gamma_p)\left(\boldsymbol{p}^n(t) - \boldsymbol{p}_k^n\right) \tag{5.4}$$

where

$$\boldsymbol{R}_z(\alpha_p) = \begin{bmatrix} \cos(\alpha_p) & -\sin(\alpha_p) & 0 \\ \sin(\alpha_p) & \cos(\alpha_p) & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathrm{SO}(3) \tag{5.5}$$

$$\boldsymbol{R}_y(\gamma_p) = \begin{bmatrix} \cos(\gamma_p) & 0 & \sin(\gamma_p) \\ 0 & 1 & 0 \\ -\sin(\gamma_p) & 0 & \cos(\gamma_p) \end{bmatrix} \in \mathrm{SO}(3) \tag{5.6}$$

and $\boldsymbol{\varepsilon}(t) = [s(t), e_c(t), e_v(t)]^T \in \mathbb{R}^3$ is respectively the along-track distance, cross-track error and vertical-track error.

By setting $\gamma_p = 0$ in Equation 5.4 the projection of the along-track distance onto the xy-plane can be written as:

$$S_{xy}(t) = \cos(\alpha_p)(x(t) - x_k) + sin(\alpha_p)(y(t) - y_k) \tag{5.7}$$

By expanding Equation 5.4 and inserting for the expression in Equation 5.7, the vertical-track error can be written as:

$$\begin{aligned} e_v &= \cos(\alpha_p)sin(\gamma_p)(x(t) - x_k) + sin(\alpha_p)sin(\gamma_p)(y(t) - y_k) \\ &\quad + cos(\gamma_p)(z(t) - z_k) \end{aligned} \tag{5.8}$$

$$e_v = (z(t) + tan(\gamma_p)S_{xy} - z_k)cos(\gamma_p) \tag{5.9}$$

$$e_v = (z(t) - z_{ref})cos(\gamma_p) \tag{5.10}$$

where z_{ref} is defined as:

$$z_{ref} = -\tan(\gamma_p)S_{xy} + z_k \tag{5.11}$$

The associated control objective for the longitudinal line-of-sight guidance is

$$\lim_{t\to\infty} e_v(t) = 0 \tag{5.12}$$

which means that the UAV has converged to the glideslope path.

In order to achieve the control objective the flight path angle γ will be controlled. A longitudinal LOS guidance law that achieves the control objective is designed as

$$\gamma_{desired} = \gamma_p + \gamma_{LOS} \tag{5.13}$$

where

$$\gamma_{LOS} = arctan\left(\frac{K_{ph}e_v + K_{ih}\int e_v}{\Delta}\right) \tag{5.14}$$

K_{ph} and K_{ih} represents here the proportional and integral gain, and Δ is the look-ahead distance. In order to use the guidance law to track a curved path, γ_p should be replaced with a time-varying value representing the tangent of the curve that describes the curved path. The LOS guidance law is illustrated in Figure 5.4.

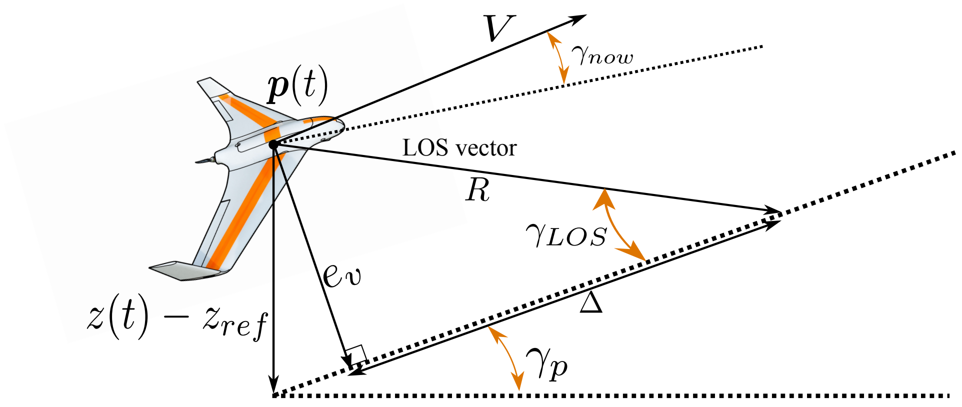


Figure 5.4: Longitudinal LOS guidance law

The longitudinal LOS guidance law is similar to the one used in You et al. (2012), but the look-ahead distance used there is equal to $\Delta = K_p V_a$. The drawback of using this expression is that for constant speed, the look-ahead distance is also constant independent of the vertical-track error. Inspired by the time-varying look-ahead distance often used for marine crafts, a time-varying look-ahead distance is instead used.

Based on the geometric relationship $e_v(t)^2 + \Delta(e_v)^2 = R^2$, which can be seen in Figure 5.4, a time-varying look-ahead distance is designed as

$$\Delta(e_v) = \sqrt{R_{max}^2 - e_v(t)^2} \tag{5.15}$$

where $e_v(t)$ is trimmed based on a value R_{min} such that R_{min} and R_{max} are the minimum and maximum allowed values for $\Delta(e_v)$ respectively. The idea behind a varying look-ahead distance is that a small Δ is assigned when the UAV is far from the desired path, thus resulting in a more aggressive behaviour that decreases the vertical-track error faster. While a larger Δ is assigned when the UAV is close to the path and overshooting needs to be avoided.

Longitudinal LOS extension

The longitudinal LOS gives a desired flight path angle, γ_d, while the input to the low-level APM:Plane PID is desired pitch, θ_d. The longitudinal LOS is therefore extended by a controller that converts the desired flight path angle to desired pitch. The controller is derived by using the backstepping technique, and is based on Jarzebowska (2012, Ch.5.2) and Harkegard and Glad (2000).

In Harkegard and Glad (2000) the flight path angle dynamics is expressed as

$$\dot{\gamma} = \frac{1}{mV_a}(L(\alpha) + F_T sin\alpha - mgcos\gamma_d) \tag{5.16}$$

$$\triangleq \vartheta(\alpha - \alpha_0) \tag{5.17}$$

where $\vartheta(0) = 0$, and α_0 is the angle of attack at steady state, defined through $\dot{\gamma} = 0$. Using sign properties, it can be concluded that

$$\alpha\vartheta(\alpha) > 0, \quad \alpha \neq 0 \tag{5.18}$$

Introducing the control error

$$z_1 = \gamma - \gamma_d$$

whose dynamics are given by

$$\dot{z}_1 = \vartheta(\alpha - \alpha_0) \tag{5.19}$$

for a constant reference value. A control Lyapunov function

$$V_1 = \frac{1}{2}z_1^2$$

can be used to determine a stabilizing function, θ_d, considering θ as the control input of Equation 5.19.

$$\dot{V}_1 = z_1\vartheta(\alpha - \alpha_0) \tag{5.20}$$

$$= z_1\vartheta(\theta - z_1 - \gamma_d - \alpha_0) \tag{5.21}$$

$$= z_1\vartheta(-(1 + c_1)z_1 + \theta + c_1 z_1 - \gamma_d - \alpha_0) \tag{5.22}$$

$$= z_1\vartheta(-(1 + c_1)z_1) \; < 0, \quad \mathrm{z}_1 \neq 0 \tag{5.23}$$

is achieved by selecting

$$\theta_d = -c_1 z_1 + \gamma_d + \alpha_0, \quad c_1 > -1 \tag{5.24}$$

Since $c_1 = 0$ is a valid choice, γ feedback is not necessary for the sake of stabilization, but it provides an extra degree of freedom for tuning the closed loop performance.

From Beard and McLain (2012) we have that

$$\dot{h} = V_g sin(\gamma) \tag{5.25}$$

$$\dot{h} = u\sin\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta \tag{5.26}$$

γ can therefore be estimated as

$$\gamma = \sin^{-1}\left(\frac{u\sin\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta}{V_g}\right) \tag{5.27}$$

Finding α_0 in Equation 5.24 is not a trivial task, and is considered an optimization problem with no analytical solution. A numerical solution is presented in Beard and McLain (2012, p.280-282) using a gradient descent algorithm. However, the algorithm requires a accurate and precise model of the aircraft, and since the current available model of the X8 is not verified yet, and wind tunnel tests are at the moment being conducted, some simplifications are made.

Instead α_0 is solved for the trim condition, $\gamma = 0$ at $V_a = 18\frac{m}{s}$, and is used for all γ_d. Flying a glideslope within $\pm$ 4 degrees, a small steady-state error will be introduced, but is eliminated with the integral term in the longitudinal LOS shown in Equation 5.14.

The longitudinal LOS extension controller, converting desired flight path angle, γ_d to desired pitch, θ_d, is therefore implemented as

$$\theta_d = -c_1(\gamma - \gamma_d) + \gamma_d + \alpha_{trim} \tag{5.28}$$

5.2.2 Lateral Line-of-Sight Guidance

The lateral guidance used in this thesis is described in Fortuna and Fossen (2015), and is developed and implemented in DUNE by J.Fortuna. It consist of a cascaded system, where a nonlinear sliding mode controller gives a desired roll angle, ϕ_d , based on references generated by a LOS guidance algorithm. It is designed for small fixed-wing UAVs, and in order to minimize the effect of wind, it relies on controlling the course of the aircraft instead of the heading. The cascaded system is illustrated in Figure 5.5.

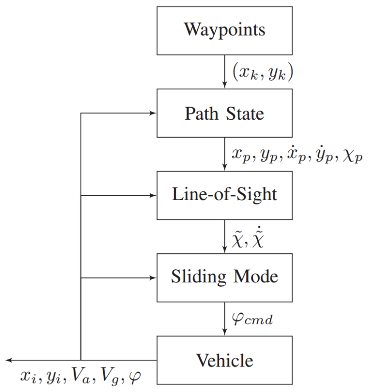

Waypoints
(x_k, y_k)
Path State
$x_p, y_p, \dot{x}_p, \dot{y}_p, \chi_p$
Line-of-Sight
$\tilde{\chi}, \dot{\tilde{\chi}}$
Sliding Mode
φ_{cmd}
Vehicle
$x_i, y_i, V_a, V_g, \varphi$

Figure 5.5: Cascaded lateral LOS system. *Image courtesy of J.Fortuna*

The lateral guidance scheme used in Frølich (2015) was also tested. Here a integral LOS guidance law based on Caharija et al. (2012) is used. The output from the LOS is desired heading, and a PID controller called bank-to-turn (BTT) was developed by Frølich in order to convert the desired heading to desired roll. Based on several simulation tests, the controller by J.Fortuna had overall much better performance than the one used in Frølich (2015), and was therefore chosen as the lateral LOS guidance used in the landing system.

5.3 Speed Controller

Since FBWA was previously not supported in DUNE, and support is first added as part of this thesis, no previously speed controller where available in DUNE. A speed controller sending desired throttle to APM:Plane is therefore developed and implemented.

The objective of the speed controller is to make the UAV track a desired speed profile. For this a PI-controller with two feed-forward terms is implemented. The speed controller is shown in Equation 5.29, where $\tilde{V}_a = V_a - V_{a,desired}$ is the airspeed error.

$$\delta_{thr} = K_{p,v_a}\tilde{V}_a + K_{i,v_a}\int_0^t \tilde{V}_a(\tau)d\tau + \delta_{thr,trim} + K_{p,e_v}e_v(t) \tag{5.29}$$

The first feed-forward term, $\delta_{thr,trim}$, is the throttle required for flying at trim conditions at $\gamma = 0$ and $V_a = 18m/s$. Based on the model by Gryte (2015) and reviewing flight-data logs from flights with the X8 conducted in 2015, this is value is estimated to 44% throttle.

Since the main objective of the guidance and control system is to accurately track a desired height, the last term, $K_{p,e_v}e_v(t)$, is added in order to improve the vertical-tracking done by the longitudinal LOS guidance. The idea behind this term is that when the UAV is either above or below the desired altitude, the throttle can be reduced or increased in order to reduce the vertical-tracking error faster.

Chapter 6

Tuning the APM:Plane PID

The equation of motion for an UAV consist of 12 nonlinear, coupled first-order, ordinary differential equations (Beard and McLain, 2012, p.60). Because of their complexity they are usually not used directly in developing controllers, but instead linearized and decoupled into reduced-order transfer functions and state-space models which are more suited to design controllers. These transfer functions are described in more details in Chapter 2.5.

APM:Plane is designed to be used by many different types of airframes, and without the need for advanced knowledge of the system dynamics. A mode called AUTOTUNE is often first used to tune the controllers, and produces reasonable tuning parameters for most airframes. For increasing the performance further the pilot must manually tune the gains according to a tuning guide. The tuning guide is based on the same principles as the Ziegler–Nichols tuning method, and in addition to be very time consuming, the method has no guarantee to be optimal or perfect.

As described in Chapter 1.2 a mathematical model of the X8 was developed by Gryte (2015), using airfoil analysis. Using the aerodynamic coefficients of the model, we can use this to analytically tune the controller gains for the system by combining the transfer functions models with the roll and pitch controller in AMP:Plane. The model by Gryte is still under development, but the results presented in this chapter can easily be updated with new aerodynamic coefficients when the model is fully developed.

As described in Chapter 3.1.1 both roll and pitch controllers scale their controller gains according to the UAV airspeed, V_a, where $scaler = \frac{15}{V_a}$. Since the coefficients of the transfer functions models also changes with V_a, the tuning is performed with $V_a = 15m/s$.

6.1 Roll Controller

By combining the roll dynamics described in Chapter 2.5 and the roll controller in APM:Plane a model of the closed-loop roll dynamics is implemented in Simulink as shown in Figure 6.1.

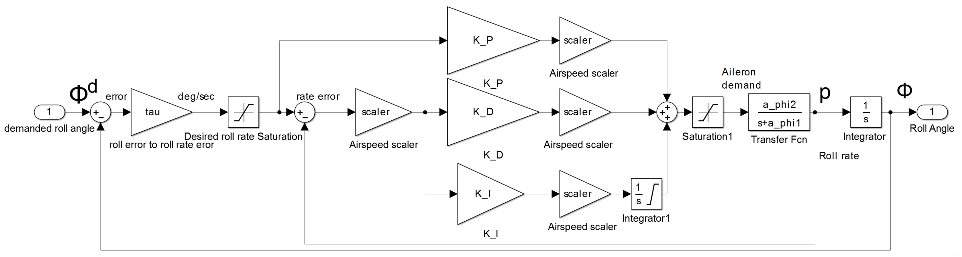


Figure 6.1: Roll controller

By setting $K_i = 0$, the closed-loop transfer function from ϕ^d to ϕ can be written as a second-order system

$$\frac{\phi}{\phi^d} = \frac{\omega_{n_\phi}^2}{s^2 + 2\zeta_\phi \omega_{n_\phi} s + \omega_{n_\phi}^2} \tag{6.1}$$

where

$$\omega_{n_\phi}^2 = a_{\phi 2}\tau(K_d + K_p) \tag{6.2}$$

$$2\zeta_\phi \omega_{n_\phi} = (a_{\phi 1} + K_d a_{\phi 2}) \tag{6.3}$$

Solving Equation 6.3 for K_d gives

$$K_d = \frac{2\zeta_\phi \omega_{n_\phi} - a_{\phi 1}}{a_{\phi 2}} \tag{6.4}$$

Solving Equation 6.2 and 6.3 for K_d and then combining them, ω_{n_ϕ} can be expressed as:

$$\omega_{n_\phi} = \sqrt{-a_{\phi 1}\tau + a_{\phi 2}K_p\tau + {\zeta_\phi}^2\tau^2} + \zeta_\phi \tau \tag{6.5}$$

where ζ_ϕ is a design parameter.

As described in Beard and McLain (2012, p.100), the controller gains should be selected such that the response to a step input giving a roll error $e_\phi^{\max}$, the ailerons will saturate at $\delta_a^{\max}$, where $e_\phi^{\max}$ is a design parameter.

Since the roll error, e_ϕ, in this case is converted to roll rate error, e_p, before being multiplied with the controller gains, and then saturated by $e_p^{\max}$, the ailerons should therefore be saturated when the roll rate error is saturated at $e_p^{\max}$. The controller gains should therefore be chosen as

$$K_p + K_d = \frac{\delta_a^{\max}}{e_p^{\max}} \tag{6.6}$$

Combining this expression with 6.4 and 6.5, this can be expressed as:

$$K_p + \frac{2\zeta_\phi(\sqrt{-a_{\phi 1}\tau + a_{\phi 2}K_p\tau + {\zeta_\phi}^2\tau^2} + \zeta_\phi\tau) - a_{\phi 1}}{a_{\phi 2}} = \frac{\delta_a^{\max}}{e_p^{\max}} \tag{6.7}$$

Solving this for K_p gives:

$$K_p = \frac{-2\sqrt{a_{\phi 2}}\zeta_\phi\sqrt{\frac{\delta_\phi^{\max}}{e_p^{\max}}}\sqrt{\tau} + a_{\phi 2}\frac{\delta_\phi^{\max}}{e_p^{\max}} + a_{\phi 1}}{a_{\phi 2}} \tag{6.8}$$

With integral effect, $K_i > 0$, the closed-loop transfer function becomes:

$$\frac{\phi}{\phi^d} = \frac{a_{\phi 2}\tau(K_d + K_p)s + K_i a_{\phi 2}\tau_i}{s\left(s^2 + (a_{\phi 1} + K_d a_{\phi 2})s + a_{\phi 2}(K_i + K_p\tau + K_d\tau)\right) + K_i a_{\phi 2}\tau} \tag{6.9}$$

For $K_i = 0$ it can be seen that the s in the denominator and numerator cancels, and the equation becomes equal to Equation 6.1.

The closed-loop poles of the system are given by

$$s^3 + (a_{\phi 1} + K_d a_{\phi 2})s^2 + a_{\phi 2}(K_i + K_p\tau + K_d\tau)s + K_i a_{\phi 2}\tau = 0 \tag{6.10}$$

which can be placed in Evans form as

$$1 + K_i\frac{a_{\phi 2}s + a_{\phi 2}\tau}{s^3 + (a_{\phi 1} + K_d a_{\phi 2})s^2 + a_{\phi 2}(K_p\tau + K_d\tau)s} = 0 \tag{6.11}$$

In Figure 6.2 the root locus of the characteristic equation is plotted as a function of K_i. Note that the pole near origin is never equal zero and only exist for $K_i > 0$. For $K_i = 1$ this pole is equal to -0.963.

The system is stable for all values of K_i, but for $K_i > 0$ the two left most poles becomes complex conjugates. As explained in Balchen et al. (2003, p.146-147) this will introduce oscillations in the system, and for a increasing K_i the figure shows that the imaginary part of the poles also increases with K_i, which will increase the oscillations. The oscillations introduced with K_i should be removed by increasing ζ_ϕ until the imaginary part of the poles is removed.

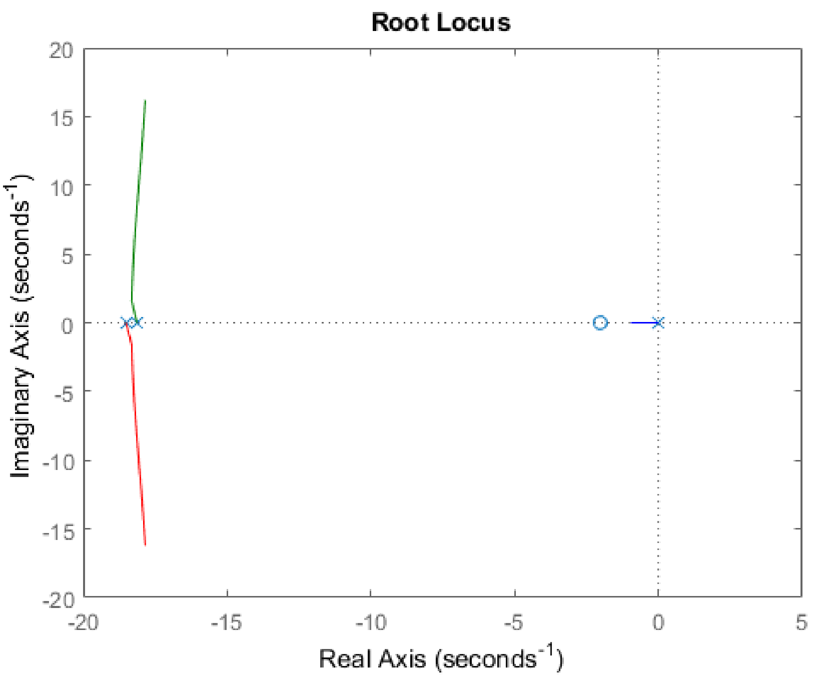


Figure 6.2: Root Locus for $K_i \in [0..1]$ with $K_p = 0.5667$, $K_d = 0.0333$ and $\zeta_\phi = 1$

6.2 Pitch Controller

The same tuning approach as for the roll controller can be applied to the pitch controller. The pitch dynamics described in Chapter 2.5 is combined with the pitch controller in APM:Plane, and a model of the closed-loop pitch dynamics is implemented in Simulink as shown in Figure 6.3. The roll angle is assumed to be zero in the transfer function model, and the the roll compensation is therefore not considered.

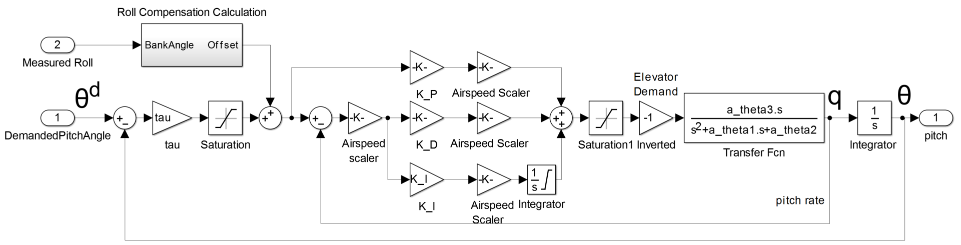


Figure 6.3: Pitch controller

By setting $K_i = 0$, the closed-loop transfer function can be written as a second-order system.

$$\frac{\theta}{\theta^d} = \frac{K_{\theta_{DC}} \omega_{n\theta}^2}{s^2 + 2\zeta_\theta \omega_{n_\theta} s + \omega_{n\theta}^2} \tag{6.12}$$

where

$$\omega_{n_\theta}^2 = a_{\theta 2} - a_{\theta 3}\tau(K_d + K_p) \tag{6.13}$$

$$2\zeta_\theta \omega_{n_\theta} = (a_{\theta 1} - a_{\theta 3} K_d) \tag{6.14}$$

$$K_{\theta_{DC}} = \frac{-a_{\theta 3}\tau(K_d + K_p)}{\omega_{n_\theta}^2} \tag{6.15}$$

Note that for the pitch dynamics a DC gain, $K_{\theta_{DC}}$, is introduced that is less than one. This means that the system will have a steady-state error, and the actual pitch will not converge to the desired pitch. This is solved by adding integral effect with $K_i > 0$.

Solving Equation 6.14 for K_d gives

$$K_d = \frac{a_{\theta 1} - 2\zeta_\theta \omega_{n_\theta}}{a_{\theta 3}} \tag{6.16}$$

Solving Equation 6.13 and 6.14 for K_d and then combining them, ω_{n_θ} can be expressed as:

$$\omega_{n_\theta} = \sqrt{-a_{\theta 1}\tau + a_{\theta 2} - a_{\theta 3} K_p \tau + {\zeta_\theta}^2 \tau^2} + \zeta_\theta \tau \tag{6.17}$$

where ζ_θ is a design parameter.

The controller gains should be chosen based on the same logic as explained for the roll controller. For pitch this means that the elevator should be saturated at $\delta_e^{\max}$, when the pitch rate error, $e_q^{\max}$, is saturated.

$$K_p + K_d = \frac{\delta_e^{\max}}{e_q^{\max}} \tag{6.18}$$

Solving this for K_p gives

$$K_p = \frac{-2\sqrt{a_{\theta 2}{a_{\theta 3}}^2{\zeta_\theta}^2 - {a_{\theta 3}}^3{\zeta_\theta}^2\frac{\delta_e^{\max}}{e_q^{\max}}\tau} - a_{\theta 1}a_{\theta 3} + {a_{\theta 3}}^2\frac{\delta_e^{\max}}{e_q^{\max}}}{{a_{\theta 3}}^2} \tag{6.19}$$

With integral effect, $K_i > 0$, the closed-loop transfer function becomes:

$$\frac{\theta}{\theta^d} = \frac{-a_{\theta 3}\tau(K_p + K_d)s - a_{\theta 3}\tau K_i}{s^3 + (a_{\theta 1} - a_{\theta 3}K_d)s^2 + (-a_{\theta 3}K_i + a_{\phi 2} - a_{\phi 3}\tau(K_p + K_d)s - a_{\phi 3}\tau K_i} \tag{6.20}$$

The DC gain, $K_{\theta_{DC}}$, is no longer present, and the pitch will reach the desired pitch.
For $K_i = 0$ it can be seen that the s in the denominator and numerator cancels, and the equation becomes equal to Equation 6.12.

The closed-loop poles of the system are given by

$$s^3 + (a_{\theta 1} - a_{\theta 3}K_d)s^2 + (-a_{\theta 3}K_i + a_{\phi 2} - a_{\phi 3}\tau(K_p + K_d)s - a_{\phi 3}\tau K_i = 0 \tag{6.21}$$

which can be placed in Evans form as

$$1 + K_i\frac{-a_{\theta 3}s - a_{\phi 3}\tau}{s^3 + (a_{\theta 1} - a_{\theta 3}K_d)s^2 + (a_{\phi 2} - a_{\phi 3}\tau(K_p + K_d)s} = 0 \tag{6.22}$$

In Figure 6.4 the root locus of the characteristic equation is plotted as a function of K_i. Note that the pole near origin is never equal zero and only exist for $K_i > 0$. How the poles change for increasing K_i is the same as for the roll controller, where the two left most poles becomes complex conjugates with increasing imaginary parts for a increasing K_i.

In comparison to the roll controller, the pitch controller should have a much higher value for K_i in order to eliminate the steady-state error.

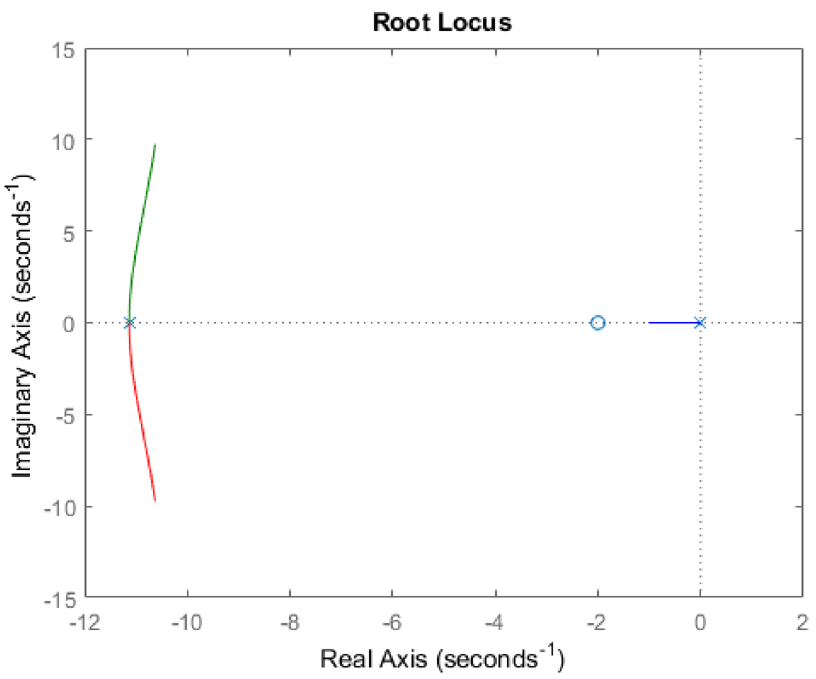


Figure 6.4: Root Locus for $K_i \in [0..1]$ with $K_p = 0.1491$, $K_d = 0.1794$ and $\zeta_\theta = 1$

6.3 Tuning Values for Roll and Pitch Controller

The coefficients of the transfer functions models are calculated based on the aerodynamics parameters in Gryte (2015, p.53) using $V_a = 15m/s$. The calculated coefficient values are listed in Table 6.1.

Parameter	Value
$a_{\phi 1}$	27.3318
$a_{\phi 2}$	279.7885
$a_{\theta 1}$	3.3686
$a_{\theta 2}$	54.7470
$a_{\theta 3}$	-105.3376

Table 6.1: Transfer function coefficient for roll and pitch with $V_a = 15m/s$

Using $\delta_a = \delta_e = 35$ and $e_\phi^{\max}$=$e_\theta^{\max} = 75$ and $\zeta_\phi = 1.008$, $\zeta_\theta = 1.05$, the gains are calculated using the equations in the previous sections. The values are listed in Table 6.2.

Parameter	Value
K_{p_ϕ}	0.4479
K_{i_ϕ}	0.0150
K_{d_ϕ}	0.0188
K_{p_θ}	0.2520
K_{i_θ}	0.1500
K_{d_θ}	0.2147

Table 6.2: Calculated gains for the roll and pitch controllers

As described in Appendix B, the values in Table 6.2 is related to the APM:Plane parameters by the equations listed in Appendix B. Using the equations the corresponding APM:Plane parameter values for the gains listed in Table 6.2, is calculated and given in Table 6.3.

Parameter	Value
RLL2SRV_P	0.9484
RLL2SRV_I	0.03
RLL2SRV_D	0.0188
PTCH2SRV_P	1.0834
PTCH2SRV_I	0.3
PTCH2SRV_D	0.2147

Table 6.3: Calculated APM:Plane parameter gains for the roll and pitch controllers

Chapter 7

Landing Path

This chapter presents the the different parts of the coordinated landing path. More details about the implementation is presented in Chapter 8.1.

7.1 Landing Path

A typical UAV flight is often divided into five phases (Valavanis, 2007; Engel, 2010).

- Phase 1: Take off. The UAV performs a takeoff.
- Phase 2: Reaching target: The UAV flies along a designated route to a designated altitude and location.
- Phase 3: Perform mission. The UAV completes the mission objectives. E.g. performs sensor measurements, monitoring, object tracking, etc.
- Phase 4: Approach landing area: The UAV flies along a designated route to the landing area.
- Phase 5: Landing: The UAV performs a landing.

The whole landing path from the UAV mission objectives are finished to the landing is done, can be described as the path of both phase 4 and phase 5. The autonomous landing system developed in this thesis takes care of both these phases, where phase 5 in this thesis is the coordinated landing between the fixed-wing UAV and the multirotor UAVs.

7.1.1 Approach Landing Area Path

In order to generate a path from any initial UAV position to the start of the coordinated landing path, the Dubins path API described in Sørbø (2016) is used. The Dubins path provided by the API is the shortest path from the initial UAV position to the start of the coordinated landing path. How the landing system interacts with the API is explained in more detail in Chapter 8.1.

7.1.2 Coordinated Landing Path

The coordinated landing path is defined by five waypoints and one loiter point. Based on these points the path is divided into four parts: waiting loiter, approach, glideslope and final approach. The path is illustrated in Figure 7.1 and Figure 7.2.

The first part of the coordinated landing path is the waiting loiter. Here the fixed-wing UAV will circle around a point with radius r at a specified altitude. The idea behind this waiting loiter is to introduce an easy-to-use synchronization mechanism for the operator, in order to synchronize the landing between the fixed-wing UAV and the multirotor UAVs. When the fixed-wing UAV is flying in this waiting loiter, the multirotors have time to prepare and get ready for the landing. First when both the fixed-wing UAV and the multirotors are ready, the operator can start the landing by pressing a *exit loiter*-button that is implemented (Chapter 8.2).

While flying in the loiter the X8 is banking at an approximately constant roll angle. The approach phase after the waiting loiter insures that the roll angle is close to zero and any potential oscillations is eliminated before the glideslope. This way the elevons can be used mainly as elevators along the whole glideslope, which increases the vertical tracking performance.

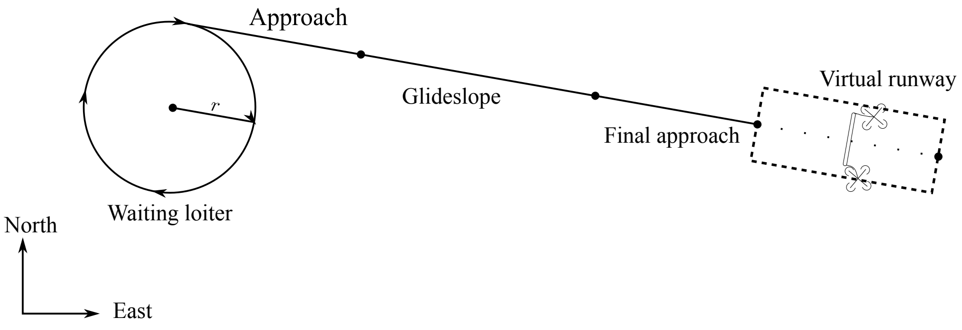


Figure 7.1: Lateral Path Illustration

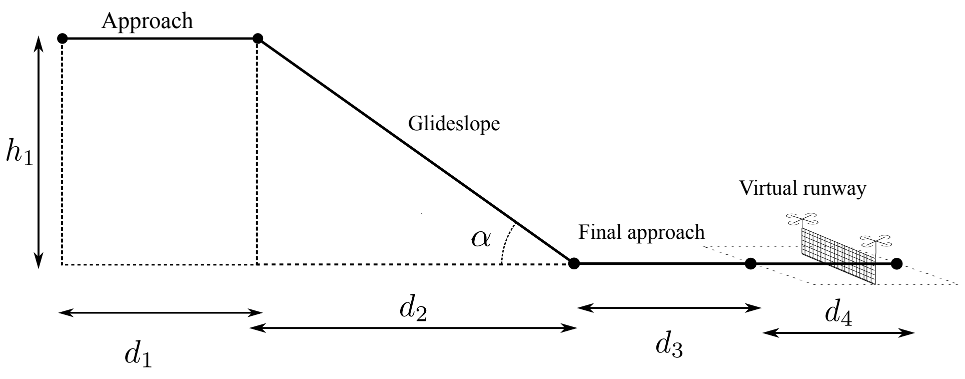


Figure 7.2: Longitudinal Path Illustration

Variable	Brief description
h_1	Relative height above the virtual runway
d_1	Horizontal approach distance from loiter to glideslope
d_2	Horizontal length of glideslope
d_3	Horizontal distance from glideslope to virtual runway
d_4	Horizontal length of the virtual runway
α	Descend angle during the glideslope phase
r	Radius of the waiting loiter circle

Table 7.1: Description of variables in Figure 7.1 and Figure 7.2

The purpose of the glideslope is to gradually lose height with a constant flight path angle. While the final approach phase is added for a flare-path that is discussed in the next section. The fixed-wing UAV will aim at the middle of the virtual runway, and at a desired net-center height, while the multirotor system in addition tries to compensate for any height or cross-track error by moving the net.

7.1.3 Feasible Glideslope and Flare Path

A straight-line path has a discontinuous first derivative at the location of the waypoints, and it is therefore not possible for the fixed-wing UAV to achieve a smooth transition between two straight lines. The fixed-wing UAV does not manage to change its flight path angle instantaneously. As mentioned in Chapter 2.4, the most common path planning solution for fixed-wing UAVs is to introduce a flare-path in order to achieve a smooth transition to the runway.

In order to generate a flare-path the longitudinal LOS guidance is extended with a option to use a 3rd-order low-pass filtering reference model when switching waypoints. The reference model used is the same as in Fossen (2011, p.250), and is included in Appendix A.1. The time-constant in the reference model, τ, must be chosen so that the flare-path converges to the height of the runway during the final approach distance, d_3. How τ is calculated in shown in Equation 7.1. t_{WP} is here a *time-of-arrival factor* in seconds used by the waypoint algorithm to define a circle of acceptance, for when to switch to the next waypoint.

$$\tau = \frac{t_{WP} + \frac{d_3}{V_{desired}}}{10} \tag{7.1}$$

A landing path with two different values of τ is shown in Figure 7.3. The advantage of using the reference model over the exponential function described in Chapter 2.4, is that it can also be used to generate a smooth transition into the glideslope. Moreover, when there are no special convergence requirements, it can also be used to generate a smooth path for all types of paths, with no prior knowledge of the path in advance.

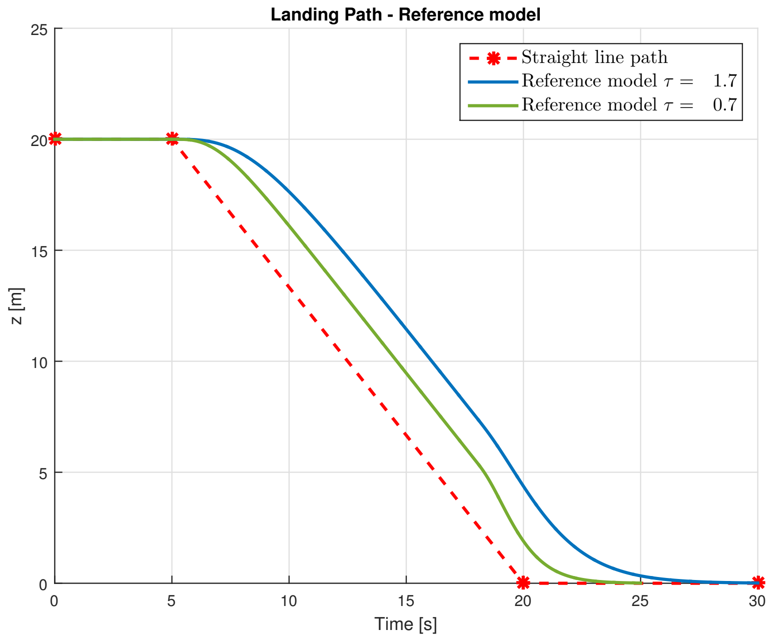


Figure 7.3: Landing path with reference model.

Chapter 8

Implementation

This chapter describes the implementation of the LOS guidance and controllers, the coordinated path planning, the FBWA-mode and the Neptus plugin. All of the implementation is done in C++ and Java using the LSTS framework.

8.1 Coordinated Landing Path

In order to coordinate a landing path between the multirotors and the X8 Skywalker, a new task in DUNE called NetRecovery is created. The task is run on both the master multirotor and the X8, and is responsible for creating the landing path for the X8. The landing path starts at the current position of the X8 and ends at the virtual runway where the net carried by the multirotors is located. A block diagram illustrating the interactions and system architecture is shown in Figure 8.1.

In Neptus the operator can graphically place a virtual runway at a desired location. A figure of the virtual runway in Neptus is shown in Figure 8.2. When the runway is placed, the operator can specify the height of the runway, and which multirotors and fixed-wing UAVs that are part of the landing. The virtual runway is then uploaded to the master multirotor in a IMC plan database message called IMC::PlanDB.

The NetRecovery task on the multirotor receives all types of PlanDB messages. If one these PlanDB messages contains information about the virutal runway, a new IMC message called CoordinatedNetRecovery is sent to the NetRecovery task on the X8. This IMC message contains all the information about the virtual runway needed to create a landing path for the X8. This includes information such as the height of the virtual runway and latitude/longitude coordinates of the start and end point of the virtual runway.

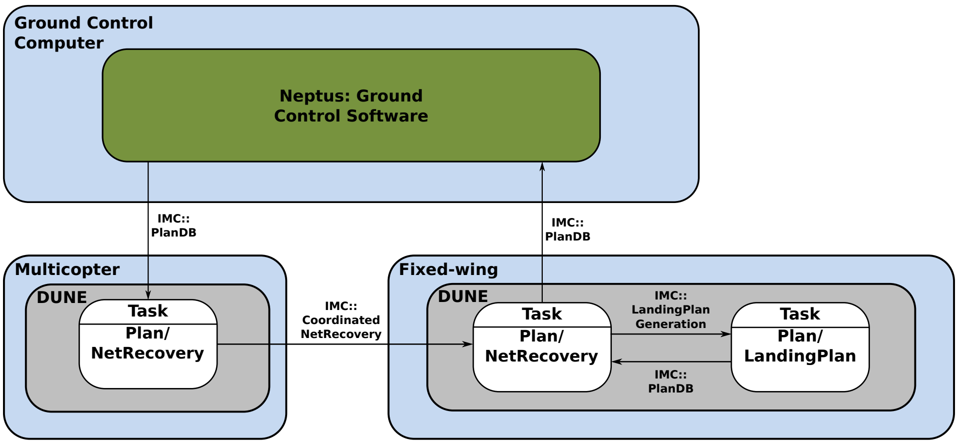


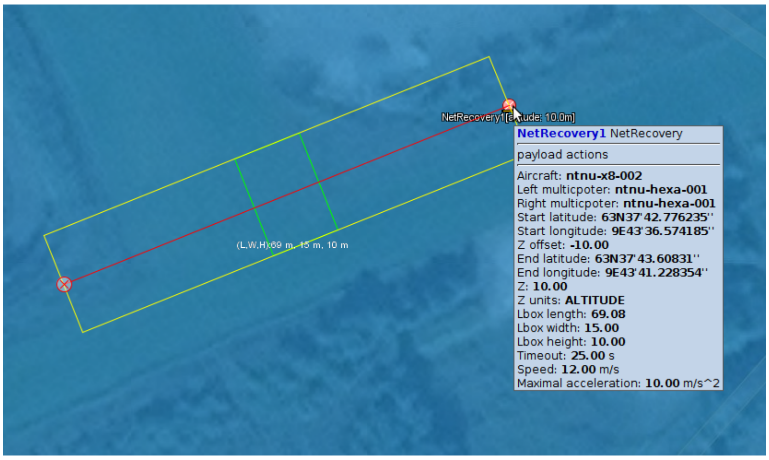
Figure 8.1: Block diagram illustrating the architecture of the coordinated landing path implementation.

When the IMC::CoordinatedNetRecovery message is received by the fixed-wing NetRecovery task, the bearing and range of the runway is calculated. Based on the bearing and range, a specified glideslope angle and the length of the final-approach and approach phase, a IMC::LandingPlanGeneration message is created. This message is sent to a task called LandingPlan created by Sørbø (2016). The LandingPlan task creates a path from the initial fixed-wing UAV position to the waiting loiter. The path is then sent as a IMC::PlanDB message from the LandingPlan task back to the fixed-wing NetRecovery task.

The final IMC:Plan message is then created by the fixed-wing NetRecovery task. The path defined in this ICM:Plan message consists of the following

- Path from initial fixed-wing UAV position to waiting-loiter
- A waypoint to approach phase
- A waypoint to glideslope phase
- A waypoint to final approach phase
- A waypoint to start of virtual runway
- A waypoint to end of virtual runway

The IMC:Plan message is then sent to Neptus, where the operator can inspect and approve the plan before the plan is started. A image of an example of the coordinated landing plan in Neptus is shown in Figure 8.3.



NetRecovery1 NetRecovery

payload actions

Aircraft: **ntnu-x8-002**
Left multicpoter: **ntnu-hexa-001**
Right multicpoter: **ntnu-hexa-001**
Start latitude: **63N37'42.776235''**
Start longitude: **9E43'36.574185''**
Z offset: **-10.00**
End latitude: **63N37'43.60831''**
End longitude: **9E43'41.228354''**
Z: **10.00**
Z units: **ALTITUDE**
Lbox length: **69.08**
Lbox width: **15.00**
Lbox height: **10.00**
Timeout: **25.00** s
Speed: **12.00** m/s
Maximal acceleration: **10.00** m/s^2

Figure 8.2: Example of a virtual runway in Neptus.

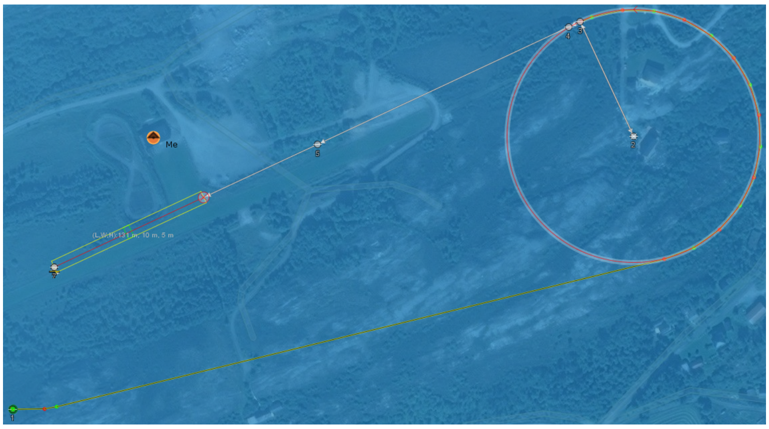
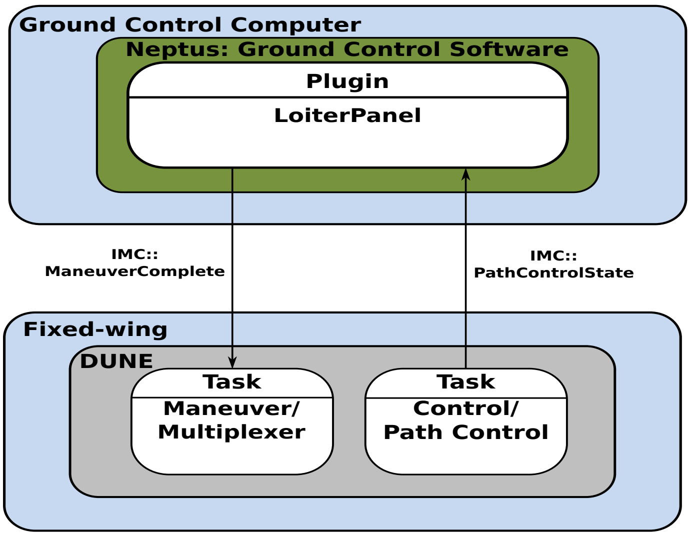


Figure 8.3: Example of a coordinated landing plan in Neptus.

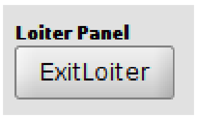
8.2 Waiting Loiter Plugin

In order for the operator to easily exit the waiting loiter explained in section 7.1.2, a plugin in Neptus is developed. The plugin acts as an easy-to-use synchronization mechanism for the operator, in order to synchronize the landing between the fixed-wing UAV and the multirotors. First when both the fixed-wing UAV and the multirotors are ready, the operator can start the landing by pressing the implemented *exit loiter*-button shown in Figure 8.4b. The system architecture behind the functionality of the button is shown in Figure 8.4a.

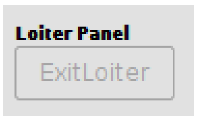
The longitudinal and lateral guidance systems, illustrated as *Path Control* in the figure, periodically sends a IMC message called IMC::PathControlState. In this message a flag called FL_LOITERING is active if the UAV performs a loiter maneuver. The plugin is implemented to receive the IMC::PathControlState message, and only when the FL_LOITERING flag is active, the plugin button is enabled. This is illustrated in Figure 8.4b-8.4c.



(a) Simplified block diagram illustrating the system architecture of the LoiterPanel plugin.



(b) Enabled when UAV is in loiter.



(c) Disabled when UAV is not in loiter.

Figure 8.4: System architecture and picture of the button.

In order to implemented the ability to exit the loiter in DUNE, a new IMC message called IMC::ManeuverComplete is created. When the operator click on the *ExitLoiter*-button in Neptus, this IMC message is sent to DUNE. Here the message is received in a task called Maneuver-Multiplexer, which then calls a function `signalCompletion()`, which signals a supervisor task that the loiter maneuver is completed. The UAV will then exit the loiter and continue to the next maneuver in the active mission list. If there are no maneuvers in the mission list, the UAV will stay in the same loiter.

8.3 Longitudinal LOS and Speed Controller

The system architecture for the longitudinal LOS, longitudinal LOS extension and speed controller is shown in Figure 8.5.

The longitudinal LOS is implemented in the task called LongitudinalLOS, while the longitudinal LOS extension and speed controller is implemented together in the same task. With this system architecture, the longitudinal LOS can also be used in FBWB mode by converting the desired flight path angle to desired climb-rate (Equation 5.26).

Two of the key IMC messages used by the guidance and control system is IMC::EstimatedState and IMC::DesiredPath. The EstimatedState message contains information about the UAV position, orientation and velocities, while the the DesiredPath message provides the system with information about the start and end waypoints. A very simplified pseudo-code of how the longitudinal LOS calculates the desired flight path angle is shown in Algorithm 2.

The desired flight path angle is then sent to the longitudinal LOS extension where the desired pitch is calculated. Here the speed controller also calculates the desired throttle, based on a desired speed specified for the current waypoint. Both desired pitch and desired throttle is then sent to a task called Ardupilot, further explained in Section 8.4.

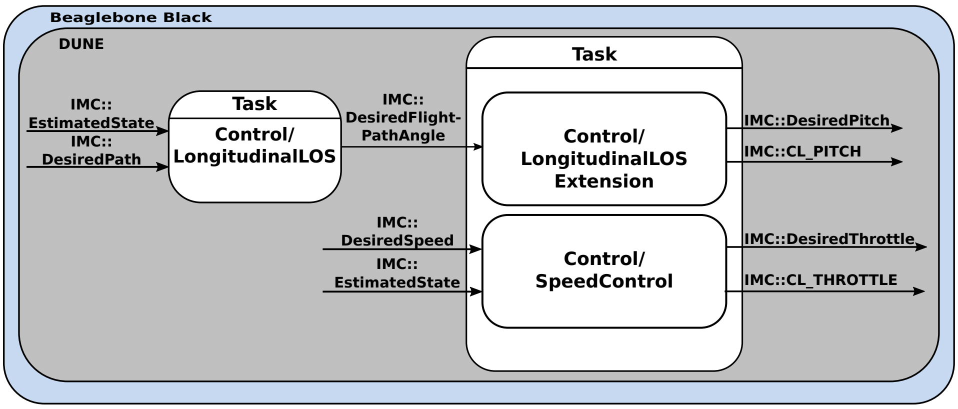


Figure 8.5: Block diagram illustrating the architecture of the longitudinal LOS guidance and speed controller in DUNE.

Algorithm 2: Longitudinal LOS guidance

```
  Input  : IMC::EstimatedState, IMC::DesiredPath
  Output: Desired flight path angle,$\gamma_d$
1 while Longitudinal LOS is active do
2     $\gamma_p$ = calculateGlideslopeAngle(IMC::DesiredPath)
3     z_desired = calculateZdesired(IMC::EstimatedState,IMC::DesiredPath,$\gamma_p$)
4     z_error = (z_desired - z_now)*cos($\gamma_p$)
5     $\gamma_{LOS}$=calculateLOSAngle(z_error)
6     return $\gamma_d = \gamma_p + \gamma_{LOS}$
7 end
```

8.4 Fly By Wire A Mode

DUNE only supported the APM:Plane modes Guided and Fly-by-wire B (FBWB). The evaluation of these modes in Chapter 4, showed that none of them where suited for high precision landings, and that support for the Fly-by-wire A (FBWA) mode was needed.

In FBWA desired pitch, throttle and roll is sent to APM:Plane using the Micro Air Vehicle Communication Protocol (MAVLink, 2016). The DUNE task responsible for this is called Ardupilot, and it receives the desired roll, pitch and throttle from the different controller tasks in DUNE, and sends the associated value to APM:Plane using a mavlink packet.

The Ardupilot task is also responsible for setting the correct APM:Plane mode. This is done based on the active control-loops. When e.g. a pitch controller that sends a desired pitch is activated, it sends a control-loops IMC message called IMC::CL_PITCH with an activate flag, telling the Ardupilot task that pitch control should be activated. Based on the active control-loops, the Ardupilot task can automatically determine which APM:Plane mode that should be active. FBWA is for instance the only APM:Plane mode that receives a desired pitch, and when the Ardupilot task receives the IMC::CL_PITCH with an activate flag, it is implemented so that it automatically switches over to FBWA mode. A block diagram illustrating the architecture is shown in Figure 8.6.

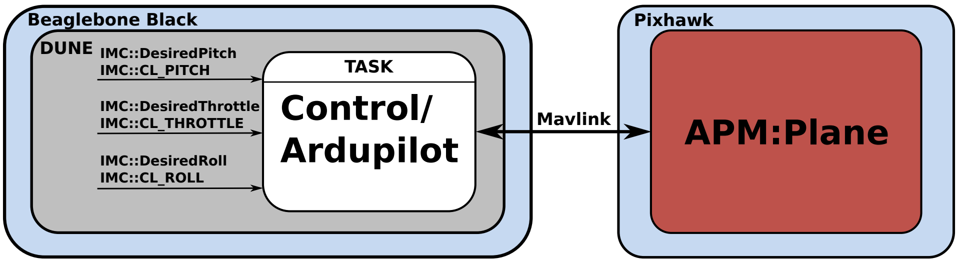


Figure 8.6: Block diagram illustrating the architecture of the FBWA implementation in DUNE

Since FBWB already uses desired roll, a total of 4 new IMC messages was created.

- IMC::DesiredPitch
- IMC::CL_PITCH
- IMC::DesiredThrottle
- IMC::CL_THROTTLE

In order to send the desired values to APM:Plane, they are mapped into PWM values and sent with a mavlink packet called `mavlink_msg_rc_channels_override_pack`. The mavlink packet used to set APM:Plane mode is `mavlink_msg_set_mode_pack`.

Part III

Results

Chapter 9

Simulations: Software in The Loop

In this chapter the results from the simulations done with the landing system is presented. The chapter starts with describing the simulator architecture in Section 9.1, and the results of the simulations is presented and discussed in Section 9.2 and Section 9.3.

9.1 Simulator Architecture

APM:Plane has a built-in Software-in-The-Loop (SITL) simulator that allows running the actual autopilot code without the hardware. In order to simulate the flight dynamics of the X8, a model of the X8 created and implemented by Gryte (2015) is used. The model is implemented in JSBSim, which is an open source, platform-independent, 6-degrees-of-freedom (DOF) Flight Dynamic Model (FDM) library written in C++ (JSBSim, 2013).

The SITL architecture is shown together with DUNE and Neptus in Figure 9.1. JSBSim is connected with APM:Plane over UDP, and continuously outputs flight dynamics data, i.e. position, attitude, angular velocity and acceleration in structs called `SITL_FDM` based on inputs from AMP:Plane, i.e. aileron/elevator deflection, wind and throttle. In order to configure APM:Plane, the minimalistic ground control station software MAVProxy is used.

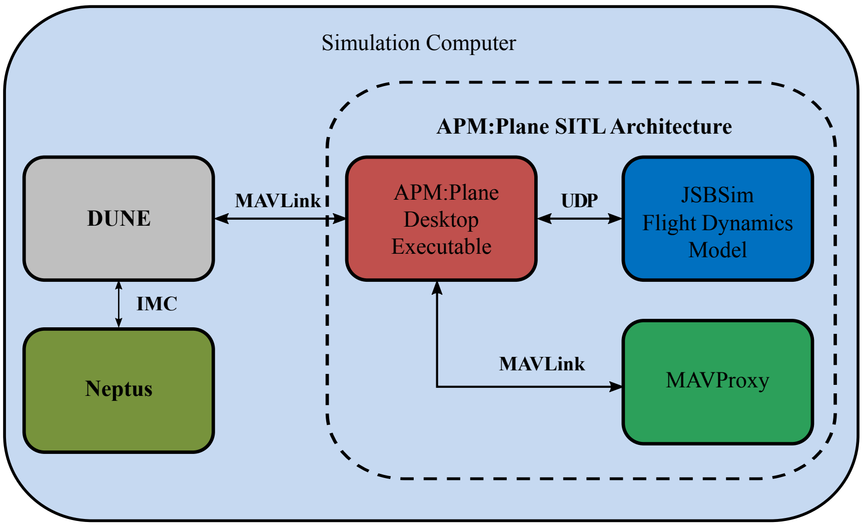

Simulation Computer
APM:Plane SITL Architecture
DUNE
MAVLink
APM:Plane Desktop Executable
UDP
JSBSim Flight Dynamics Model
IMC
Neptus
MAVLink
MAVProxy

Figure 9.1: Software-in-the-loop architecture

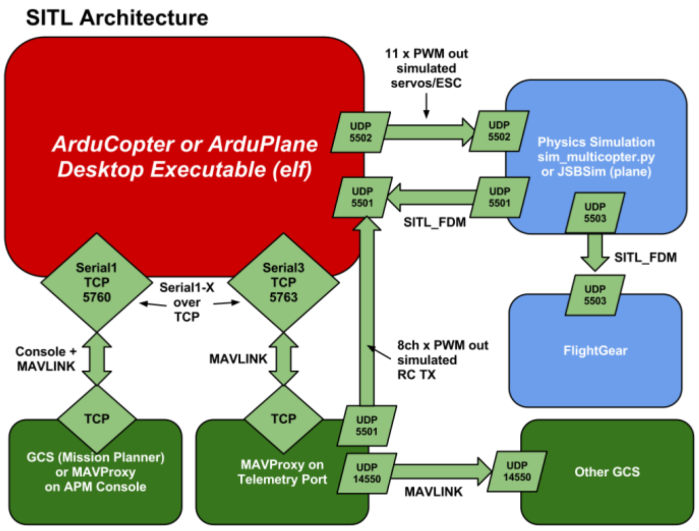

SITL Architecture
ArduCopter or ArduPlane Desktop Executable (elf)
11 x PWM out simulated servos/ESC
UDP 5502
UDP 5502
Physics Simulation sim_multicopter.py or JSBSim (plane)
UDP 5501
UDP 5501
SITL_FDM
UDP 5503
SITL_FDM
UDP 5503
FlightGear
Serial1 TCP 5760
Serial1-X over TCP
Serial3 TCP 5763
Console + MAVLINK
MAVLINK
8ch x PWM out simulated RC TX
TCP
GCS (Mission Planner) or MAVProxy on APM Console
TCP
MAVProxy on Telemetry Port
UDP 5501
UDP 14550
MAVLINK
UDP 14550
Other GCS

Figure 9.2: APM:Plane Software-in-the-loop architecture. *Image courtesy of ardupilot.org*

9.2 Coordinated Landing Simulation, No Wind

This section presents simulation results where the autonomous fixed-wing landing system developed in this thesis is tested together with the multirotor recovery system described in Moe (2016). All tests are performed five times to give enough samples to say something about the general performance. In the plots with the time, time $t = 0$ is the time when the fixed-wing UAV crosses the straight-line between the two multirotors, acting as the location of the landing net.

9.2.1 Results

In Figure 9.3 the landing path, virtual runway and fixed-wing UAV trajectory is projected on the Nort-East plane for one of the flights. The same landing path is used in all the five tests.

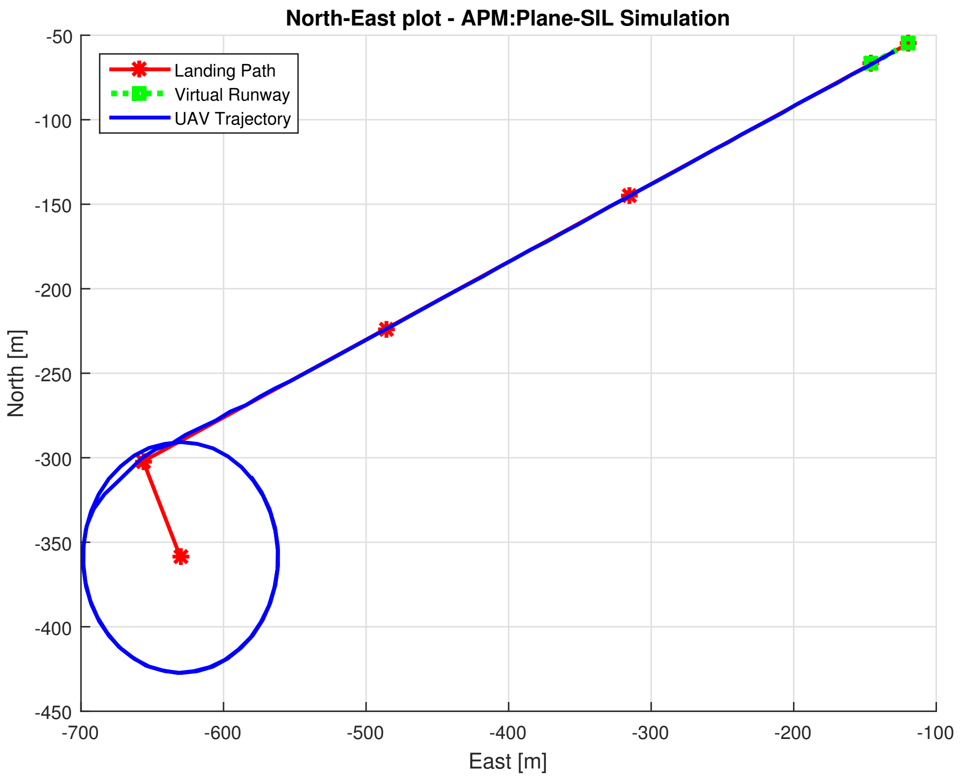


Figure 9.3: North-East trajectory of one flight, no wind

In the figure the UAV is first flying in the waiting loiter. After one round the *exit-loiter* is pressed, and the fixed-wing UAV starts the landing. The first section after the loiter is the approach phase, followed by the glideslope phase and the final approach phase. The last section is the virtual runway, where the UAV lands in the net approximately in the middle of the virtual runway.

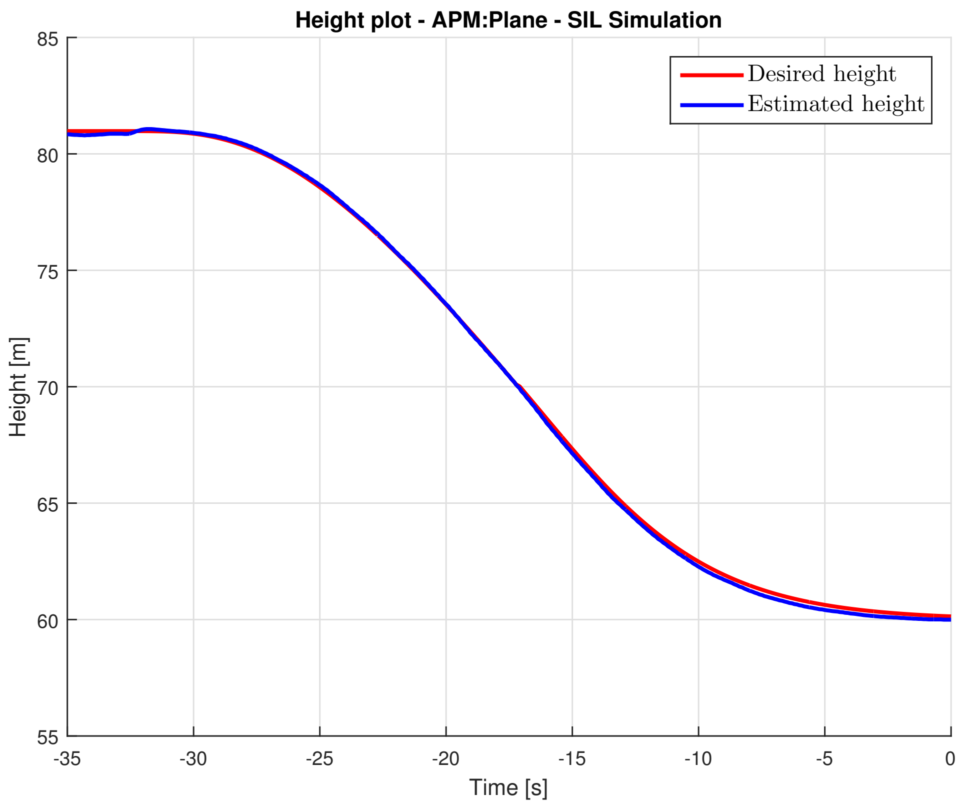
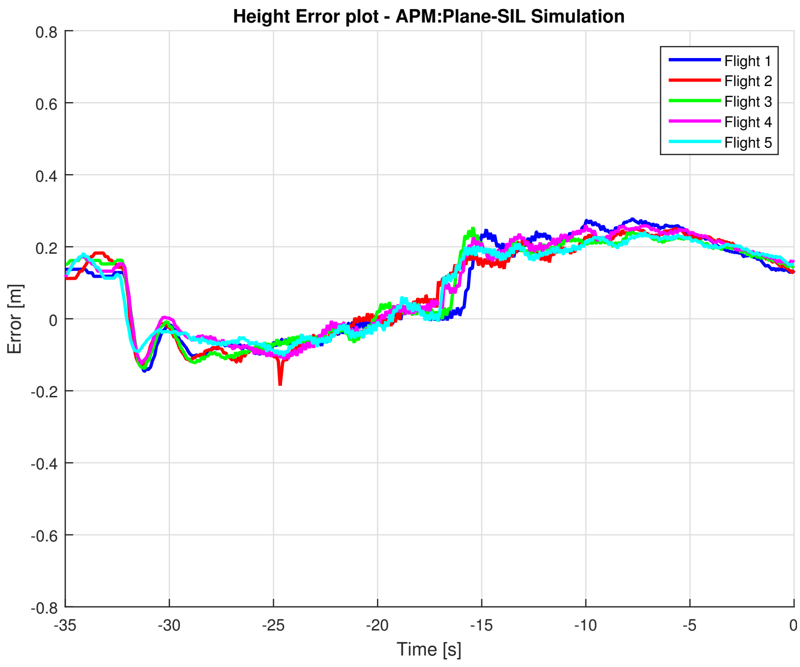


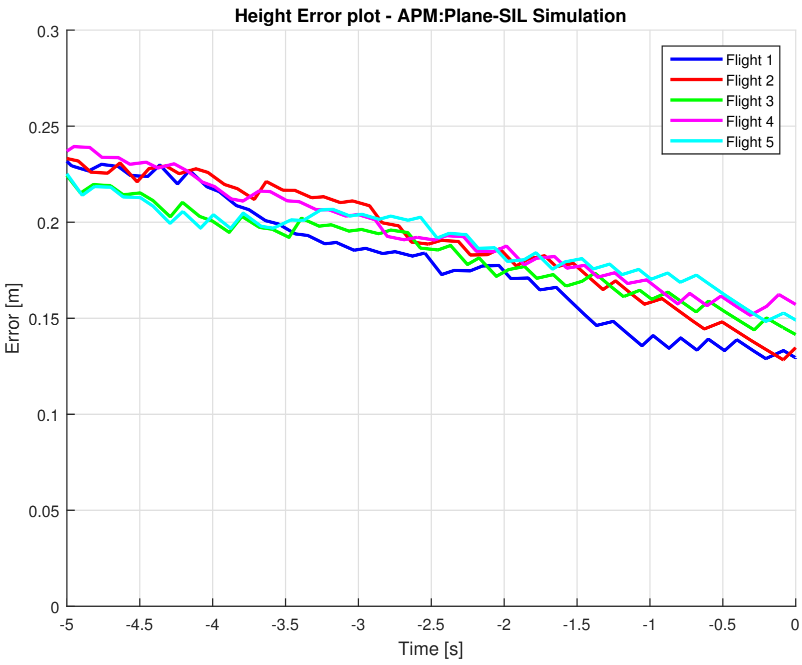
Figure 9.4: Height plot of one flight, no wind

In Figure 9.4 the desired height and the height of the UAV is shown. The glideslope path starts at $t = -30$ and ends at $t = -15$. At $t = -15$ the flare-path starts, and slowly decreases the UAV rate of descent, providing a feasible and smooth path to the virtual runway. The virtual runway is located at $60\,m$ height.

In Figure 9.5 the height error for all the five flights is shown. The performance is approximately equal for all flights, and the mean absolute error along the whole path for all flights is $13.25\,cm$, while the mean absolute error at time $t = 0$ is equal to $13.8\,cm$. For the glideslope path the error is below 10 cm, but for the curved flare-path a small increase in error is seen at $t = -15$. The small increase introduced is close to constant as the desired flight path angle changes along the flare-path. At the end of the flare-path the error is slowly decreasing.



(a) Height error of 5 flights, no wind



(b) Height error of 5 flights, last 5 seconds, no wind

Figure 9.5: Height error of 5 flights, no wind

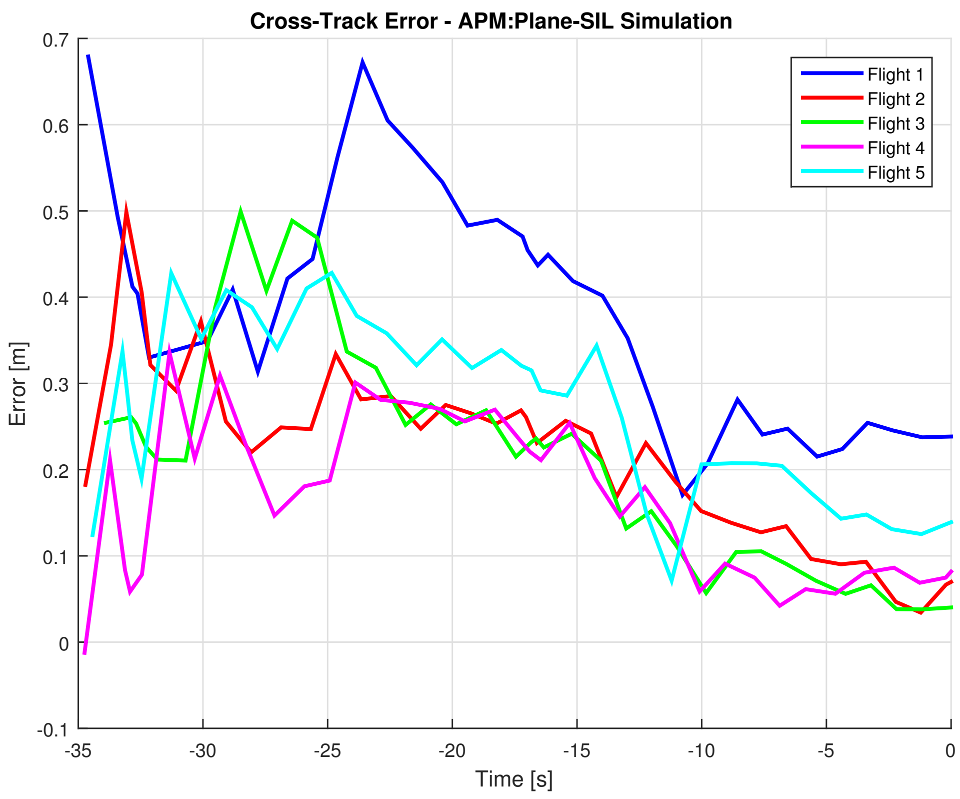
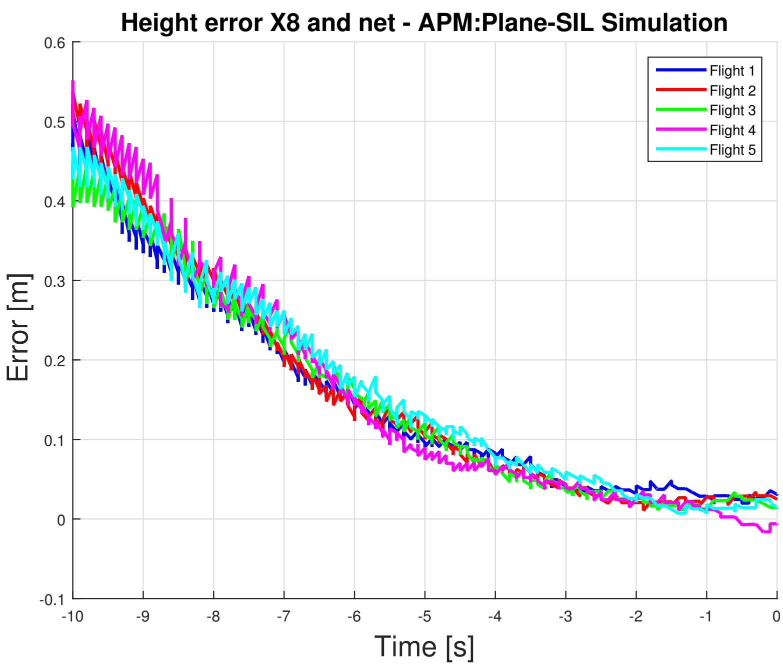


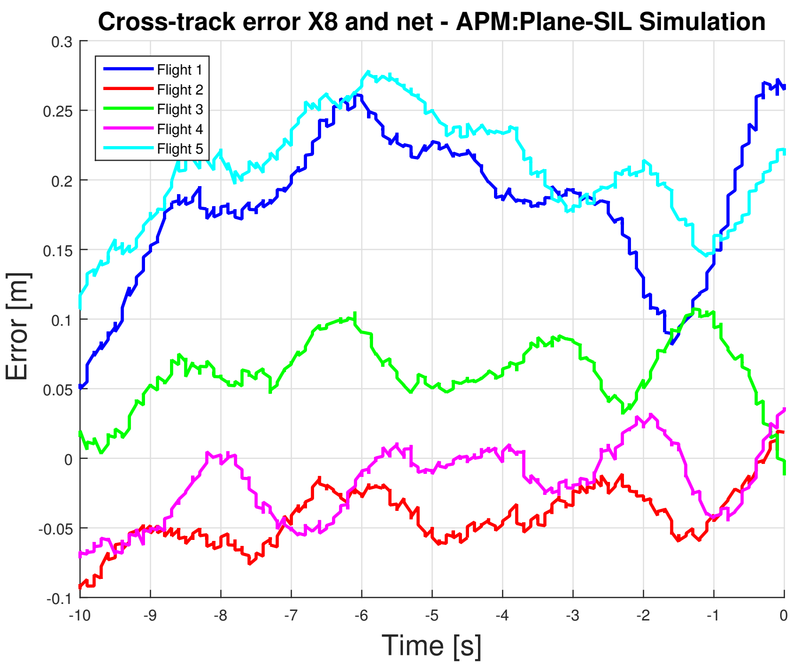
Figure 9.6: Cross-Track Error of 5 flights, no wind

In Figure 9.6 the cross-track error along the landing path is shown. The performance is also here consistent for all flights, but with some reduced performance for flight 1. The mean absolute error for the flights is $0.25\,m$. While at time $t = 0$ the mean absolute error is $0.15\,m$ for the flights.

For completeness purposes the results with the multirotor recovery system is shown in Figure 9.7. While the fixed-wing UAV aims for the middle of the virtual runway, the multirotor system tries to compensate for the fixed-wing height and cross-track errors by moving the net. The height error and cross-track error between the net center and the X8 is shown in Figure 9.7a and Figure 9.7b, respectively. Particularly the height error is reduced to a very low number, from $13\,cm$ to only $3\,cm$.

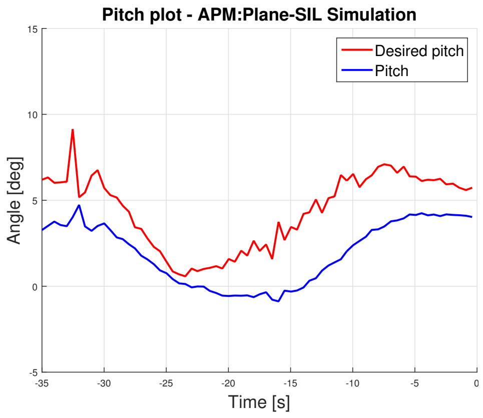


(a) Height error between X8 and net center, no wind

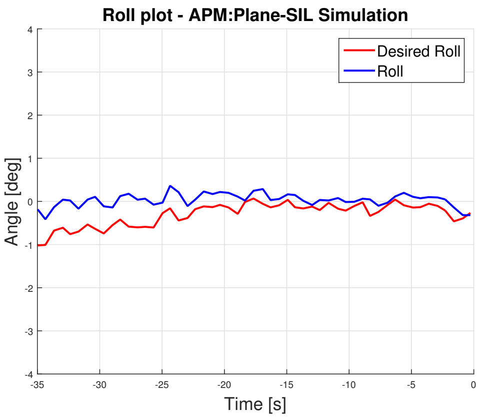


(b) Cross-track error between X8 and net center, no wind

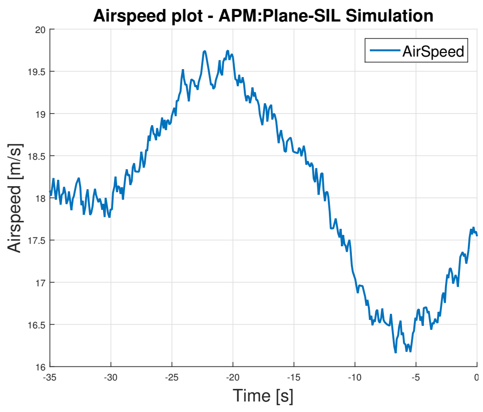
Figure 9.7: Height and cross-track error between X8 and net center.



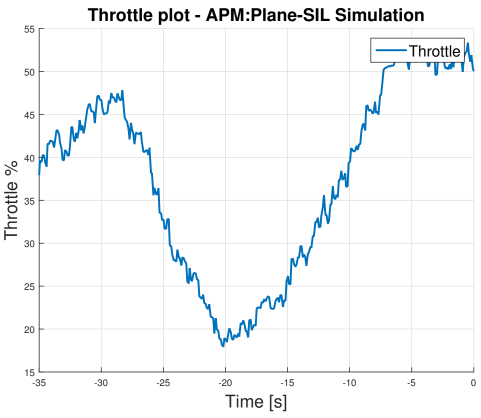
(a) Pitch and desired pitch along the landing path.



(b) Roll and desired roll along the landing path.



(c) Airspeed along the landing path.



(d) Throttle along the landing path.

Figure 9.8: Pitch, roll, airspeed and throttle along the landing path, one flight.

In Figure 9.8b both roll and desired roll is close to 0 degree. Some small oscillations can be seen, but the roll is overall close 0 degree along the whole landing path.

In Figure 9.8a some deviation between desired pitch and pitch is seen. Compared with the height plot, the biggest deviation is seen along the flare-path, when the longitudinal LOS tries to reduce the rate of descent by pitching the fixed-wing UAV upwards.

The desired airspeed is set to $18\,m/s$. At the start of the glideslope at $t = -30$, the fixed-wing UAV starts to gain airspeed. At the same time the throttle, shown in Figure 9.8d, is reduced to compensate for the increased airspeed. At $t = -20$ the airspeed starts to decrease, and at the same time the throttle is slowly increased.

9.2.2 Discussion

The SITL simulations with no wind shows overall very good results. Along the landing path the mean absolute error in height is $13.25\,cm$, while the mean absolute error in cross-track is $25\,cm$. When the fixed-wing UAV crosses the straight-line between the multirotors, and lands in the net, the mean absolute error in height is $13.8\,cm$ and $15\,cm$ for the cross-track, with respect to the virtual runway.

The largest error in height is seen during the flare-path, where the desired flight-path angle is also constantly changing, and the error reaches $25\,cm$ at most. The fact that the largest error is along the flare-path can to some degree be explained by the architecture of the longitudinal LOS extension. Here the simplification with using α_{trim} instead of α_0 reduces the performance when the desired flight path angle is constantly changing, since the simplification introduces a small steady-state error that is reliant on the integral effect in the longitudinal LOS to be eliminated.

Nevertheless, the performance of the guidance and control system is very good, with small error in both height and cross-track, and with consistent performance in all the flights.

9.3 Coordinated Landing Simulation, With Wind

The SITL simulator have the option to enable wind in the simulations for the fixed-wing UAV. Here the wind direction, speed and turbulence can be changed in order to test their effect on the flight behaviour. In this section a landing simulation was performed with wind speed set to $12\,m/s$ from north with a turbulence of $3\,m/s$. This is classified as strong breeze on the Beaufort wind force scale. Since the multirotor simulator don't have support for wind, results with the multirotor recovery system is not included. The same landing path used in the simulation with no wind, is also used in the simulation with wind.

9.3.1 Results

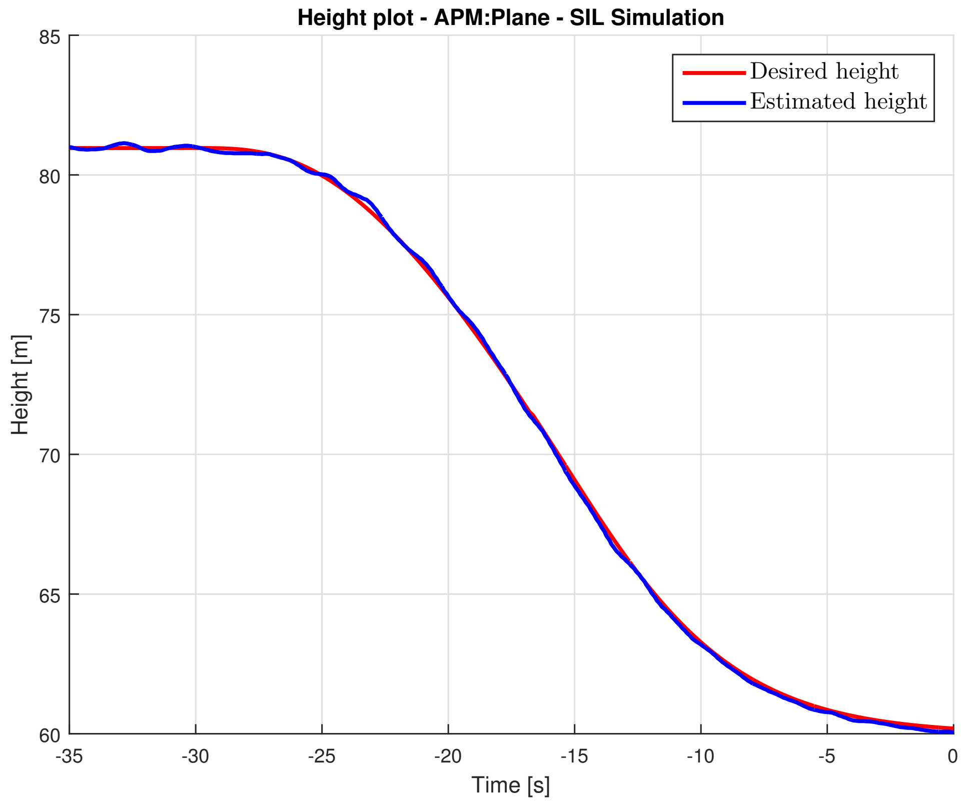
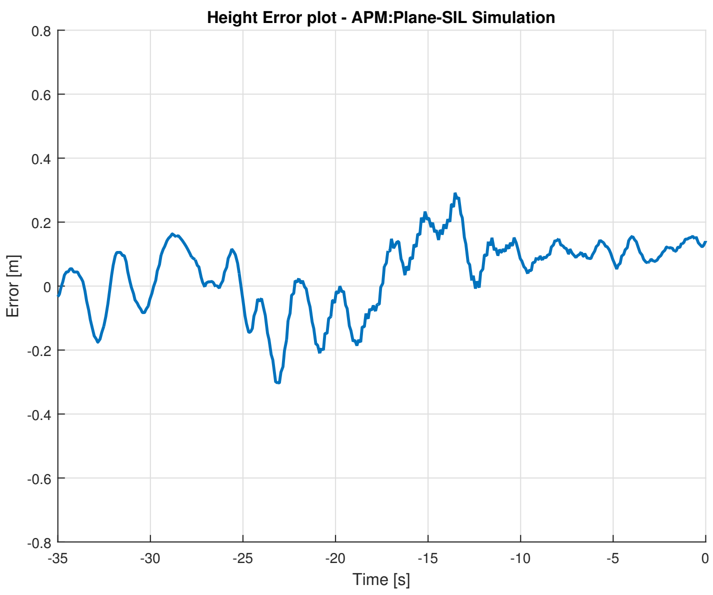
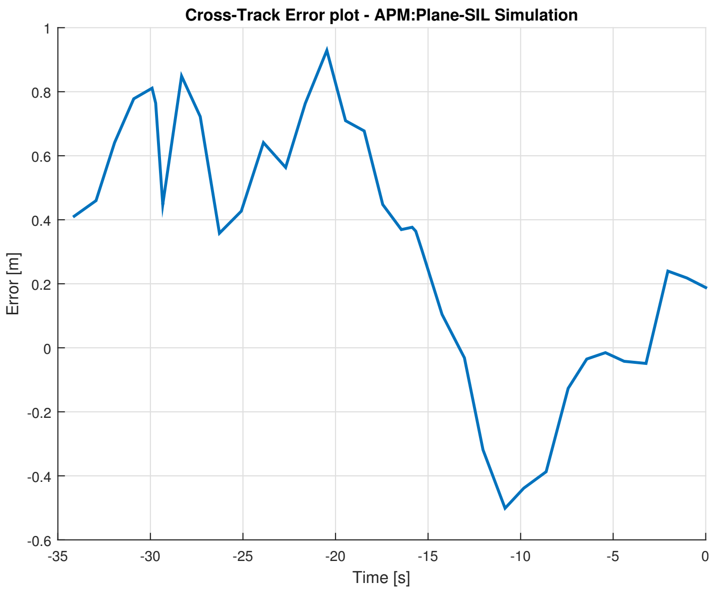


Figure 9.9: Height plot of flight with wind

In the height plot in Figure 9.9 the UAV still follows the reference quite accurately. Compared with the height plot without wind in Figure 9.4, some deviation can now be seen along the glideslope path.



(a) Height error, flight with wind

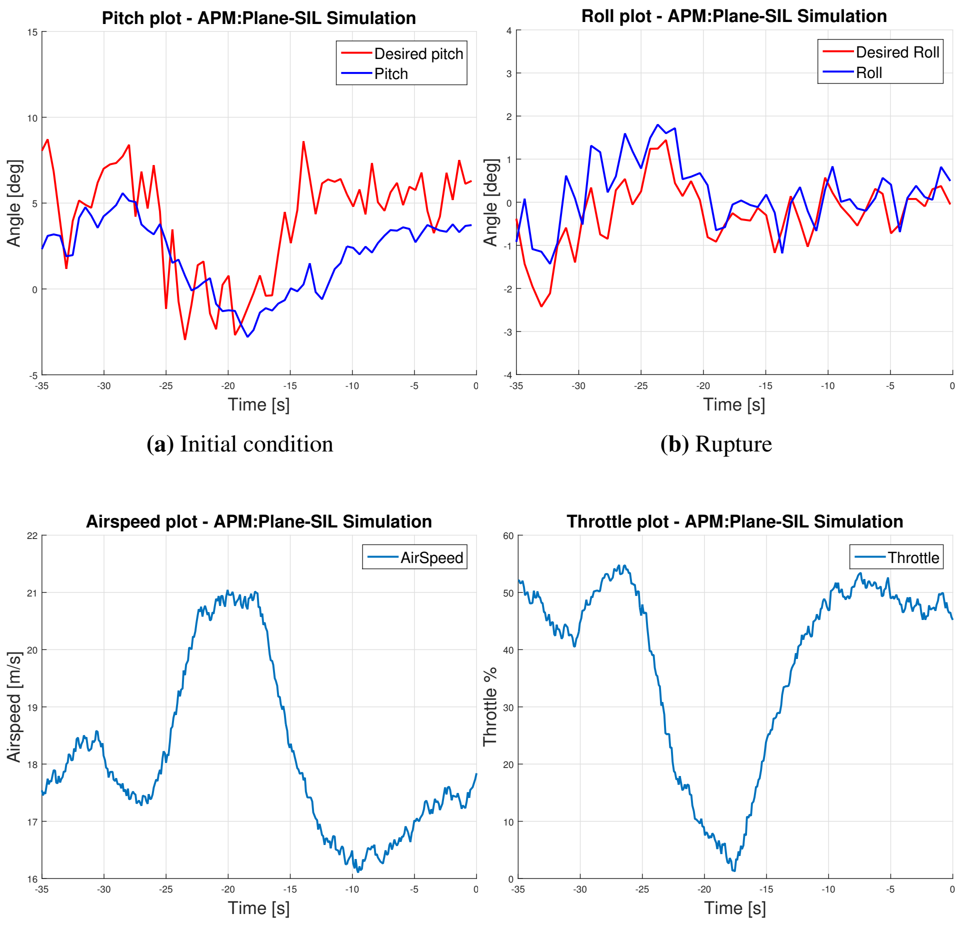


(b) Cross-track error, fight with wind

Figure 9.10: Height and cross-track error, flight with wind.

As seen in Figure 9.10a the height error now oscillates much more than in the flights without wind. However, the maximum error is for the most time within $\pm 0.2\,m$ which is approximately the same as for the flights without wind.

The cross-track error seen in Figure 9.10b is now noticeable larger compared with the simulations without wind. While the error without wind was for most of the time between $0\,m$ and $0.5\,m$, the error now varies between $0.9\,m$ and $-0.5\,m$.



(a) Initial condition

(b) Rupture

(c) DFT, Initial condition

(d) DFT, rupture

Figure 9.11: Pitch, roll, airspeed and throttle along the landing path, one flight.

In the pitch and roll plot increased oscillations is shown compared with the flights without wind. While the pitch dynamics is still slow, the the roll clearly follows the desired roll despite the increased oscillations. At the start of the glideslope path the airspeed quickly reaches $21\,m/s$, whereas the reference speed is set to $18\,m/s$. In order to compensate for the increased airspeed the throttle is dropped to almost $0\,\%$. This drop in throttle happens at the same time as the airspeed increases.

9.3.2 Discussion

The results from the simulations with wind shows that the guidance and control system handles the strong wind overall very well. This is especially true for the longitudinal LOS where the maximum error in height did not increase noticeably. For the lateral LOS some increase in the cross-track error is seen, but the overall performance of the guidance and control system is very good considering the wind speed at $12\,m/s$ and with $3\,m/s$ turbulence.

Chapter 10

Experiments

10.1 Agdenes Airfield

The flight experiments was conducted at Agdenes airfield. The airfield is located about 90 km southwest of Trondheim, and is the primary test field used by the UAV-Lab. The airfield is shown in Figure 10.1. The final flight experiments of the autonomous landing system was done May 31 and June 1. All tests were conducted as Visual Line of Sight (VLOS) operations.

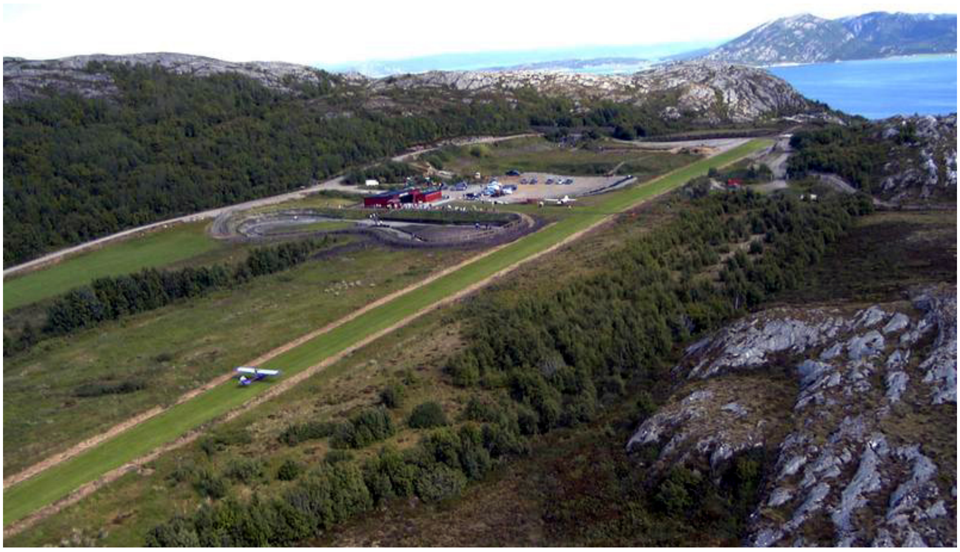


Figure 10.1: Aerial photo of Agdenes Airfield. *Image courtesy of norskeflyplasser.no*

10.2 Coordinated Landing

This section presents the results of two coordinated landing flights performed at Agdenes. Most of the available flight time was used to tune the high-level guidance system to a level where the performance of the system was mostly limited by the tuning of the low-level PID. Furthermore, these two flights presented in this section are the two last flights performed with the final tuning parameters at June 1.

10.2.1 Results

Both flights presented here uses the same landing path. The landing path and the fixed-wing UAV trajectory projected onto the North-East plane is shown in Figure 10.2 for the first flight. The UAV starts at the indicated start point, and flies half a loiter and turns towards the virtual runway. After the turning circle, the path consists of a approach phase, glideslope phase and final approach phase before it reaches the virtual runway. In the plots with the time, time $t = 0$ is the time when the fixed-wing UAV crosses the straight-line between two multirotor UAVs which are located at the virtual runway.

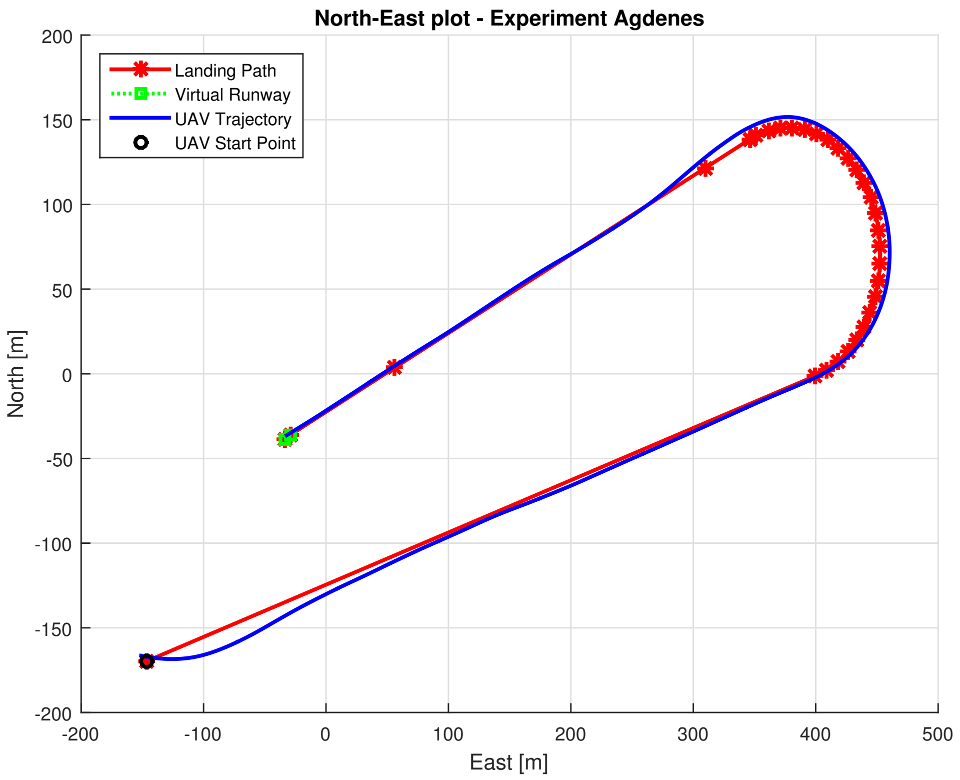
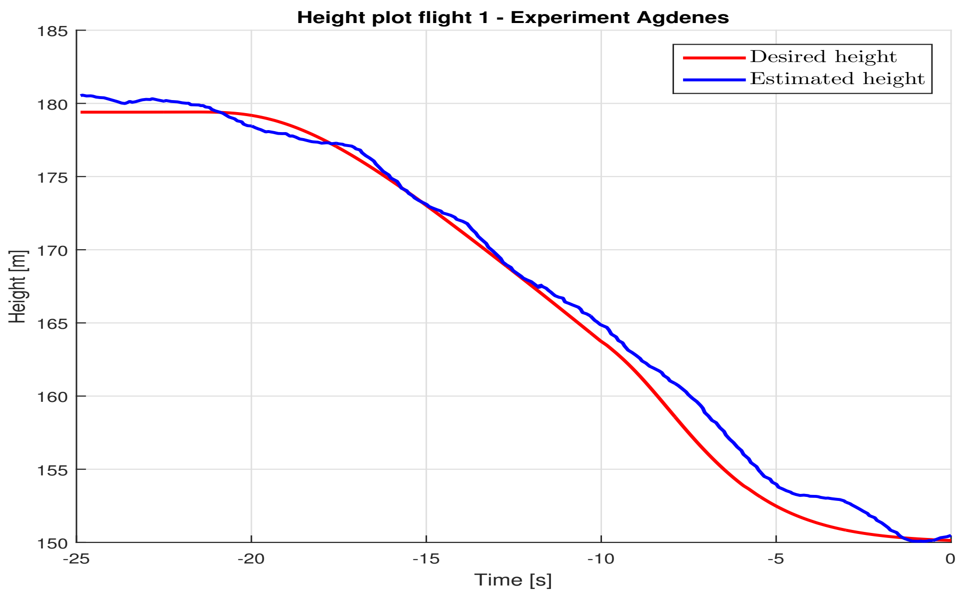
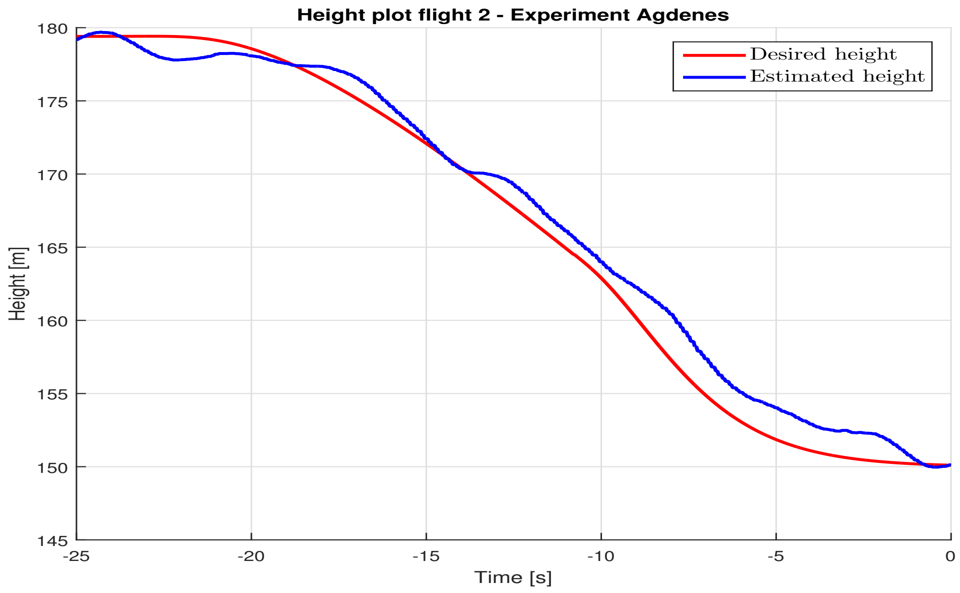


Figure 10.2: North-East Trajectory



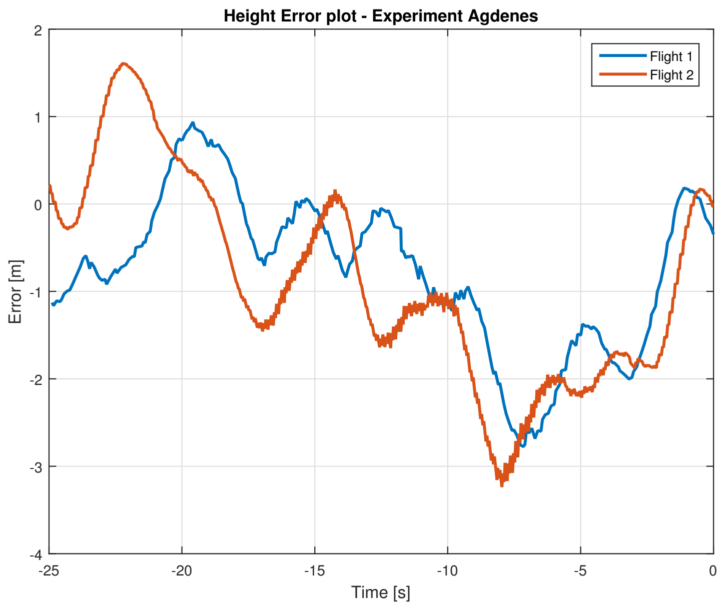
(a) Height and desired height for flight 1



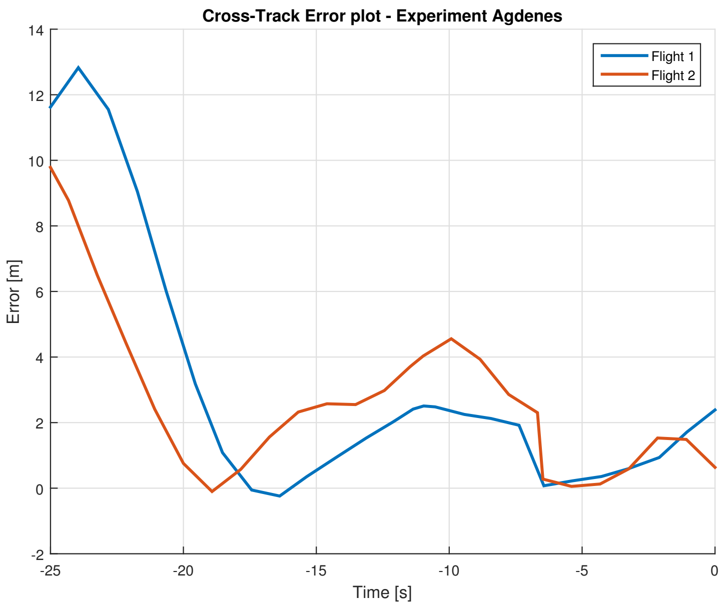
(b) Height and desired height for flight 2

Figure 10.3: Height and desired height for flight 1 and flight 2.

Figure 10.3 shows the height and desired height for both flights. Comparing the two flights, the performance along the glideslope is a little better for flight 1, while increased deviation from the reference is seen for both flights along the flare-path that starts at $t = -9$. At time $t = 0$ the deviation is minimal for both flights.



(a) Height error along landing path.



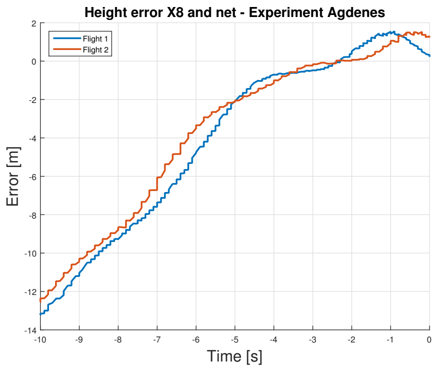
(b) Cross-track error along landing path.

Figure 10.4: Height error and cross-track error along landing path.

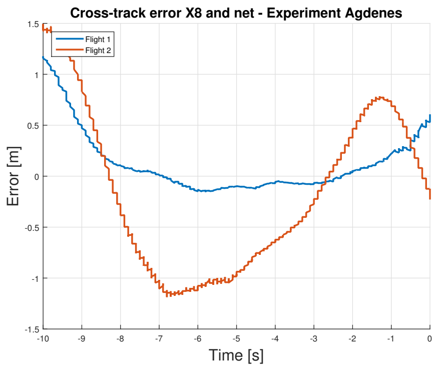
Figure 10.4a shows the height error for both flights. The mean absolute error for flight 1 is $0.9\,m$ and $1.2\,m$ for flight 2. At $t = -10$ the increased error for the flare-path is seen. When the flare-path ends, and the fixed-wing UAV enters the virtual runway the error is quickly reduced, and at time $t = 0$ the height error for flight 1 is $0.345\,m$ and $0.028\,m$ for flight 2.

Figure 10.4b shows the cross-track error along the landing path. At $t = -25$ the fixed-wing UAV is still turning towards the landing path, and struggles with following the final parts of the turning circle. This is best illustrated in the North-East plot in Figure 10.2. When tracking the straight line landing path from $t = -20$, the error quickly decreases. From $t = -20$ to $t = 0$, the mean absolute error is $1.6\,m$ for flight 1 and $1.9\,m$ for flight 2. For flight 1 a sudden increase in the cross-track error is seen at the very end of the landing path. At $t = 0$ the error is $2.4\,m$ for flight 1 and $0.6\,m$ for flight 2.

For completeness purposes two plots with the multirotor recovery system is shown in Figure 10.5. As explained previously, the fixed-wing UAV aims for the middle of the virtual runway, while the multirotor system tries to compensate for the fixed-wing UAV height and cross-track errors by moving the net. Especially the cross-track error is here greatly reduced. Information about the multirotor recovery system can be found in Moe (2016).

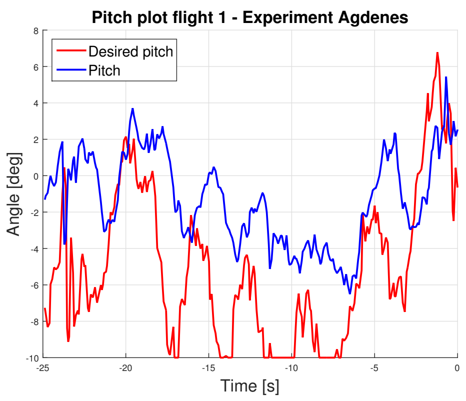


(a) Height error between X8 and net center.

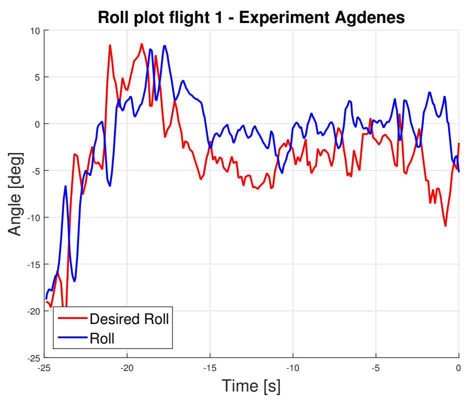


(b) Cross-track error between X8 and net center.

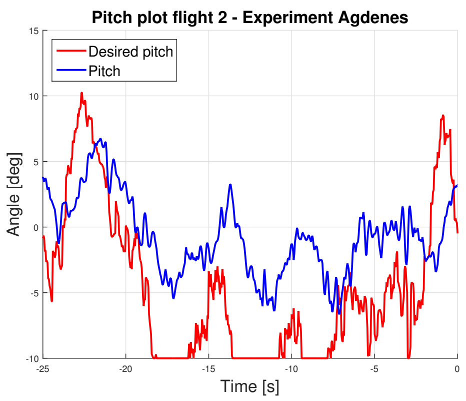
Figure 10.5: Height error and cross-track error between X8 and net-center.



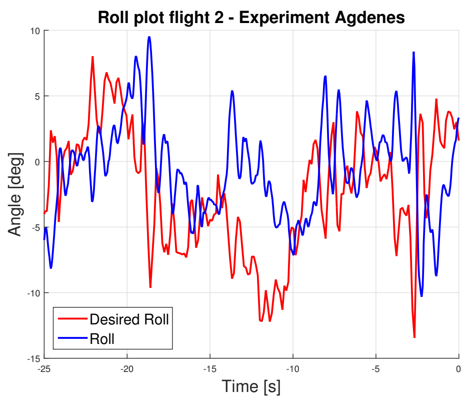
(a) Pitch and desired pitch for flight 1



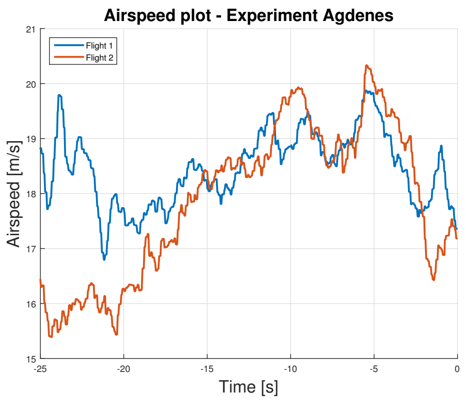
(b) Roll and desired roll for flight 1



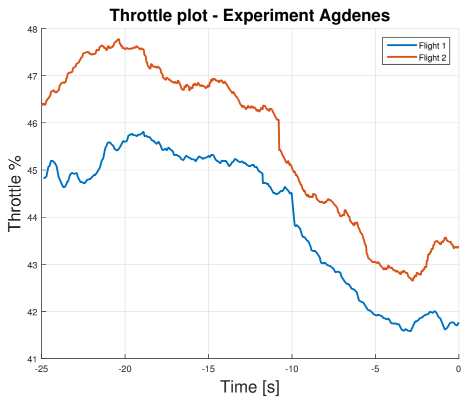
(c) Pitch and desired pitch for flight 2



(d) Roll and desired roll for flight 2



(e) Airspeed for flight 1 and flight 2



(f) Throttle for flight 1 and flight 2

Figure 10.6: Pitch, roll, airspeed and throttle along the landing path for flight 1 and flight 2.

10.2.2 Discussion

In Figure 10.6a and Figure 10.6c the pitch and desired pitch for flight 1 and flight 2 is shown. The deviation is generally large, and the desired pitch sometimes saturates at -10 degrees over several seconds waiting for the pitch to follow. From the plots the desired pitch may appear aggressive, but tuning down the aggressiveness lead to a significantly increase in the height error.

Looking at the rate of the signals, the response is much faster. Furthermore, the low-level PID seems to prioritize following the desired pitch rate over the actual desired pitch. Considering the control architecture of the low-level PID in Figure 3.2, this behaviour is not surprising. The PID is to some extent actually a rate-controller, and as explained in Section 3.1.1, the pitch error is converted to pitch rate error by multiplying it with a time constant, and then used in the inner-loop of the PID with feedback from the measured pitch rate. In order to increase the weighting on the pitch error over the pitch rate error, the K_P gain shown in Figure 3.2 must be increased. Furthermore, since tuning of the low-level PID is very time consuming, and with a limited time to perform tests, there was unfortunately no time to tune the low-level PID at Agdenes.

Some deviation is also seen in the roll dynamics. The biggest problem here is that the roll lags about 0.6 seconds behind the desired roll, which sometimes causes the roll and desired roll to be in antiphase, causing large deviations. For flight 1 this is most prominent in the last 3 seconds of the flight, where the roll and desired roll goes into antiphase, causing a 10 degrees error between the roll and desired roll. The result of this is clearly seen in Figure 10.4b, where the cross-track error at the same time rapidly increases to $2.4\,m$. Furthermore, the same tendency with following the rate over the actual desired angle is seen in the roll dynamics. With respect to Figure 3.1, a strategy to tune the PID would involve increasing the K_P gain. Another thing that can be done to decrease the cross-track error is to introduce a time-varying look-ahead distance in the lateral LOS, instead of the fixed-distance currently used. This will result in a less aggressive desired roll when the cross-track error is small.

Despite sub-optimal tuning of the low-level PID, the mean absolute error for the height is approximately $1\,m$, while the cross-track mean absolute error is about $1.7\,m$. The landing net used at Agdenes is $3\,m$ high, and $5\,m$ long, meaning that the error is within these limits. Furthermore, tuning of the low-level PID would have increased the performance of the guidance and control system significantly.

Nevertheless, the fixed-wing UAV successfully manages to fly both the glideslope and flare path in a coordinated landing path with the multirotor UAVs, demonstrating both the capabilities and feasibility of the guidance and control system developed in this thesis.

Figure 10.7: Picture of the X8 fixed-wing UAV and one multirotor UAV at Agdenes.

Part IV

Discussion and Conclusions

Chapter 11

Conclusion and Closing Discussions

11.1 Conclusion and Closing Discussions

The main goal of this thesis was to develop an autonomous landing system for a fixed-wing UAV in order to land in a net suspended by multirotor UAVs. A decoupled guidance and control system was developed and tuned, together with a system for coordinating the landing path between the fixed-wing UAV and the multirotor UAVs. The final system was tested in both simulations and flight experiments, where successful tests demonstrated both the feasibility and the performance of the system.

In the SITL simulations the system shows excellent performance, where the fixed-wing UAV successfully hit the target within 15 cm in the vertical direction, and 25 cm in the horizontal direction. This includes both simulations without wind, and simulation with $12\,m/s$ wind. Along the landing path the mean absolute error in height is 13.25 cm, while the mean absolute error in cross-track is 25 cm. Compared with the system developed by Frølich (2015), where both height error and cross-track error was within ± 1 m, the performance is significantly better with the system developed in this thesis.

The system is also tested in flight experiments at Agdenes airfield. Due to sub-optimal tuning of the low-level PID in APM:Plane, some decrease in the performance is seen. In the presented flights the mean absolute error in height is about 1m, while the mean absolute error in cross-track is about 1.7m. The conclusion drawn from the flight experiments is that the tuning of low-level PID in APM:Plane is crucial for increasing the performance and accuracy. Since tuning of the low-level PID is very time consuming, and with a limited time to perform tests, there was unfortunately no time to tune the low-level PID at Agdenes. Analytical expressions for how these controllers can be tuned based on a model is found, but the uncertainties in the current mathematical model of the X8 is too high for them to be used in real flights at the moment.

11.2 Future Work

An important step in further development of guidance and control systems for the X8 is improving the mathematical model of the X8 described in Gryte (2015). Wind tunnel tests have just recently been done, and by combining the wind tunnel data with the flight data from Agdenes, system identification techniques should be used in order to identify the aerodynamic model parameters.

Furthermore, in order to increase the performance of the guidance and control system the parameters in the low-level PID controller in APM:Plane used in the experiments at Agdenes should be tuned further. The parameter values used in the experiments are a result of the APM:Plane mode autotune. For better performance, manual tuning is a described as a must in the autotune documentation. When the final mathematical model of the X8 is fully developed, the analytical tuning expressions developed in this thesis should be tested in real flights, potentially saving several hours and days of tuning compared to manual tuning.

A fully developed mathematical model of the X8 also enables the use of control and guidance systems developed based on state-space control design methods. The methods described in Lavretsky and Wise (2013), such as the LQR and model reference adaptive control (MRAC), have great potential in increasing the performance even further.

Bibliography

3DRobotics, 2016. 3DRobotics: Pixhawk.
URL https://store.3dr.com/products/3dr-pixhawk

APM:Plane Dev Team, 2016. APM:Plane.
URL http://ardupilot.org/plane/

Balchen, J. G., Andresen, T., Foss, B. A., 2003. Reguleringsteknikk.

BeagleBoard Foundation, 2016. BeagleBone Black.
URL https://beagleboard.org/black

Beard, R., McLain, T., 2012. Small unmanned aircraft: Theory and practice. Princeton University Press.

Caharija, W., Candeloro, M., Pettersen, K. Y., Sørensen, A. J., 2012. Relative velocity control and integral los for path following of underactuated surface vessels.

Engel, A., 2010. Verification, Validation, and Testing of Engineered System. John Wiley & Sons.

Etkin, B., Reid, L. D., 1996. Dynamics of Flight: Stability and Control; Third Edition. Journal of Guidance, Control, and Dynamics.

Fortuna, J., Fossen, T. I., 2015. Cascaded Line-of-Sight Path-Following and Sliding Mode Controllers for Fixed-Wing UAVs.

Fossen, T. I., 2011. Handbook of Marine Craft Hydrodynamics and Motion Control.

Frølich, M., 2015. Automatic Ship Landing System for Fixed-Wing UAV.

Gautam, A., Sujit, P. B., Saripalli, S., 2014. A survey of autonomous landing techniques for UAVs. In: 2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings. pp. 1210–1218.

González, P. J., Boschetti, P. J., Cárdenas, E. M., Rodríguez, M., 2013. Design of a Landing Control System which considers Dynamic Ground Effect for an Unmanned Airplane.

Gryte, K., 2015. High Angle of Attack Landing of an Unmanned Aerial Vehicle.

Harkegard, O., Glad, S. T., 2000. A backstepping design for flight path angle control.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=912259

Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J., 2001. Global positioning system : theory and practice.

How, J. P., Frazzoli, E., Chowdhary, G. V., 2015. Linear Flight Control Techniques for Unmanned Aerial Vehicles. In: Handbook of Unmanned Aerial Vehicles. Springer, pp. 529—-576.

Jarzebowska, E., 2012. Model-based tracking control of nonlinear systems, 1st Edition. Chapman and Hall/CRC.

JSBSim, 2013. JSBSim.
URL http://jsbsim.sourceforge.net/JSBSim/index.html

Ju, H. S., Tsai, C. C., 2008. Glidepath command generation and tracking for longitudinal autolanding.

Klausen, K., Moe, J. B., van den Hoorn, J. C., Gomola, A., Fossen, T., Johansen, T., 2016. Coordinated Control Concept for Recovery of a Fixed-Wing UAV on a Ship using a Net Carried by Multirotor UAVs.

Lavretsky, E., Wise, K., 2013. Robust and Adaptive Control With Aerospace Applications.

LSTS, 2015a. LSTS — IMC.
URL http://lsts.pt/toolchain/imc

LSTS, 2015b. LSTS — Neptus.
URL http://lsts.fe.up.pt/toolchain/neptus

LSTS, 2015c. LSTS: Toolchain.
URL http://lsts.pt/

Martins, R., Marques, E. R. B., Sousa, J. B., Dias, P. S., Pinto, J., Pereira, F. L., 2009. IMC: A communication protocol for networked vehicles and sensors.
URL http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=5278245

MAVLink, 2016. MAVLink Micro Air Vehicle Communication Protocol.
URL http://qgroundcontrol.org/mavlink/start

Moe, J. B., 2016. Autonomous landing of Fixed-Wing UAV in net suspended by Multirotor UAVs: A Multirotor recovery system.

Nawrat, A. M., 2014. Innovative Control Systems for Tracked Vehicle Platforms.
URL http://link.springer.com/10.1007/978-3-319-04624-2

Skulstad, R., Syversen, C. L., 2014. Low-Cost Instrumentation System for Recovery of Fixed-Wing UAV in a Net.

SNAME, 1950. Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. Technical and Research Bulletin 1-5, (The Society of Naval Architects and Marine Engineers).

Sørbø, K. H., 2016. Autonomous landing of fixed wing uav in a stationary net: Path and navigation system.

Stevens, B. L., Lewis, F. L., 2003. Aircraft control and simulation.
URL http://www.worldcat.org/oclc/51751879

Valavanis, K. P., 2007. Advances in Unmanned Aerial Vehicles. Vol. 33. Springer Science & Business Media.
URL http://link.springer.com/book/10.1007/978-1-4020-6114-1

Yoon, S., Kim, H. J., Kim, Y., 2010. Spiral Landing Guidance Law Design for Unmanned Aerial Vehicle Net-Recovery. Journal of Aerospace Engineering, Proceedings of the Institution of Mechanical Engineers, Part G.
URL http://dx.doi.org/10.1243/09544100JAERO744

You, D., Jung, Y., Cho, S., Shin, H., 2012. A Guidance and Control Law Design for Precision Automatic Take-off and Landing of Fixed-Wing UAVs. AIAA Guidance, Navigation and Control Conference.
URL http://arc.aiaa.org/doi/abs/10.2514/6.2012-4674

Part V

Appendix/Appendices

Appendix A

Low-Pass Filtering Reference Model

A.1 3rd-Order Low-Pass Filtering Reference Model

In order to ensure that z_{ref}, $\dot{z}_{ref}$ and $\ddot{z}_{ref}$ are all sufficiently smooth a filter of 3rd can be used. This can be achived by cascading a mass-damper-spring system with a 1st-order low-pass filter, which gives the transfer function

$$\frac{z_{ref}}{z_d} = \frac{\omega^2}{(1+Ts)(s^2+2\zeta\omega s+\omega^2)} \tag{A.1}$$

where $T = 1/\omega$ is the time constant of the 1st-order low-pass filter. This transfer function can be written

$$\frac{z_{ref}}{z_d} = \frac{\omega^3}{s^3+(2\zeta+1)\omega s^2+(2\zeta+1)\omega^2 s+\omega^3} \tag{A.2}$$

and can be realized with the following state-space matrix and vector

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\omega^3 & -(2\zeta+1)\omega^2 & -(2\zeta+1)\omega \end{bmatrix} \tag{A.3}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \omega^3 \end{bmatrix} \tag{A.4}$$

Appendix B

APM:Plane PID Parameters

A list over the parameters in the low-level roll and pich PID in APM:Plane is here shown, together with the mapping to the block diagrams in Chapter 3.1.1.

B.1 APM:Plane Roll and Pitch PID

Parameter	Display Name	Description	Range	Default
RLL2SRV_TCONST	Roll Time Constant	This controls the time constant in seconds from demanded to achieved bank angle.	0.4 - 1.0	0.5
RLL2SRV_P	Proportional Gain	This is the gain from bank angle error to aileron	0.1 - 4.0	0.4
RLL2SRV_D	Damping Gain	This is the gain from roll rate to aileron. This adjusts the damping of the roll control loop	0 - 0.1	0.02
RLL2SRV_I	Integrator Gain	This is the gain from the integral of bank angle to aileron	0 - 1.0	0.04
RLL2SRV_RMAX	Maximum Roll Rate	This sets the maximum roll rate that the controller will demand (degrees/sec)	0 - 180	75
RLL2SRV_IMAX	Integrator limit	This limits the number of degrees of aileron in centi-degrees over which the integrator will operate	0 - 4500	3000
RLL2SRV_FF	Feed forward Gain	This is the gain from demanded rate to aileron output	0.1 - 4.0	0

Table B.1: Parameter list for the roll controller

Parameter	Display Name	Description	Range	Default
PTCH2SRV_TCONST	Pitch Time Constant	This controls the time constant in seconds from demanded to achieved pitch angle.	0.4 - 1.0	0.5
PTCH2SRV_P	Proportional Gain	This is the gain from pitch angle to elevator.	0.1 - 3.0	0.4
PTCH2SRV_D	Damping Gain	This is the gain from pitch rate to elevator.	0 - 0.1	0.02
PTCH2SRV_I	Integrator Gain	This is the gain applied to the integral of pitch angle.	0 - 0.5	0.04
PTCH2SRV_RMAX_UP	Pitch up max rate	This sets the maximum nose up pitch rate that the controller will demand (degrees/sec).	0 - 100	0
PTCH2SRV_RMAX_DN	Pitch down max rate	This sets the maximum nose down pitch rate that the controller will demand (degrees/sec).	0 - 100	0
PTCH2SRV_RLL	Roll compensation	This is the gain term that is applied to the pitch rate offset calculated as required to keep the nose level during turns.	0.7 - 1.5	1
PTCH2SRV_IMAX	Integrator limit	This limits the number of centi-degrees of elevator over which the integrator will operate.	0 - 4500	3000
PTCH2SRV_FF	Feed forward Gain	This is the gain from demanded rate to elevator output.	0.1 - 4.0	0

Table B.2: Parameter list for the pitch controller

Due to changes in the controller over time, and in order to keep the tuning parameters for old versions of APM:Plane still compatible with the newer version, the gains in Table B.1 and Table B.2 is related to the gains in Figure 3.1 and Figure 3.2 as followed

Roll

$$K_p = (\texttt{RLL2SRV\_P- RLL2SRV\_I}\star\texttt{RLL2SRV\_TCONST})\star\texttt{RLL2SRV\_TCONST} \\ \texttt{-RLL2SRV\_D+RLL2SRV\_FF} \tag{B.1}$$

$$K_i = \texttt{RLL2SRV\_I}\star\texttt{RLL2SRV\_TCONST} \tag{B.2}$$

$$K_d = \texttt{RLL2SRV\_D} \tag{B.3}$$

Pitch

$$K_p = (\texttt{PTCH2SRV\_P-PTCH2SRV\_I}\star\texttt{PTCH2SRV\_TCONST})\star\texttt{PTCH2SRV\_TCONST} \\ \texttt{-PTCH2SRV\_D+PTCH2SRV\_FF} \tag{B.4}$$

$$K_i = \texttt{PTCH2SRV\_I}\star\texttt{PTCH2SRV\_TCONST} \tag{B.5}$$

$$K_d = \texttt{PTCH2SRV\_D} \tag{B.6}$$