



Pol Padilla Moure

Terrain Following – Low Level Fixed Altitude Flying

Faculty of Engineering and Applied Sciences
Autonomous Vehicle Dynamics and Control

MSc
Academic Year: 2024–2025

Supervisor: Dr Namhoon Cho, Prof Antonios Tsourdos
August 2025



Faculty of Engineering and Applied Sciences
Autonomous Vehicle Dynamics and Control

MSc

Academic Year: 2024–2025

Pol Padilla Moure

Terrain Following – Low Level Fixed Altitude Flying

Supervisor: Dr Namhoon Cho, Prof Antonios Tsourdos
August 2025

This thesis is submitted in partial fulfilment of the
requirements for the degree of MSc.

© Cranfield University 2025. All rights reserved. No part of
this publication may be reproduced without the written
permission of the copyright owner.

Academic Integrity Declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

This thesis studies a terrain following capability for fixed-wing tier 1 UAS ($< 25\text{ kg}$), designing a system to perform very low-level altitude flights adapting to the terrain contour and avoiding obstacles. It is the first work on terrain following combining uncertain digital elevation models (DEM) and real-time observations with risk-awareness. An innovative dual-stage system is proposed, using a cubic B-spline curve to generate an upper envelope combining DEM datasets and direct ground mapping measurements through onboard sensors, and a non-linear model predictive control (NMPC) to track the reference envelope with altitude range constraints. The system is designed for real-time implementation, employing moving window predictions, and an aggressiveness modulation to improve solver times while safely overcoming obstacles. The cubic B-spline DEM-Obstacle envelope is a geometric object that is found through solving a quadratic program with guaranteed convergence. The NMPC uses the full non-linear longitudinal dynamic model of the UAS to provide optimal vertical guidance and control to the non-linear underactuated platform, tracking the envelope. The performance is critically sensitive to the rangefinder angular uncertainty, forcing higher flight paths while maintaining minimal collision risk. Chance constraint formulation in the envelope allows improvements through moderate risk allowance, balancing a trade-off between risk and performance. Although the obstacle avoidance sensors are essential, the best performance is achieved using both a quality elevation dataset and sensor suite, employing LiDAR DEM and LiDAR rangefinders.

Keywords:

avoidance, basis spline (B-spline), control, digital elevation model (DEM), envelope, ground, guidance, navigation, non-linear model predictive control (NMPC), nap-of-the-earth (NOE), obstacle, path planning, risk, sensors, unmanned aerial system (UAS).

Acknowledgements

This thesis was made possible through the support of several individuals who, in different ways, contributed to my personal and academic journey.

I am deeply grateful to my academic supervisor, Dr Namhoon Cho, for his invaluable guidance and feedback throughout the course of this research, helping to steer it in the right direction. I also wish to acknowledge the support of Prof Antonios Tsourdos and the Centre for Autonomous and Cyber-Physical Systems, whose vision and encouragement have been essential throughout this academic year.

My sincere thanks are extended to my industry supervisor, Mr Andrew Tebay, and to BAE Systems for their support. Their feedback and insight were essential to understanding the current needs and challenges faced in the engineering field explored in this study.

I am also grateful to my friends: to Carlos, for his friendship and excellent work ethic; to Max, for keeping the beat and sharing the joy of music; and to Dr Juanjo Alins, who has since become a friend, and whose guidance has shaped my growth for the better during these years at university.

Finally, I owe a special debt of gratitude to my family for their unwavering support, patience, and love, without which this thesis would not have been possible. They have always been there, taking care of me while I focused on my studies. This thesis marks the close of one chapter of my journey, and the beginning of the next.

A mis abuelos, Fernando y Gerardo.

Contents

Academic Integrity Declaration	i
Abstract	ii
Acknowledgements	iii
Contents	iv
List of Figures	vi
List of Tables	x
List of Abbreviations	xi
1 Introduction	1
1.1 Object	1
1.2 Scope	1
1.3 Requirements	2
1.4 Rationale	3
2 Literature Review	5
2.1 Terrain Following Capabilities	5
2.2 Rotary-Wing Terrain Following Methods	6
2.3 Fixed-Wing Terrain Following Methods	9
2.3.1 Design Based on Geometric Methods	9
2.3.2 Design Based on Optimal Control Theory	11
2.4 TF Method with DEM and Obstacle Sensors	14
3 Background	15
3.1 Fixed-Wing Tier 1 UAS System and Dynamics	15
3.1.1 Dynamic Model of the Fixed-Wing UAS	16
3.1.2 Trimming and Dynamic Stability Modes	20
3.1.3 Model Verification	22
3.2 Digital Elevation Models	25
3.3 Real-Time Direct Ground Mapping	28
3.3.1 Navigation Sensors	29
3.3.2 Obstacle Avoidance Sensors	31
3.4 Model Predictive Control	33

3.4.1	Linear MPC	34
3.4.2	Non-Linear MPC	35
4	Methodology	37
4.1	Terrain Following System and Architecture	37
4.2	Simulation Environment	38
4.3	Cubic B-Spline DEM-Obstacle Envelope	42
4.3.1	Cubic B-spline Curves	43
4.3.2	Chance Constraint	44
4.3.3	Convex OCP Definition	46
4.4	TF Non-Linear Model Predictive Control	47
4.4.1	Non-Convex OCP Definition	48
5	Results	50
5.1	Evaluation of the Stair and Cubic B-spline Envelope	50
5.1.1	Stair Envelope Simulation	52
5.1.2	Cubic B-spline Envelope Simulation	54
5.2	Sensitivity Analysis on Uncertainty Degradation	57
5.2.1	DEM Uncertainty	59
5.2.2	Navigation Horizontal Position Uncertainty	60
5.2.3	Navigation Vertical Position Uncertainty	61
5.2.4	Navigation Pitch Angle Uncertainty	62
5.2.5	Sensor Distance Uncertainty	63
5.2.6	Sensor Angle Uncertainty	64
5.3	Risk Allowance Effect on Terrain Following	65
5.3.1	DEM Risk Allowance	66
5.3.2	Obstacle Measurements Risk Allowance	67
6	Evaluation	68
6.1	Capabilities of the Terrain Following System	68
6.2	Sensor Suite Selection for a Terrain Following UAS	70
6.2.1	Terrain Following Simulation of Tier 1 UAS	72
7	Conclusions	73
A	Ethical Approval Letter	80
B	Software Codes	81
B.1	Cubic B-spline Envelope	81
B.2	UAS Longitudinal Dynamics	85
B.3	Terrain Following NMPC	89

List of Figures

1.1	Panavia Tornado terrain following operation and envelope generation [2].	3
1.2	Visual size comparison diagram between a Panavia Tornado jet aircraft and a generic tier 1 UAS platform (adapted from Wikimedia Commons, public domain).	4
2.1	Terrain Following (TF) problem representation.	6
2.2	Terrain Avoidance (TA) problem representation.	6
2.3	Diagram of the laser gimbal system showing the geometric relations to the ground for terrain following control of a quadrotor [9].	7
2.4	Terrain following control loop and sensor fusion altitude estimation for a quadrotor [10].	7
2.5	Terrain following control based on digital elevation model for quadrotors [11].	8
2.6	Path planning for a quadrotor using a digital surface model (DSM) [13]. .	8
2.7	Spline RRT variant with dynamical feasibility trajectory generation in two scenarios, with minimum cost path solution (red) [16].	9
2.8	Terrain following trajectories and altitude profile through Dubins RRT with dynamic feasibility and ICS avoidance, between hovering circles over waypoints [14].	10
2.9	Vertical terrain following smoothing path planning trajectory examples. .	10
2.10	Geometric terrain following path planning trajectories through pull-up and pull-down manoeuvres [15].	11
2.11	Transformation from physical DEM (a) to a reference plane state (b) and terrain following profile (c) [19].	11
2.12	Path planning trajectory for terrain following through optimal control method using Direct Collocation to solve NLP, using two different integration schemes [7].	12
2.13	Optimal control method through linearised flight dynamics and preview control to follow reference waypoints [20].	13
2.14	Terrain following MPC for real-time reference trajectory tracking and obstacle avoidance, through altitude and region constraints [8].	13
3.1	Skywalker X8 flying wing UAV [22].	15
3.2	Inertial Earth axis ($\vec{x}$, $\vec{z}$) and body axis ($\vec{x}_b$, $\vec{z}_b$) of small fixed-wing UAS. .	18
3.3	Pole-zero plot of the linearised longitudinal dynamics around the trimmed steady level flight at 17 m/s of the Skywalker X8 UAS model.	22

3.4	Short-period oscillation mode of the Skywalker X8 UAS longitudinal motion model.	23
3.5	Phugoid mode —long period— of the Skywalker X8 UAS longitudinal motion model.	23
3.6	Dynamic response to an increase in positive —downward— elevator deflection angle of the Skywalker X8 UAS longitudinal motion model. . . .	24
3.7	Dynamic response to an increase of thrust level of the Skywalker X8 UAS longitudinal motion model.	24
3.8	Example graph of a 2D Digital Surface Model (DSM).	25
3.9	Example graph of a 2D Digital Terrain Model (DTM).	25
3.10	Digital elevation model visualization of a geotiff tile from the GMTED2010 database, over the Mediterranean Sea.	26
3.11	Digital elevation model visualization of a geotiff tile from the IFSAR Alaska DTM database, in north-west Alaska.	27
3.12	Digital elevation model visualization of a geotiff tile from the 3DEP DTM database, over the mountainous Teton Range.	27
3.13	Direct ground mapping and obstacle measurement.	29
3.14	Generated trajectory and speed profile with MPC for a quadcopter. Black circles account for obstacles and dashed lines for the constrained region around them [43].	34
3.15	MPC definition and performance diagram [41].	35
3.16	Optimisation problem types and graphical representation of state-control space domain for prediction [41].	35
3.17	Tracking performance comparison of an oblique trajectory between LMPC and NMPC on an underactuated non-linear physical plant [45].	36
4.1	Dual-stage terrain following system architecture.	37
4.2	Dual-stage terrain following system graphical representation.	38
4.3	Flowchart of the simulations for the terrain following system.	39
4.4	Virtual terrain generated through the Diamond-Square fractal algorithm, with a projected NOE flight between a start and end waypoints.	40
4.5	Elevation profile and slope angle of the projected NOE path.	41
4.6	Online simulation environment implementing elevation virtual terrain and additional obstacles, with three rangefinder sensors onboard and their measurements.	41
4.7	Stair DEM-Obstacle terrain envelope.	42
4.8	Cubic b-spline DEM-Obstacle terrain envelope graph, with high and low aggressiveness effect.	43
4.9	Uniform cubic basis functions example for $n + 1 = 7$ control points [56]. . .	44
4.10	Risk allowance scenarios on the DEM-Obstacle envelope.	46
4.11	DEM-Obstacle envelope generated through QP using CasADi.	47
4.12	Terrain following NMPC graphical representation.	47
4.13	Terrain following NMPC prediction horizon through SQP using CasADi. . .	49
5.1	Terrain following simulation instance using stair DEM-Obstacle envelope. .	53
5.2	Terrain following simulation instance using stair DEM-Obstacle envelope, with NMPC solver failure.	53

5.3	Terrain following simulation instance using stair DEM-Obstacle envelope, after NMPC convergence and recovery.	53
5.4	Trajectory profiles from the stair DEM-Obstacle envelope simulation. . .	54
5.5	State and control logs from the stair DEM-Obstacle envelope simulation. .	54
5.6	Terrain following simulation instance using cubic B-spline DEM-Obstacle envelope.	55
5.7	Terrain following simulation instance using cubic B-spline DEM-Obstacle envelope, with newer measurements on the tree obstacle.	55
5.8	Terrain following simulation instance using cubic B-spline DEM-Obstacle envelope, with fitted DEM-Obstacle envelope over the tree obstacle. . . .	55
5.9	Trajectory profiles from the cubic B-spline envelope simulation.	56
5.10	State and control logs from the cubic B-spline envelope simulation. . . .	56
5.11	Terrain following simulation instance with uncertain DEM and sensors. .	58
5.12	Trajectory profiles from the DEM uncertainty sensitivity simulations. . .	59
5.13	Tracking and deviation to the baseline envelope from the DEM uncertainty sensitivity simulations.	59
5.14	Trajectory profiles from the navigation horizontal position uncertainty sensitivity simulations.	60
5.15	Tracking and deviation to the baseline envelope from the navigation horizontal position uncertainty sensitivity simulations.	60
5.16	Trajectory profiles from the navigation vertical position uncertainty sensitivity simulations.	61
5.17	Tracking and deviation to the baseline envelope from the navigation vertical position uncertainty sensitivity simulations.	61
5.18	Trajectory profiles from the navigation pitch angle uncertainty sensitivity simulations.	62
5.19	Tracking and deviation to the baseline envelope from the navigation pitch angle uncertainty sensitivity simulations.	62
5.20	Trajectory profiles from the rangefinder sensor distance uncertainty sensitivity simulations.	63
5.21	Tracking and deviation to the baseline envelope from the rangefinder sensor distance uncertainty sensitivity simulations.	63
5.22	Trajectory profiles from the rangefinder sensor angle uncertainty sensitivity simulations.	64
5.23	Tracking and deviation to the baseline envelope from the rangefinder sensor angle uncertainty sensitivity simulations.	64
5.24	Trajectory profiles from the DEM risk allowance simulations.	66
5.25	Tracking and deviation to the baseline envelope from the DEM risk allowance simulations.	66
5.26	Trajectory profiles from the obstacle measurement risk allowance simulations.	67
5.27	Tracking and deviation to the baseline envelope from the obstacle measurement risk allowance simulations.	67
6.1	Components of the commercial sensor suite for a terrain following tier 1 UAS.	71

6.2	Trajectory profiles from the tier 1 UAS with a commercial sensor suite. . .	72
A.1	CURES approval letter.	80

List of Tables

3.1	Physical properties of the Skywalker X8 UAV.	16
3.2	Atmospheric and environmental properties.	16
3.3	Propulsion motor parameters: Hacker A40-12 S V2 14-pin KV610 [24]. .	16
3.4	Aerodynamic coefficient parameters.	17
3.5	US Geographical Survey Digital Elevation Model Summary.	28
3.6	Error values in GNSS-INS navigation techniques selected for terrain fol- lowing.	31
3.7	Characteristics of passive perception and obstacle avoidance sensors for UAS.	32
3.8	Characteristics of active perception and obstacle avoidance sensors for UAS.	32
3.9	Parameters from a RADAR and LiDAR sensor adequate for low-level terrain following, with detection range 0-200 m [39].	33
5.1	Navigation and sensor suite parameters.	50
5.2	Cubic B-spline DEM-Obstacle envelope parameters.	51
5.3	Terrain following NMPC parameters.	51
5.4	Virtual terrain parameters.	52
5.5	Standard deviations used for the sensitivity analysis.	57
5.6	Uncertainty sensitivity analysis summary on worst case scenarios, provid- ing the minimum above ground level AGL_{min} heights, minimum D_{min} and maximum D_{max} deviations and RMSE to the baseline envelope profile. . .	58
5.7	Risk allowance terrain following simulation parameters.	65
5.8	Risk allowance summary, providing the minimum above ground level AGL_{min} heights, minimum D_{min} and maximum D_{max} deviations and RMSE to the baseline.	65
6.1	Possible commercial sensor suite for a terrain following tier 1 UAS. . . .	71
6.2	Standard deviations estimated from the commercial sensor suite of Table 6.1.	71
6.3	Tier 1 UAS with commercial sensor suite performance summary, provid- ing the minimum above ground level AGL_{min} heights, minimum D_{min} and maximum D_{max} deviations and RMSE to the baseline.	72

List of Abbreviations

AoA	Angle of Attack
AGL	Above Ground Level
AUV	Autonomous Underwater Vehicles
DEM	Digital Elevation Model
DoF	Degrees-of-Freedom
DRL	Deep Reinforcement Method
DSM	Digital Surface Model
DTM	Digital Terrain Model
DVL	Doppler Velocity Log
GCP	Ground Control Point
GNSS	Global Navigation Satellite System
ICS	Inevitable Collision State
INS	Inertial Navigation System
LiDAR	Light Detection and Ranging
LMPC	Linear Model Predictive Control
LQR	Linear Quadratic Regulator
LRF	Laser Rangefinder
MPC	Model Predictive Control
NMPC	Non-linear Model Predictive Control
NLP	Non-Linear Programming
NOE	Nap-of-the-Earth
OCP	Optimisation Constrained Problem
QP	Quadratic Programming
RADAR	Radio Detection and Ranging
RRT	Rapidly Exploring Random Tree
RTI	Real-Time Iteration
SLAM	Simultaneous Localisation and Mapping
SQP	Sequential Quadratic Programming
TA	Terrain Avoidance
TF	Terrain Following
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 Object

This IRP aims to study low-level, fixed-altitude terrain following capability for a tier 1 UAS platform with a maximum take-off weight of less than 25 kg. The system should autonomously follow a predefined route over rugged terrain while maintaining a constant height, or range of height, above ground level. Survivability is achieved through adequate collision avoidance and proximity to the ground, hiding from enemy detection.

The main objective is to design and evaluate a terrain following system that meets the constraints of low cost, low weight, and low complexity, in line with tier 1 UAS requirements. The project involves the development of guidance and control algorithms to minimise the tracking error to the desired trajectory. Through sensitivity analysis, the effect of sensor uncertainties and the balance between risk levels allowed and performance must be evaluated. Thereafter, adequate sensor suites for the terrain following system are selected according to the aforementioned aims.

Ultimately, the project investigates the trade-off between system complexity, sensor quality, and incursion mission survivability.

1.2 Scope

The project includes all the essential studies and work packages required to develop a vertical terrain following system for fixed-wing UAS, coupled with technical explanations needed for understanding the results. Briefly listed, the scope encompasses:

- Literature review and background for the state-of-the-art:
 - study on general terrain following techniques and examples, with survivability metrics of low-level terrain following.
 - path-planning, guidance and control algorithms for terrain following, identifying the computational complexity and amount of input data required.
 - fixed-wing tier 1 UAS vertical longitudinal dynamics, for platform vehicle definition to implement a terrain following capability.
 - navigation and obstacle avoidance sensors, including resolution, complexity, weight and price.

- Development of the Terrain Following Capability:
 - architecture of a vertical terrain following system.
 - terrain envelope generation with DEM and obstacle measurements.
 - uncertainty propagation and risk-aware chance constraint formulation.
 - non-linear MPC controller for tracking of the terrain envelope trajectory.
- Evaluation of the terrain following performance:
 - simulation of the system and validation of the algorithm safety.
 - performance sensitivity of DEM and sensor uncertainties.
 - effect of risk allowance on performance and safety margins, as a trade-off.
 - evaluation of the system's capabilities and limitations for real implementation.
 - selection of possible off-the shelf sensors for a terrain following tier 1 UAS.

Design processes related to aerodynamics, structure, and propulsion of the UAS are not studied, using instead a reference dynamic model upon which to develop the terrain following capability. This is also done with the sensor modelling, employing a simplified uncertain model with a known Gaussian distribution.

The thesis is mainly comprised in this report, containing the essential state-of-the-art, background, methodology, results, and evaluation. In addition, a presentation and an industrial poster are provided for outreach and dissemination of the study.

1.3 Requirements

The development of a terrain following capability for tier 1 UAS platforms is subjected to certain requirements, to narrow down the research objective, considering the platform type, algorithm efficiency, data availability and hardware restrictions:

- the terrain following capability must ensure vertical avoidance of obstacles while maintaining the desired altitude range over the ground.
- the selected sensor suite should be inexpensive in price, lightweight, compact (to be installed on a tier 1 UAS < 25 kg) and prioritise simplicity.
- the path planning and control tracking algorithms should provide the sufficient computational efficiency to be executed in a lightweight onboard computer.
- the tier 1 UAS must be a fixed-wing platform.
- the terrain used for validation should be non-flat.
- the onboard computer of the UAS can have a saved height-map knowledge (digital elevation model - DEM), although incomplete, to perform obstacle avoidance.
- the key objective is incursion survivability: closeness to the ground, without collision, to avoid enemy detection.

Concerning the study and project development:

- development of the project should be carried out using open source software.
- dedication threshold between 650h and 750h, part of MSc at Cranfield University.

1.4 Rationale

Terrain following (TF) aerial vehicles autonomously fly at low altitudes, tracking the ground's surface without crashing into obstacles. This technology has been developed since the late 20th century, with civilian applications like reconnaissance or search and rescue, but heavily linked to military and defence industry [1]. This has been seen on sea skimming missiles or close air support (CAS) aircraft that must evade the enemy's countermeasures during incursion missions into heavily contested airspace.

One of the most notable examples is the Panavia Tornado, an aircraft first flown in 1974 and designed through European cooperation —consortium formed by BAE Systems, Airbus and Leonardo. It is a multi-role fighter-bomber combat aircraft with a tandem (two seat) configuration, twin 3-spool turbofan engines and a distinctive variable-sweep (between 25° and 67° of sweep) wing design for subsonic and supersonic flight. The aircraft excels in low level attacks at high speeds, doted with “fly-by-wire” controls that enable the implementation of an automatic low altitude terrain following system to enhance the penetration capability into enemy airspace [2].

This remarkable terrain following system relied on a large, heavy and expensive sensor suite. It operated with two powerful forward and downward facing radars: a ground mapping radar (GMR) for long range terrain mapping, and the terrain following radar (TFR) to feed elevation data to the flight controller. Additionally, it incorporated a Doppler navigation radar for precise determination of acceleration and speed, and a radio altimeter to keep track of the aircraft's current height. The powerful sensor suite, integrated with the flight controller and the fly-by-wire autopilot, enabled a high-speed, low-level and all-weather terrain following capability within the flight envelope limits, performing collision avoidance. The terrain following system is shown in Figure 1.1, where the aircraft must keep flight above the generated CRAM envelope line —requiring manual override if breached [2].

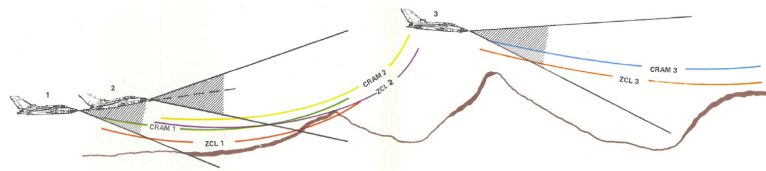


Figure 1.1: Panavia Tornado terrain following operation and envelope generation [2].

Modern aircraft, such as the B-2 Spirit or the F-35 Lightning II, are no longer optimised for low altitude flight, but for high altitude incursion. Instead of masking the radar signature behind terrain clutter, they rely on stealth technology to keep the aircraft safe and undetected. This technology is very expensive, due to the manufacturing process and the coating materials involved, and not suitable for smaller unmanned aircrafts.

During the last decades, and heavily accelerated during recent conflicts, the use of unmanned aerial systems (UAS) has grown exponentially. This rapid development has opened possibilities for a variety of civilian and defence industry applications, specially related with surveillance, search and inspection. These vehicles, compared to traditional aircrafts can have very reduced sizes and weights and, consequently, costs [3].

As such, the inexpensive nature of small tier 1 UAS (less than 25 kg) makes them unsuitable for stealth technology application due to its prohibitive cost in expendable platforms. As an alternative, to obtain penetration and incursion capabilities with tier 1 UAS, terrain following can be employed. Additionally, an UAS with active obstacle avoidance also presents many civilian advantages, as the current allowed airspace (beneath 120 m AGL) is cluttered with dangers and structures [4].

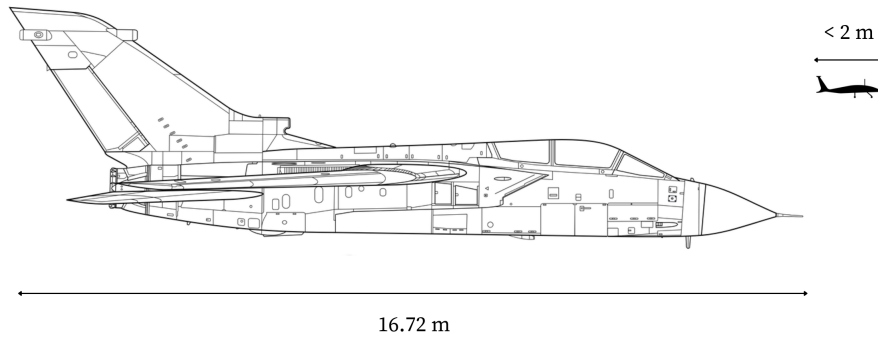


Figure 1.2: Visual size comparison diagram between a Panavia Tornado jet aircraft and a generic tier 1 UAS platform (adapted from Wikimedia Commons, public domain).

In pursuit of a terrain following system for UAS, complex multi-sensor systems such as those seen on the Panavia Tornado would be too costly and heavy. This highlights the need for researching efficient ways in which to achieve safe low-level flight, using lightweight and cost-effective hardware, to maintain the advantages that the autonomous systems have over the complex manned aircrafts. Possible UAS platforms that could implement this terrain following capability are the Puma 3AE (7.1 kg) [5] or the ScanEagle (28 kg) [6], showcasing their difference in size with fighter jets in Figure 1.2.

This thesis is centred in the study of an efficient real-time terrain following system, to address vertical ground following and the longitudinal dynamics of fixed-wing UAS. The avionic suite is reduced to a flight controller with a GNSS/INS navigation module and a single obstacle avoidance sensor (LiDAR or RADAR). An innovative trajectory planning method is proposed, based on a cubic B-spline envelope fitting, combining digital elevation model (DEM) data and additional obstacles detected, with risk-awareness. The tracking is carried out through non-linear model predictive control (NMPC), solving an optimisation constrained problem (OCP) defined for terrain following. The combined system provides a safe, efficient and fast algorithm for real-time terrain following, introducing obstacle knowledge updates, with control frequencies fast enough for real implementation.

Through simulations, the effect of the sensors' uncertainties and risk levels allowed can be evaluated, identifying the trade-off between the system complexity, danger levels and performance. Avionic suites and hardware sensor components adequate for implementation are selected to target the desired altitude ranges, based on the sensitivity analysis.

Chapter 2

Literature Review

2.1 Terrain Following Capabilities

The terrain following problem can be defined as the capability development of an aerial vehicle to plan and track a trajectory maintaining a desired altitude range above ground level (AGL) whilst performing collision avoidance. The terrain following problem involves guidance, navigation and control (GNC):

- Guidance: determination of the terrain following trajectory within the altitude range and incorporating obstacle avoidance.
- Navigation: estimation of current states and position, with emphasis on the AGL height and identification of obstacles.
- Control: actuator signals to control the dynamic system platform—the UAS—and follow the terrain hugging trajectory.

In literature and academic research, the terrain following problem is categorized into Terrain Following (TF) and Terrain Avoidance (TA) [7] [8]. The Terrain Following problem focuses on the vertical guidance, surpassing obstacles and terrain from above to keep a target altitude h , as exemplified in Figure 2.1. Complimentary, the Terrain Avoidance problem is an extension of Terrain Following, introducing the lateral movement to maintain separation d from obstacles, as seen in Figure 2.2. When introducing Terrain Avoidance, a TF/TA ratio is used to modulate the priority of the vertical tracking versus the lateral tracking of the reference trajectory.

These flights can consider a range of altitudes, from wide margins of safety to very low heights. For high-speed and very low-level flights the denomination used is Nap-of-the-Earth (NOE), modulating airspeed and height to adapt to the ground's contour to avoid detection. This is perhaps one of the most dangerous flight environment conditions, as it becomes a highly constrained guidance and control problem, and the use of Digital Elevation Models (DEM) provide incomplete obstacle maps. Due to the unaccounted obstacle and dangers, a NOE capability requires onboard obstacle avoidance sensors to update the environment perception in real-time [8].

This study focuses on a pure Terrain Following (TF) capability for NOE, restricting the fixed-wing UAV dynamics to the 2D longitudinal plane, and performing collision avoidance through overhead flight trajectories on both terrain and obstacles. A Terrain

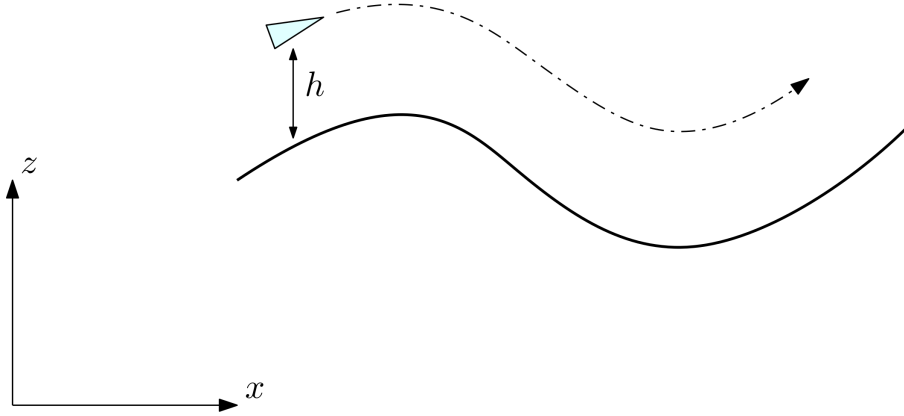


Figure 2.1: Terrain Following (TF) problem representation.

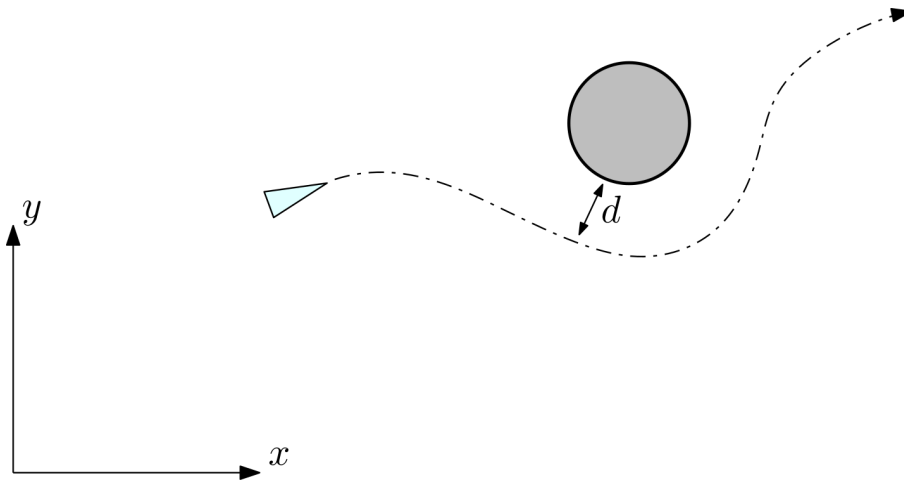


Figure 2.2: Terrain Avoidance (TA) problem representation.

Avoidance (TA) capability remains as possible further expansion and research. The developed system introduces the combination of DEM and obstacle avoidance sensors, innovating in a prominent research gap in the literature, whereas current TF techniques rely exclusively on DEM or sensor measurements, solely.

Terrain following systems can vary greatly depending on the UAS platform employed. To correctly understand and select adequate algorithms and sensor suites, the terrain following guidance control and path planning techniques for rotary-wing and fixed-wing aircraft are evaluated.

2.2 Rotary-Wing Terrain Following Methods

Rotary-wing UAS are not the selected platform of this thesis study, but terrain following techniques applied still share common elements with fixed-wing UAS and provide knowledge on the capability development and state-of-the-art.

Quadrotor, and other rotary-wing UAS, enable precise hover and control over the vertical speed. This is the main distinction between fixed-wing and rotary-wing in terrain following, as the dynamics of the rotary vehicles are not as constraining, enabling a

precise motion control at lower speeds on the horizontal and vertical planes, reaching hover capabilities. This situation is given most of the time as rotary-wing UAS operate close to their deployment zone and avoid high-speeds or velocities to extend the endurance.

A common element to terrain following systems for quadrotors is the use of PID to control the vertical speed. Studies propose a TF system for quadrotors using a simple proportional controller based on altitude measurements with a gimballed laser rangefinder [9]. This method enables faster forward movement speeds with a tilt angle on the LiDAR, to evaluate the change of height in front of the vehicle and adapt the vertical speed in the projected path, as seen in Figure 2.3.

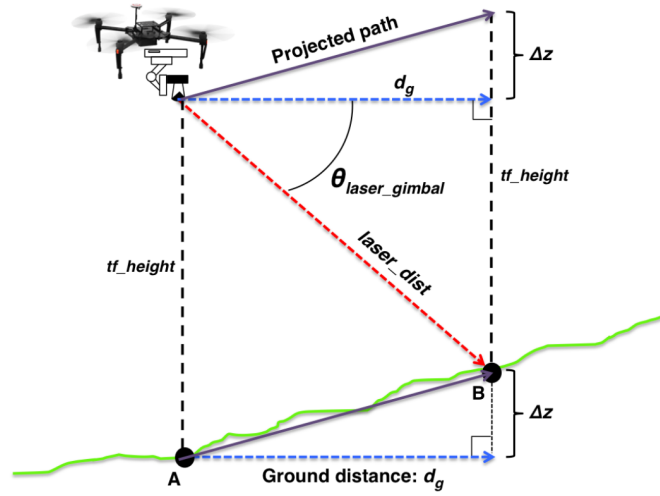
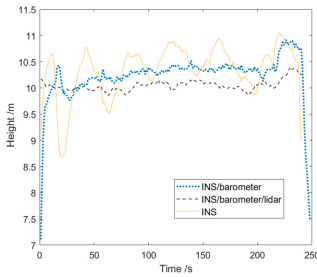
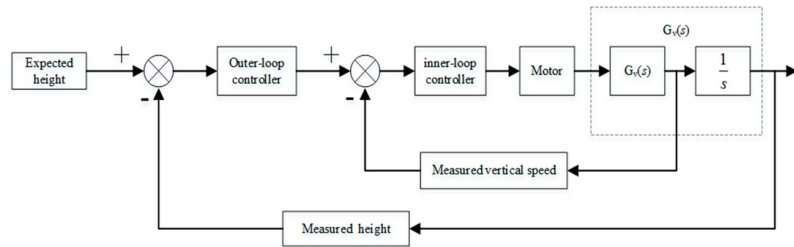


Figure 2.3: Diagram of the laser gimbal system showing the geometric relations to the ground for terrain following control of a quadrotor [9].

Introducing multiple information sources, techniques using a cascade PID can be implemented for terrain following for altitude control [10]. Using an EKF estimation of height combining an INS, barometer and LiDAR, a more stable and precise estimation and flight can be achieved, reducing sensitivity issues that appear when using a single sensor for altitude evaluation. The controller and altitude performance can be seen in Figure 2.4.



(a) Height estimation using EKF sensor fusion.



(b) Structure of the cascade PID control loop for altitude and velocity control, using the height estimation as outer feedback.

Figure 2.4: Terrain following control loop and sensor fusion altitude estimation for a quadrotor [10].

As previously mentioned, altitude information can also be obtained from Digital Elevation Models (DEM). Vertical speed PID controllers based on DEM data are also studied, implementing a look ahead altitude measurement to minimise tracking error at moderate forward speeds [11]. With maximum speeds of 15 m/s, the terrain following tracking is minimised when measuring height at a forward distance of 12-14 m. Because of the DEM incomplete data, this method does not enable to fly at very low heights, targeting 40 m AGL, safe from most unaccounted obstacles. Altitude flight results and the controller are presented in Figure 2.5.

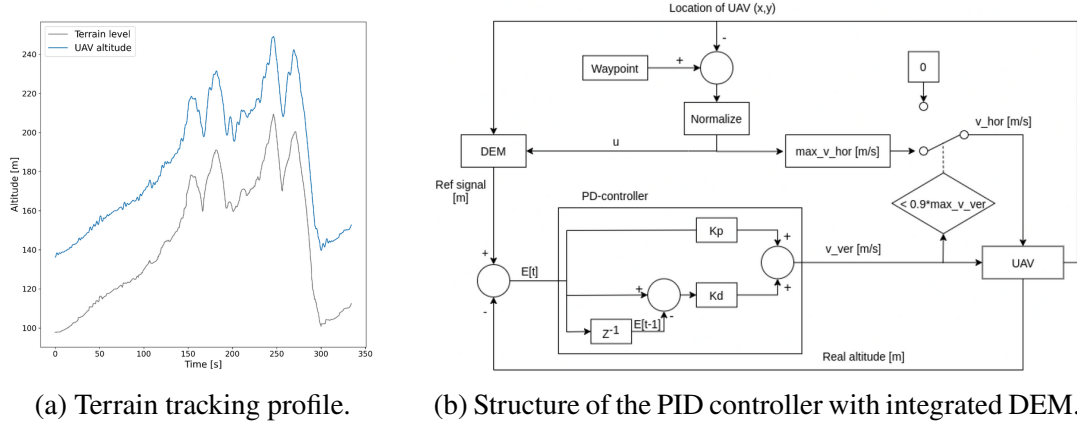


Figure 2.5: Terrain following control based on digital elevation model for quadrotors [11].

There are similar terrain following techniques used for Autonomous Underwater Vehicles (AUV), to inspect the ocean floor at low speeds. An approach uses a 4-beam Doppler Velocity Log (DVL) to estimate a gradient-based sea floor altitude [12]. Thereafter, vertical depth speed is controlled by a PI with a feed-forward term.

Terrain following research on quadrotors is centred on controller architecture or sensor fusion techniques to acquire an accurate altitude estimation. Although, path planning is also studied using waypoint navigation. Over the DEM, the altitude profile between the waypoints is obtained, and the speed of the quadrotor is planned using linear segments with parabolic blends (LSPB) [13]. The waypoint planning and altitude profile using this technique is in Figure 2.6.

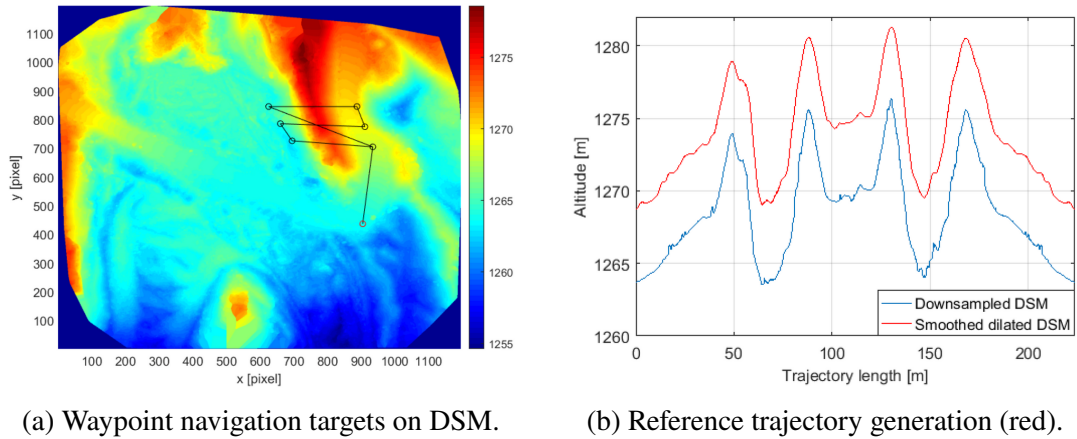


Figure 2.6: Path planning for a quadrotor using a digital surface model (DSM) [13].

2.3 Fixed-Wing Terrain Following Methods

Rotary-wing terrain following techniques are not directly applicable to fixed-wing UAS. Fixed-wing aircraft are underactuated non-linear dynamic systems, flying at higher speeds than rotary-wing vehicles. The use of PID implemented directly on vertical speed control is not adequate for terrain following, neither is a simple waypoint path planning. Instead, fixed-wing UAS terrain following techniques are heavily oriented towards dynamically constrained path planning to avoid inevitable collision states (ICS) [14], with the use of model predictive control (MPC) for real-time implementation. Look-ahead techniques in rotary-wing —using gimbaled LiDAR or feedforward terms— also appear in fixed-wing terrain following, now through forward-looking obstacle avoidance sensors and optimal control to anticipate the movement while considering its dynamic limitations.

The terrain following systems can be divided between the geometric and the optimal control theory based methods [15], explained in this section.

2.3.1 Design Based on Geometric Methods

Design based on geometric methods for terrain following are commonly reliant on the use of splines to generate and plan terrain following trajectories. Due to their analytical nature, spline trajectories can be computed in short amounts of time and modulated to the terrain features.

Splines can be used to find 3D paths to minimise the overall altitude, with search tree methods for randomised exploration of possible trajectories between the initial point and the target destination [8]. Rapidly exploring random tree (RRT) spline methods can be modified to evaluate the dynamical feasibility of the trajectory, in addition to minimising the altitude and the path length [16], as seen in the RRT spline generation of Figure 2.7. These methods provide a fast initial guess for a terrain following reference trajectory when lateral and vertical manoeuvring is allowed.

Path planning methods for fixed-wing UAS are specially important to avoid the aforementioned inevitable collision states (ICS), defined as the state in which the aircraft will subsequently crash without regard for any possible combination of control inputs. A sampler RRT can use the Dubins airplane space to compute dynamically feasible trajectories through ascending or descending curves within the allowed altitude range [14]. Computed terrain following trajectories with this method appear in Figure 2.8.

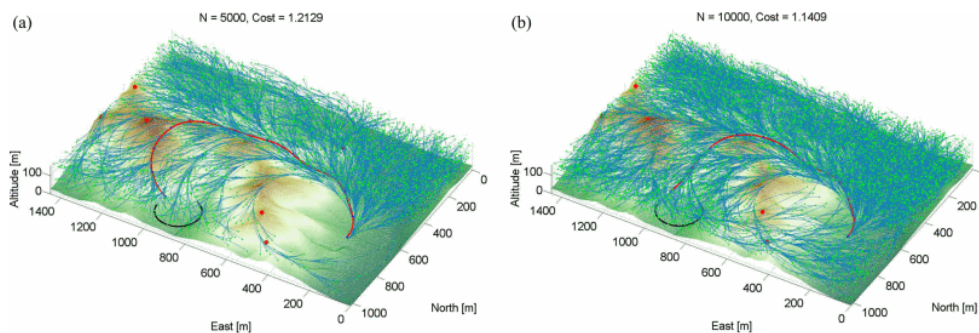


Figure 2.7: Spline RRT variant with dynamical feasibility trajectory generation in two scenarios, with minimum cost path solution (red) [16].

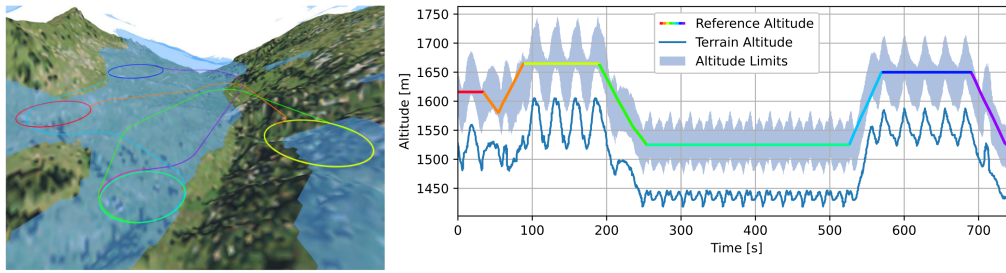
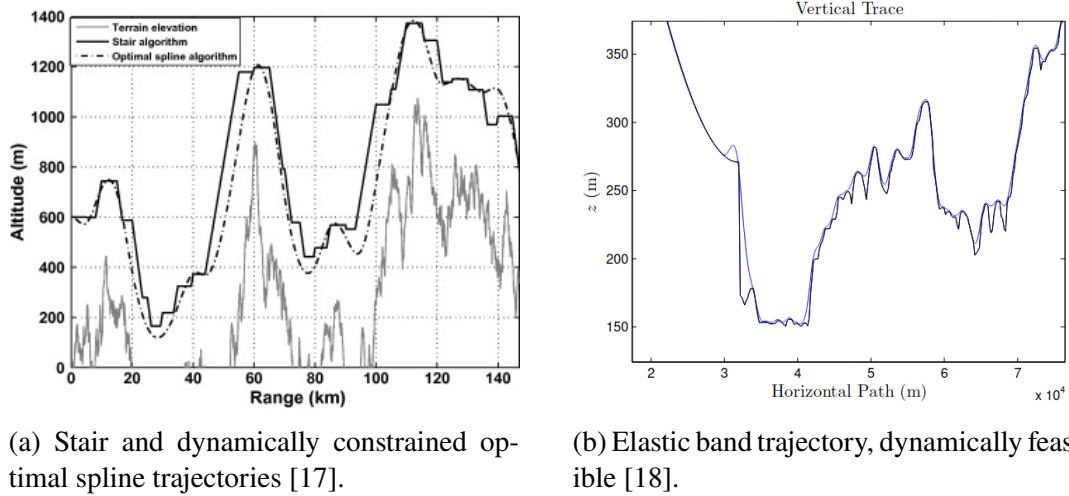


Figure 2.8: Terrain following trajectories and altitude profile through Dubins RRT with dynamic feasibility and ICS avoidance, between hovering circles over waypoints [14].

The use of splines, when restricted to 2D vertical terrain following —this thesis study case—, allows to generate terrain envelopes for a set of data. Compared to simple stair envelopes, optimal spline fitting through sequential quadratic programming (SQP) —minimising terrain deviation while ensuring non-linear dynamic constraints— enable a better tracking through smoother reference trajectories [17]. A similar method uses elastic band theory to assimilate the nodes of the spline to linear springs interconnected, obtaining an optimal solution that minimises the elastic energy of the spring chain while keeping the nodes above the terrain [18]. Both planning trajectory methods are shown in Figure 2.9.

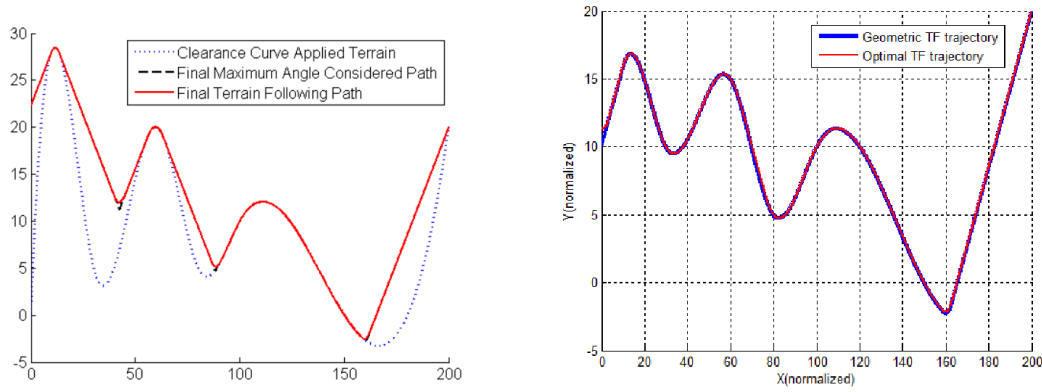


(a) Stair and dynamically constrained optimal spline trajectories [17].

(b) Elastic band trajectory, dynamically feasible [18].

Figure 2.9: Vertical terrain following smoothing path planning trajectory examples.

The study of terrain following with purely geometrical methods has been studied for cruise missiles —with a more stable airspeed. The use of pull-up and pull-down manoeuvres with a constant load factor enable the design of a geometrical path planning, based solely on a known DEM [15]. This technique is promising as it can reduce computational times to find a dynamically feasible path planning, very similar to optimal control solutions at constant flight speeds. The trajectory over the ground terrain is seen in Figure 2.10, limiting descent angles and showcasing pull-up trajectories before reaching the bottom of a steep valley where the aircraft could not manoeuvre out —avoiding an inevitable collision state (ICS).



(a) Terrain following geometrical trajectory over a terrain clearance DEM profile.

(b) Geometrical and optimal control trajectories comparison.

Figure 2.10: Geometric terrain following path planning trajectories through pull-up and pull-down manoeuvres [15].

Employing geometrical methods, the use of a PI regulator for altitude control has been proposed. Given a DEM, the physical state can be transformed to a reference state where the terrain following now has to track a horizontal plane, in feedback equivalence as the dynamics of the UAV are invariant under the transformation [19]. Notwithstanding, the method entails certain complications, as terrain models are not 100% differentiable, so the transformed reference state has errors and is not perfectly planar. The complex transformation also restricts obstacle updates to the DEM map. The change from physical to reference plane is presented in Figure 2.11.

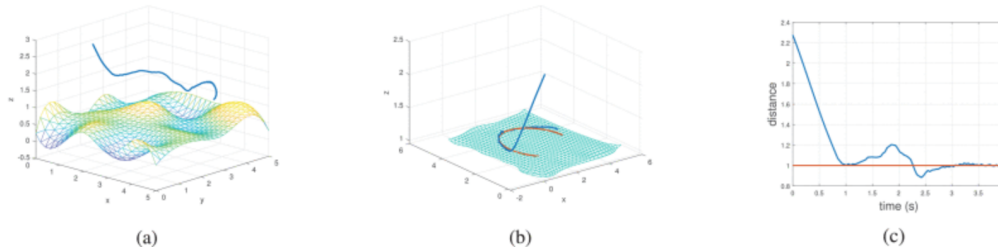


Figure 2.11: Transformation from physical DEM (a) to a reference plane state (b) and terrain following profile (c) [19].

2.3.2 Design Based on Optimal Control Theory

While geometric methods provide path-planning solutions that can be checked for dynamical feasibility, optimal control methods provide planning and control solutions that minimise a cost function, while complying with equality and inequality constraints. This constitutes an optimisation constrained problem (OCP). The dynamics of the system are usually defined as equality constraints. Thereafter, the optimal control problem must minimise the cost function through a dynamically feasible trajectory. The cost function definition depends on the problem tackled and the desired characteristics of the solution, in which the choice of optimisation parameters gain relevance.

Complete terrain following path planning from end-to-end destinations through optimal control theory can be computed using the Direct Collocation method and digital elevation models (DEM). The direct methods, opposite to indirect Hamiltonian bound, solve a non-linear programming problem (NLP) defining both physical and control states of the aircraft. Planning a complete trajectory with sufficient accuracy requires a high number of optimisation nodes, creating a large scale NLP. The Direct Collocation method, discretizing the highly non-linear dynamics into algebraic equations, can be employed to obtain a precomputed 3D terrain following trajectory and solve the TF and TA problems. The optimal control problem can be defined to solve the minimum length path—for time and fuel consumption savings—, while satisfying the aircraft's dynamics and constraining the altitude within the minimum and maximum ceiling allowed [7]. Even with the high discretisation of the path, the long distance of the trajectory requires large timesteps between nodes, making the choice of the dynamics' integrator very important. Explicit methods such as Euler's are fast but do not provide accuracy with large time differentials, and more complex methods like Runge-Kutta are time consuming for the large-scale NLP. Instead, Trapezoidal and Hermite-Simpson methods can be used as seen in Figure 2.12.

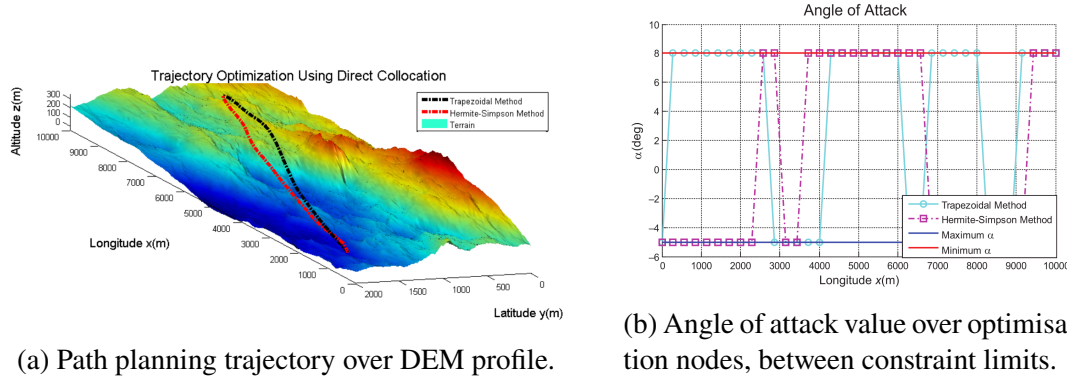


Figure 2.12: Path planning trajectory for terrain following through optimal control method using Direct Collocation to solve NLP, using two different integration schemes [7].

Optimal control can also be tackled through linearised dynamics. This method assumes the small angles of attack, pitch and flight path, so as to obtain a space-state representation of fixed-wing aircraft dynamics. The terrain following method uses a laser rangefinder (LRF) to obtain waypoints directly above the ground measurements, and uses preview control to solve a quadratic cost function through the algebraic Riccati equation to minimise deviation from the reference waypoints [20]. This method is similar to the linear quadratic regulator (LQR), but it anticipates in the cost function the future trajectory in a moving window prediction horizon. The linearised model allows to obtain analytical models for performance prediction, and to study the effect of the sensor and error propagation. Figure 2.13 shows the controller approach and the reference waypoints.

Although linearised models offer faster solving times for the optimal control problem, the performance is limited because the aircraft must operate very closely to the trimming point. At very low-level flight, to perform a successful NOE, aggressive manoeuvres—highly non-linear—can be required to avoid the terrain features. A linearised model can not correctly predict such manoeuvres with sufficient accuracy. This is therefore achieved through model predictive control (MPC).

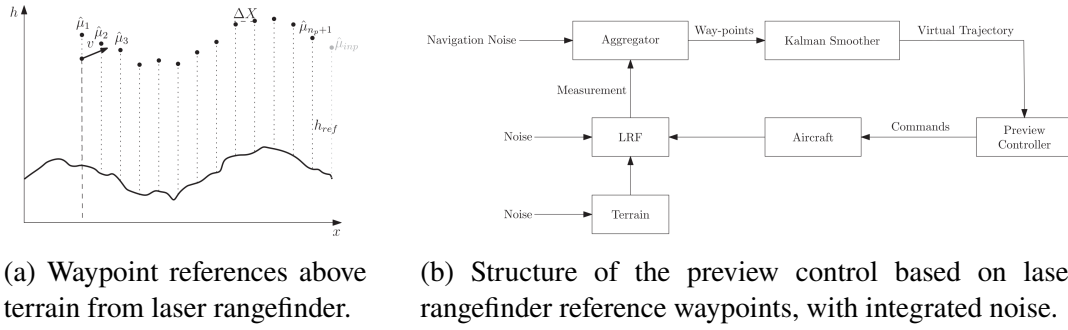


Figure 2.13: Optimal control method through linearised flight dynamics and preview control to follow reference waypoints [20].

Past studies have used MPC for terrain following, to perform NOE flight, employing DEM with complete obstacle information —although it is acknowledged that the obstacle information should be acquired with a sensor as no DEM has a complete obstacle map [8]. The MPC method allows to correctly avoid sudden terrain obstacles, pursuing a reference trajectory. This method is similar to the path-planning presented previously [7] solving an NLP, although it is applied to track a reference trajectory in a moving window with a finite prediction horizon for real-time implementation —like the preview control method [20]. As an OCP, the prediction solutions can be constrained to avoid lateral and vertical regions around obstacles, effectively performing TF and TA. This study has shown that a 10 s prediction horizon is required for stable guidance and control at 30 kts (15.4 m/s) of speed and until 30 degrees of terrain slope. The effect of the prediction horizon length on the terrain following tracking, developing oscillations at lower values, and the reference trajectory over obstacles is shown in Figure 2.14. Before the appearance of faster non-linear MPC solvers, the solution computation could be reduced through perturbation linearisation of the dynamics and modeling of the control input as basis functions. Recently, studies look into the use of cubic spline trajectories fitting radar ground measurements to improve MPC tracking performance [21].

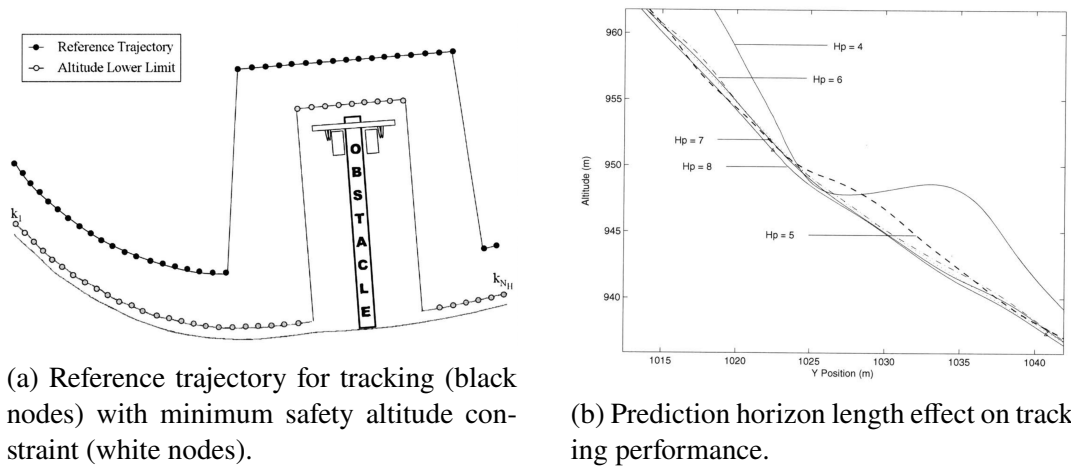


Figure 2.14: Terrain following MPC for real-time reference trajectory tracking and obstacle avoidance, through altitude and region constraints [8].

2.4 TF Method with DEM and Obstacle Sensors

Terrain following systems and methods in the current state-of-the-art delve deeply into guidance and control methods for vertical and lateral avoidance. Research on rotary-wing UAS shows a prevalence for PID-based control architectures with altitude measurements from onboard distance sensors. Meanwhile, research on fixed-wing UAS terrain following is focused on dynamically feasible path planning methods.

The field of fixed-wing terrain following path planning, studied in this thesis, provides a range of geometrical and optimal control methods to obtain precomputed reference trajectories. In both types of methods, path planning and guidance control is based mostly on digital elevation models (DEM) or, in a few cases, on onboard sensors, exclusively.

Current literature presents a research gap in the combination of DEM and onboard sensor measurements to perform NOE terrain following flights. Relying exclusively on DEM datasets is not feasible because of inaccuracies and incomplete obstacle information. And sensor-based flight, at very low altitudes, is not able to anticipate the terrain profile behind obstacles and hill crests. At the moment of this thesis development, methods and studies into the combination of DEM and obstacle sensor measurements are missing from the terrain following field of research.

This study focuses on the first vertical terrain following system for fixed wing UAS that combines DEM and obstacle information provided by onboard sensors. The proposed system, developed in the Methodology Chapter 4, uses spline-based geometrical methods for the generation of a DEM-Obstacle envelope trajectory, and optimal control theory for guidance control and reference tracking.

Before the Methodology, in the following Background Chapter 3, the dynamics of the fixed-wing UAS are defined, along with the digital elevation models and sensors required for obstacle detection, jointly with the use of model predictive control for obstacle avoidance and guidance and control for UAS.

Chapter 3

Background

3.1 Fixed-Wing Tier 1 UAS System and Dynamics

The terrain following system proposed in this study is designed for fixed-wing tier 1 UAS, below 25 kg. Among the existing aircraft platforms, dynamic models of defence products are not available to the public due to national security concerns. Thereafter, this thesis uses a commercially available UAS as the reference platform to implement the terrain following algorithms. As an advantage, a future extension of the research with aims of implementing software and hardware in the loop becomes more straightforward, as well as flight tests.

The selected platform is the Skywalker X8 [22], a readily available commercial flying wing UAV with two elevons and a pusher propeller. This vehicle fits well within the tier 1 UAS category, with a weight range between 2500-3000 g depending on configuration and payload add-ons, while boasting cruising speeds between 65-70 km/h (18-20 m/s)—similar to the speeds used in NOE terrain following MPC guidance and control for fixed-wing UAS [8].

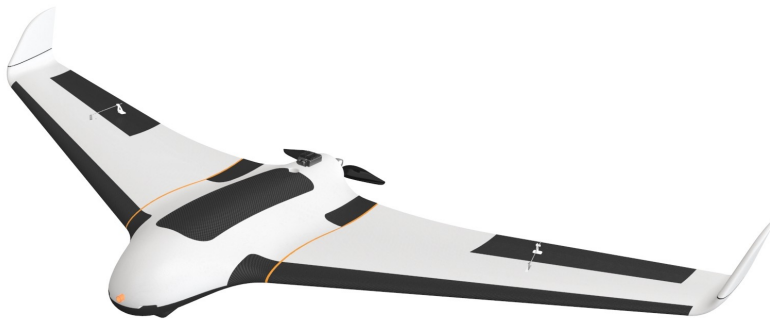


Figure 3.1: Skywalker X8 flying wing UAV [22].

According to the scope and requirements of this project —focused on the terrain following algorithms and sensor suite— we consider a complete knowledge of the dynamic model of the aircraft, used as ground truth for the model predictive control and for dynamic update in simulations, without uncertainties. The dynamics are extracted from a modeling and identification analysis of the Skywalker X8, using a hybrid output error method through experimental studies [23].

3.1.1 Dynamic Model of the Fixed-Wing UAS

The analytic aerodynamic modelling provides the full 6 degrees-of-freedom (DoF), together with propulsion and control actuators. This study, for vertical Terrain Following (TF), makes use of the following assumptions:

- longitudinal dynamics: all lateral effects are considered negligible.
- small altitude variations: air density is considered constant.
- no wind perturbations: airspeed is equal to the ground speed.
- fast actuators: throttle level and elevons do not have lag between the command signal and the actuator position.

Restricted to the presented assumptions, the 3-DoF longitudinal dynamic model of the reference UAS platform is developed from the 6-DoF system identification analysis on the Skywalker X8 [23]. The aircraft's physical properties and environment's are shown in Table 3.1 and 3.2, respectively. Following, the propulsion and aerodynamic coefficient parameters are presented in Table 3.3 and 3.4.

Table 3.1: Physical properties of the Skywalker X8 UAV.

Property	Value
m — Mass	3.364 kg
I_y — Inertia	0.140 kg m ²
$\bar{c}$ — Mean aerodynamic chord	0.36 m
b — Wingspan	2.10 m
S — Surface	0.75 m ²

Table 3.2: Atmospheric and environmental properties.

Property	Value
ρ — Atmosphere air density	1.225 kg/m ³
g — Gravity acceleration	9.81 m/s ²

Table 3.3: Propulsion motor parameters: Hacker A40-12 S V2 14-pin KV610 [24].

Parameter	Value
k_E — Back-EMF constant	0.0157 V/(rad/s)
R — Motor resistance	0.017 Ω
U_b — Battery/motor voltage	14.9 V
I_m — Motor current	49 A
D — Propeller diameter	0.3556 m
I_p — Propeller inertia	3.46·10 ⁻⁴ kg m ²

Table 3.4: Aerodynamic coefficient parameters.

Parameter		Value
Thrust coefficients	C_{T0}	0.1400
	C_{T1}	-0.0300
	C_{T2}	-0.2370
	C_{T3}	0.0847
Drag coefficients	C_{D0}	0.058
	C_{Dq}	0.480
	C_{DCT}	-0.217
	C_{Dk1}	-0.034
	C_{Dk2}	0.225
Lift coefficients	C_{L0}	-0.077
	$C_{L\alpha}$	2.573
	C_{Lq}	17.119
	$C_{L\delta e}$	1.369
Pitch moment coefficients	C_{m0}	0.027
	$C_{m\alpha}$	-0.274
	C_{mq}	-1.608
	$C_{m\delta e}$	-0.276

The dynamic and kinematic equations of the UAS model use a state “ $\mathbf{x}$ ” and control “ $\mathbf{u}$ ” variables, presented in Equation 3.1. Given the inertial Earth frame and the body frame of Figure 3.2, the state is composed by the forward body axis speed u , the downward body axis speed w , the pitch rate q , the Euler pitch angle θ , and the inertial Earth position x_s and z_s of the aircraft system. Because we are considering the longitudinal dynamics, lateral lineal and rotational speeds and positions are not considered. As well, the control inputs of the aircraft are the elevator angle command δ_e —identical action from the elevons— and throttle level δ_t for the motor.

$$\mathbf{x} = \begin{bmatrix} u \\ w \\ q \\ \theta \\ x_s \\ z_s \end{bmatrix} ; \quad \mathbf{u} = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \quad (3.1)$$

The definition of the state variable time derivatives $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ is achieved through 6 differential equations: 3 dynamic equations for the velocities and rates derivatives ($\dot{u}$, $\dot{w}$, $\dot{q}$), and 3 kinematic and navigation equations to describe the angle and position derivatives ($\dot{\theta}$, $\dot{x}_s$, $\dot{z}_s$). These constitute the motion model of the fixed-wing UAS platform, to constraint our flight to feasible trajectories and to track the navigation position over the terrain.

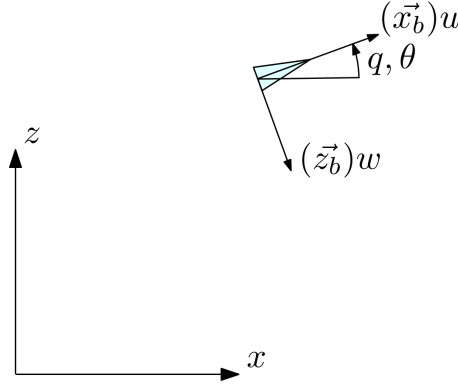


Figure 3.2: Inertial Earth axis $(\vec{x}, \vec{z})$ and body axis $(\vec{x}_b, \vec{z}_b)$ of small fixed-wing UAS.

From the properties and parameters, jointly with the state variables, new variables are defined to ease notation of the equations. Among the new variables: the aerodynamic speed V_a —the norm of body speeds, because no winds are considered by assumptions $(u_a, w_a = u, w)$ —, non-dimensional pitch rate q^* , angle of attack (AoA) α and dynamic pressure $\bar{q}$.

$$V_a = \sqrt{u^2 + w^2} \quad (3.2)$$

$$q^* = q \frac{\bar{c}}{2V_a} \quad (3.3)$$

$$\alpha = \tan^{-1} \frac{w}{u} \quad (3.4)$$

$$\bar{q} = \frac{1}{2} \rho V_a^2 \quad (3.5)$$

Before applying Netwon's second law to obtain the dynamic equations, all aerodynamic and propulsion force and moment coefficients are defined: thrust C_T , lift C_L , drag C_D and pitch moment C_m .

The thrust force T is considered to be aligned with the forward body axis $\vec{x}_b$. The thrust coefficient C_T is modelled as a function of the advance ratio J , that is in turn based on the motor speed Ω_p . Assuming a slow variation of the motor voltage, allowing to discard the inductance, the motor speed is a direct expression regulated by the voltage level U , controlled through the thrust level δ_t :

$$U = \delta_t U_b \quad (3.6)$$

$$\Omega_p = \frac{U - I_m R}{k_E} \quad (3.7)$$

$$J = \frac{2\pi V_a}{\Omega_p D} \quad (3.8)$$

$$C_T = C_{T0} + C_{T1} J + C_{T2} J^2 + C_{T3} J^3 \quad (3.9)$$

Aerodynamic forces are defined through the lift L , drag D and pitch moment M , aligned over the stability axis —coinciding with the airspeed incidence direction through the angle of attack. Their coefficients, without lateral effects due to our longitudinal

dynamics assumptions, depend on the angle of attack α , the non-dimensional pitch rate q^* , elevator deflection control angle δ_e and the thrust coefficient C_T :

$$C_L = C_{L0} + C_{L\alpha} \alpha + C_{Lq} q^* + C_{L\delta_e} \delta_e \quad (3.10)$$

$$C_D = C_{D0} + C_{Dq} |q^*| + C_{DCT} |C_T| + C_{Dk1} |C_L| + C_{Dk2} C_L^2 \quad (3.11)$$

$$C_m = C_{m0} + C_{m\alpha} \alpha + C_{mq} q^* + C_{m\delta_e} \delta_e \quad (3.12)$$

Through the coefficients, the total forces can be defined in the body axis. Thrust T is directly aligned with $\vec{x}_b$, but lift L and drag D must be transformed from stability to body axis with a rotation by the angle of attack α . The pitch moment M is already properly aligned with the body axis because lateral effects are not considered. The computation of the force and moment magnitudes:

$$T = \frac{\rho D^4}{4\pi^2} C_T \Omega_p^2 \quad (3.13)$$

$$L = \bar{q} S C_L \quad (3.14)$$

$$D = \bar{q} S C_D \quad (3.15)$$

$$M = M_{aero}^b = \bar{q} S \bar{c} C_m \quad (3.16)$$

Completing the forces actuating on the aircraft, the gravity force is applied in the inertial vertical axis, and must be also transformed to the body axis through a rotation of the pitch angle θ . And the body forces definition:

$$F_{prop}^b = \begin{bmatrix} T \\ 0 \end{bmatrix} \quad (3.17)$$

$$F_{aero}^b = \begin{bmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{bmatrix}^T \begin{bmatrix} -C_D \\ -C_L \end{bmatrix} \quad (3.18)$$

$$F_{grav}^b = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}^T \begin{bmatrix} 0 \\ mg \end{bmatrix} \quad (3.19)$$

Joined together we obtain the total body force and moment. The moment is purely the aerodynamic moment, as the propeller motor does not cause pitch directly as it is aligned with the forward body axis through the centre of gravity:

$$F^b = F_{prop}^b + F_{aero}^b + F_{grav}^b \quad (3.20)$$

$$M^b = M_{aero}^b \quad (3.21)$$

Newton's second law of motion, applied to the non-inertial body axis, provides the linear and angular accelerations expressions through the forces and moments applied on the system:

$$\dot{v}^b = \frac{1}{m} F^b - \omega^b \times v^b \quad (3.22)$$

$$\dot{\omega}^b = I^{-1} (M^b - \omega^b \times I \omega^b) \quad (3.23)$$

The 3 dynamic equations obtained through Newton's second law of motion, applying the longitudinal dynamic assumptions, are therefore:

$$\dot{u} = du/dt = \frac{F_x^b}{m} - q w \quad (3.24)$$

$$\dot{w} = dw/dt = \frac{F_z^b}{m} + q u \quad (3.25)$$

$$\dot{q} = dq/dt = \frac{M^b}{I_y} \quad (3.26)$$

Afterwards, the 3 kinematic and navigation equations provide tracking of the inertial Euler pitch angle and Earth frame position, from the pitch rate and body speeds, respectively:

$$\dot{\theta} = d\theta/dt = q \quad (3.27)$$

$$\dot{x}_s = dx_s/dt = u \cos \theta + w \sin \theta \quad (3.28)$$

$$\dot{z}_s = dz_s/dt = u \sin \theta - w \cos \theta \quad (3.29)$$

Together, the dynamic, kinematic and navigation equations complete the longitudinal motion model of the fixed-wing UAS:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{x}_s \\ \dot{z}_s \end{bmatrix} = f(\mathbf{x}, \mathbf{u}) \quad (3.30)$$

3.1.2 Trimming and Dynamic Stability Modes

Necessary for the validation and verification of the motion model, the dynamic modes and motion response of the model must be assessed. The dynamic modes can be evaluated through linearised dynamics, using classical control theory, because Lyapunov and Lagrangian analysis directly on the non-linear dynamics is not straightforward.

The linearisation is carried around a trimming point, an equilibrium in which states will remain constant or return to after a perturbation in a stable system. Around the equilibrium point, the dynamics can be linearised through Taylor series to the 1st order, neglecting terms of 2nd order or higher to obtain a simplified set of equations respecting the principle of superposition.

The first step is the trimming of the dynamic equations. The trimming variables are both states “x” and control “u”, but without the inertial navigation positions:

$$\mathbf{z}_{trim} = [u \ w \ q \ \theta \ \delta_e \ \delta_t]^T \quad (3.31)$$

After removing the kinematic and navigation equations, we are left with the 3 dynamic equations and with 6 variables. To this purpose, the trimming conditions includes additional constraints formulated as differential equations. These additional constraints are

the rate of climb $\dot{X}_{ROC}$, the flight speed $\dot{X}_{FS}$ and the pull-up rate $\dot{X}_{PU}$. The constraints are defined through the trim flight angle γ_{trim} , the trim speed V_{trim} and trim pitch rate $\dot{\theta}_{trim}$:

$$\dot{X}_{ROC} = \sin \gamma_{trim} - \cos \alpha \sin \theta + \sin \alpha \cos \theta \quad (3.32)$$

$$\dot{X}_{FS} = V_{trim} - V_a \quad (3.33)$$

$$\dot{X}_{PU} = \dot{\theta} - \dot{\theta}_{trim} \quad (3.34)$$

Together with the 3 dynamic equations, the 3 constraints provide 6 equations to solve the 6 variables of the trimming state z_{trim} . For terrain following simulations and verification of the model, the reference trim point is a steady horizontal level flight at 17 m/s ($\gamma_{trim} = 0$, $V_{trim} = 17$ and $\dot{\theta}_{trim} = 0$). The solution is provided when solving for null derivative in the trim equations, for equilibrium:

$$\dot{x}_{trim} = f(z_{trim}) = \begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{X}_{ROC} \\ \dot{X}_{FS} \\ \dot{X}_{PU} \end{bmatrix} = 0 \rightarrow z_{trim} = \begin{bmatrix} 16.8053 \\ 2.5652 \\ 0.0000 \\ 0.1515 \\ -0.0526 \\ 0.5835 \end{bmatrix} \quad (3.35)$$

Linearising the longitudinal dynamics around the trimming point, the natural dynamics matrix $[A]$ and control matrix $[B]$ can be obtained through the Jacobian of the motion equations $f(x, u)$ —excluding the 2 inertial navigation equations. Defined for small displacements from the equilibrium solution:

$$\delta \dot{x} = [A] \delta x + [B] \delta u \quad (3.36)$$

$$[A] = \left. \frac{\delta f(x, u)}{\delta x} \right|_{x_{trim}, u_{trim}} \quad [B] = \left. \frac{\delta f(x, u)}{\delta u} \right|_{x_{trim}, u_{trim}} \quad (3.37)$$

Applying eigenvalues to the natural dynamics $[A]$ yields two complex pairs of poles corresponding to the short-period oscillation and the phugoid mode. Both dynamic modes are stable, because of the negative real part. The poles are seen graphically in Figure 3.3, finding the phugoid modes close to the real axis with a small imaginary component, indicating the slow period and low attenuation of the potential-kinematic energy exchange; while the short-period poles have a large negative real part and an important imaginary value, denoting the characteristic fast oscillations and high damping of the mode.

The short-period and phugoid mode poles, their frequency and damping ratio are:

$$\lambda_{sp} = -5.9662 \pm 7.3341i \rightarrow \omega_{sp} = 9.4544 \text{ rad/s} ; \zeta_{sp} = 0.6311 \quad (3.38)$$

$$\lambda_{ph} = -0.2813 \pm 0.7547i \rightarrow \omega_{ph} = 0.8054 \text{ rad/s} ; \zeta_{ph} = 0.3492 \quad (3.39)$$

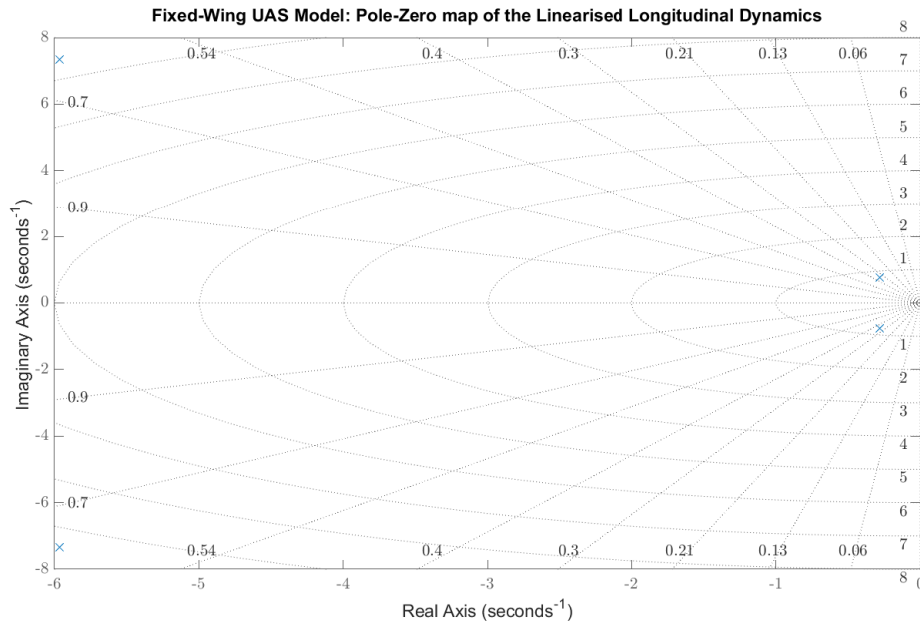


Figure 3.3: Pole-zero plot of the linearised longitudinal dynamics around the trimmed steady level flight at 17 m/s of the Skywalker X8 UAS model.

3.1.3 Model Verification

After analytically assessing the dynamical stability of the aircraft model using classical control theory on linearised dynamics, the response is evaluated with the full non-linear equations of motion.

In Figure 3.4, the short-period mode is excited through an initial pitch rate increase from the trim state, forcing fast and highly damped oscillations in the first 2 seconds. The oscillations affect mostly the pitch rate and the vertical body speed—causing fast changes in the angle of attack.

The phugoid mode in Figure 3.5 was forced through a starting speed increase from the trim condition, producing long oscillation in the dynamic and kinematic variables around the trim state, lasting around 20 seconds as potential and kinematic energy are slowly exchanged.

Having assessed the natural dynamic stability, the control input effect is also evaluated and verified. An elevator command increase—downward deflection—response is shown in Figure 3.6. It creates an expected pitch-down movement as a negative moment is applied. The pitch-down movement slows down as the aircraft descends and gains speed, eventually leading to a phugoid motion.

Otherwise, a throttle increase response is presented in Figure 3.7. The thrust creates an immediate acceleration in the forward body speed. Similarly to the previous manoeuvre, the speed increase slows down as the aircraft gains altitude and pitch, evolving into the phugoid motion.

From both an analytical evaluation of the dynamic stability and a reasonable non-linear response of states and control inputs, we assess the motion model to be verified and fit for use in the terrain following system and simulations developed in this study.

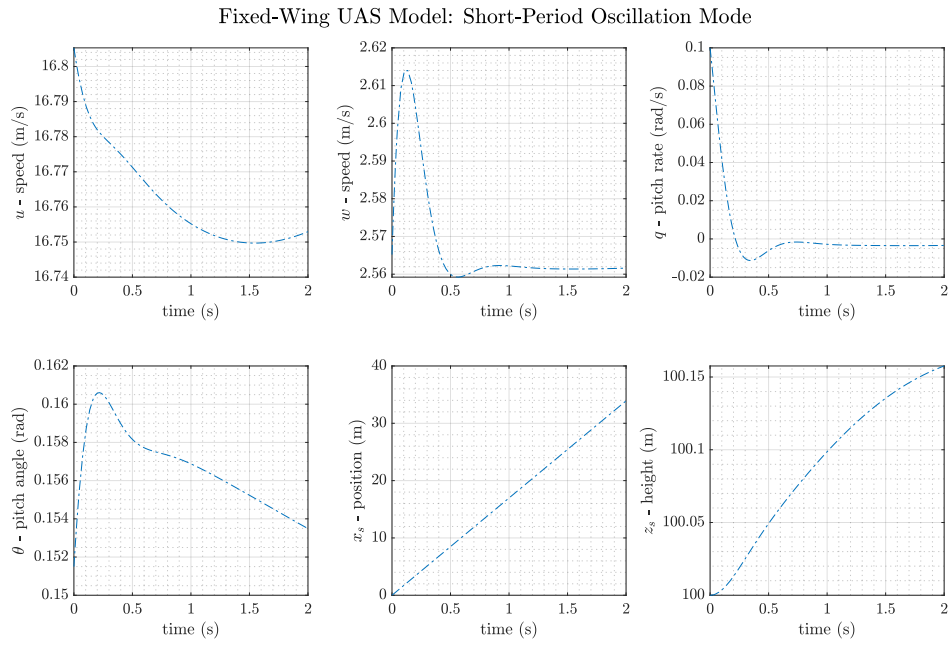


Figure 3.4: Short-period oscillation mode of the Skywalker X8 UAS longitudinal motion model.

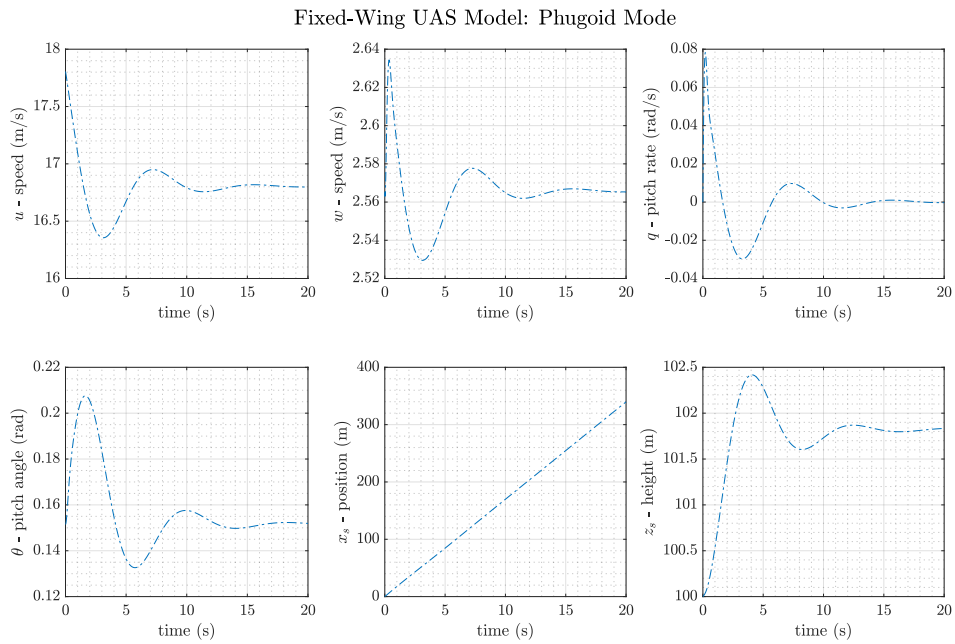


Figure 3.5: Phugoid mode —long period— of the Skywalker X8 UAS longitudinal motion model.

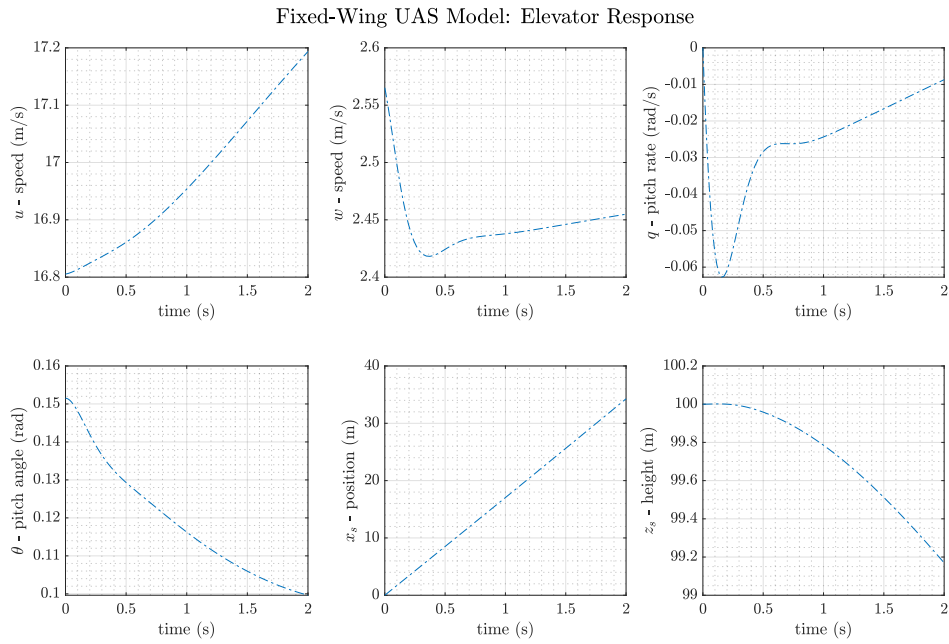


Figure 3.6: Dynamic response to an increase in positive —downward— elevator deflection angle of the Skywalker X8 UAS longitudinal motion model.

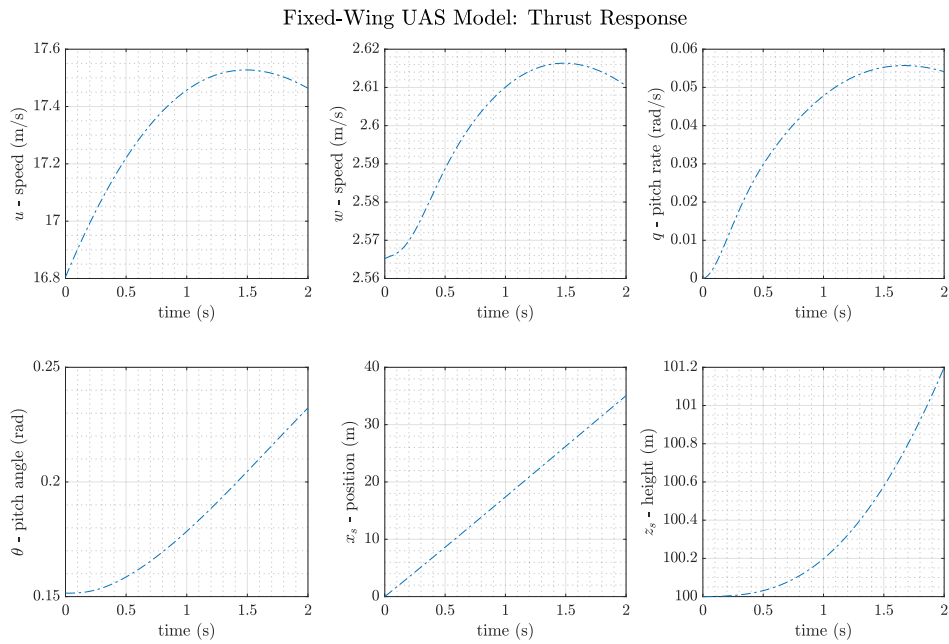


Figure 3.7: Dynamic response to an increase of thrust level of the Skywalker X8 UAS longitudinal motion model.

3.2 Digital Elevation Models

The use of digital elevation models (DEM) for terrain following has been extensively reported throughout the state-of-the-art presented in the Literature Review Chapter 2. A DEM is digital file with elevation information associated with a set of discrete coordinates. It is mostly used in path planning methods, but has also been used in guidance and control techniques. Improvements on the quality of DEM, increasing the levels of resolution, accuracy and obstacle representation provide a better groundwork to safely navigate at low altitude levels.

Depending on the terrain features incorporated digital elevations models can be classified into digital surface models (DSM) and digital terrain models (DTM) [11]. A digital surface model (DSM) measures the altitude of any surface object securely attached to the ground —buildings, trees, etc.—, whilst the digital terrain model (DTM) is a bare-earth representation that has processed data to fit only the ground without any kind of obstacle.

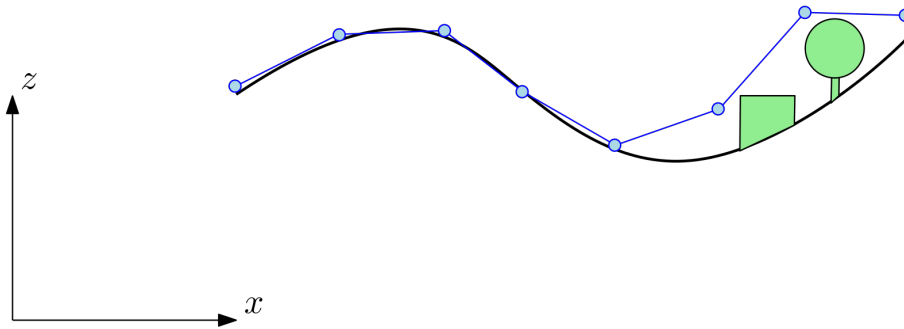


Figure 3.8: Example graph of a 2D Digital Surface Model (DSM).

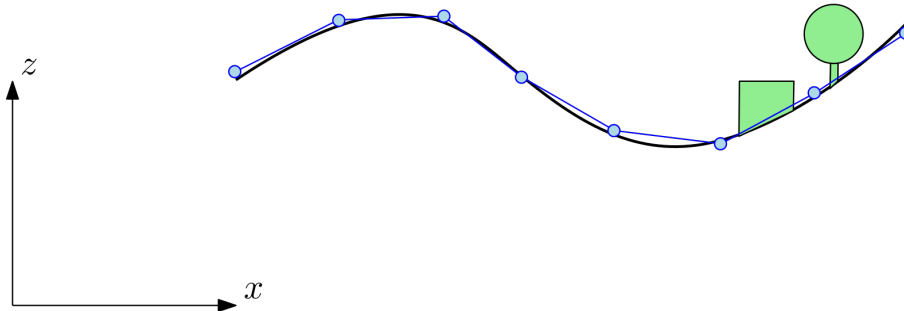


Figure 3.9: Example graph of a 2D Digital Terrain Model (DTM).

Usually, digital elevation models are 3D, providing matrices with East-North-Up elevation information in “.geotiff” file format. Nevertheless, for vertical terrain following purposes, 2D DEMs can be generated through interpolation with only horizontal position and height information along the flight path. This data is graphically exemplified in Figures 3.8 and 3.9 in which the blue points are the DEM values, depicting a DSM and a DTM, respectively.

Digital elevation models are also classified based on the resolution quality and coverage, highly dependant on the type of sensor used to measure the terrain elevation. The types fall into:

- Global DEM: continental coverage at very low resolutions.
- RADAR DEM: regional coverage at metre-level resolution and accuracy.
- LiDAR DEM: local coverage at high resolution and centimetre-level accuracy.

Because digital elevation models are also sources of uncertainty and incompleteness, the quality of the three types of DEMs from the U.S. Geographical Survey (USGS) are evaluated. These DEMs are available for use and download, shared into public domain.

One of the most recent global DEMs provided by the USGS is the Global Multi-resolution Terrain Elevation Data 2010 (GMTED2010) [25]. It provides a complete DEM coverage between 84°N to 90°S latitudes. Nevertheless, the elevation raster files have a horizontal spatial resolution between 30 and 7.5 arc-seconds —926 m to 231.5 m at the equator— and a vertical RMSE between 42 m and 26 m depending on the spatial resolution available. The datasets are provided in files spanning $30^{\circ}\times 20^{\circ}$ longitude-latitude patches of the Earth, enough to fit a large swathes of a continent, as seen in the processed DEM of Figure 3.10.

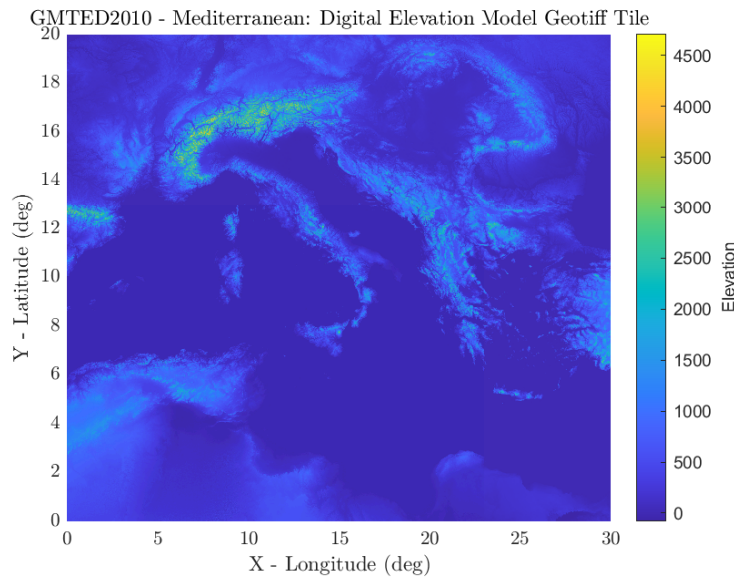


Figure 3.10: Digital elevation model visualization of a geotiff tile from the GMTED2010 database, over the Mediterranean Sea.

Following, the USGS has a RADAR DEM of Alaska [26]. The Alaska Mapping Initiative (AMI) is obtained through an interferometric synthetic aperture radar (IFSAR), in raster file tiles of 15 minutes —15 by 30 km due to the conical projection. With an improved spatial resolution of 5 m, the IFSAR DEM provides both distinct DSM and DTM, that can have substantial differences given tall forests and vegetation masses in the region. The vertical accuracy has also improved, down to 3 meters at 90% of confidence. A loaded example of the data is delivered in Figure 3.11.

Lastly, the USGS is working on the 3D Elevation Program (3DEP) [27]. This LiDAR DEM database has a 1 m spatial resolution, and the vertical accuracy is below 20 cm at 95% confidence. The original point-cloud data has only been post-processed into a highly detailed and accurate bare-earth DTM, so it is lacking many surface obstacles. Additionally, this DEM program is incomplete, as many other LiDAR DEM projects world-wide. It is only available in certain regions of the US in 10 by 10 km tiles, and even some of the files provided can have missing data within. The detail achieved from the 3DEP files is shown in Figure 3.12.

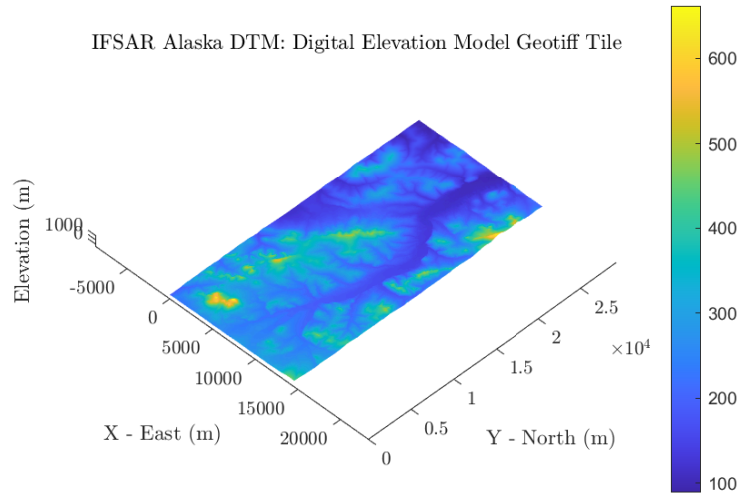


Figure 3.11: Digital elevation model visualization of a geotiff tile from the IFSAR Alaska DTM database, in north-west Alaska.

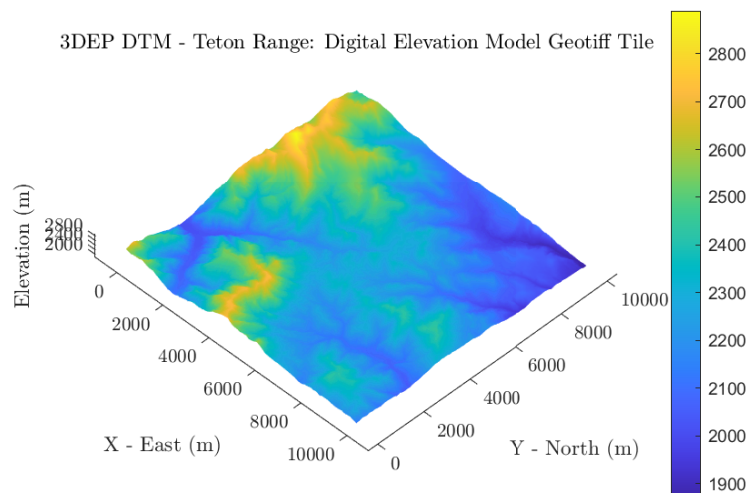


Figure 3.12: Digital elevation model visualization of a geotiff tile from the 3DEP DTM database, over the mountainous Teton Range.

Overall, a summary from the DEM types is provided in Table 3.5. Global DEMs are not suitable for low-level terrain following due to the low spatial resolution and high vertical inaccuracy that does not provide enough terrain-obstacle information. Both RADAR and LiDAR DEM have a suitable spatial resolution to perform terrain following, although the RADAR's has a vertical inaccuracy several orders of magnitude higher than LiDAR's and its effect is studied in this thesis. Another factor to consider is the memory size of files. Because of the detailed resolution of LiDAR DEMs, even small tiles have very large sizes—hundreds of MBs—that increase computational times. This inefficiency could be fixed with an adequate pre-processing to sub-select the region in which the algorithm computations are taking place and discard redundant data.

Table 3.5: US Geographical Survey Digital Elevation Model Summary.

DEM	GMTED2010	IFSAR Alaska	3DEP
Type	Global	RADAR	LiDAR
Coverage	84°N - 90°S	Regions of Alaska	Regions of US
Spatial Resolution	30 to 7.5 arc-seconds	5 m	1 m
Vertical Accuracy	42 to 26 m RMSE	±3 m (90%)	±18.57 cm (95%)
Tile Square	30° Lon. × 20° Lat.	14590 × 29205 m	10012 × 10012 m
Tile File Size	~ 280 MB	~65 MB	~230 MB

3.3 Real-Time Direct Ground Mapping

Effective obstacle avoidance for NOE flights, in terrain following, requires the use of on-board sensor measurements to detect obstacles missing from the incomplete and uncertain DEMs. A constant ground measurement, below and in front of the vehicle is a real-time ground mapping.

Ground mapping can be either direct or indirect [28]. Indirect 3D mapping takes place when undertaking simultaneous localisation and mapping (SLAM): measuring ground features and estimating the UAS position and attitude, based on previously known ground control points (GCP), usually landmarks.

For a terrain following UAV, at very low altitudes, vision SLAM would not provide sufficient field of view to identify features and LiDAR-based odometry is challenging for UAVs, with flight length barely reaching 100 m [29]. Because of the incursion mission concept of a terrain following UAV, it will probably not re-visit and improve estimations. Additionally, the reliance of SLAM on GCP—from the DEM— fails in its purpose of identifying features not present in the DEM.

As an alternative, direct 3D mapping can be applied. This technique relies on the UAS position knowledge, acquired independently, to obtain a point cloud of relative data through distance measurement sensors. Direct ground mapping, although less complex than SLAM, requires both a precise navigation sensor to obtain the UAS position and attitude, and a precise obstacle avoidance sensor to measure distance and bearing to the terrain features.

The use of real-time direct ground mapping is selected for the terrain following system in this thesis, as it provides a fast method to gather a point cloud of terrain and obstacle measurements to be used afterwards in the guidance and control for an adequate path planning in conjunction with DEM information. The diagram of the direct ground mapping in the vertical terrain following UAS is shown in Figure 3.13, showcasing the relationship between the vehicle position x_s, z_s and attitude θ with the obstacle sensor distance d and bearing angle ϵ —could be fixed to the airframe or gimbaled. The resulting relative measuring expression for the obstacles' position x_o, z_o :

$$\begin{aligned} x_o &= x_s + d \cos(\theta - \epsilon) \\ z_o &= z_s + d \sin(\theta - \epsilon) \end{aligned} \quad (3.40)$$

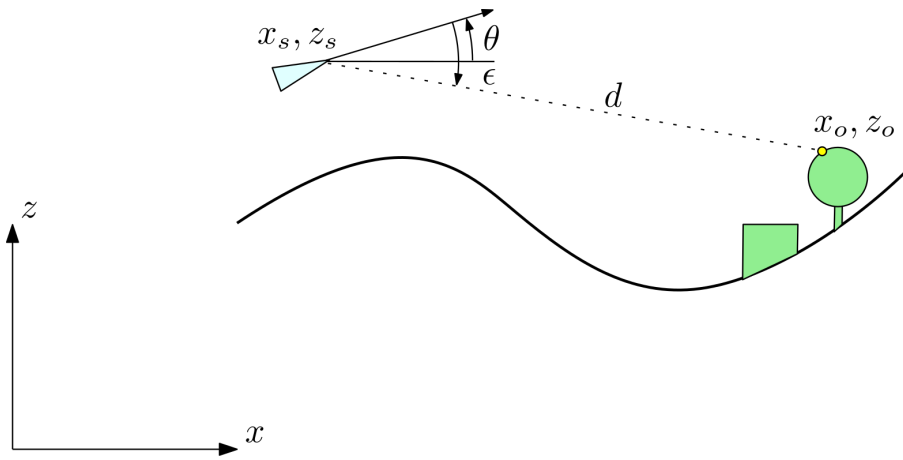


Figure 3.13: Direct ground mapping and obstacle measurement.

Smaller tier 1 UAS are restricted beneath 25 kg of weight. Accordingly, the sensor suite and payloads are constrained in size, complexity and cost. This limits the navigation and obstacle avoidance sensors that can be installed. As such, the available sensors' characteristics and performance for lightweight unmanned aircraft are evaluated for their use in the terrain following system.

3.3.1 Navigation Sensors

Navigation sensors are used to acquire and estimate the UAS position and attitude, to perform direct ground mapping.

In this regard, GNSS-denied methods such as visual and LiDAR SLAM, or aided by deep reinforcement methods (DRL) are insufficient to obtain an adequate precision and operate in lengthy flights [29]. From a review on UAV avionic systems architecture [30], modern navigation commonly relies on GNSS and inertial navigation as the main techniques for outdoor navigation.

Global Navigation Satellite Systems (GNSS) —GPS or GALILEO— are renown positioning systems in both ground and aerial vehicles, although the signal can be obstructed and do not provide information about the attitude of the vehicle. To complement GNSS receivers, inertial navigation systems (INS) are employed. They work with inertial measurement units (IMU), usually with 6-DoF: 3-axis accelerometer and 3-axis gyroscope,

with potentially having an additional 3-axis magnetometer. IMUs are employed for the vehicle attitude orientation determination, but position can be briefly estimated through dead reckoning for short-path navigation [30].

Combining the data through multi-sensor fusion provides a complete knowledge of the vehicle state using an augmented navigation kinematic system through GNSS/INS fusion. Deterministic estimators are adequate when relying on high-grade sensors, whilst low-cost avionics of an expendable tier 1 UAS are bound to have higher uncertainty. Alternatively, linear and non-linear stochastic estimators can be employed. The later, non-linear estimators, are composed by extended Kalman filters (EKFs), unscented Kalman filters (UKFs), particle filters (PF) or Lyapunov-based filters [30].

Although Lyapunov-based filters can provide better results, it is an active field of research not yet widely seen in commercial avionic systems for UAS. In consideration, the terrain following system developed in this study uses commonly seen Kalman filters for navigation estimation, as they are widely available, used in affordable components and provide adequate accuracies. Many commercial flight controller units for UAS already provide IMU sensors and fast integration with GNSS receivers.

Among the GNSS/INS integration techniques for vehicles evaluated from reviews [31], loosely coupled and tightly coupled systems are the most used. The loosely coupled is more simple, as it uses the position and velocity measurements of the GNSS to estimate the error of the INS into the Kalman filter. In absence of GNSS signal, the loosely coupled filter will work purely as an INS, degrading over time. On the other hand, tightly coupled Kalman filters use the GNSS to obtain a complementary measurement vector through the carrier signal phase so that even with partial GNSS signal the states are updated. As a downside, it is more complex to numerically implement, because it requires computation of the ephemeris and receiver clock.

Loosely coupled filters are the most commonly seen in avionic devices, and can be implemented with any GNSS receiver and IMU sensor. Among the reviewed techniques, the most successful improves on the performance of a standard loosely coupled Kalman filter. Using a post-processing smoothing, the standard deviation can be reduced from a metre level order of magnitude to centimetre [32]. This method also improves resistance to partial loss of GNSS signal—when less than 4 satellites are received—and helps to withstand for longer periods of time incomplete position and velocity measurements.

As a possibility for short-range flight, real-time kinematics (RTK) GNSS can be used for precise estimation of a UAV position and perform direct ground mapping [28]. RTK-GNSS is a differential GNSS that corrects the errors of an onboard receiver on the UAS with a ground static receiver. This method is very cost-effective—employed with low-cost single-frequency receivers—and provides good accuracies close to the ground station, but performance degrades with distance and obstacle cluttering. To fix this, the use of multiple onboard RTK-GNSS receivers is proposed to improve positioning accuracy.

The uncertainty levels from the most promising, available and cost-effective navigation methods are shown in Table 3.6. For long-range incursion flights, a loosely coupled Kalman filter with post-processing smoothing is deemed more adequate, although to achieve the specified performance UIMU-LCI defence-grade navigation sensors were employed [33]. As a low-cost back-up for short-range flights, RTK-GNSS provides a millimetre level performance.

Table 3.6: Error values in GNSS-INS navigation techniques selected for terrain following.

Navigation Technique		GNSS/INS Loosely Coupled Kalman Filter	GNSS/INS Loosely Coupled Kalman Filter Post Smoothing	RTK-GNSS Multiple Receiver
Position RMSE	N (m)	1.087	0.010	0.0048
	E (m)	1.422	0.020	0.0053
	U (m)	0.807	0.011	0.0111
Velocity RMSE	X (m/s)	0.031	0.009	—
	Y (m/s)	0.073	0.007	—
	Z (m/s)	0.014	0.007	—
Attitude RMSE	Roll (°)	0.014	0.005	0.202
	Pitch (°)	0.013	0.005	0.174
	Yaw (°)	0.247	0.006	0.066

3.3.2 Obstacle Avoidance Sensors

The obstacle avoidance sensors perform the range and bearing detection of terrain features, undertaking the second stage of the direct ground mapping.

Supplementing the DEM to obtain a more complete obstacle map, the information provided feeds into the terrain following system designed in this thesis. This terrain following system is based on MPC to take advantage of the non-linear range of the UAS to perform evasive manoeuvres at low level flight. Previous studies into MPC guidance for low-level terrain following with sharp obstacles require a prediction horizon of up to 10 seconds for an adequate tracking and avoidance [8]. Considering the reference speed of our modelled UAS —of 17 m/s— the required sensor detection range is therefore:

$$d > V_{trim} H_p = 17 \cdot 10 = 170 \text{ m} \quad (3.41)$$

Obstacle avoidance techniques and sensors for UAS are thoroughly studied and reviewed in literature. Detection of objects can be undertaken with passive camera-based systems and active sensors [34] [35]. The understanding and limitation of these sensors are critical for a terrain following system. Passive sensors capture data emitted by the environment itself, making them stealthy to undesired surveillance and detection systems. Whilst active sensors receive echoes from energy sources emitted by the sensor and reflected by the environment, enabling longer detection ranges with the downside of making themselves more easily detected.

Camera-based systems, such as monocular or infrared cameras do not provide a direct distance estimation, relying instead on post-processing to identify depth and distance to objects, usually done through machine learning agents [36]. Additionally, standard monocular cameras are highly affected by lighting and weather conditions, while infrared is affected by interference in a cluttered environment —such as in low-level flights. These elements make them unsuitable choices for direct ground mapping applications.

As an alternative passive sensor, stereo cameras provide depth perception through focus up to distances of a 100 m. These sensors are still highly affected by weather or lighting conditions, so they are restricted to clear daylight use [37]. Nevertheless, the restricted range is not sufficient for fixed-wing UAS terrain following, although they could be used in some rotary-wing low-speed terrain following applications. A summary of the passive perception sensors' characteristics for UAVs is provided in Table 3.7

Table 3.7: Characteristics of passive perception and obstacle avoidance sensors for UAS.

Sensor Type	Monocular Camera	Stereo Camera	Infrared Camera
Distance	No	$d < 100$ m	No
Accuracy	—	Low	—
Velocity	No	Depth-based	No
Light sensitivity	High	High	Medium
Weather sensitivity	High	High	Medium
Size	Small	Small	Small
Power	Low	Low	Low
Processing	Medium	High	Medium
Cost	Low	Low	Low

Table 3.8: Characteristics of active perception and obstacle avoidance sensors for UAS.

Sensor Type	LiDAR	RADAR	Ultrasonic
Distance	Medium ($d < 500$ m)	Long ($m < d < km$)	Short ($d < 5$ m)
Accuracy	High	Medium	Low
Velocity	Distance-based	Doppler	No
Light sensitivity	No	No	No
Weather sensitivity	Medium	No	Low
Size	Small	Large	Small
Power	Low	High	Low
Processing	High (3D) Medium (2D) Low (1D)	Medium	Low
Cost	Medium	High	Low

Active sensors, comprised by light detection and ranging (LiDARs), radio detection and ranging (RADARs) and ultrasonic sensors, are commonly used in unmanned vehicles [35]. The use of ultrasonic sensors is immediately discarded due to their very low detection ranges and accuracies. This leaves LiDAR and RADAR as the two main candidates for direct ground mapping for fixed-wing terrain following.

LiDARs use laser beams whilst RADAR employ high-frequency electromagnetic waves as their active principle. The range of LiDARs is usually smaller than RADARs, but still able to reach the necessary distance for fixed-wing terrain following. Lightweight LiDAR sensors are nowadays more common for UAS, and prices can oscillate—in the same way as computer processing—if it is a 3D point-cloud lidar or a simple 1D laser rangefinder (LRF) [37]. The use of LiDARs could be a key decision as it has been theorised their capability to detect up to 5 mm wires at 500 m—a capability much needed to avoid anti-drone nets deployed in its low-level path—, given a scanning pattern and post-processing of signal [4].

On the other hand, the use of RADARs on tier 1 UAS is not as common, given their higher cost. They have a further range than LiDAR but it is essentially overkill in a low-level terrain following application. Critically they have worse accuracies, specially important in the angle bearing precision. The main advantage over LiDARs is their fully all-weather operation capability, as LiDARs suffer some performance loss under fog or rain precipitation [37]. The active sensors' characteristics are summarised in Table 3.8.

This thesis study evaluates a sensitivity analysis of the sensor's uncertainty on the proposed terrain following system. Official datasheet values from the manufacturers are acquired under certain conditions that do not reflect the actual use environment, leading to worse performance [38]. Because of it, a RADAR and a LiDAR tested values from a study [39] are selected as reference values for the proposed terrain following system. The sensors and parameters are described in Table 3.9.

Table 3.9: Parameters from a RADAR and LiDAR sensor adequate for low-level terrain following, with detection range 0-200 m [39].

Sensor	IMST sR-1200e	Velodyne ULTRA Puck
Type	RADAR	LiDAR
FOV	65° Az. × 24° El.	360° Az. × 40° El.
Scan rate	10 Hz - 200 Hz	5 Hz - 20 Hz
Detection range	307 m	200 m
Sensing accuracy/resolution	range ± 0.6 m velocity ± 6.25 m/s angle $\pm 3^\circ$	range ± 3 cm angle $\pm 0.4^\circ$
Operating frequency/wavelength	24.00-24.25 GHz	903 nm

3.4 Model Predictive Control

Model predictive control (MPC) is part of optimal control theory, to minimise a cost criterion. The control law is obtained through a set of differential equations defining states and control inputs, upon which the cost function depends. In purely linear dynamics, if the cost function is described through quadratic functions, the optimal problem is a Linear Quadratic Regulator (LQR) and can be solved through the Riccati equation. Nevertheless,

when constraints are applied to the solution —specially in terrain following when the altitude region is bounded— the solution can no longer be obtained directly through LQR, even in linear systems. Solving optimisation constrained problems (OCP) is the purpose of MPC [40].

The use of MPC is becoming increasingly important for real control systems, with focus on complex non-linear and highly constrained systems, providing path planning guidance and control. It's research for autonomous systems control is heavily oriented towards collision avoidance, minimising effort while applying safety distance constraints to obstacles, applied to ground [41] and aerial vehicles [42] [43], as showcased in Figure 3.14. Terrain following is a specific variation of obstacle avoidance MPC, whereas all obstacles must be avoided from above whilst minimising the altitude, and —as seen in the Literature Review Chapter 2— predictive control requires different cost and constraints depending on the formulation and objective [8] [21].

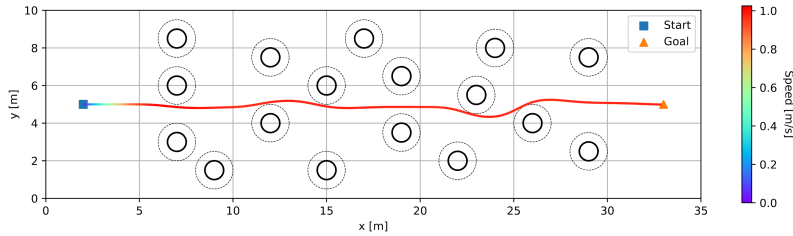


Figure 3.14: Generated trajectory and speed profile with MPC for a quadcopter. Black circles account for obstacles and dashed lines for the constrained region around them [43].

MPC algorithms and solvers can be either linear MPC (LMPC) or non-linear MPC (NMPC), solving convex and non-convex OCP, respectively. Every technique has advantages and draw-backs, that are reviewed to select the most adequate MPC technique for the proposed terrain following system of this thesis.

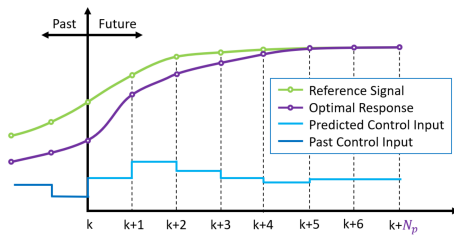
3.4.1 Linear MPC

Linear MPC (LMPC) theory is used to solve convex optimisation problems. This is only possible with a linear state and control system —or linearised from the original non-linear state around an equilibrium point.

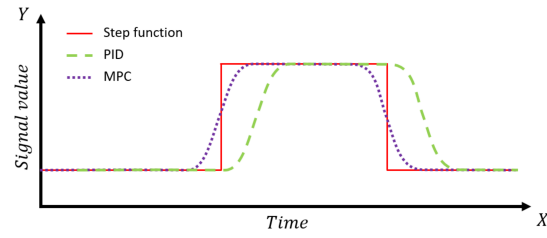
The approach for MPC is based on quadratic programming (QP). From the current initial state k , the solver finds optimal control solutions for a discretized finite horizon N_p in an open-loop problem, shown in Figure 3.15a. After convergence, this provides a control sequence for the discretized horizon, and the first controls of that sequence can be applied to the physical plant [40]. Figure 3.15b provides a comparison between a classic PID control lagging behind the reference state and correcting against disturbance in the feedback, while MPC anticipates with the prediction horizon to provide an optimal path in the reference tracking [41].

Linear MPC has important advantages. First and foremost, a guaranteed convergence as it is a convex problem with a single global minimum, as seen in Figure 3.16a. Additionally, quadric programming is very computationally efficient, able to be programmed in many low-cost devices. As an example, LMPC solvers programmed in field programmable gate

arrays are able to achieve execution times within the order of microseconds (μs), becoming ideal for efficient and fast implementation on real control systems [44].

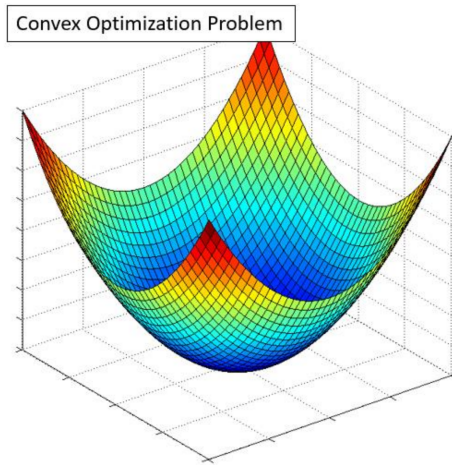


(a) MPC prediction horizon.

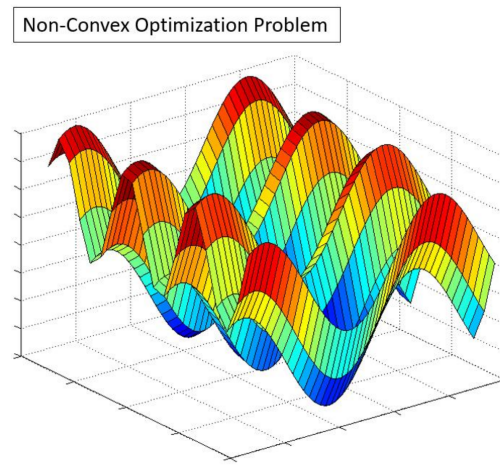


(b) PID and MPC step response.

Figure 3.15: MPC definition and performance diagram [41].



(a) Convex optimisation problem.



(b) Non-convex optimisation problem.

Figure 3.16: Optimisation problem types and graphical representation of state-control space domain for prediction [41].

3.4.2 Non-Linear MPC

The main limitation of linear MPC is their performance levels when applied to non-linear systems. This is specially relevant when applied to underactuated highly non-linear plants, with less control signals than degrees of freedom. To consider the non-linear range and provide adequate performance non-linear MPC is needed (NMPC) [45]. A comparison on an underactuated vessel system between a LMPC applied with the linearised dynamics and NMPC on the non-linear model is shown in Figure 3.17.

Importantly, NMPC solvers rely on sequential quadratic programming (SQP) to solve the non-convex NLP —without guaranteed convergence, as it could get stalled in a local minima shown in Figure 3.16b—, and work on the non-linear dynamics and constraints explicitly. The SQP starts from an initial guess of the whole prediction horizon —not just the current state as in LMPC— and solves for every sampling instant a QP problem. It employs a numerical integration scheme to simulate the dynamics, with

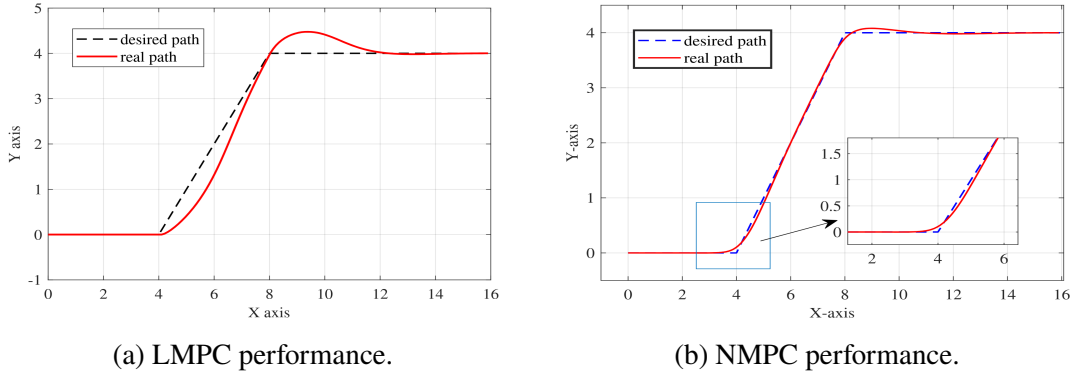


Figure 3.17: Tracking performance comparison of an oblique trajectory between LMPC and NMPC on an underactuated non-linear physical plant [45].

online linearisation of the states and control predictions, rather than the whole reference. This makes the problem much more time-consuming than a LMPC solved with a single QP instance, as every prediction iteration has several inner iterative procedures computed—interior point, gradient, etc. [46].

The implementation of NMPC for real-time iteration (RTI) presents challenges as the system plant keeps evolving as the solution is computed. This can be addressed using a prediction of the initial state to match the expected finish time of the solver, while more recent techniques update the solver during the computation [46]. Similarly, a pre-compensator can be employed [47]. As an alternative, successive online linearized-based NMPC (SOL-NMPC) can be used to reduce the computational times by using the current state of the system as the trimming state and linearising to rapidly solve the complete prediction horizon through a single QP [48]. This method enables coding on FPGA or STM controllers, but still lacks prediction on the full non-linear range as it keeps linearising a non-linear system to use LMPC.

During the last decade, more advanced and fast NMPC solvers have been developed, tailored for real-time implementation of non-linear model predictive control, with increased computational efficiency and speeds. Among the new solvers, the most common are CasADi [49] and PANOC [50]. Whilst PANOC uses a line search FBS (forward-backward-envelope) without matrix operations to perform on embedded systems, CasADi offers a development interface with several NMPC solvers. Evolving from CasADi towards embedded systems, the use of ACADOS is recommended [51]. Nevertheless, these solvers are able to compute large NLP problems within milliseconds (*ms*), enabling the use of NMPC in control loops.

The use of NMPC is deemed very important at low-level flight, with a small error margin in a environment cluttered by obstacles. And with the use of the newer solvers, NMPC is capable of adequately predicting and tracking an optimal terrain following trajectory with altitude range constraints better than LMPC, handling the underactuated non-linear longitudinal motion model of the UAS—with 3-DoF and 2 control signals (δ_e, δ_t). Because of the aforementioned reasons, NMPC is selected in the proposed terrain following system of this thesis.

Chapter 4

Methodology

4.1 Terrain Following System and Architecture

The terrain following system proposed in this thesis uses a dual-stage approach for real-time implementation. The first stage develops the path planning through a geometric method, combining the knowledge from digital elevation models (DEMs) and from on-board obstacle avoidance sensors, with cubic B-spline fitting. Afterwards, the guidance and control stage employs a non-linear MPC (NMPC) for tracking the reference path generated with altitude range and dynamic constraints. Thereafter, the architecture of the terrain following system is presented in Figure 4.1.

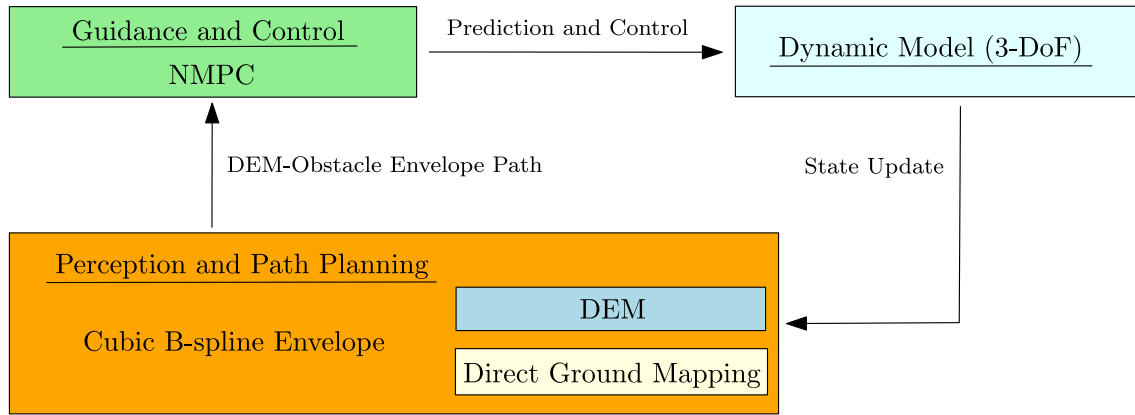


Figure 4.1: Dual-stage terrain following system architecture.

The graphical visualisation of the system is shown in Figure 4.2. In the diagram, the DEM data is represented by the blue points, and the sensor and the direct ground mapping measurements by the dotted black line and the yellow points, respectively. Above both the DEM and measurements, the orange cubic B-spline has been fitted to generate the DEM-Obstacle envelope. Lastly, the envelope is used as the reference trajectory for the green NMPC prediction for optimal guidance and control in low-level terrain following for fixed-wing tier 1 UAS.

The combination of DEM and sensor range measurements is the main innovation point of this thesis, as identified from the state-of-the-art and explained in the Literature Review Chapter 2. This is achieved through the fitting of a cubic B-spline curve above

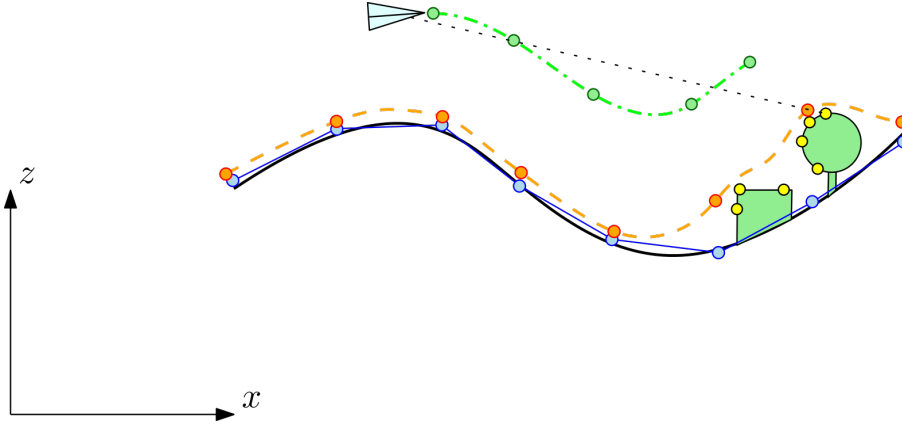


Figure 4.2: Dual-stage terrain following system graphical representation.

the DEM and direct ground mapping measurements, to provide an upper DEM-Obstacle envelope that acts as the reference trajectory for the NMPC. Because the NMPC solver already takes longer computational times to solve the non-convex OCP —incorporating dynamic constraints—, the fitting of the B-spline is formulated as a convex OCP and is solved through fast QP. The cubic B-spline DEM-Obstacle envelope is explained in detail in Section 4.3, and the terrain following NMPC in Section 4.4.

Pursuing real-time implementation, the spline envelope is computed around the UAS in a moving window, just as the NMPC prediction horizon. But while the convex spline envelope ensures a convergence and generates a reference path, the NMPC is non-convex and in some instances the solution may diverge. To maintain the real-time focus, the NMPC has a maximum number of iterations allowed to converge, after which the last converged optimal prediction solution is propagated and re-used. This fallback option allows a continued terrain following system in face of singular instances of non-convergence.

The pseudocode flowchart of the simulation code implementing the terrain following system is shown in Figure 4.3. It implements the dual-stage terrain following system, starting with the envelope that feeds the NMPC —visualising the fallback to the last previously converged optimal predictions from the NMPC. Additionally, because the sensor measurement update f_{sensor} is higher than the NMPC update f_{NMPC} , there is an inner dynamic update loop running at the faster sensor frequency to provide adequate direct ground mapping sampling.

4.2 Simulation Environment

The simulation environment provides the virtual terrain, sensor behaviour and ray-tracing interaction with obstacles, and the aircraft dynamics. Through the simulation environment, the dual-stage terrain following system is tested.

For the virtual terrain of the simulation, a DEM available to the open public could be used, but for the initial testing of this method a fast virtual terrain is generated through a random-based technique. Fractal object algorithms, such as the Diamond-Square algorithm, can quickly generate realistically-looking terrains from random seed values at the corners and center, developing the surface thereafter [52]. Python code for a virtual terrain has been adapted from an available Diamond-Square algorithm example [53].

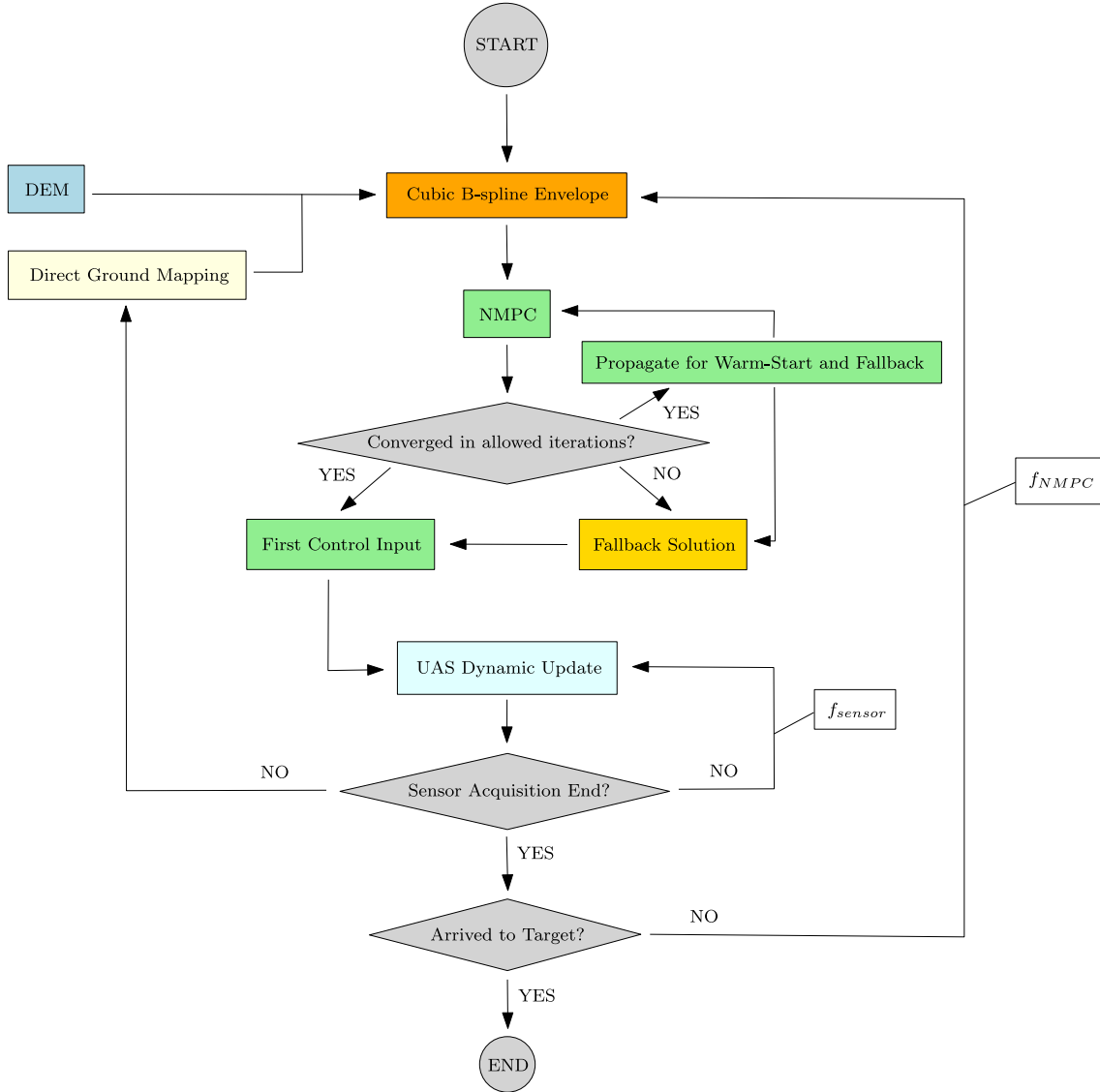


Figure 4.3: Flowchart of the simulations for the terrain following system.

On the resulting Diamond-Square virtual terrain, a start and an end waypoint destinations can be selected. These waypoints define the terrain following trajectory, as seen in Figure 4.4. Interpolating over the virtual terrain surface, the 2D elevation map and its slope at every instance can be obtained, showed in Figure 4.5. The resulting 2D elevation is used as the ground truth, and a DEM is generated from it by applying a certain white Gaussian noise with a standard deviation σ_{dem} to be used in the TF algorithms.

The 2D terrain is populated, afterwards, with additional obstacles that the DEM will not have —missing obstacle data from DTM. These have been modelled as rectangular boxes for buildings and a circle with a thin rectangle for trees.

Because the DEM does not provide information on the additional obstacles, a range-finder sensor capability is implemented in the simulation. This enables the UAS to scan the terrain with ray-tracing sensors with a certain elevation angle ϵ and maximum range d_{max} —depending on the parameters and uncertainty values a LiDAR or RADAR performance is obtained. Although the simulation environment allows for several sensors scanning

Virtual Terrain and NOE Flight Path

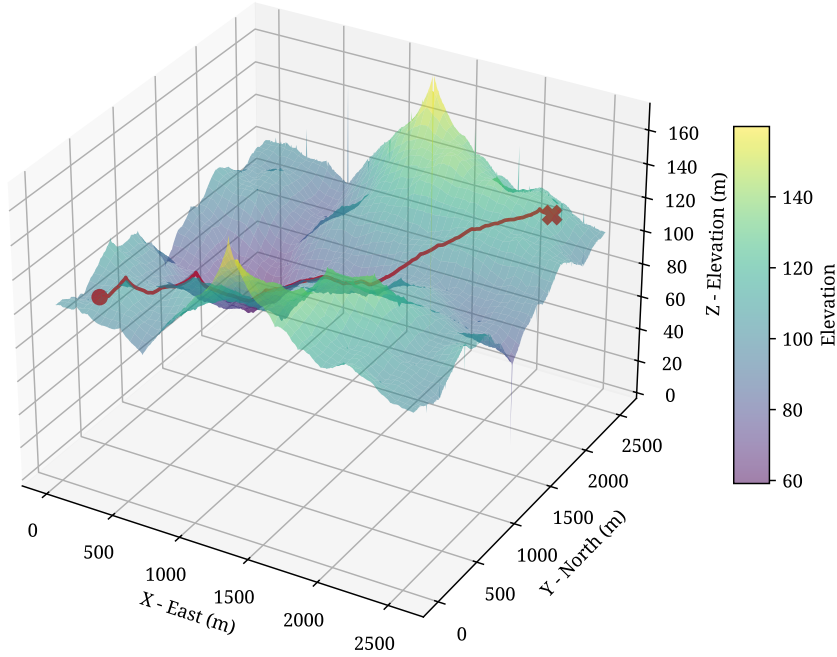


Figure 4.4: Virtual terrain generated through the Diamond-Square fractal algorithm, with a projected NOE flight between a start and end waypoints.

simultaneously, the performance and sensitivity analysis of the TF system is undertaken with a single sensor to demonstrate the effectiveness on a low-cost sensor suite.

In the simulations, all uncertainty is transferred to the DEM and direct ground mapping, using the position of the UAS and the dynamics as ground truth for the envelope and NMPC. The DEM uncertainty σ_{dem} is directly applied to the elevation data, but the ground measurements' uncertainty must be propagated from Equation 3.40. Employing a 1st order Taylor's series for uncertainty propagation of a function $f(x)$:

$$\sigma_f^2 = \sum_i \left(\frac{df}{dx_i} \right)^2 \sigma_{x_i}^2 + \sum_{i \neq j} 2 \frac{df}{dx_i} \frac{df}{dx_j} \sigma_{x_{ij}} \quad (4.1)$$

And considering that all parameters are independent without covariance ($\sigma_{x_{ij}} = 0$), the uncertainty on the ground measurements $\sigma_{x_o}, \sigma_{z_o}$ can be defined by the navigation and obstacle avoidance sensors' standard deviations:

$$\begin{aligned} \sigma_{x_o}^2 &= \sigma_{x_s}^2 + (\cos(\theta - \epsilon))^2 \sigma_d^2 + (d \sin(\theta - \epsilon))^2 (\sigma_\theta^2 + \sigma_\epsilon^2) \\ \sigma_{z_o}^2 &= \sigma_{z_s}^2 + (\sin(\theta - \epsilon))^2 \sigma_d^2 + (d \cos(\theta - \epsilon))^2 (\sigma_\theta^2 + \sigma_\epsilon^2) \end{aligned} \quad (4.2)$$

This expression already provides us with insight on the effect of the parameters contributing to the total vertical obstacle uncertainty σ_{z_o} . Considering forward looking sensors and measurements ($\theta - \epsilon \approx 0$) and simplifying accordingly, it is observed that it is highly sensitive to the angular uncertainty as it scales quadratic to the distance—an effect visible in the sensitivity analysis performed in this study:

$$\sigma_{z_o}^2 \approx \sigma_{z_s}^2 + d^2 (\sigma_\theta^2 + \sigma_\epsilon^2) \quad (4.3)$$

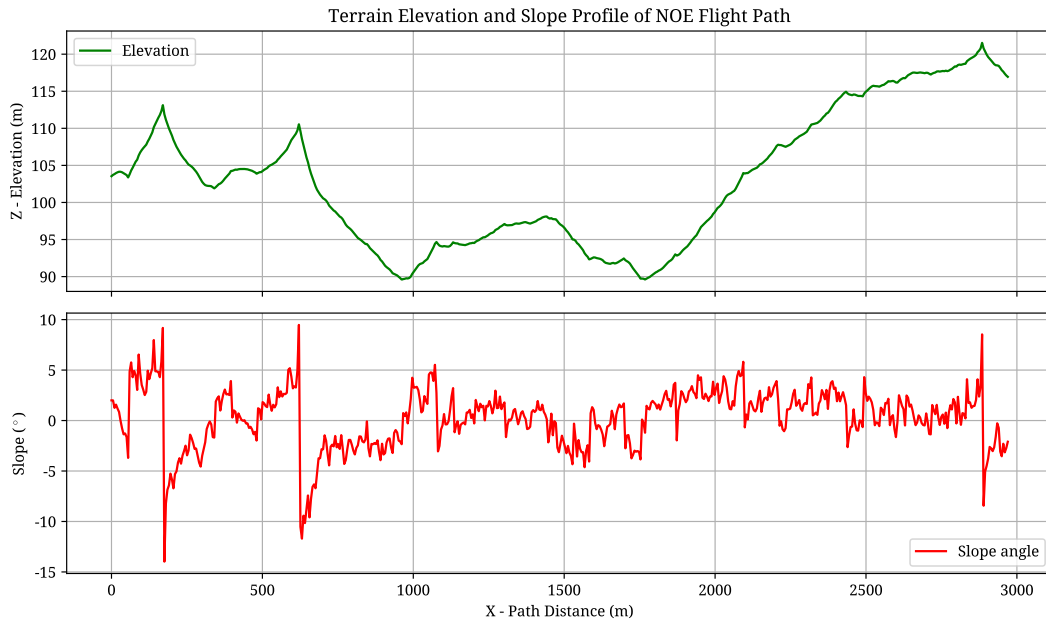


Figure 4.5: Elevation profile and slope angle of the projected NOE path.

Finally, the dynamic integrator implemented is an implicit Runge-Kutta, employing the motion model defined in Section 3.1. The complete simulation environment can be seen in Figure 4.6, depicting a simulation instance of the UAV with several rangefinder sensors performing direct ground mapping over the terrain and additional obstacles. In the snapshot of the running simulation, the effect of distance on the uncertainty is appreciated in the larger dispersion of measurement points (yellow dots) on the far-away tree, while the closer building has more accurate sensor measurements.

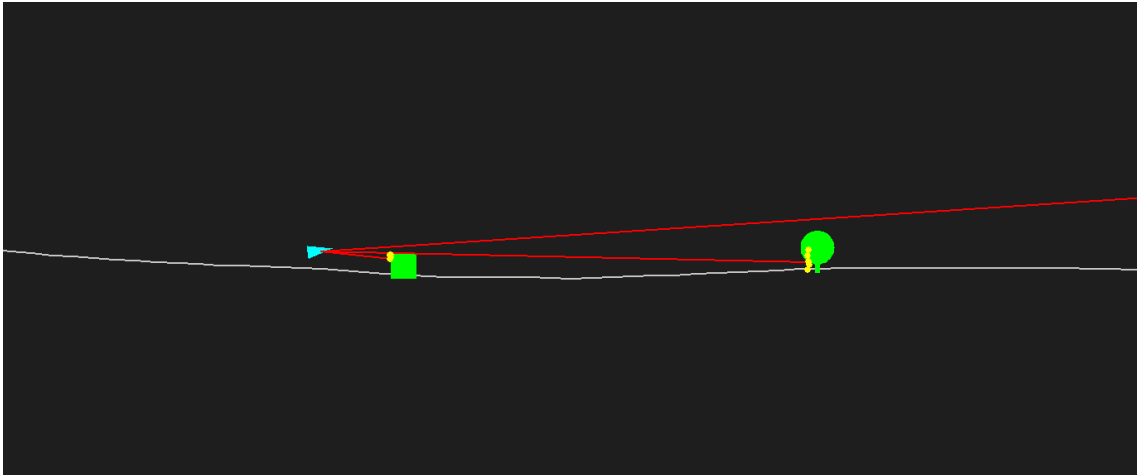


Figure 4.6: Online simulation environment implementing elevation virtual terrain and additional obstacles, with three rangefinder sensors onboard and their measurements.

4.3 Cubic B-Spline DEM-Obstacle Envelope

The DEM-Obstacle envelope is the first part of the dual-stage terrain following system proposed. It is the main innovation of this thesis in terrain following, combining the obstacle knowledge from DEM and direct ground mapping through onboard sensors.

Collision avoidance requires the combination of data from the DEM and the direct ground mapping to cover all obstacles. This can be done through a simple stair envelope, that raises values of the DEM that border an obstacle measurement to the same height—shown in Figure 4.7. Otherwise, and showing a better tracking performance even for NMPC, the use of a spline envelope is advantageous [17] [21].

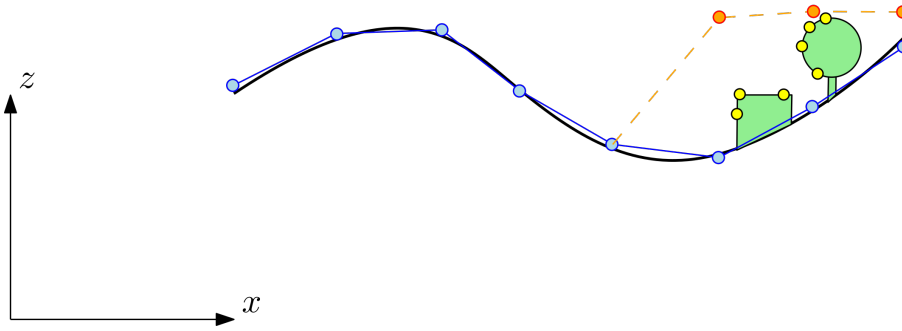


Figure 4.7: Stair DEM-Obstacle terrain envelope.

The envelope is generated through a geometric spline method. The use of splines for terrain following has been discussed in the Literature Review Chapter 2, although they are solved with dynamic constraints requiring a SQP solver for the NLP [17]. Instead, the proposed method leaves the dynamic feasibility to the NMPC and computes the envelope with simpler cubic B-splines modulating the 2nd derivative aggressiveness to smooth the trajectories. Coupled with constraints to ensure a fitting above both DEM and obstacle measurements, it can be defined as a convex OCP formulation able to be solved through QP—leaving computational time for the tracking to the NMPC. The cubic B-spline optimisation can also restrict the derivatives of the curves, maintaining its convex properties [54].

The advantages of the proposed cubic B-spline are:

- continuity to the 1st and 2nd derivative.
- aggressiveness modulation of the 2nd derivative ($0 \leq a \leq 1$).
- OCP formulation to ensure an upper envelope over all obstacle data.
- convex OCP problem with fast QP solvers and guaranteed convergence.
- risk-awareness through chance constraint formulation and safety margins over DEM and obstacle measurements with risk allowance ($0 \leq \Delta_{dem}, \Delta_o \leq 1$).

The envelope is to be fitted in a moving window of length L around the UAS, with $1/4$ of its length behind the vehicle to ensure the reference trajectory is maintained in descents after an obstacle has been overcome. The resulting envelope, with high aggressiveness $a \leq 1$ and low smooth aggressiveness $a \geq 0$, is presented in Figure 4.8. It visualises

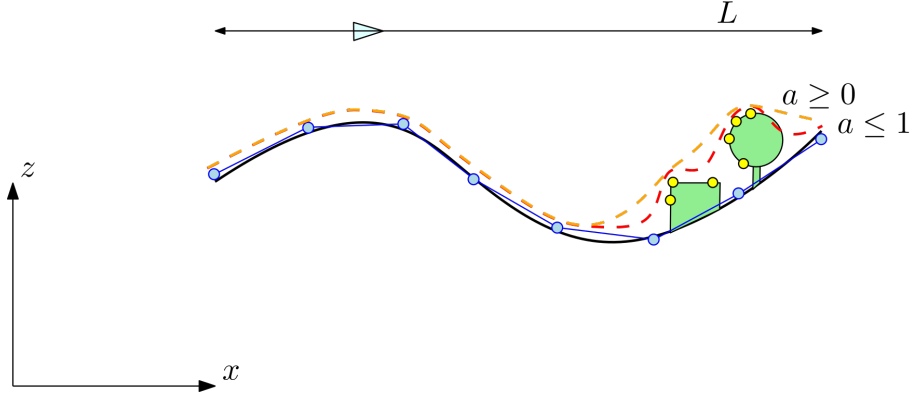


Figure 4.8: Cubic b-spline DEM-Obstacle terrain envelope graph, with high and low aggressiveness effect.

how the orange DEM-Obstacle envelope is fitted but constrained above all blue DEM and yellow obstacle measurement points.

The use of cubic B-spline curves for UAV trajectory is applied in this thesis for a vertical 2D envelope. This implementation addresses the Terrain Following (TF) problem, while the addition of Terrain Avoidance (TA) would require the modelling of a 3D space and a surface envelope. The cubic B-spline method on surfaces is capable to be used for surface modelling and fitting over point cloud datasets, through least squares approximation [55]. The selection of the cubic B-spline DEM-Obstacle envelope ensures expansion capabilities into the TF/TA with full 6-DoF dynamics.

4.3.1 Cubic B-spline Curves

The fundamentals of Basis Splines are explained and presented in detail in the renown NURBS book [56]. A B-spline curve $C(u)$ is a piecewise rational curve composed by polynomial basis functions $N_{i,p}(u)$:

$$C(u) = \sum_{i=0}^n N_{i,p}(u) P_i \quad (4.4)$$

The $n+1$ polynomial basis functions $N_{i,p}(u)$, of p^{th} degree, are modulated by associated control point coefficients P_i over a knot domain u , that define the shape of the piecewise rational B-spline curve $C(u)$.

The most commonly used basis functions $N_{i,p}(u)$ are the uniform cubic ($p=3$) basis functions spanning all domain u , defined over $n+p+2$ total knots clamped at the extremes of the range of u (additional p -times at each end). An example is provided in Figure 4.9. As a property of the basis functions, without coefficients, the sum at any point over the non-clamped domain is unitary:

$$\sum_{i=0}^n N_{i,p}(u) = 1 \quad \forall u \in [u_p, u_{n+1}] \quad (4.5)$$

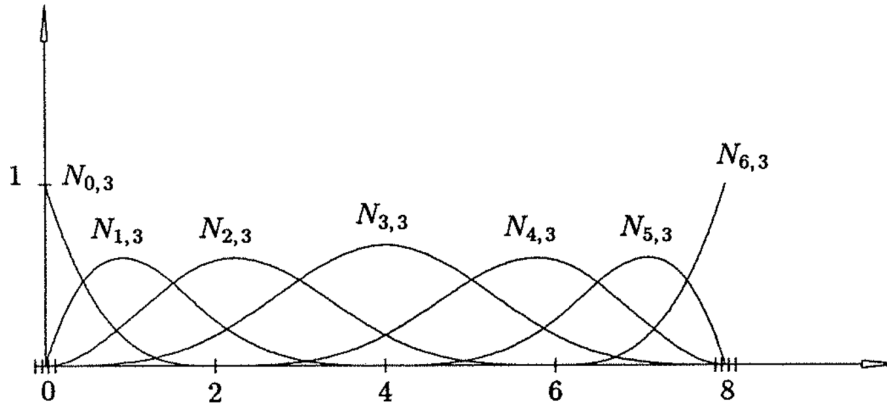


Figure 4.9: Uniform cubic basis functions example for $n + 1 = 7$ control points [56].

The derivatives of B-spline curves can be obtained from the reduced order basis functions $N_{i,p-k}(u)$, the control point coefficients P_i and knot positions u_i . Defining the k^{th} derivative of a B-spline curve $C^{(k)}$:

$$C^{(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u) P_i^{(k)} \quad (4.6)$$

With:

$$P_i^{(k)} = \begin{cases} P_i & k = 0 \\ \frac{p-k+1}{u_{i+p+1} - u_{i+k}} \left(P_{i+1}^{(k-1)} - P_i^{(k-1)} \right) & k > 0 \end{cases} \quad (4.7)$$

Because the cubic B-splines ($p = 3$) are used for the DEM-Obstacle envelope, a cost is desired on the 2nd derivative ($k = 2$) to smooth the curvature. The reduced first order basis functions $N_{i,1}(u)$ —a set of triangular hat functions— evaluate to 1 over their peak point x_i , and 0 at the peaks of all others. The peak points x_i of the linear basis functions are $n + 1$ uniformly spaced positions starting and ending in the boundaries of the non-clamped knot domain (u_p, u_{n+1}). If the knots have equal intervals ($u_{i+p+1} - u_{i+k} = \text{const}$), then the 2nd derivative expression at the peak points x_i depends only on the corresponding neighbouring control point coefficients P_i of the i^{th} basis peak:

$$C^{(2)}(x_i) = \text{const} \cdot (P_{i+2} - 2P_{i+1} + P_i) \quad (4.8)$$

The resulting expression, without the constant term, can be used for a fast convex OCP cost on the 2nd derivative at the evaluated terrain points of the DEM-Obstacle envelope, based only on the control points that constitute the optimisation variables for the OCP.

4.3.2 Chance Constraint

The OCP definition for the envelope provides safety margins based on the risk allowance considered for an inequality constraint, formulated as chance constraint [57]. The DEM-Obstacle envelope curve must have an elevation z_u higher than the DEM points z_{dem} and the direct ground mapping obstacles z_o . These are two inequality constraints, and each has an associated level of risk allowance Δ_{dem}, Δ_o .

The definition of the chance constraint is exemplified with the obstacle measurements from the direct ground mapping, but is defined in the same manner for the DEM. Starting from the inequality constraint, in which the envelope z_u is above the true obstacle measurement $z_{o,true}$:

$$z_u(x) \geq z_{o,true}(x) \quad (4.9)$$

Although the uncertain measurement captured z_o is equal to the nominal elevation and adding a noisy value ζ :

$$z_o = z_{o,true} + \zeta \quad (4.10)$$

Thereafter, the inequality constraint can be redefined in terms of the uncertain measurement:

$$z_u \geq z_o - \zeta \quad (4.11)$$

Reformulated into a chance constraint, it is considered as the probability of the the envelope being higher than the true value. Satisfying:

$$\mathbb{P}(z_u \geq z_o - \zeta) \geq 1 - \Delta_o \quad (4.12)$$

Reordering the terms from the left-hand side of the equation, we can use the cumulative distribution function ϕ to define the probability of an uncertain term ζ with a standard deviation σ_o —assuming a Gaussian normal distribution— and zero mean to be higher than a certain value:

$$\mathbb{P}(\zeta \geq z_o - z_u) = 1 - \phi\left(\frac{z_o - z_u}{\sigma_o}\right) \geq 1 - \Delta_o \quad (4.13)$$

Developing the expression, and using the quantile —inverse cdf— equation assuming $\phi^{-1}(\phi(x)) = x$, the chance constraint is obtained. The value of the envelope elevation z_u is based on the measurement z_o , the vertical standard deviation σ_o and the risk allowed Δ_o :

$$z_u \geq z_o - \sigma_o \phi^{-1}(\Delta_o) \quad (4.14)$$

In parallel, the chance constraint for the DEM points follows the same procedure. Arriving to the expression:

$$z_u \geq z_{dem} - \sigma_{dem} \phi^{-1}(\Delta_{dem}) \quad (4.15)$$

The effect of the risk allowance on the chance constraint formulation is visualised in Figure 4.10. Three main scenarios are presented:

- low risk ($\Delta < 0.5 \rightarrow \phi^{-1}(\Delta) < 0$): the envelope will be fitted at a higher elevation than the measurements, as it can't risk the chance that most of the points have been estimated below the true value.
- neutral risk ($\Delta = 0.5 \rightarrow \phi^{-1}(\Delta) = 0$): the envelope is fitted just above the highest measurement. This is the default risk setting for the terrain following simulations.
- high risk ($\Delta > 0.5 \rightarrow \phi^{-1}(\Delta) > 0$): the envelope is allowed to be fitted below the points. This situation can provide an envelope closer to the true value if the sampling of uncertain measurements has points higher than the nominal value.

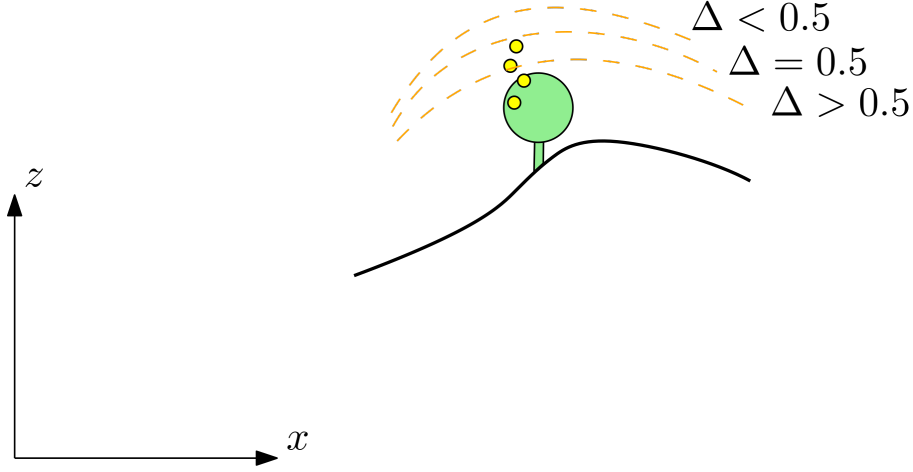


Figure 4.10: Risk allowance scenarios on the DEM-Obstacle envelope.

4.3.3 Convex OCP Definition

The DEM-Obstacle envelope $z_u(x)$ is obtained through the optimisation of the control point coefficients P_i of a cubic B-spline curve. The cubic basis functions $N_{i,3}$ are defined over $n+1$ terrain points x_i from the DEM dataset —knot domain boundaries and peaks for the linear basis functions $N_{i,1}$ —, and evaluated over the obstacle measurements coordinates x_o stored from the direct ground mapping that are found within the moving window L .

The cost J is composed by a quadratic function that balances the height deviation between the envelope elevation and the DEM J_h , and the 2nd derivative values at the terrain points J_d —recalling Equation 4.8— through an aggressiveness parameter a .

Finally, the curve elevation points are given as an equality constraint providing the B-spline definition. While the inequality chance constraints are imposed to ensure clearance over both obstacles and the DEM, defined in Equations 4.14 and 4.15.

OCP minimise $J = a J_h + (1 - a) J_d$

$$J_h = \sum_{i=0}^n (z_u(x_i) - z_{dem}(x_i))^2 \quad (\text{deviation cost})$$

$$J_d = \sum_{i=0}^{n-2} (P_{i+2} - 2P_{i+1} + P_i)^2 \quad (\text{derivative cost})$$

$$\text{subject to } z_u(x) = \sum_{i=0}^n N_{i,3}(x) P_i \quad (\text{eq. cstr., cubic B-spline})$$

$$z_u(x_i) \geq z_{dem}(x_i) - \sigma_{dem} \phi^{-1}(\Delta_{dem}) \quad (\text{ineq. cstr., DEM clearance})$$

$$z_u(x_o) \geq z_o(x_o) - \sigma_o \phi^{-1}(\Delta_o) \quad (\text{ineq. cstr., obstacle clearance})$$

The QP for the convex OCP has been implemented through CasADi, using the OSQP solver and employing C++ compilation and execution to improve solver speeds [49]. An example of the DEM-Obstacle envelope fitting is presented in Figure 4.11. Because of the low obstacle risk, the envelope leaves a higher margin over the uncertain obstacle measurements. On the other hand, the high-risk allowance for the DEM allows the envelope to lower beneath the DEM values.

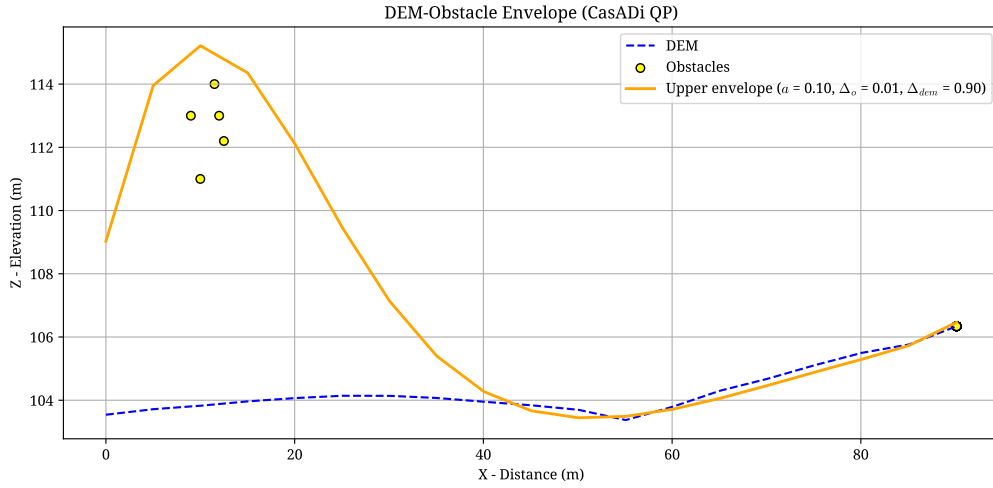


Figure 4.11: DEM-Obstacle envelope generated through QP using CasADi.

4.4 TF Non-Linear Model Predictive Control

The non-linear model predictive control (NMPC) provides the guidance and control input prediction for an adequate tracking of the DEM-Obstacle envelope generated. It is the second part of the dual-stage terrain following system proposed.

The terrain following NMPC provides dynamic feasibility, constrained with the longitudinal motion model of the fixed-wing UAS, from Equation 3.30. It is used for real-time implementation, computing a prediction horizon of optimal states “ x ” and controls “ u ”—refer back to Equation 3.1—in a moving window. The prediction horizon is computed using $m + 1$ discretized timesteps with a fixed dt . The cost function minimises deviation to the desired height h_{ref} above the DEM-Obstacle envelope, while constraints enforce an h_{min} and h_{max} altitudes. The NMPC prediction is exemplified in Figure 4.12.

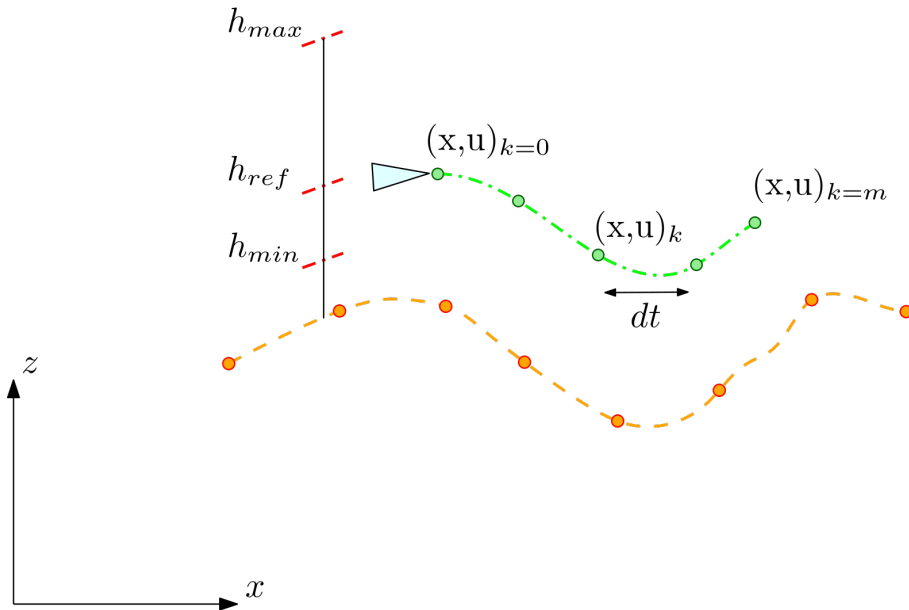


Figure 4.12: Terrain following NMPC graphical representation.

The main characteristics of the terrain following NMPC are:

- optimal control prediction horizon ($H_p = m \, dt$) for real-time moving window.
- evaluation of full non-linear dynamics ensuring performance and feasibility.
- non-convex OCP solved through SQP, without guaranteed convergence.
- minimisation of tracking error for DEM-Obstacle envelope and control jerk.
- constrained altitude range over DEM-Obstacle envelope, state and control limits.

4.4.1 Non-Convex OCP Definition

The NMPC's OCP definition is based on a quadratic cost J evaluating the deviation from h_{ref} for the envelope tracking J_a , a speed V_{ref} cost tracking J_s , a pitch rate cost J_q and a control jerk penalty divided into elevator $J_{j,e}$ and throttle $J_{j,t}$. The main interest other than the altitude cost, is the control jerk penalty to provide a smooth trajectory —well conditioned—, but it depends on values from different timesteps, making it inefficient for the solver. Applying a velocity and rate costs improves the computational times, because they depend only on a single timestep. These costs contribute to lower the control jerk on the input derivatives. Every cost has an associated weight ($w_a, w_s, w_q, w_{j,e}, w_{j,t}$) for balancing the NMPC objective J .

The dynamic feasibility is imposed through equality constraints, as well as the initial state $\mathbf{x}_0$. The remaining inequality constraints limit the altitude range over the DEM-Obstacle envelope, and the state and control values are bounded by upper and lower limits defining the flight envelope of the aircraft.

OCP minimise $J = J_a + J_s + J_q + J_{j,e} + J_{j,t}$

$$J_a = w_a \sum_{i=0}^m (z_{k+i|k} - z_u(x_{k+i|k}) - h_{ref})^2 \quad (\text{altitude cost})$$

$$J_s = w_s \sum_{i=0}^m \left(\sqrt{u_{k+i|k}^2 + w_{k+i|k}^2} - V_{ref} \right)^2 \quad (\text{speed cost})$$

$$J_q = w_q \sum_{i=0}^m (q_{k+i|k})^2 \quad (\text{rate cost})$$

$$J_{j,e} = w_{j,e} \sum_{i=1}^m \left(\frac{\delta_{e_{k+i|k}} - \delta_{e_{k+i-1|k}}}{dt} \right)^2 \quad (\text{elevator jerk cost})$$

$$J_{j,t} = w_{j,t} \sum_{i=1}^m \left(\frac{\delta_{t_{k+i|k}} - \delta_{t_{k+i-1|k}}}{dt} \right)^2 \quad (\text{throttle jerk cost})$$

subject to $\mathbf{x}_{k+i+1|k} = f(\mathbf{x}_{k+i|k}, \mathbf{u}_{k+i|k})$ (eq. cstr., dynamics)

$\mathbf{x}_{i=0} = \mathbf{x}_0$ (eq. cstr., current state)

$h_{min} \leq z_{k+i|k} - z_u(x_{k+i|k}) \leq h_{max}$ (ineq. cstr., AGL range)

$\mathbf{x}_{l,lim} \leq \mathbf{x}_{k+i|k} \leq \mathbf{x}_{u,lim}$ (ineq. cstr., state limits)

$\mathbf{u}_{l,lim} \leq \mathbf{u}_{k+i|k} \leq \mathbf{u}_{u,lim}$ (ineq. cstr., control limits)

The SQP for the terrain following NMPC has also been implemented through CasADi, using the IPOPT solver [49]. The interior point method solver is very versatile and capable of solving a prediction horizon $H_p = 2$ s ($m = 20$, $dt = 0.1$ s) within 60 milliseconds (ms) when compiled in C++ in a single core of an Intel N200 CPU with a maximum clock speed of 1GHz. This performance is achieved through multiple shooting formulation, declaring intermediate optimisation variables for states and controls at every timestep of the prediction for better conditioning, and an adequate sparse structure. Single shooting only declares intermediate control variables and states are used implicitly, reducing the OCP dimensionality, but it provides a worse performance in large stiff NLP. The computational times, although already enough for a control loop, could still be lowered with other NMPC solvers, designed for real-time implementation on embedded devices.

The cold-start initialisation of the NMPC uses the trim condition obtained in Equation 3.35 to provide a level flight initial guess. Afterwards the NMPC uses a warm-start cycle reusing the previous prediction horizon, advancing one step and propagating the last one using the last predicted control inputs. As the non-convex OCP doesn't guarantee performance, the solver has an allocated maximum number of iterations to converge. If it fails, the propagated warm-start guess is used as the fallback solution for control. This mechanism was already explained and shown in Figure 4.3.

An example of a converged terrain following NMPC prediction horizon is shown in Figure 4.13. It showcases the values of the states —blue graphs— and the controls —green plots. The state and control values include the limits —red boundaries— and always stay within, as they were imposed through inequality constraints in the OCP definition. The resulting prediction maintains a balance between the altitude tracking and smoothness of the trajectory, as the control input jerk cost is effective in slowing down the variation. Tracking the envelope, the UAS initially performs a descent and stabilises over the h_{ref} but relaxes the trajectory without aggressively following the terrain contour. The speed and rate cost are also appreciated, as the velocity components perform a stable oscillation around their reference value, and the pitch rate q returns towards zero after the initial dive of the UAS to reach the desired altitude.

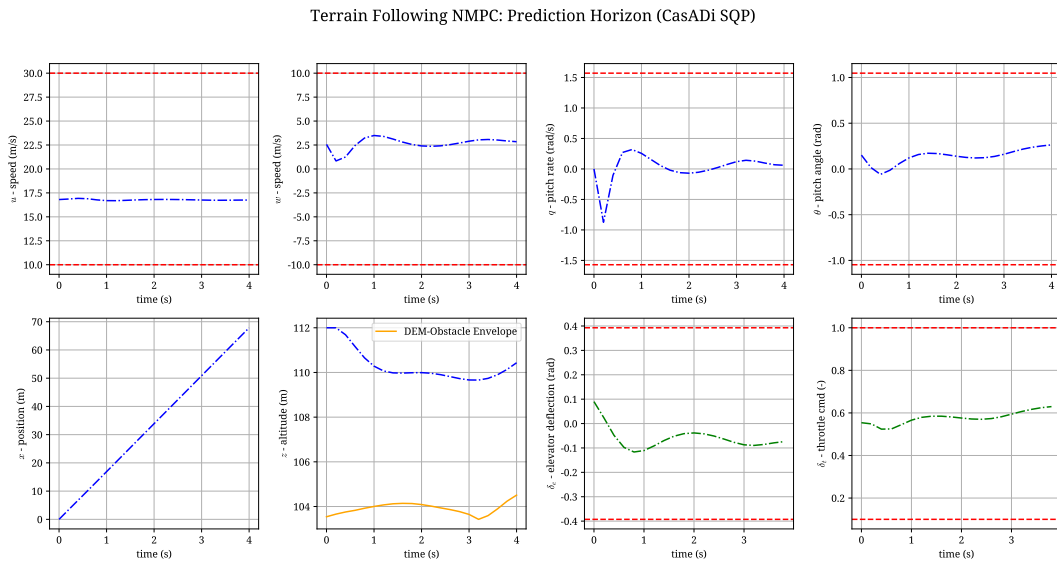


Figure 4.13: Terrain following NMPC prediction horizon through SQP using CasADi.

Chapter 5

Results

5.1 Evaluation of the Stair and Cubic B-spline Envelope

The dual-stage terrain following system proposed —involving a path planning DEM-Obstacle envelope, and a guidance and control NMPC for the envelope altitude tracking—is initially tested through a comparison between a simple stair envelope and the cubic B-spline envelope. This comparison allows to highlight the merits and advantages of the developed cubic B-spline envelope in front of the more simple stair envelope.

Recalling the definitions, the stair envelope raises the DEM values around an obstacle measurement to the same level, while the cubic B-spline envelope fits a piecewise rational curve to provide an upper envelope with aggressiveness modulation. Return to Section 4.3 to revise the full description.

For the simulation, the UAS employs a simple sensor suite with a single forward facing rangefinder sensor, without presence of navigation or obstacle avoidance uncertainties. The values of the sensor suite and uncertainties are shown in Table 5.1. The effect of the uncertainties are described, later on, in the sensitivity analysis in Section 5.2.

Table 5.1: Navigation and sensor suite parameters.

Navigation and Sensor Parameters		Value
Rangefinder sensor	Number (-)	1
	f_{sensor} (Hz)	50
	Range d_{max} (m)	200
	Angle ϵ ($^{\circ}$)	10
DEM uncertainty	DEM σ_{dem} (m)	0.0
Navigation uncertainty	Hor. pos. σ_{x_s} (m)	0.0
	Vert. pos. σ_{z_s} (m)	0.0
	Pitch σ_{θ} ($^{\circ}$)	0.0
Obstacle avoidance uncertainty	Distance σ_d (m)	0.0
	Angle σ_{ϵ} ($^{\circ}$)	0.0

The cubic B-spline DEM-Obstacle envelope parameters for the first stage of the terrain following system are presented in Table 5.2, and the envelope tracking NMPC parameters are presented in Table 5.3. These values stay fixed throughout all the simulations in this study, except for the risk allowance parameters from the cubic B-spline envelope OCP. After the sensitivity analysis, the effect of the risk allowance is evaluated as means to improve terrain following performance under presence of uncertainty, in Section 5.3.

Lastly, all simulations are performed over the same half kilometre stretch (500 m) of non-flat terrain generated through a Diamond-Square fractal algorithm. Over that terrain there is an unknown building and a tree, measuring 6 m and 10 m, respectively. The environment terrain parameters are described in Table 5.4.

Table 5.2: Cubic B-spline DEM-Obstacle envelope parameters.

Cubic B-spline Parameters		Value
Envelope Fitting	Solver	CasADi OSQP
	Moving window L (m)	250 (1/4 aft, 3/4 fore)
	Aggressiveness a (-)	0.01
Risk allowance	Δ_{dem} (-)	0.5
	Δ_o (-)	0.5

Table 5.3: Terrain following NMPC parameters.

NMPC Parameters		Value
Prediction horizon	Solver	CasADi IPOPT
	Max. iter. (-)	100
	f_{NMPC} (Hz)	10
	dt (s)	0.1
	m (-)	20
Cost weights	w_a (-)	1
	w_s (-)	1
	w_q (-)	1
	$w_{j,e}$ (-)	3
	$w_{j,t}$ (-)	0.2
Reference values	h_{ref} (m)	5
	V_{ref} (m/s)	17
Altitude range	h_{min} (m)	2
	h_{max} (m)	15
State and control limits	$x_{l,lim}$	$[10, -10, -\pi/2, -\pi/3, -, -]$
	$x_{u,lim}$	$[30, 10, \pi/2, \pi/3, -, -]$
	$u_{l,lim}$	$[-\pi/8, 0.1]$
	$u_{u,lim}$	$[\pi/8, 1.0]$

Table 5.4: Virtual terrain parameters.

Simulation Environment Parameters		Value
Terrain generation	Source	Diamond-Square Fractal
	Spacing (m)	5
	Total Drop (m)	11.123
	Max Slope (°)	9.176
	Distance (m)	500
Building obstacle	Height (m)	6
	Width (m)	6
	Position (m)	300
Tree obstacle	Height (m)	10
	Width (m)	8
	Position (m)	400

5.1.1 Stair Envelope Simulation

The terrain following simulation using a stair DEM-Obstacle envelope is exemplified through Figures 5.1, 5.2 and 5.3. The progression showcases how the envelope — represented by the dashed orange curve— influences the NMPC prediction —given by the green dashed trajectory.

Initially, in Figure 5.1, the stair envelope only includes the building obstacle, because it is the only measured, while no knowledge is yet gathered on the tree. The NMPC prediction is enough to compute a possible avoidance path over the building obstacle. Nevertheless, in Figure 5.2, the newly detected tree causes a much taller stair on the DEM-Obstacle envelope and the NMPC solver is not able to converge on such an aggressive manoeuvre in the allocated iterations, causing fallback and propagation —seen on the ever upwards trajectory prediction. After several fallbacks, an optimal solution converges and the UAS continues to follow its path, as seen in Figure 5.3.

The resulting flight trajectory and elevation profiles are presented in Figure 5.4. From the trajectory profile we can observe that there were 19 NMPC solver failures, concentrated around the obstacles. It is normal that the elevation to the DEM increases over the obstacles, but instead of a stable envelope AGL height around the target h_{ref} the elevation dips down to the minimum level h_{min} over the tree obstacle.

The state and control log is presented in Figure 5.5 The stair envelope forces the NMPC to perform a very aggressive and harsh tracking, with high oscillations in the control inputs, forcing the aircraft to its flight envelope limits and reaching the maximum pitch rate q . Exemplified by the magenta bars, the solver failures concentrate on the harsh manoeuvre regions with high pitch rate values, denoting increasing non-linear regions that don't provide an adequately conditioned problem for the NMPC solver.

Concluding, the stair envelope is unsafe: harsh manoeuvres with low minimum altitude over obstacles, causing many solver failures. A longer NMPC prediction horizon is required to compensate the effects of the stairs, but that would increase the computational times and decrease its real-time capability.

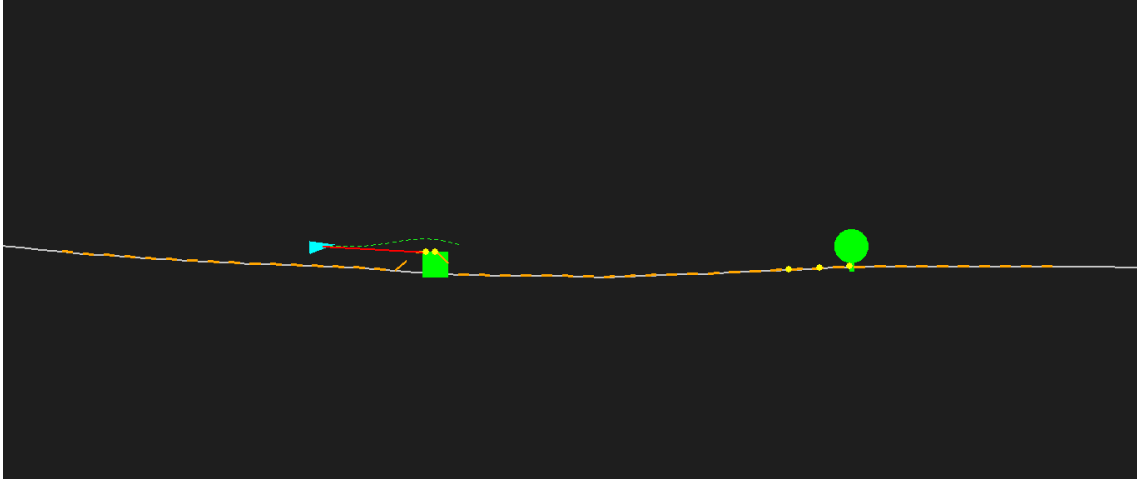


Figure 5.1: Terrain following simulation instance using stair DEM-Obstacle envelope.

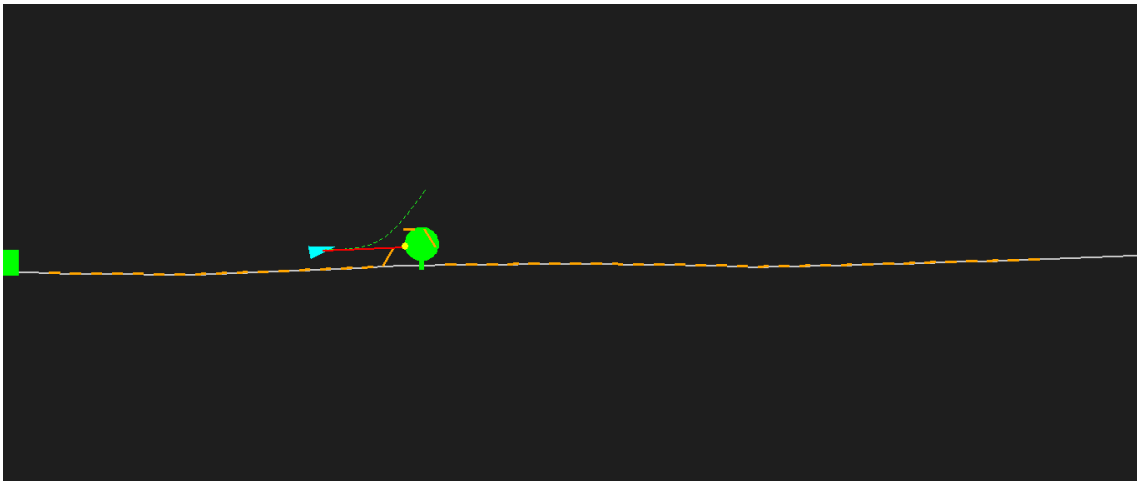


Figure 5.2: Terrain following simulation instance using stair DEM-Obstacle envelope, with NMPC solver failure.

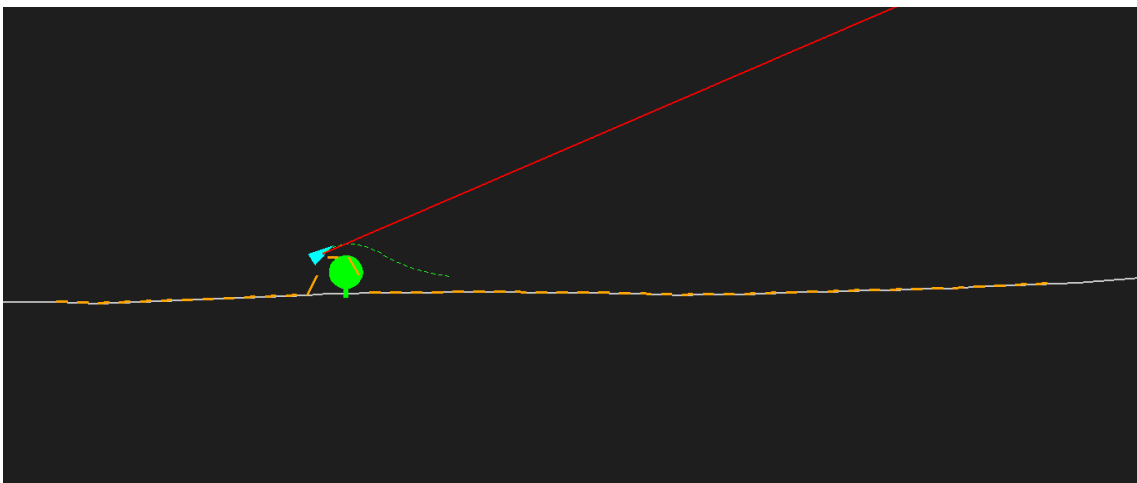


Figure 5.3: Terrain following simulation instance using stair DEM-Obstacle envelope, after NMPC convergence and recovery.

Terrain Following Simulation: UAS Trajectory and AGL Profiles

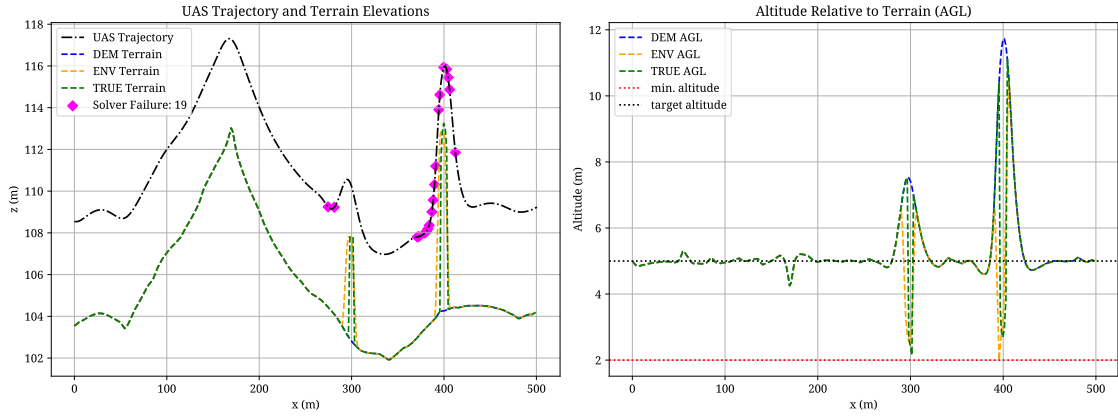


Figure 5.4: Trajectory profiles from the stair DEM-Obstacle envelope simulation.

Terrain Following Simulation: State/Control Logs (Solver Failure: 19)

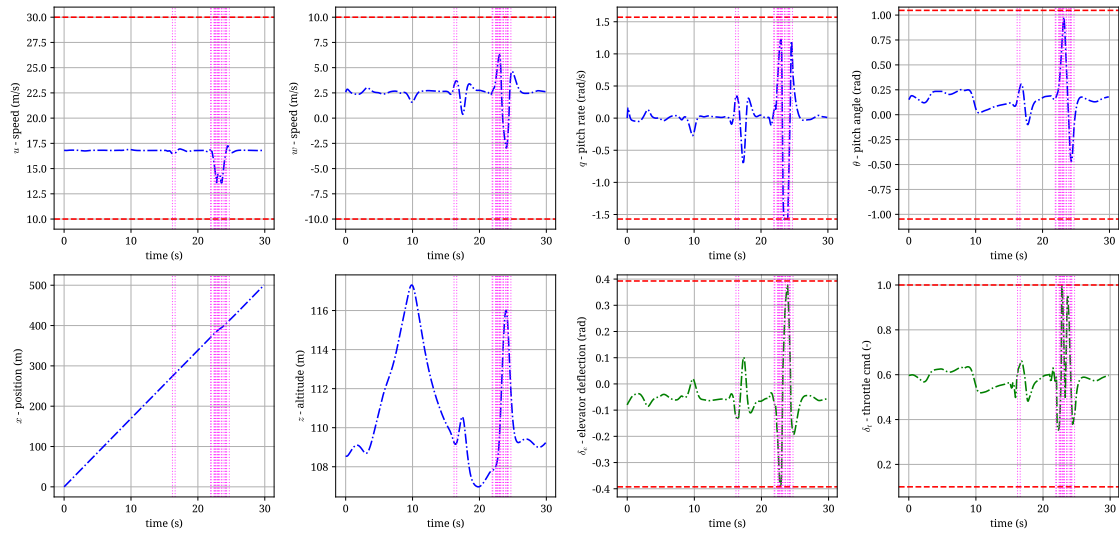


Figure 5.5: State and control logs from the stair DEM-Obstacle envelope simulation.

5.1.2 Cubic B-spline Envelope Simulation

The cubic B-spline DEM-Obstacle envelope yields an improved performance to the terrain following system, providing a better conditioned terrain path to be tracked by the NMPC. Because of the aggressiveness modulation, it can smooth out terrain features and anticipates pull-up and pull-down manoeuvres. The NMPC now plans ascents and descents over the cubic envelope before the prediction reaches the obstacle—opposite to the stair envelope—, reducing the strain on the SQP solver and decreasing the number of solver failures as a smoother and faster control is provided.

The process is shown in Figures 5.6, 5.7 and 5.8. The cubic envelope is generated as data from the sensor is gathered—when the NMPC pitches upwards the sensor follows and estimates the full height of the obstacle—and the spline grows to be fitted.

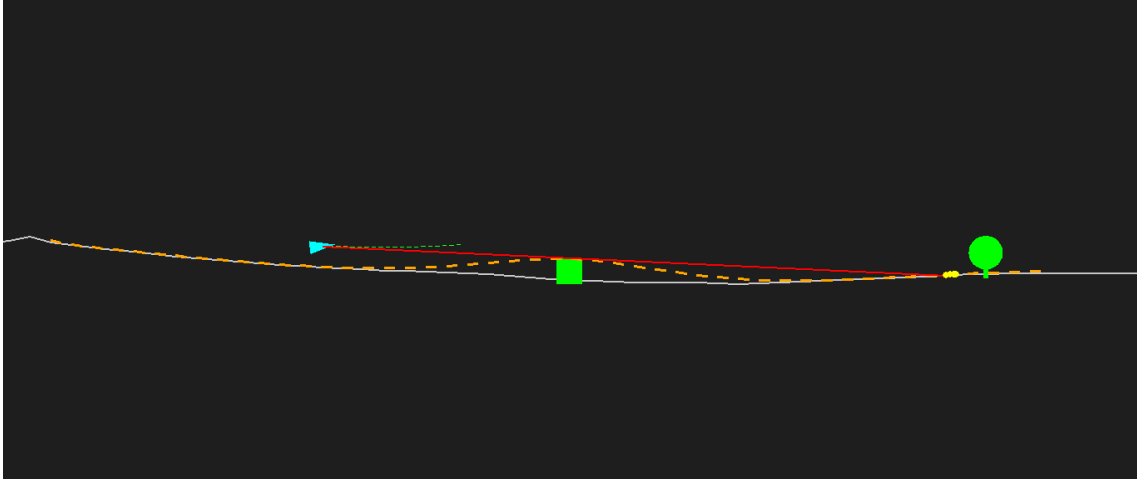


Figure 5.6: Terrain following simulation instance using cubic B-spline DEM-Obstacle envelope.

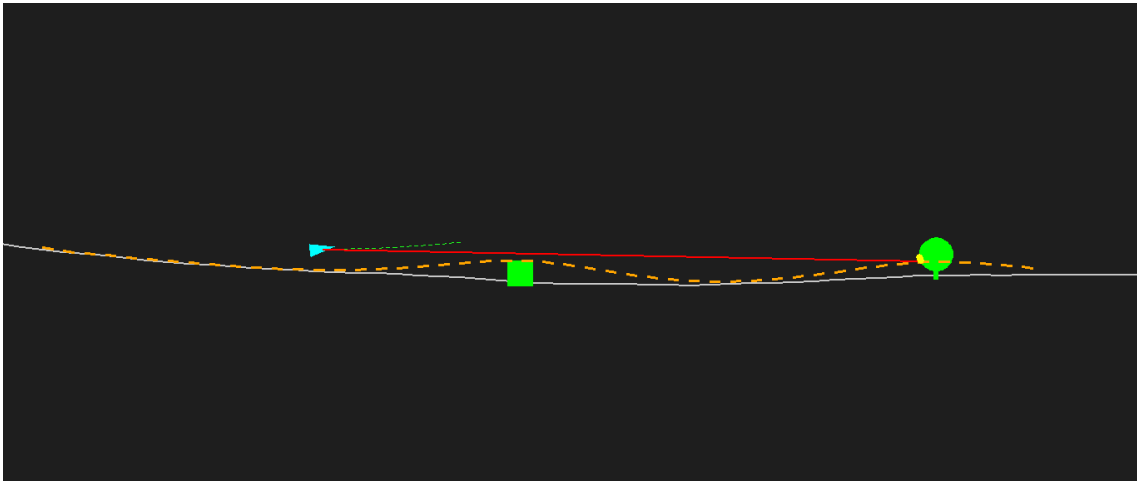


Figure 5.7: Terrain following simulation instance using cubic B-spline DEM-Obstacle envelope, with newer measurements on the tree obstacle.

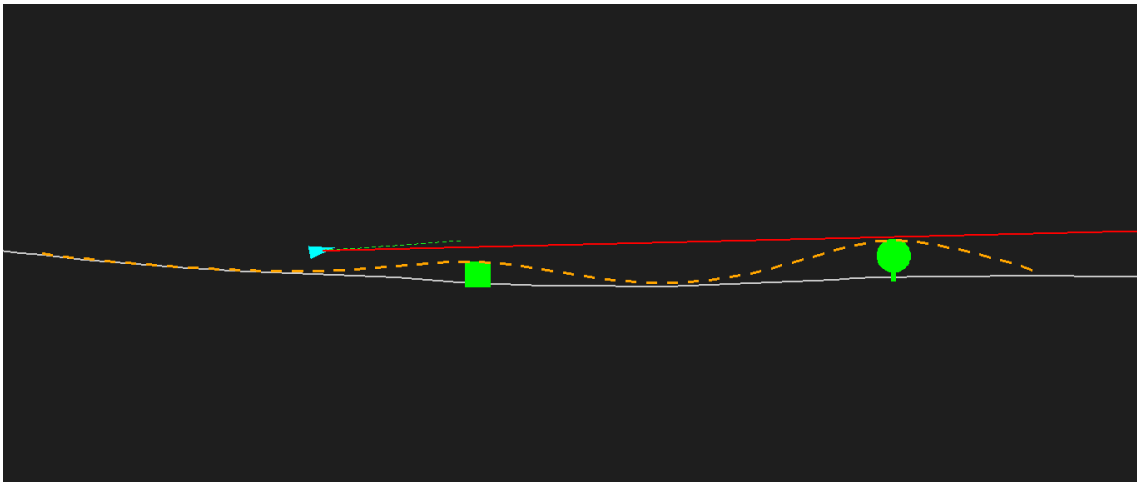


Figure 5.8: Terrain following simulation instance using cubic B-spline DEM-Obstacle envelope, with fitted DEM-Obstacle envelope over the tree obstacle.

The resulting trajectory, available in Figure 5.9, is stable and smooth, able to surpass the obstacles without incurring in NMPC solver failures. Even more remarkable, the envelope elevation tracking is much more stable, keeping within the range h_{ref} . The true AGL on obstacles are also surpassed at h_{ref} , without collision risks due to low altitudes.

The stable terrain following trajectory is also appreciated in the state logs of Figure 5.10, where values are kept around the trim conditions and away from the flight envelope limits.

Summarising, the cubic B-spline DEM-Obstacle envelope provides a safer performance than the stair envelope by adequately tracking and surpassing the obstacles without solver failures. The NMPC is able to be computed with shorter prediction horizon ($H_p = 2$ s) while the convex envelope provides the longer foresight, enabling a real-time performance.

Terrain Following Simulation: UAS Trajectory and AGL Profiles

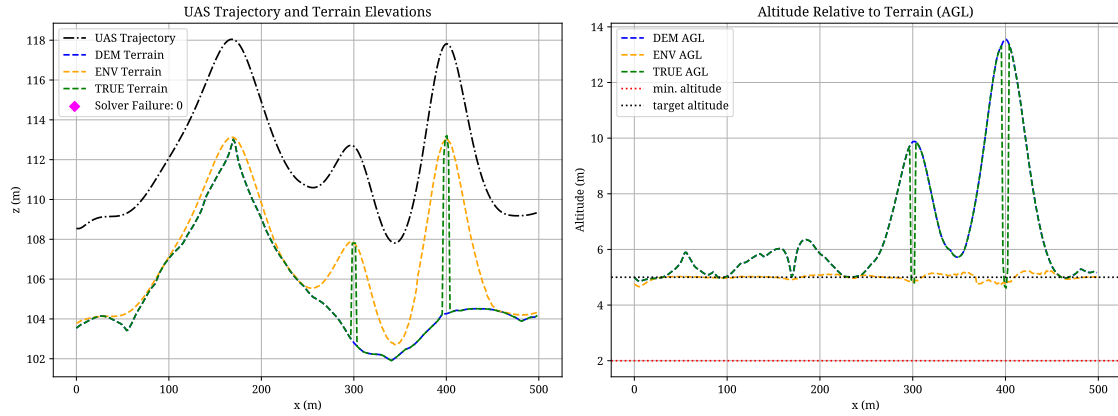


Figure 5.9: Trajectory profiles from the cubic B-spline envelope simulation.

Terrain Following Simulation: State/Control Logs (Solver Failure: 0)

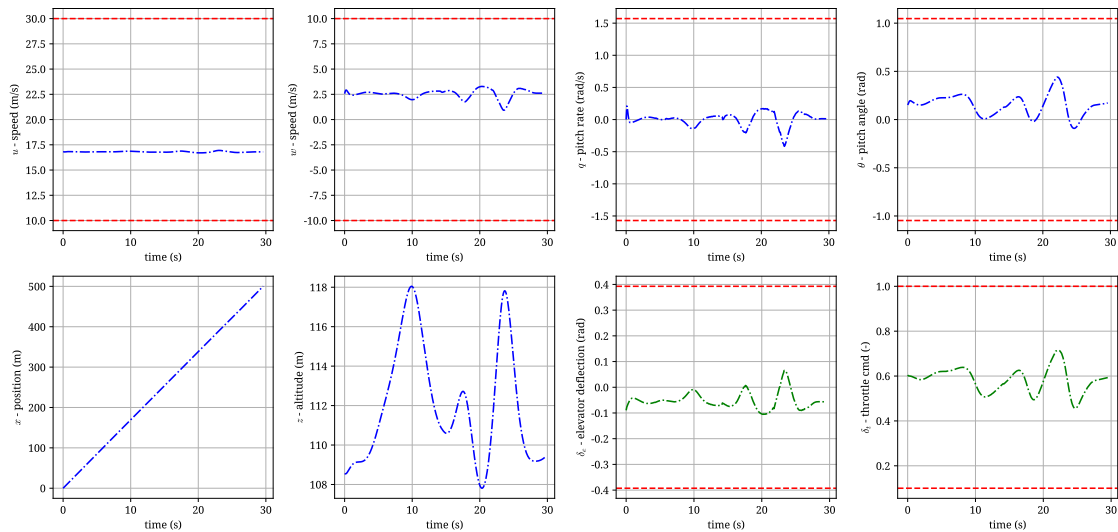


Figure 5.10: State and control logs from the cubic B-spline envelope simulation.

5.2 Sensitivity Analysis on Uncertainty Degradation

This study proposes a dual stage terrain following system, based on a cubic B-spline DEM-Obstacle envelope and a tracking NMPC. The performance of the terrain following system depends on the quality of the data provided. Thereafter, a sensitivity analysis is performed on the system, evaluating the effect of the uncertainty of the DEM data and the navigation and obstacle avoidance sensors.

Each simulation for the sensitivity analysis maintains the same parameters used in the previous cubic B-spline simulation, specified in Tables 5.1, 5.2, 5.3, and 5.4. Singularly, each sensitivity analysis focuses on a specific parameter, performing 5 simulations between a minimum —null— and a maximum uncertainty. The uncertainty values for the sensitivity analysis are provided in Table 5.5. In the study, for lack of information, all noise is assumed to be Gaussian.

Table 5.5: Standard deviations used for the sensitivity analysis.

Uncertainty Parameter	Min. std	Max. std
DEM σ_{dem} (m)	0.0	1.824
Navigation horizontal position σ_{x_s} (m)	0.0	1.422
Navigation vertical position σ_{z_s} (m)	0.0	0.807
Navigation pitch angle σ_{θ} ($^{\circ}$)	0.0	0.013
Sensor distance σ_d (m)	0.0	0.365
Sensor angle σ_{ϵ} ($^{\circ}$)	0.0	1.824

The maximum values considered for the uncertainties are extracted from the Background Chapter 3, from the evaluation of DEM, navigation sensors and obstacle avoidance sensors to perform terrain following. They are the worst values still considered useful for a terrain following UAS. The maximum uncertainty for the DEM corresponds to the RADAR DEM, extracted from Table 3.5 —estimated $\sigma_{dem} = 1.824\text{m}$ from a $\pm 3\text{m}$ at 90% confidence. The maximum navigation uncertainty values are those from a common standard GNSS/INS loosely coupled Kalman filter, as provided in Table 3.6. Lastly, the rangefinder sensor's maximum uncertainties are extracted from the terrain following RADAR of Table 3.9 —considering $\pm 3^{\circ}$ at 90% confidence.

In the presence of uncertainty, the values of the DEM and obstacle measurements are no longer representing the true environment. Although there are values both higher and lower than the nominal, the DEM-Obstacle will be fitted above the taller points that become the active constraints —with neutral risk allowance ($\Delta = 0.5$)—, because it is an upper envelope by definition. This feature enables a low collision risk with uncertainty as the minimum altitudes stay stable, leading to more chances of enemy detection as the maximum values increase. The elevated envelope fitting is appreciated in Figure 5.11.

Because the uncertainty generates different envelopes, a fixed reference is needed for performance evaluation. Using the true ground altitude can be confusing, as there will be noticeable jumps as the UAS pulls up to avoid an obstacle, because the NMPC is tracking the envelope. Alternatively, the method selected is to use the DEM-Obstacle envelope

generated in a no-uncertainty case as the baseline tracking reference to compare all other flight paths. The baseline envelope is the same found in Figure 5.9.

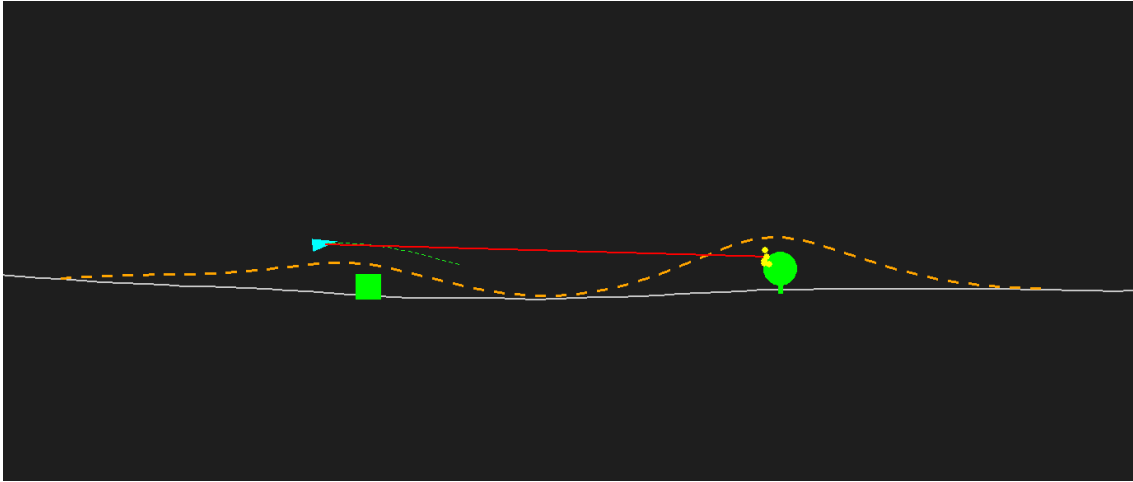


Figure 5.11: Terrain following simulation instance with uncertain DEM and sensors.

The results from the sensitivity analysis are summarised in Table 5.6, showcasing the deviation values at the maximum uncertainty cases. It can fully be appreciated that the most critical parameter that increases the RMSE and maximum deviation is the sensor angle uncertainty —as it was already surmised from the uncertainty propagation expression in Equation 4.2—, followed by the DEM and the navigation vertical positioning. The remaining parameters have deviations beneath 1 m. Horizontal uncertainties do not have important effects on the tracking performance, nor the pitch angle as its standard deviation is quite small. Importantly, the safety of the upper envelope technique is showcased, as the minimum altitude and deviation values remain very close to the baseline. No collision risk is incurred, only detection risk from the higher altitudes over the terrain.

Table 5.6: Uncertainty sensitivity analysis summary on worst case scenarios, providing the minimum above ground level AGL_{min} heights, minimum D_{min} and maximum D_{max} deviations and RMSE to the baseline envelope profile.

Case	Max. std	AGL_{min} (m)	D_{min} (m)	RMSE (m)	D_{max} (m)
Baseline	—	4.611	-0.340	0.105	0.231
DEM	1.824 m	4.599	-0.302	1.374	2.609
Hor. Pos.	1.422 m	4.662	-0.340	0.170	0.520
Vert. Pos.	0.807 m	4.855	-0.340	0.743	1.490
Pitch	0.013 °	4.623	-0.340	0.113	0.260
Distance	0.365 m	4.507	-0.366	0.141	0.361
Angle	1.824 °	4.855	-0.340	3.775	8.784

An individual review of each uncertainty sensitivity analysis case is provided below, showcasing the resulting trajectories and the performance degradation as the levels of uncertainty rise depending on the parameter affected.

5.2.1 DEM Uncertainty

The effect of increasing uncertainty levels on the DEM data used in the terrain following system is appreciated in Figures 5.12 and 5.13. Higher uncertainty levels on the DEM means terrain values can be above and below the nominal. Because the DEM-Obstacle envelope acts as an upper boundary, the envelope is raised over the higher values and the flight level is subsequently raised.

From the flight profile, in Figure 5.12, it can be seen that as the uncertainty increases the UAS trajectory is also elevated, with clear and distinguished flight profiles. The only exception occurs over the obstacles, a region of the DEM-Obstacle envelope that is defined by the rangefinder sensor measurements —unaffected by the DEM uncertainty in these sensitivity simulations. Over other regions, the uncertain DEM values are higher than the direct ground mapping, so the envelope is fitted on them.

Compared to the baseline envelope, the tracking deviations and RMSE are observed in Figure 5.13. There is a noticeable linear increase in the maximum and RMSE values depending directly on the uncertainty. But the collision safety of the upper envelope method is ensured as the minimum deviation remains stable over the baseline value.

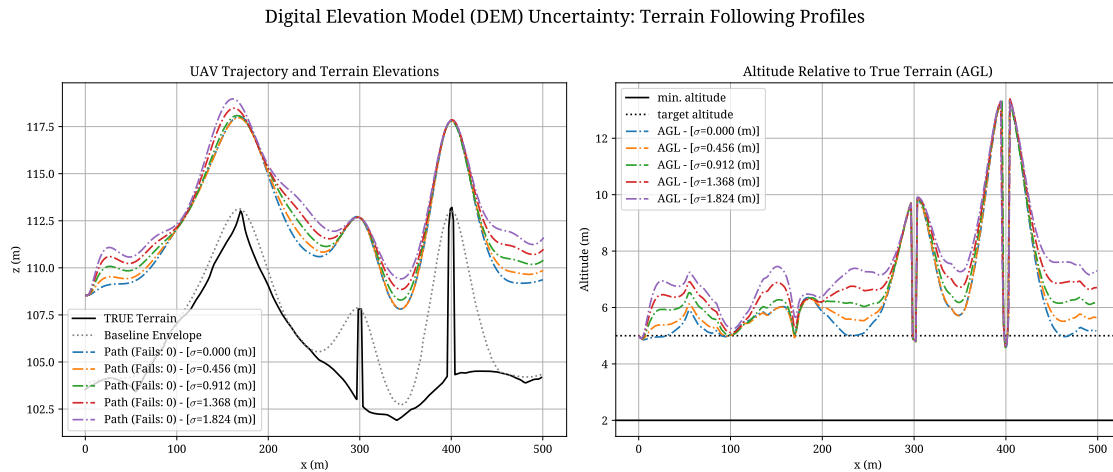


Figure 5.12: Trajectory profiles from the DEM uncertainty sensitivity simulations.

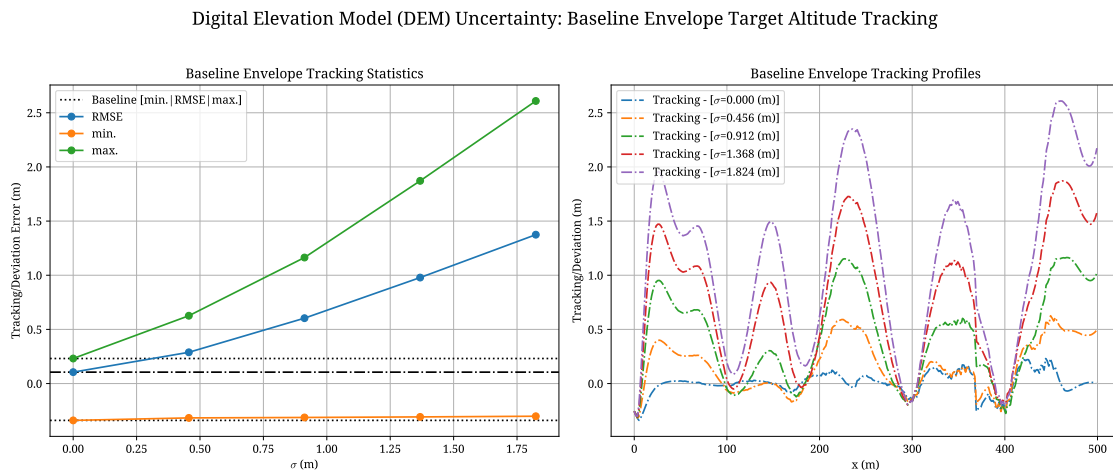


Figure 5.13: Tracking and deviation to the baseline envelope from the DEM uncertainty sensitivity simulations.

5.2.2 Navigation Horizontal Position Uncertainty

Switching over to the navigation sensors, the horizontal position uncertainty is presented in Figures 5.14 and 5.15. An uncertainty in the horizontal position estimation translates into displaced sensor measurements, forward and backwards, that could affect the DEM-Obstacle envelope.

Nevertheless, as seen from the flight profiles in Figure 5.14, the trajectories are almost exactly identical. The horizontal uncertainty does not greatly affect the terrain following performance, although small changes sometimes causes sparse NMPC failures that are compensated by the fallback prediction.

The small differences are only seen when observing the tracking, in Figure 5.15. The maximum deviation and the RMSE to the baseline marginally increase with higher degrees of uncertainty, but the low effect is seen as the RMSE stays at 0.170 m with a much greater $\sigma_{x_s} = 1.422$ m. No collision risks are present, with stable minimum deviations.

Navigation Horizontal Uncertainty: Terrain Following Profiles

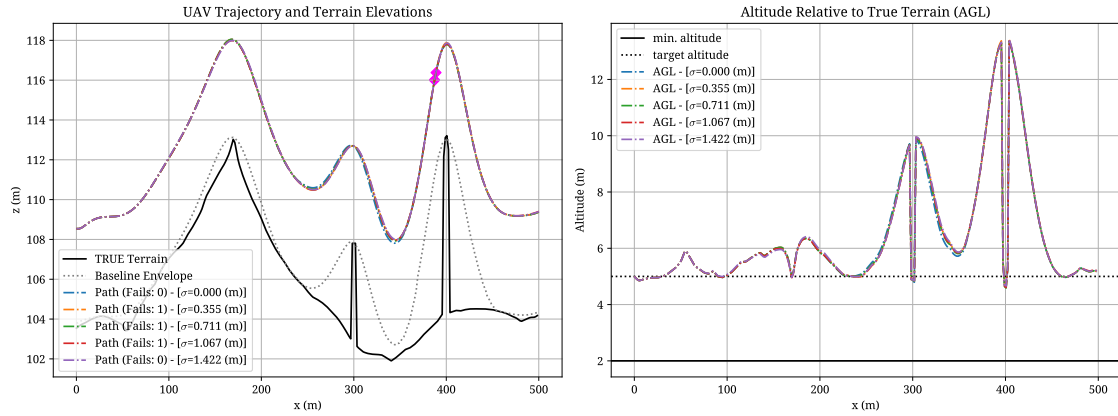


Figure 5.14: Trajectory profiles from the navigation horizontal position uncertainty sensitivity simulations.

Navigation Horizontal Uncertainty: Baseline Envelope Target Altitude Tracking

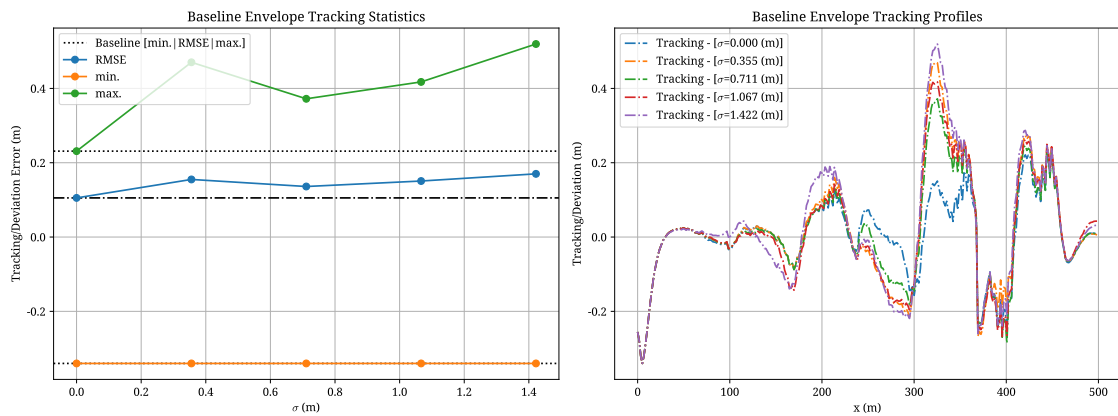


Figure 5.15: Tracking and deviation to the baseline envelope from the navigation horizontal position uncertainty sensitivity simulations.

5.2.3 Navigation Vertical Position Uncertainty

The vertical navigation uncertainty is provided in Figures 5.16 and 5.17. This parameter affects the vertical estimation of the rangefinder and obstacles in direct ground mapping.

Similar to the DEM, the vertical parameter has noticeable effects on the terrain following performance, seen in the flight paths from figure 5.16. The solver failures are very few and sparse, easily compensated through the fallback predictions. Contrary to the DEM uncertainty, this uncertainty affects all regions as the direct ground mapping estimates higher points everywhere and the envelope is fitted over those.

Although the vertical navigation uncertainty $\sigma_{z_s} = 0.807\text{m}$ is smaller than the horizontal $\sigma_{x_s} = 1.422\text{m}$, it has a much larger effect on the terrain following performance, as seen in Figure 5.17. Both RMSE and maximum deviation steadily increase from the baseline values, until reaching a RMSE of 0.743 —similar in magnitude to σ_{z_s} — and a large deviation just shy of 1.5 m. Still, using a RADAR DEM is worse than a standard loosely coupled Kalman filter.

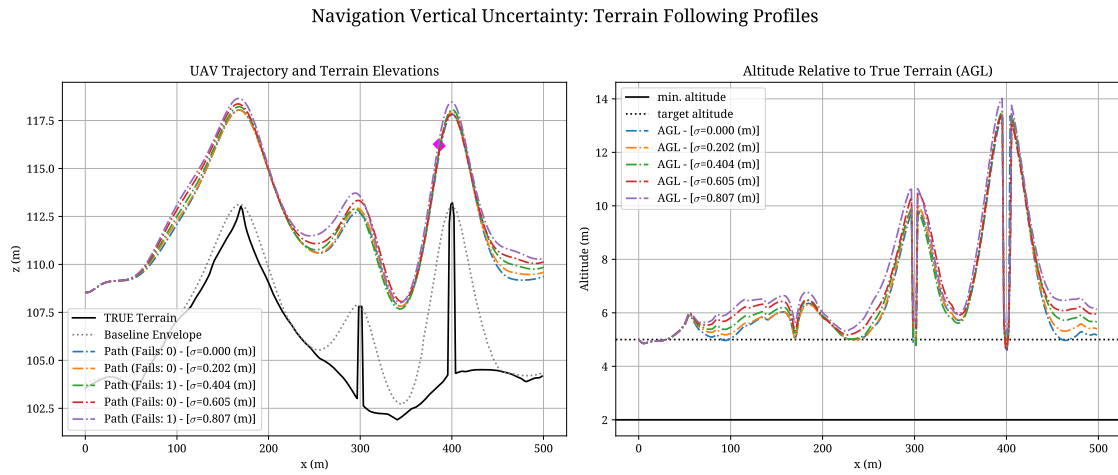


Figure 5.16: Trajectory profiles from the navigation vertical position uncertainty sensitivity simulations.

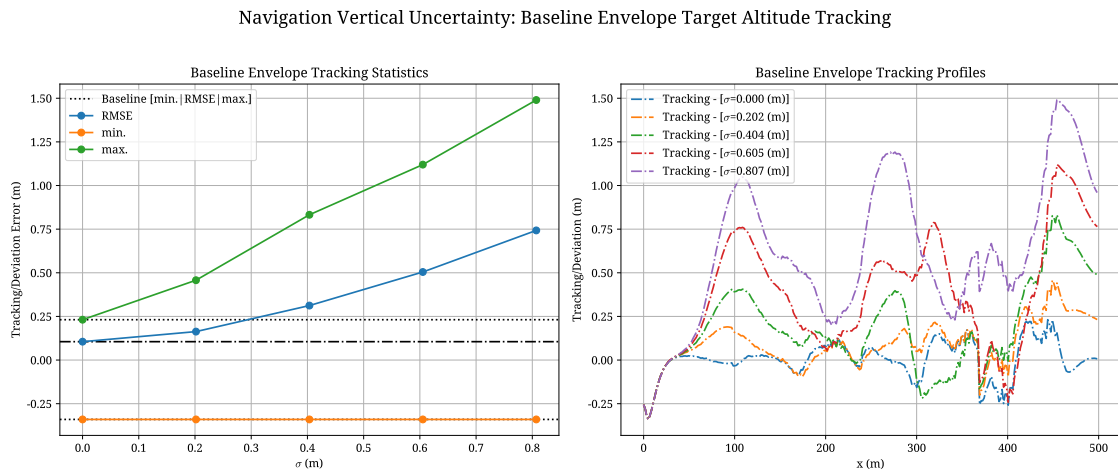


Figure 5.17: Tracking and deviation to the baseline envelope from the navigation vertical position uncertainty sensitivity simulations.

5.2.4 Navigation Pitch Angle Uncertainty

The last navigation parameter is the pitch angle uncertainty, evaluated in Figures 5.18 and 5.19. The pitch angle uncertainty is an angular noise that affects measurements, able to displace their position vertically—a critical effect at distance, from the propagation Equation 4.2—and horizontally.

Upon inspection of the flight profiles, in Figure 5.18, no discernable difference is appreciated. Again, some singular solver failures—due to shifting envelope conditions—are handled by the fallback without issues.

This apparent resilience to the angular uncertainty is due because the maximum pitch uncertainty evaluated comes from a standard loosely coupled Kalman filter, that already provides very good pitch angle accuracy: $\sigma_\theta = 0.013^\circ$. No clear tendency can be observed in Figure 5.19, and variations can be attributed to just random noise values.

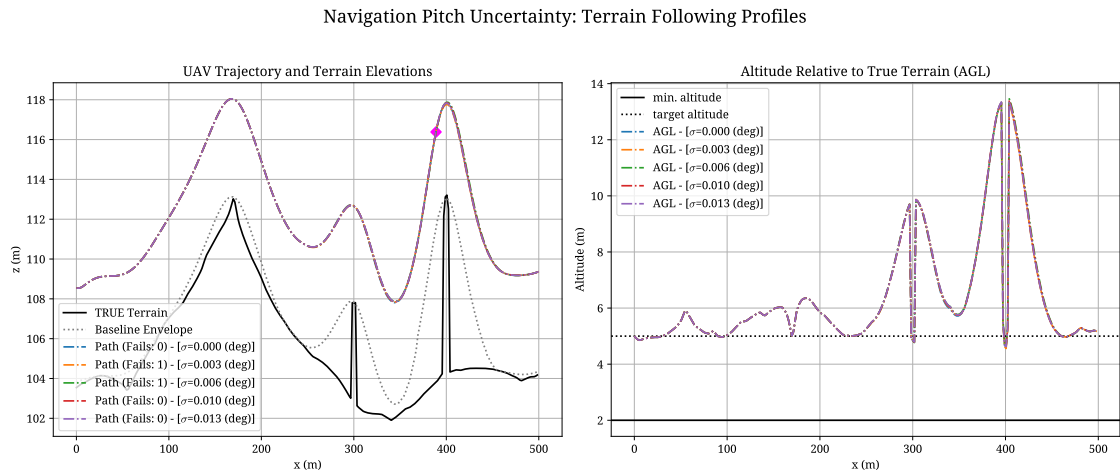


Figure 5.18: Trajectory profiles from the navigation pitch angle uncertainty sensitivity simulations.

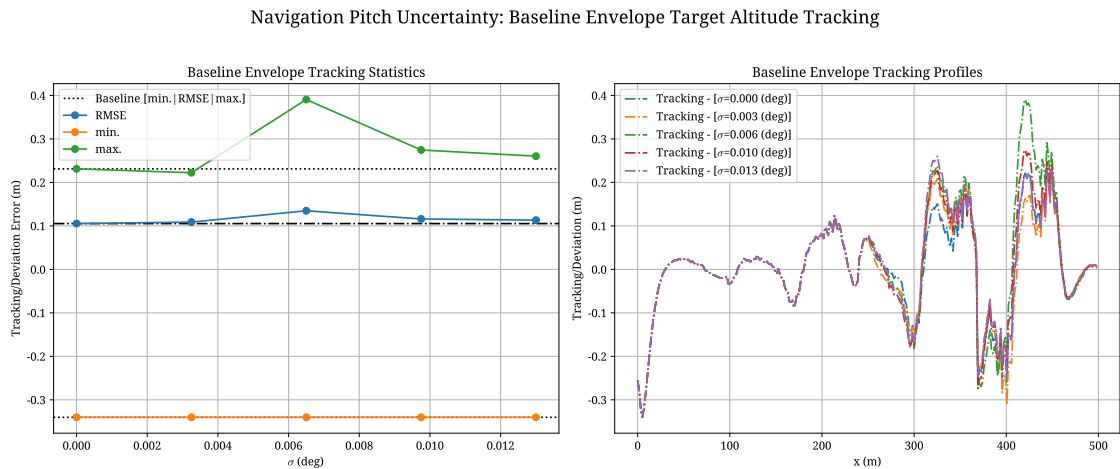


Figure 5.19: Tracking and deviation to the baseline envelope from the navigation pitch angle uncertainty sensitivity simulations.

5.2.5 Sensor Distance Uncertainty

Moving on from the navigation to the rangefinder sensor, the distance uncertainty sensitivity results are shown in Figures 5.20 and 5.21. The distance uncertainty can affect both vertical and horizontal estimation of obstacles. Nevertheless, in a forward looking sensor, the effect is mostly horizontal.

Again, the effect on the flight trajectories is not noticeable with horizontal uncertainties. This is visualised in Figure 5.20, that showcases identical flight paths for all distance uncertainty levels, with manageable solver failure instances.

The evaluation of the tracking results, from Figure 5.21, confirms the low effect of a large distance uncertainty $\sigma_d = 0.365$ m from a RADAR, only resulting in a RMSE of 0.141 m. Coupled with the navigation horizontal position uncertainty, these results allow to conclude that horizontal uncertainties have a low sensitivity on terrain following performance, while direct vertical uncertainties have a clear, quasi linear, correlation.

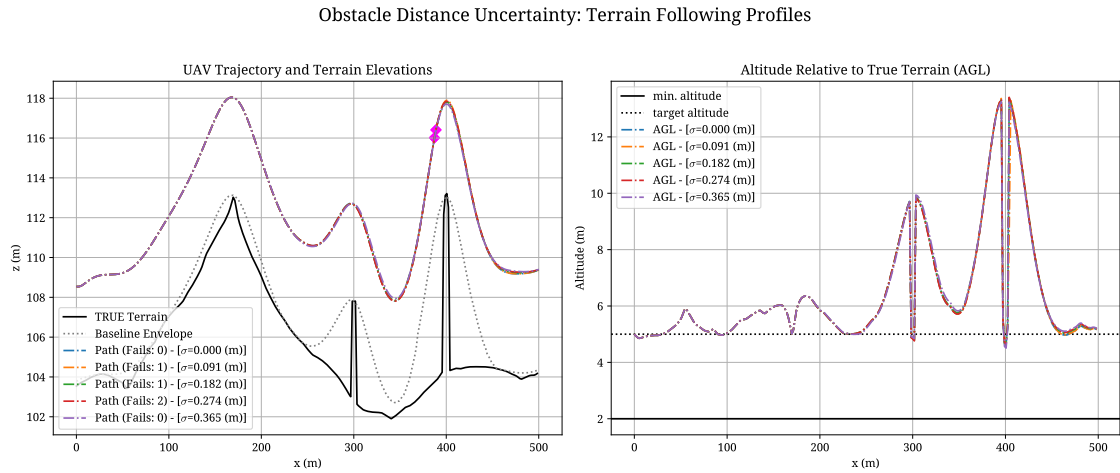


Figure 5.20: Trajectory profiles from the rangefinder sensor distance uncertainty sensitivity simulations.

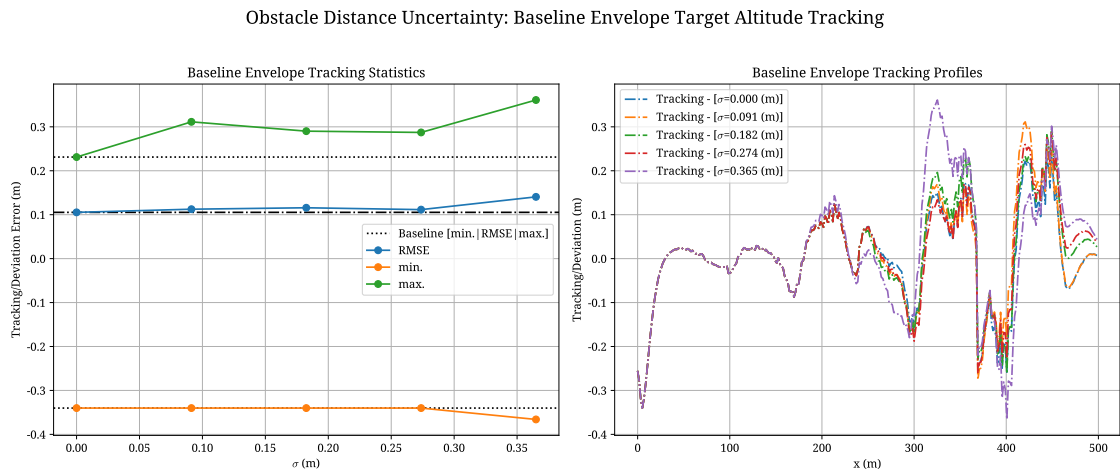


Figure 5.21: Tracking and deviation to the baseline envelope from the rangefinder sensor distance uncertainty sensitivity simulations.

5.2.6 Sensor Angle Uncertainty

The last parameter evaluated in the sensitivity analysis is the rangefinder sensor angle uncertainty, presented in Figures 5.22 and 5.23. The angular noise in the sensor, through quadratic distance propagation, produces a very large vertical obstacle uncertainty.

The flight paths of Figure 5.22 demonstrate the large effect of the growing angular uncertainty on the terrain following trajectories, with ever higher paths affecting all regions mapped by the sensor. The increase in altitude is such that a growing number of NMPC failures appear—up to 12—, although the fallback prediction handles them.

The highest sensor angular uncertainty is that of a RADAR sensor $\sigma_\epsilon = 1.824^\circ$. Reaching this maximum value, the RMSE increases to 3.775 m, with a maximum deviation peak of almost 9 m. Even under such uncertainty values, the terrain following system maintains a stable minimum deviation and remains free from collision risk, as seen in Figure 5.23. The sensor angular uncertainty is confirmed as the most critical parameter in the sensitivity analysis, forcing the highest flights prone to detection and deteriorating the NMPC solver performance, although safe from crashes with the ground.

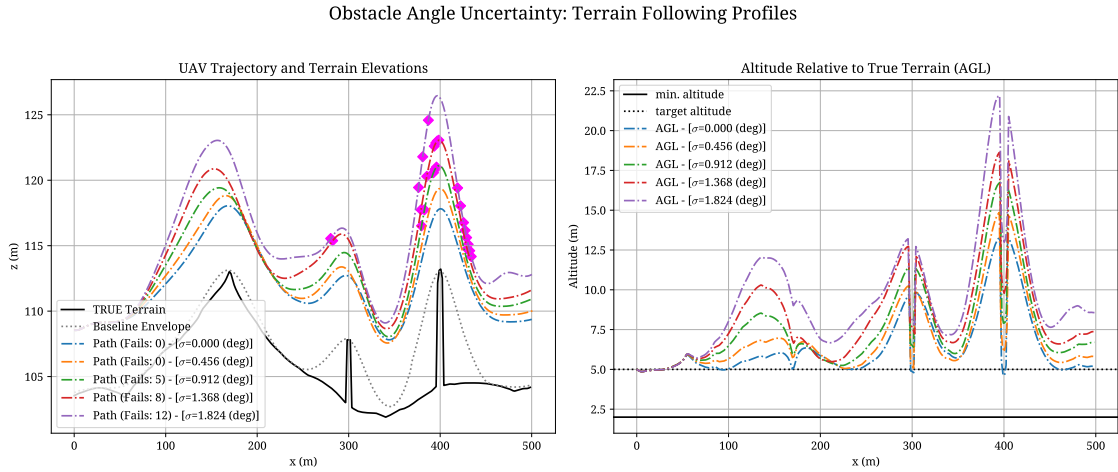


Figure 5.22: Trajectory profiles from the rangefinder sensor angle uncertainty sensitivity simulations.

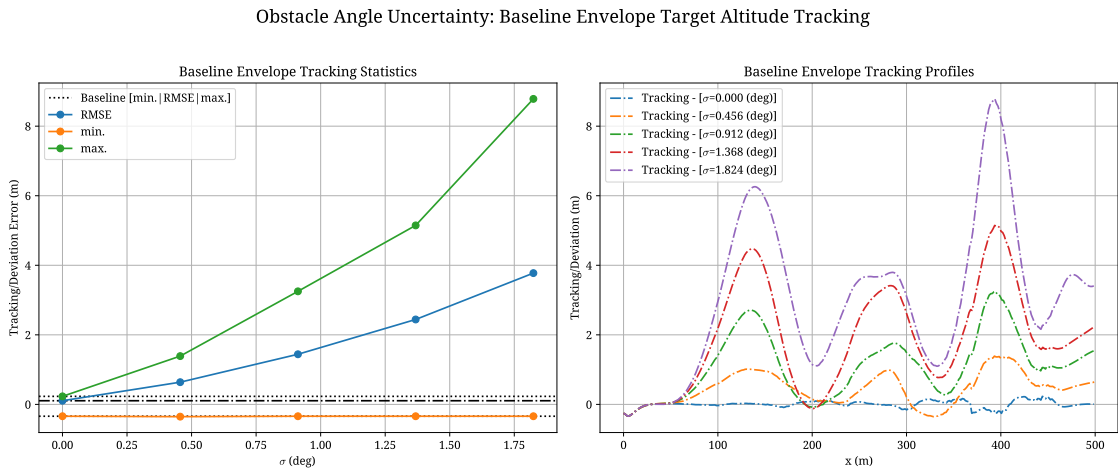


Figure 5.23: Tracking and deviation to the baseline envelope from the rangefinder sensor angle uncertainty sensitivity simulations.

5.3 Risk Allowance Effect on Terrain Following

The presence of uncertainty on the DEM and the sensors can greatly affect the performance of the terrain following system. The performance has a low sensitivity to horizontal uncertainty parameters, a direct correlation to vertical components and a critical effect from angular uncertainty —propagated to a large obstacle vertical uncertainty through forward-looking distance.

The flight paths from all the sensitivity simulations indicate that the uncertainty increases the maximum deviation and RMSE, while keeping a stable minimum deviation. This collision-free risk provided by the upper DEM-Obstacle envelope with uncertainties allows the use of risk allowance through the chance constraints of the envelope —risk-awareness formulated in Section 4.3. Evaluating the risk-to-performance trade-off can help to identify the levels of risk acceptable to improve the terrain following flight levels, fitting the envelope below highly uncertain points, without increasing the chances of a collision with the ground.

The chance constraints are applied to the two data sources used for the envelope fitting: DEM and obstacle measurements from the direct ground mapping. The two simulations evaluate a range of risk allowances between the neutral 50% and 99%, while using the maximum worst-case uncertainties selected for the sensitivity analysis. All other simulation parameters have been specified in Tables 5.1, 5.2, 5.3, and 5.4. The risk allowance simulations' specific parameters are presented in Table 5.7, summarising the results in Table 5.8. The detailed evaluation is found in the subsections below.

Table 5.7: Risk allowance terrain following simulation parameters.

Risk Allowance		Min. Risk $\Delta_{\min}$	Max. Risk $\Delta_{\max}$
DEM	$\sigma_{dem} = 1.824 \text{ m}$	$\Delta_{dem} = 0.50$	$\Delta_{dem} = 0.99$
Obstacle	$\sigma_{x_s} = 1.422 \text{ m}$	$\Delta_o = 0.50$	$\Delta_o = 0.99$
	$\sigma_{z_s} = 0.807 \text{ m}$		
	$\sigma_{\theta} = 0.013^\circ$		
	$\sigma_d = 0.365 \text{ m}$		
	$\sigma_{\epsilon} = 1.824^\circ$		

Table 5.8: Risk allowance summary, providing the minimum above ground level $AGL_{\min}$ heights, minimum $D_{\min}$ and maximum $D_{\max}$ deviations and RMSE to the baseline.

Case	Risk Δ	$AGL_{\min}$ (m)	$D_{\min}$ (m)	RMSE (m)	$D_{\max}$ (m)
Baseline	—	4.611	-0.340	0.105	0.231
DEM	0.50	4.599	-0.302	1.374	2.609
DEM	0.99	4.635	-0.333	0.232	0.739
Obstacle	0.50	4.855	-0.340	3.535	7.082
Obstacle	0.87	4.855	-0.340	1.150	3.048
Obstacle	0.99	2.921	-1.903	0.599	0.234

5.3.1 DEM Risk Allowance

The DEM risk allowance simulations are presented in Figures 5.24 and 5.25. Increasing the risk allowance Δ_{dem} enables the DEM-Obstacle envelope to be fitted below the uncertain DEM points. Because it is an upper envelope, it is fitted beneath uncertain points higher than the true ground, so the envelope provides a better representation of the terrain.

Contrary to the DEM uncertainty sensitivity analysis, the flight paths from Figure 5.24 show an improvement as the risk allowance is increased. The lower trajectories are still safe, as ground and objects are avoided around the target altitude, never dipping close to the minimum altitude allowed.

The clear improvement is seen over the tracking analysis of Figure 5.25. With increasing risk allowance, both maximum deviation and RMSE decrease, while the minimum deviation is kept stable and collision-free. At 99% risk the envelope is fitted beneath almost all DEM points, so the upper envelope is mostly fitted to clean rangefinder obstacle points. Because the performance does not reach the baseline values, it can be concluded that the rangefinder sensor can compensate a highly uncertain DEM, but the best performance is achieved with both quality DEM and sensor measurements.

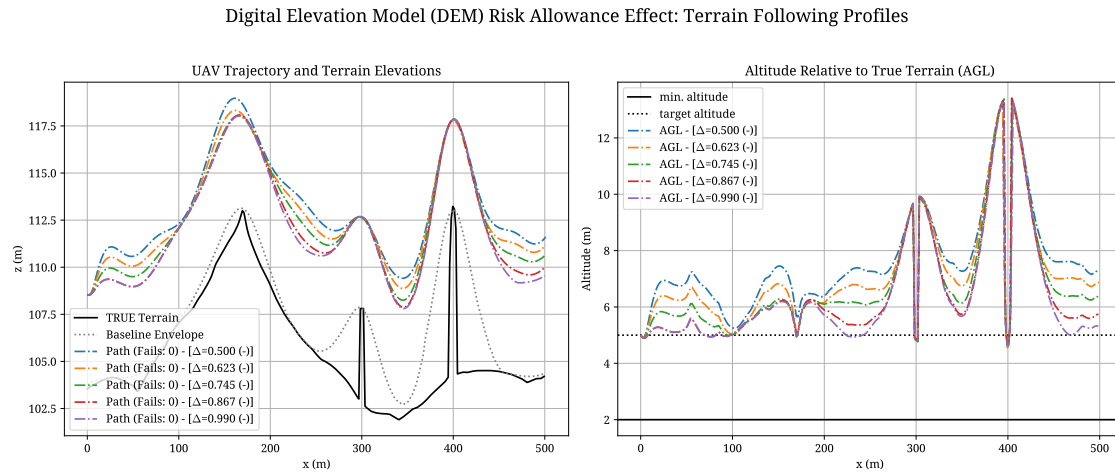


Figure 5.24: Trajectory profiles from the DEM risk allowance simulations.

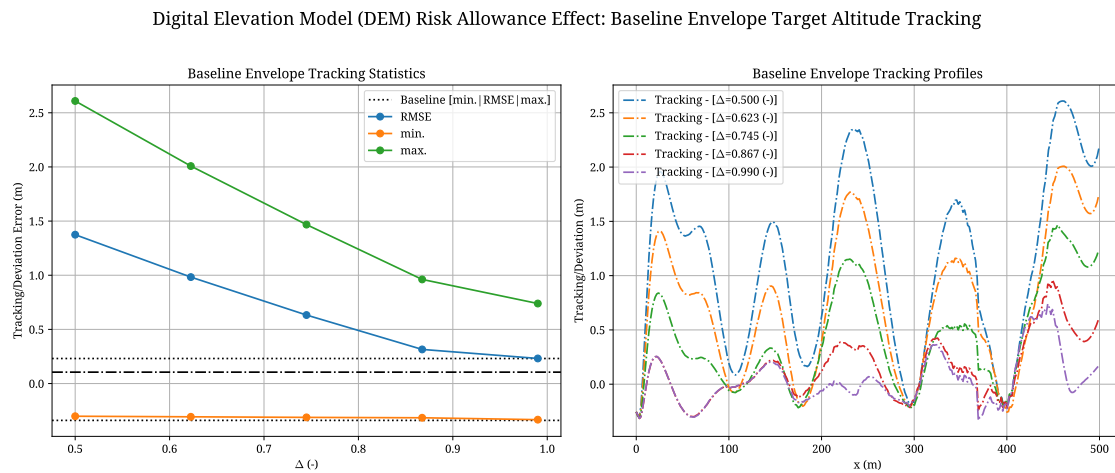


Figure 5.25: Tracking and deviation to the baseline envelope from the DEM risk allowance simulations.

5.3.2 Obstacle Measurements Risk Allowance

The obstacle measurement risk allowance effect on terrain following is shown in Figures 5.26 and 5.27. In the same manner, the risk allowance Δ_o relaxes the constraints on the DEM-Obstacle envelope to be fitted beneath—with high risk—the uncertain obstacle measurement points.

The evaluation of the flight paths, from Figure 5.26, allows to see a visual reduction in the flight altitudes over the terrain. Still, because of the high angular uncertainty and vertical error there are NMPC solver failures, handled by the fallback. The AGL heights lower close to the minimum altitude over the obstacles at 99% risk.

The improvement in performance is seen on the tracking statistics of Figure 5.27. Until 87% risk, the minimum deviation is stable while the maximum deviation is reduced by 57% and the RMSE by 67%. Further increasing the risk, the minimum deviation already lowers almost up to -2 m, causing potential crashes and collisions. This happens over obstacles, as the envelope is fitted too low, and the DEM has no knowledge to compensate. It can be concluded that the sensor is essential for terrain following, and only moderate risk is allowed to still be able to safely overcome obstacles not shown in the DEM.

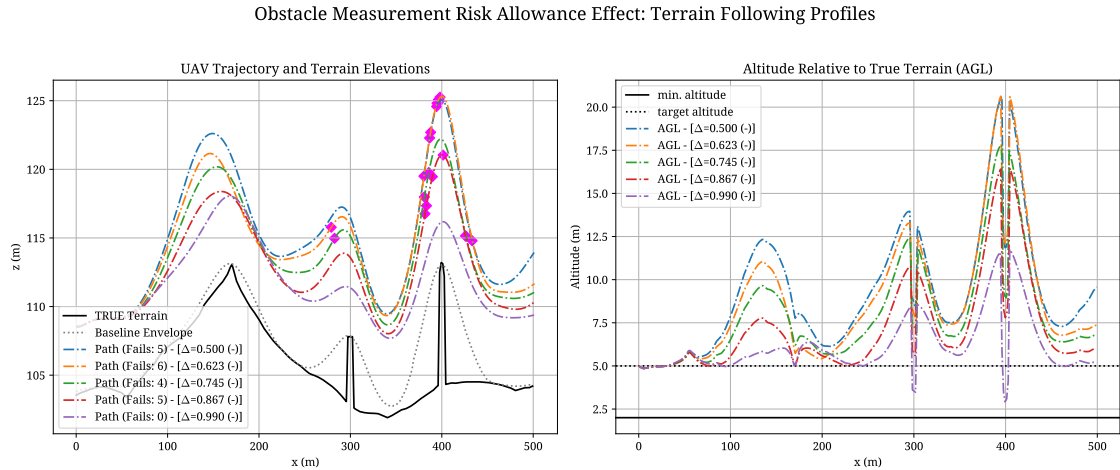


Figure 5.26: Trajectory profiles from the obstacle measurement risk allowance simulations.

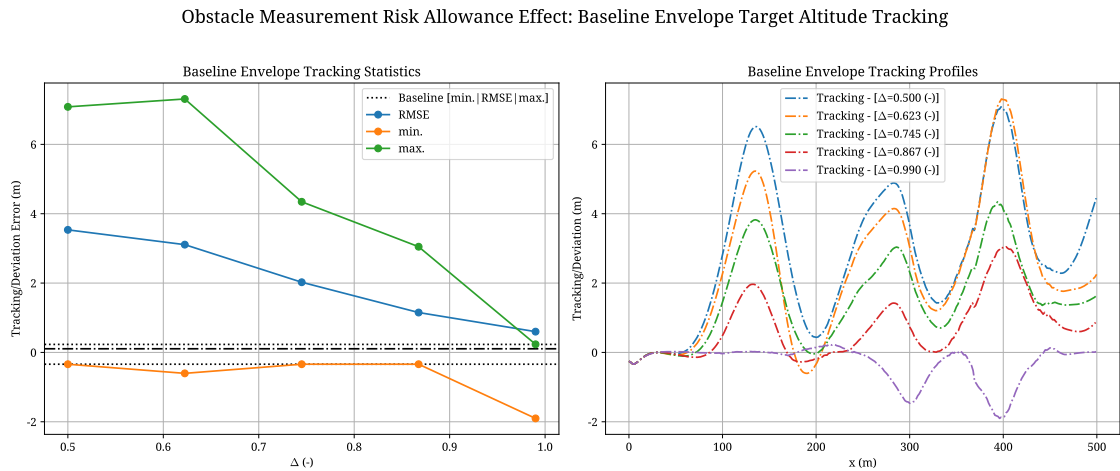


Figure 5.27: Tracking and deviation to the baseline envelope from the obstacle measurement risk allowance simulations.

Chapter 6

Evaluation

6.1 Capabilities of the Terrain Following System

The dual-stage terrain following system proposed in this thesis provides a TF capability specifically designed to tailor real-time performance and combination of digital elevation models (DEM) and direct ground mapping with onboard sensors. The system's capabilities and advantages also have limitations and points of further research and study to improve and complete. The main system's capabilities are:

- vertical path-planning, guidance and control of a fixed-wing UAS in NOE flights.
- first stage: risk-aware combination of DEM and direct ground mapping through a convex cubic B-spline envelope in a moving window, providing obstacle avoidance.
- second stage: optimal terrain following tracking prediction for the envelope, with altitude range and dynamic constraints, through a non-convex NMPC.

The system advantages are:

- real-time computational efficiency, with a long convex DEM-Obstacle envelope foresight window and a smaller NMPC prediction horizon.
- smooth and stable optimal control prediction, with high NMPC convergence — employing a warm-start and fallback procedure— rates improved by the cubic B-spline envelope reference trajectory with aggressiveness modulation.
- safe tracking and obstacle avoidance over the terrain at the desired height, with the upper DEM-Obstacle envelope guaranteeing clearance over all measured obstacles, using chance constraints under uncertainty.
- demonstrated simulated performance, in an Intel N200 low-cost CPU, with a virtual terrain model and a single forward-looking rangefinder sensor.

The sensitivity of the terrain following performance to uncertainty on data and sensors:

- increasing levels of uncertainty degrade terrain following performance by increasing flight altitudes and making the UAS visible to enemy detection, but the upper envelope active constraints maintain stable minimum heights to avoid collisions.

- low sensitivity to horizontal parameters, such as navigation horizontal position and forward rangefinder distance uncertainty.
- direct affectation by the vertical parameters, such as the DEM uncertainty and the navigation vertical position of the UAS, increasing the maximum deviation and RMSE to the baseline envelope.
- critical sensitivity to angular dispersion, from the rangefinder, because it propagates to a large vertical obstacle uncertainty through the distance, increasing altitudes.

The risk tolerance and its effect on the terrain following system performance:

- the best performance is only achieved through quality and non-uncertain DEM and sensors, a baseline performance used for benchmark providing ± 0.34 m of deviation.
- in the presence of a highly uncertain DEM, a quality rangefinder can provide acceptable performance, still worse than the baseline but under ± 1 m of deviation.
- the rangefinder sensor is essential to the system, and only moderate risk allowance ($< 90\%$) can be applied under uncertainty to guarantee stable minimum deviations to avoid collisions from obstacles not represented in the DEM.

The terrain following system is currently developed to perform strict Terrain Following (TF), a vertical avoidance problem in 2D. Nevertheless, the system has the capability to be expanded to provide also the lateral Terrain Avoidance in 3D. This can be achieved through an update from cubic B-spline curves to surfaces —still maintaining their convex OCP fitting properties— and incorporating the lateral dynamics to the NMPC. The increased complexity would most probably require more powerful CPU processors to solve the problem for real-time implementation, and obstacle avoidance sensors would need to have a certain degree of scanning or panoramic view to adequately measure the 3D environment.

Leaving out the 3D problem, the main limitation of the system is its inability to provide end-to-end path planning between the start location and the destination of the UAS. This capability is critical to adequately plan safe trajectories over mountainous regions, where the UAS could fall in a large inevitable collision state because of a gorge or tall ridge. Because of the real-time focus of the developed terrain following system, both the DEM-Obstacle envelope and tracking NMPC use a moving window prediction that can not fit large terrain features, at geographical scale.

A possible further research scope would be the combination of this thesis dual-stage terrain following system with a path-planning method. Safe path planning methods — among those provided in the Literature Review Chapter 2— can be employed to provide a precomputed reference horizontal trajectory, while the developed dual-stage terrain following system performs the real-time guidance and vertical avoidance execution.

Lastly, the terrain following flight will always be a trade-off between risk and performance. This risk-aware method has implemented overall risk allowances on the DEM points and the obstacle measurements for the generation of the cubic B-spline envelope. Because of the continuous trajectories, some constraints are inactive while others critically define the upper envelope. Future research could focus on risk allocation strategies, such as iterative risk allocation, to shift risk between inactive and active constraints and improve the performance while maintaining safe minimum altitude margins.

6.2 Sensor Suite Selection for a Terrain Following UAS

The results and analysis performed on the terrain following system proposed provide insight to select an adequate sensor suite for a tier 1 UAS, low-cost and lightweight. Specially, from the sensitivity analysis, several conclusions can be extracted.

Starting, for a DEM dataset, whose uncertainty has a direct effect on the terrain following performance, it is recommended to obtain the DEM with the lowest possible uncertainty. Among the models available, LiDAR DEMs have an edge over RADAR DEMs, because they are orders of magnitude more precise. If flying in a region available, the 3DEP LiDAR DEM from the USGS can provide a centimetre level accuracy instead of the RADAR DEM metre level accuracy, as seen from Table 3.5. The models provided by the USGS and other geographical departments are available to the public, so no direct cost is incurred other than processing and design costs.

For navigation sensors, the vertical position uncertainty is very important, and also keeping a minimal pitch uncertainty. An improvement over the standard loosely coupled Kalman filter would be desired to improve the vertical accuracy. This can be achieved through a post-processing smoothing of the filter, increasing the system's precision to centimetre level, as visualised in Table 3.6. Because it is based on a loosely coupled Kalman filter, the method can be implemented with any IMU sensor and GNSS receiver, expanding the range of possibilities for use.

Designed specifically for UAS, the HerePro GNSS receiver [58] is a commercial-grade sensor with acceptable performance, and has a direct integration with the Cubepilot Orange flight controller [59] that has triple redundant integrated ICM 45686 IMUs [60]. Although these components do not have the precision of military-grade devices, and might have some performance loss, they provide a low-cost alternative for expendable tier 1 UAS.

Lastly, from the sensitivity analysis it is clear that the rangefinder sensor angular uncertainty is the most critical parameter. Between a RADAR and a LiDAR sensor, the difference in angular precision is noticeable, as seen in Table 3.9. Although a RADAR could provide complete all-weather capability, the trade-off in performance is too steep. Thereafter, a LiDAR rangefinder is recommended to perform terrain following using the developed system.

There is an emerging market for terrain following LiDARs for small UAS. Among the options, the Lightware SF30/D [61] provides a proclaimed measuring range of 200 m with ± 5 cm accuracy and a beam divergence beneath $< 0.2^\circ$ ($\pm 0.1^\circ$). Remarkably, it boasts high sensor rates between 49 Hz and 20 kHz. This low-cost sensor is also exceptionally lightweight and compact, ideal for tier 1 UAS.

Table 6.1 provides a summary of the possible sensor suite discussed, with components—shown in Figure 6.1—chosen according to the sensitivity analysis performed. The prices are referenced in British pounds (£), without VAT, at the time of writing the thesis.

From the components selected, a new set of uncertainties can be evaluated in our terrain following simulations to understand the performance levels to be achieved. If not specified, the statistical distribution is assumed to be Gaussian and confidences of 90%—considering $X \approx z_{0.95} \sigma$. The sensor suite uncertainties are presented in Table 6.2.

Based on the comparison between the commercial GNSS receiver and IMU sensor and the military-grade UIMU-LCI sensor [33], a certain degradation is applied to the GNSS/IMU post-processing smoothing uncertainty values. The Cubepilot accelerometer

has more sensitivity ($\pm 4000^\circ/s$) than the UIMU-LCI ($\pm 800^\circ/s$). But the HerePro GNSS has a single-point accuracy (± 1.5 m) 25% worse than the UIMU-LCI (± 1.2 m) so, heuristically, the same degradation is applied to the uncertainties from the GNSS/INS loosely coupled Kalman filter with post-processing smoothing technique of Table 3.6.

Table 6.1: Possible commercial sensor suite for a terrain following tier 1 UAS.

Component	Model	Price	Weight	Dimensions
DEM	USGS 3DEP	N/A	N/A	N/A
GNSS	HerePro	£665	101 g	15×78 mm
IMU	Cube Orange (ICM-45686)	£312	73 g	$38.4 \times 38.4 \times 22$ mm
Rangefinder	Lightware SF30/D	£295	31.53 g	$30 \times 56 \times 50$ mm

Table 6.2: Standard deviations estimated from the commercial sensor suite of Table 6.1.

Uncertainty Parameter	Standard Deviation
DEM σ_{dem} (m)	0.095
Navigation horizontal position σ_{x_s} (m)	0.025
Navigation vertical position σ_{z_s} (m)	0.014
Navigation pitch angle σ_θ ($^\circ$)	0.006
Sensor distance σ_d (m)	0.030
Sensor angle σ_ϵ ($^\circ$)	0.061



Figure 6.1: Components of the commercial sensor suite for a terrain following tier 1 UAS.

6.2.1 Terrain Following Simulation of Tier 1 UAS

The use of the proposed sensor suite of Table 6.1, with the uncertainty parameters shown in Table 6.2, provides an exceptional terrain following performance. The generated envelope and flight path is showcased in Figure 6.2. The orange cubic B-spline envelope is fitted above the terrain features captured by the precise DEM—small uncertainties can be seen in the blue elevation—and the obstacles measured through the navigation and rangefinder sensors. Because of the constant and smooth trajectories, no NMPC solver failures are registered. The system is able to properly avoid the obstacles, overcoming their apexes at the target altitude, and maintaining a close envelope tracking at the h_{ref} height.

The simulation has been undertaken with neutral risk ($\Delta_{dem} = \Delta_o = 0.5$), achieving performance degradations to the baseline case beneath 2% in the RMSE, as seen from the results in Table 6.3. With an adequate sensor suite, there is no need to increase the risk allowance, as the centimetre level uncertainties provide a remarkable performance.

As stated before, this simulation implements a vertical 2D terrain following system. Real-world sensing and DEM datasets can have greater uncertainties, with overconfident datasheet values. The use of LiDARs presents challenges in 3D environments, requiring higher processing requirements to evaluate the complex obstacles, dealing with glares or reflections at high angles of incidence—critical for a forward-looking sensor. Nevertheless, it is the recommended option to achieve the best performance, not possible through the poor angular resolution of a RADAR rangefinder.

Table 6.3: Tier 1 UAS with commercial sensor suite performance summary, providing the minimum above ground level AGL_{min} heights, minimum D_{min} and maximum D_{max} deviations and RMSE to the baseline.

Case	Risk Δ	AGL_{min} (m)	D_{min} (m)	RMSE (m)	D_{max} (m)
Baseline	—	4.611	-0.340	0.105	0.231
Tier 1 UAS	0.50	4.610	-0.335	0.107	0.232
Relative (%)	—	-0.022%	-1.641%	1.753%	0.294%

Terrain Following Simulation: UAS Trajectory and AGL Profiles

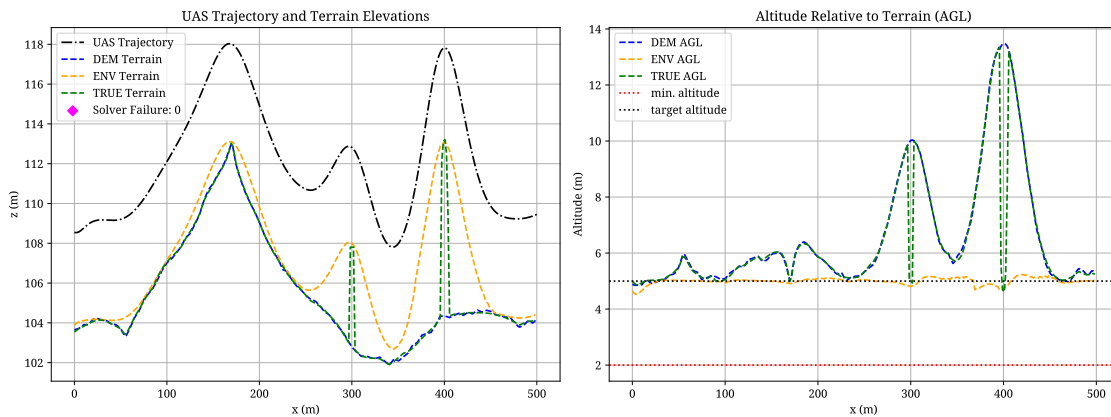


Figure 6.2: Trajectory profiles from the tier 1 UAS with a commercial sensor suite.

Chapter 7

Conclusions

The terrain following capability allows aircrafts to perform low-level altitude flights, adapting to the contour of the ground whilst avoiding obstacles. Terrain following techniques and vehicles were developed during the last decades of the XXth century as means of avoiding detection by enemy surveillance radars. Through the first decades of the XXIst century, stealth and furtive aircraft were developed as an alternative, but at greater cost. The exponential increase of smaller UAS since the last decade has renewed the interest in the terrain following capability, providing these expendable systems with a low-cost solution for incursion and avoiding enemy detection.

Studies on terrain following for fixed-wing UAS are mostly centred on geometrical and optimal control methods for path planning. These techniques rely either on sensor data, or digital elevation models (DEM), exclusively. The reliance only on precomputed paths is a clear risk when aiming for a very low-level flight, also known as nap-of-the-earth (NOE), because DEMs have incomplete obstacle knowledge and inaccuracies. Nevertheless, there is a research gap in methods to study the combination of DEM and sensor measurements for terrain following.

The need for combining DEM datasets with real-time direct ground mapping through onboard sensors to generate a complete obstacle map for terrain following is addressed in this thesis for the first time. The proposed dual-stage vertical terrain following system employs a geometric cubic B-spline curve fitted through a convex OCP to provide a DEM-Obstacle upper envelope for path-planning, and a tracking NMPC solving a non-convex OCP with altitude range constraints. Both stages are applied to a moving window, designed for real-time performance.

The cubic B-spline DEM-Obstacle envelope guarantees the computation of an upper envelope over the DEM points and the obstacle measurements, evolving as more information is acquired. Through aggressiveness modulation, the envelope provides a better tracking reference for the NMPC, safely overcoming obstacles and allowing for a shorter 2 s prediction horizon. For the rare occasions that the NMPC fails to converge, a fallback solution is implemented to keep the system running and avoid stalling.

A sensitivity analysis is conducted over DEM, navigation and obstacle avoidance sensor uncertainty parameters, selecting worst case scenarios from a detailed background study. The effect is compared to a baseline scenario, without uncertainty. Rising levels of noise increase the flight altitude, with chances of enemy detection, but maintain stable minimum deviations ensuring a collision-free path. This effect is per-design, as the DEM-Obstacle

envelope is actively constrained by the highest uncertain points, while lower points serve as inactive constraints.

Not all parameters are equally critical. The terrain following system has a low sensitivity to horizontal uncertainties, such as navigation position or sensor distance. There is a direct correlation to vertical uncertainties, from the DEM dataset and the navigation estimation, increasing the RMSE and maximum deviation of the flight paths. Lastly, the angular uncertainties are the most important parameter, causing large vertical obstacle uncertainties through distance propagation. This effect can produce deviations of almost 9 m if using a RADAR rangefinder with poor angular resolution. Thankfully, the pitch angle accuracy achieved from the GNSS/INS filters is high enough that the degradation is negligible.

Through the risk-aware chance constraint formulation of the DEM-Obstacle envelope, risk allowances can be implemented to the DEM and obstacle measurement points. Increasing risk enables the envelope to be fitted beneath highly uncertain points to improve tracking, as a trade-off. Under the uncertain scenarios evaluated in the sensitivity analysis, the risk tolerance of the system is studied. The terrain following system can implement high risk allowance to an uncertain DEM, achieving deviations within ± 1 m if backed by accurate obstacle avoidance sensors. Contrary to it, sensors are essential to the system and only moderate risks can be implemented to avoid lowering the minimum altitudes, because there is no DEM knowledge backup for certain obstacles.

The best performance is achieved with both quality and accurate DEM datasets and onboard sensors. The large sensitivity on angular uncertainty indicates that LiDAR rangefinders are more appropriate than RADAR for low-level terrain following, even if they are slightly affected by weather. There is an increasing range of off-the-shelf LiDAR products for UAS, lightweight, accurate and with the required range. Additionally, GNSS receivers and IMU sensors from a flight controller can be used for implementation of a GNSS/INS loosely coupled Kalman filter with post-processing smoothing. Together with a LiDAR DEM dataset, a low-cost and lightweight commercially available sensor suite for a terrain following tier 1 UAS can be obtained. With centimetre level uncertainties, these sensor suites can achieve a remarkable performance with just a 2% degradation from the baseline, at an absolute ± 0.34 m of deviation for precise and safe NOE flights.

In conclusion, this thesis proposes a new dual-stage terrain following system capable of providing real-time path planning and vertical obstacle avoidance guidance and control. It is achieved through a fast risk-aware cubic B-splines DEM-Obstacle envelope and NMPC. The system prioritises safety over performance in the presence of uncertainty, balancing a trade-off through the chance constraint formulation and the risk allowance selected. Further research and studies can progress on the work by providing end-to-end path planning capabilities to avoid inevitable collision states, incorporate lateral avoidance and dynamics through cubic surface envelopes, and develop risk allocation strategies to improve the performance. A natural progression of the research requires the implementation of software-in-the-loop (SITL) simulations integrated with a flight controller, eventually leading towards hardware-in-the-loop (HITL) and flight testing of the terrain following system.

References

1. Yurtsever MG, Büyükekiz KB, Mor ZS, Taşdemir A, Topuk Ş, Erkek BB and Baş E. Exploring the Design Philosophy and Challenges in Cyber -Physical-Human Systems: A Focus on Autonomous Terrain Following. IFAC-PapersOnLine 2024; 58. 5th IFAC Workshop on Cyber-Physical Human Systems:203–6. doi: 10.1016/j.ifacol.2025.01.181
2. GAD A. Flight Manual: PANAVIA 200 TORNADO, AER.1F-PA200-1. NATO D.G.C.A.A.S. - Direzione Assistenza Velivoli, Torino, 1990. Available from: <https://www.secretprojects.co.uk/panavia200.pdf> [Accessed on: 2025 Jul 12]
3. Islam SMR. Drones on the Rise: Exploring the Current and Future Potential of UAVs. IEEE Conference 2023. doi: 10.48550/arXiv.2304.13702
4. Ramasamy S, Sabatini R, Gardi A and Liu J. LIDAR obstacle warning and avoidance system for unmanned aerial vehicle sense-and-avoid. Aerospace Science and Technology 2016; 55:344–58. doi: 10.1016/j.ast.2016.05.020
5. AeroVironment. Puma 3AE Datasheet. Internet. 2025. Available from: https://www.avinc.com/images/uploads/product_docs/2025_Puma3AE_Datasheet_250707_v08.pdf [Accessed on: 2025 Jul 27]
6. INSITU, A Boeing Company. ScanEagle Product Card. Internet. 2024. Available from: https://www.insitu.com/wp-content/uploads/2025/04/ScanEagle_ProductCard_DU040825.pdf [Accessed on: 2025 Jul 27]
7. Maghsoudi H and Kosari A. Generation of an optimal low-altitude trajectory for a fixed-wing unmanned aerial vehicle in a mountainous area. Aviation 2021; 25(2):115–22. doi: 10.3846/aviation.2021.13291
8. Lapp TR. Guidance and control using model predictive control for low altitude real-time terrain following flight. MSc thesis. Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics, 1979. Available from: <https://dspace.mit.edu/handle/1721.1/30278>
9. Clark M and Roberts RC. Autonomous quadrotor terrain-following with a laser rangefinder and gimbal system. Proceedings of IEEE Sensors 2017; 2017-December. Cited by: 4:1–3. doi: 10.1109/ICSENS.2017.8234268
10. Yang Y, Huang Y, Yang H, Zhang T, Wang Z and Liu X. Real-time terrain-following of an autonomous quadrotor by multi-sensor fusion and control. Applied Sciences (Switzerland) 2021; 11. Cited by: 7; All Open Access, Gold Open Access, Green Open Access:1–19. doi: 10.3390/app11031065

11. Wubben J, Morales C, Calafate CT, Hernández-Orallo E, Cano JC and Manzoni P. Improving UAV Mission Quality and Safety through Topographic Awareness. *Drones* 2022; 6. doi: 10.3390/drones6030074
12. Dukan F and Sørensen AJ. Altitude estimation and control of ROV by use of DVL. *IFAC Proceedings Volumes (IFAC-PapersOnline)* 2012; 9. Cited by: 5; All Open Access, Bronze Open Access:79–84. doi: 10.3182/20120919-3-IT-2046.00014
13. Melita CD, Guastella DC, Cantelli L, Di Marco G, Minio I and Muscato G. Low-Altitude Terrain-Following Flight Planning for Multirotors. *Drones* 2020; 4. doi: 10.3390/drones4020026
14. Lim J, Achermann F, Girod R, Lawrance N and Siegwart R. Safe Low-Altitude Navigation in Steep Terrain With Fixed-Wing Aerial Vehicles. *IEEE Robotics and Automation Letters* 2024; 9:4599–606. doi: 10.1109/LRA.2024.3368800
15. Kim MK, Cho MH and Tahk MJ. Trajectory Design with Path Prediction for Terrain Following. 30th Congress of the International Council of the Aeronautical Sciences 2016; 05(1). Available from: https://www.icas.org/icas_archive/ICAS2016/data/preview/2016_0379.htm
16. Lee D and Shim DH. Spline-RRT* based optimal path planning of terrain following flights for fixed-wing UAVs. *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2014 :257–61. doi: 10.1109/URAI.2014.7057427
17. Samar R and Rehman A. Autonomous terrain-following for unmanned air vehicles. *Mechatronics* 2011; 21. Special Issue on Development of Autonomous Unmanned Aerial Vehicles:844–60. doi: 10.1016/j.mechatronics.2010.09.010
18. Jonsson R. Planning Terrain Following Flight Paths - An Elastic Band Algorithm. MSc thesis. Karlstads University, Faculty of Health, Science and Technology, 2017. Available from: <https://www.diva-portal.org/smash/get/diva2:1109259/FULLTEXT01.pdf>
19. Kyriakis P and Moustris G. Terrain Following for Fixed-Wing Unmanned Aerial Vehicles Using Feedback Equivalence. *IEEE Control Systems Letters* 2019; 3:150–5. doi: 10.1109/LCSYS.2018.2854239
20. Livshitz A and Idan M. Preview Control Approach for Laser-Range-Finder-Based Terrain Following. *IEEE Transactions on Aerospace and Electronic Systems* 2020; 56:1318–31. doi: 10.1109/TAES.2019.2933955
21. Zhu R, Wang B, Wang J, Jin B and Tang C. Online Terrain Following Scheme Combining Cubic Spline and Model Predictive Control. *Proceedings of 2023 7th Chinese Conference on Swarm Intelligence and Cooperative Control*. Ed. by Jiang GP, Wang M and Ren Z. Singapore: Springer Nature Singapore, 2024 :390–401. doi: 10.1007/978-981-97-3340-8_35
22. SkyWalker. Skywalker X8. Internet. 2015. Available from: <http://skywalkermodel.com/en/76.html> [Accessed on: 2025 Jul 30]
23. Løw-Hansen B, Hann R, Gryte K, Johansen TA and Deiler C. Modeling and identification of a small fixed-wing UAV using estimated aerodynamic angles. *CEAS Aeronautical Journal* 2025; 16:501–23. doi: 10.1007/s13272-025-00816-3

24. Hacker Motor GmbH. A40-12S V4 kv610. Internet. 2025. Available from: https://www.hacker-motor-shop.com/A40-12S-V4-kv610.htm?shop=hacker_e&a=article&ProdNr=33726608&p=310 [Accessed on: 2025 Jul 30]
25. US Geographical Survey. USGS EROS Archive - Digital Elevation - Global Multi-resolution Terrain Elevation Data 2010 (GMTED2010). Internet. 2018. Available from: <https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-global-multi-resolution-terrain-elevation> [Accessed on: 2025 Aug 1]
26. US Geographical Survey. USGS EROS Archive - Digital Elevation - Interferometric Synthetic Aperture Radar (IFSAR) - Alaska. Internet. 2018. Available from: <https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-interferometric-synthetic-aperture-radar#overview> [Accessed on: 2025 Aug 1]
27. US Geographical Survey. 1 meter Digital Elevation Models (DEMs) - USGS National Map 3DEP Downloadable Data Collection. Internet. 2025. Available from: <https://data.usgs.gov/datacatalog/data/USGS:77ae0551-c61e-4979-aedd-d797abdcde0e> [Accessed on: 2025 Aug 1]
28. Suzuki T, Inoue D and Amano Y. Robust UAV Position and Attitude Estimation using Multiple GNSS Receivers for Laser-based 3D Mapping. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019 Nov :4402–8. doi: 10.1109/iroso40897.2019.8967894
29. Chang Y, Cheng Y, Manzoor U and Murray J. A review of UAV autonomous navigation in GPS-denied environments. *Robotics and Autonomous Systems* 2023; 170:104533. doi: 10.1016/j.robot.2023.104533
30. Hashim HA. Advances in UAV avionics systems architecture, classification and integration: A comprehensive review and future perspectives. *Results in Engineering* 2025; 25:103786. doi: 10.1016/j.rineng.2024.103786
31. Boguspayev N, Akhmedov D, Raskaliyev A, Kim A and Sukhenko A. A Comprehensive Review of GNSS/INS Integration Techniques for Land and Air Vehicle Applications. *Applied Sciences* 2023; 13. doi: 10.3390/app13084819
32. Wang, Mengke, Yu, Peidong and Li, Yunzhi. Performance analysis of GNSS/INS loosely coupled integration systems under GNSS signal blocking environment. *E3S Web Conf.* 2020; 206:02013. doi: 10.1051/e3sconf/202020602013
33. SPAN. UIMU LCI: Tactical Grade, low Noise IMU delivers 3d Position, velocity and attitude solution as part of Span Technology. Internet. 2014. Available from: <https://hexagondownloads.blob.core.windows.net/public/Novatel/assets/Documents/Papers/IMU-LCI/IMU-LCI.pdf> [Accessed on: 2025 Aug 2]
34. Merei A, Mcheick H, Ghaddar A and Rebaine D. A Survey on Obstacle Detection and Avoidance Methods for UAVs. *Drones* 2025; 9. doi: 10.3390/drones9030203
35. Chandran NK, Sultan MTH, Lukaszewicz A, Shahar FS, Holovatyy A and Giernacki W. Review on Type of Sensors and Detection Method of Anti-Collision System of Unmanned Aerial Vehicle. *Sensors* 2023; 23. doi: 10.3390/s23156810

36. Mohamed SAS, Haghbayan MH, Westerlund T, Heikkonen J, Tenhunen H and Plosila J. A Survey on Odometry for Autonomous Navigation Systems. *IEEE Access* 2019; 7:97466–86. DOI: 10.1109/ACCESS.2019.2929133
37. Balestrieri E, Daponte P, De Vito L and Lamonaca F. Sensors and Measurements for Unmanned Systems: An Overview. *Sensors* 2021; 21. DOI: 10.3390/s21041518
38. Bartmiński P, Si luch M and Kociuba W. The Effectiveness of a UAV-Based LiDAR Survey to Develop Digital Terrain Models and Topographic Texture Analyses. *Sensors* 2023; 23. DOI: 10.3390/s23146415
39. Balestrieri E, Daponte P, De Vito L, Picariello F and Tudosa I. Sensors and Measurements for UAV Safety: An Overview. *Sensors* 2021; 21. DOI: 10.3390/s21248253
40. Nakagaki Y and Zhai G. A Study on Optimization Algorithms in MPC. *Journal of Physics: Conference Series* 2020 Mar; 1490:012073. DOI: 10.1088/1742-6596/1490/1/012073
41. Idman T. Path planning and trajectory generation - model predictive control. BSc thesis. KTH Royal Institute of Technology, School of Engineering Sciences, 2019. Available from: <https://www.diva-portal.org/smash/get/diva2:1380200/FULLTEXT01.pdf>
42. Lindqvist B. On the Utilization of Nonlinear MPC for Unmanned Aerial Vehicle Path Planning. PhD thesis. Lulea University of Technology, Dept. of Computer Science and Electrical Engineering, 2021. Available from: <https://www.diva-portal.org/smash/get/diva2:1613413/FULLTEXT01.pdf>
43. Bui DN, Khuat TH, Phung MD, Tran TH and Tran DL. Model Predictive Control for Optimal Motion Planning of Unmanned Aerial Vehicles. *2024 International Conference on Control, Robotics and Informatics (ICCRI)*. Los Alamitos, CA, USA: IEEE Computer Society, 2024 Jul :1–6. DOI: 10.1109/ICCRI64298.2024.00025
44. McInerney I, Constantinides GA and Kerrigan EC. A Survey of the Implementation of Linear Model Predictive Control on FPGAs. *IFAC-PapersOnLine* 2018; 51. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018:381–7. DOI: 10.1016/j.ifacol.2018.11.063
45. Li W, Zhang X, Wang Y and Xie S. Comparison of Linear and Nonlinear Model Predictive Control in Path Following of Underactuated Unmanned Surface Vehicles. *Journal of Marine Science and Engineering* 2024; 12. DOI: 10.3390/jmse12040575
46. Gros S, Zanon M, Quirynen R, Bemporad A and Diehl M. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control* 2020; 93:62–80. DOI: 10.1080/00207179.2016.1222553
47. Su Y, Tan KK and Lee TH. Computation delay compensation for real time implementation of robust model predictive control. *Journal of Process Control* 2013; 23:1342–9. DOI: 10.1016/j.jprocont.2013.09.006
48. Patne V, Ingole D and Sonawane D. Towards Fast Nonlinear Model Predictive Control for Embedded Applications. *IFAC-PapersOnLine* 2022; 55. 22nd IFAC Symposium on Automatic Control in Aerospace ACA 2022:304–9. DOI: 10.1016/j.ifacol.2023.03.051

49. Andersson JAE, Gillis J, Horn G, Rawlings JB and Diehl M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* 2019; 11:1–36. doi: 10.1007/s12532-018-0139-4
50. Stella L, Themelis A, Sopasakis P and Patrinos P. A Simple and Efficient Algorithm for Nonlinear Model Predictive Control. *IEEE 56th Annual Conference on Decision and Control* 2017. doi: 10.48550/arXiv.1709.06487
51. Verschueren R, Frison G, Kouzoupis D, Frey J, Duijkeren N van, Zanelli A, Novoselnik B, Albin T, Quirynen R and Diehl M. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation* 2021. DOI: 10.48550/arXiv.1910.13753
52. Boiangiu CA, Morosan A and Stan M. Fractal Objects in Computer Graphics. *IEEE International Conference on Application of Information and Communication Technologies*. 2015 Jun :9. Available from: https://www.researchgate.net/publication/287218131_Fractal_Objects_in_Computer_Graphics
53. Philipp K. Janert, Ph.D. The Diamond-Square Algorithm for Terrain Generation. Internet. 2022. Available from: <https://janert.me/blog/2022/the-diamond-square-algorithm-for-terrain-generation/> [Accessed on: 2025 Aug 6]
54. Sun W, Sun P, Ding W, Zhao J and Li Y. Gradient-based autonomous obstacle avoidance trajectory planning for B-spline UAVs. *Scientific Reports* 2024; 14:14458. doi: 10.1038/s41598-024-65463-w
55. Amiri-Simkooei A, Esmaeili F and Lindenbergh R. Least squares B-spline approximation with applications to geospatial point clouds. *Measurement* 2025; 248:116887. doi: 10.1016/j.measurement.2025.116887
56. Piegl L and Tiller W. The NURBS Book. 2nd ed. Monographs in Visual Communication. Originally published: 14 November 1996. eBook published: 6 December 2012. Springer Berlin, Heidelberg, 1997 :XIV, 646. doi: 10.1007/978-3-642-59223-2
57. Massachusetts Institute of Technology. Iterative Risk Allocation: A Gentle Introduction by Ben Ayton and Steve Levine. Internet. 2016. Available from: https://ocw.mit.edu/courses/16-412j-cognitive-robotics-spring-2016/d197eacf01a89659f3e05c275d9d3946_MIT16_412JS16_RR2.pdf [Accessed on: 2025 Aug 7]
58. CubePilot. HerePro. Internet. 2025. Available from: <https://www.cubepilot.com/#/herepro/specs> [Accessed on: 2025 Aug 12]
59. CubePilot. The Cube Orange Standard Set. Internet. 2025. Available from: <https://www.cubepilot.com/#/cube/specs> [Accessed on: 2025 Aug 12]
60. TDK. ICM-45686 High-Precision 6-Axis MEMS MotionTracking Device. Internet. 2025. Available from: <https://invensense.tdk.com/products/motion-tracking/6-axis/icm-45686/> [Accessed on: 2025 Aug 12]
61. Lightware. SF30/D microLiDAR sensor. Internet. 2025. Available from: <https://lightwarelidar.com/wp-content/uploads/2025/07/SF30D-Product-Guide-v3.1.pdf> [Accessed on: 2025 Aug 12]

Appendix A

Ethical Approval Letter



17 April 2025

Dear Mr Padilla Moure ,

Reference: CURES/24853/2025

Project ID: 28308

Title: Individual Research Project: Terrain Following – Low Level Fixed Altitude Flying

Thank you for your application to the Cranfield University Research Ethics System (CURES).

We are pleased to inform you your CURES application, reference CURES/24853/2025 has been reviewed. You may now proceed with the research activities you have sought approval for.

If you have any queries, please contact CURES Support.

We wish you every success with your project.

Regards,

CURES Team

Figure A.1: CURES approval letter.

Appendix B

Software Codes

The software code suite developed in this thesis is available in a private GitHub repository, provided per request. Please contact the author at Pol.PadillaMoure.151@cranfield.ac.uk, or pol.padilla@icloud.com if the university address is unavailable.

The appendix provides the codes for generation of the cubic B-spline DEM-Obstacle envelope, the UAS dynamic model, and the terrain following NMPC employing CasADi, in Python.

B.1 Cubic B-spline Envelope

```
1  #!/usr/bin/env python3
2
3  """
4  Function: QP with cubic B-splines for Obstacle-Terrain envelope update in
5  ↪ CasADi
6  Author: Pol Padilla Moure
7  """
8
9  import pickle
10 import casadi as ca
11 import numpy as np
12 from scipy.interpolate import BSpline
13 from scipy.stats import norm
14 from multiprocessing import Process
15
16 from utils.plot_tools import PlotTools
17 plts = PlotTools()
18
19 class QPenvelope:
20     def __init__(self, lenT=20, lenO=20, lenC=20, aggressiveness = 0.5,
21         ↪ delta_meas=0.5, delta_dem=0.5, sigma_dem=0.0,):
22
23         # Parameters
```

```

23     self.N = lenC                                # Number of
        ↪ control points (terrain peaks for linear basis functions)
24     self.a = aggressiveness                        # 0 =
        ↪ smooth/loose, 1 = tight to terrain
25     self.delta_meas = delta_meas                  # Obstacle
        ↪ measurement risk level
26     self.delta_dem = delta_dem                    # Digital
        ↪ Elevation Model risk level
27     self.k_meas = norm.ppf(self.delta_meas)        # Inverse cdf of
        ↪ risk level for Measurement (for chance constraint)
28     self.k_dem = norm.ppf(self.delta_dem)          # Inverse cdf of
        ↪ risk level for Digital Elevation Model (for chance
        ↪ constraint)
29
30     # CasADi Opti setup - with convex 'conic' solvers
31     self.opti = ca.Opti('conic')
32     self.c = self.opti.variable(self.N)           # Spline
        ↪ coefficients (to optimise)
33     self.z_terr = self.opti.parameter(lenT)        # DEM terrain
        ↪ samples
34     self.z_obs = self.opti.parameter(lenO)         # Measured
        ↪ obstacle samples
35     self.B_terr = self.opti.parameter(lenT, lenC)  # Basis functions
        ↪ for terrain samples
36     self.B_obs = self.opti.parameter(lenO, lenC)  # Basis functions
        ↪ for obstacle samples
37     self.sigma_meas = self.opti.parameter(lenO)   # Vertical
        ↪ standard deviation of measurements
38
39     # --- OCP PROBLEM DEFINITION --- #
40     # Smoothness penalty using second-order finite differences
41     D2 = np.zeros((self.N-2, self.N))
42     for i in range(self.N-2):
43         D2[i, i] = 1
44         D2[i, i+1] = -2
45         D2[i, i+2] = 1
46     Q_smooth = ca.mtimes([self.c.T, D2.T @ D2, self.c])
47
48     # Deviation from baseline
49     deviation = self.B_terr @ self.c - self.z_terr
50     Q_deviation = ca.dot(deviation, deviation)
51
52     # Combined quadratic cost (smoothness + deviation)
53     cost = (1-self.a) * Q_smooth + self.a * Q_deviation
54     self.opti.minimize(cost)
55
56     # Constraints (above terrain and obstacles)
57     # Uncertainty handling is done through chance constraint "k"
        ↪ on the standard deviation "sigma" (for DEM and Measurements)

```

```

58     self.opti.subject_to(self.B_terr @ self.c >= self.z_terr -
59         ↪ sigma_dem * self.k_dem )
60
61     self.opti.subject_to(self.B_obs @ self.c >= self.z_obs -
62         ↪ self.sigma_meas * self.k_meas )
63
64     # --- SOLVER DEFINITION --- #
65     # Solver settings (change 'osqp.verbose' True to see parameters of
66     ↪ solver)
67     jit_options = {"flags": ["-O3"], "verbose": True}
68     opts_setting = {'osqp.verbose': False,
69         ↪ 'error_on_fail': True,
70         ↪ "jit": True, "compiler": "shell", "jit_options":
71         ↪ jit_options}
72     self.opti.solver("osqp",opts_setting)
73     #self.opti.solver("qpoads")
74
75 def run_qp_solver(self,x_terrain,z_terrain,x_obstacle,z_obstacle,
76     ↪ sigma_meas, switch_debug =
77     ↪ False):
78     # Spline basis (uniform cubic B-spline) // respect lenT, lenO and
79     ↪ lenC
80     degree = 3
81     ↪ # Cubic spline
82     knots = np.linspace(x_terrain[0], x_terrain[-1], self.N - 2)
83     ↪ # Cubic need (FK - 2*degree) internal knots
84     full_knots = np.concatenate((knots[0]*degree, knots,
85     ↪ [knots[-1]]*degree)) # FK size = N + degree + 1
86     B_terrain = np.array([BSpline(full_knots, np.eye(self.N)[i],
87     ↪ degree)(x_terrain) for i in range(self.N)]).T # Evaluate
88     ↪ Basis Function at x_terrain
89     B_obstacle = np.array([BSpline(full_knots, np.eye(self.N)[i],
90     ↪ degree)(x_obstacle) for i in range(self.N)]).T # Evaluate
91     ↪ Basis Function at x_obstacle
92
93     # Set input parameters
94     self.opti.set_value(self.z_terr, z_terrain)
95     self.opti.set_value(self.z_obs, z_obstacle)
96     self.opti.set_value(self.B_terr, B_terrain)
97     self.opti.set_value(self.B_obs, B_obstacle)
98     self.opti.set_value(self.sigma_meas, sigma_meas)
99     # Give an initial guess
100    self.opti.set_initial(self.c, np.full(self.N, z_terrain[0]))
101
102    # Execute solver
103    sol = self.opti.solve()
104    c_opt = sol.value(self.c)
105
106    # Evaluate optimized spline
107    spline_y = B_terrain @ c_opt

```

```

93
94     # Debugging:
95     if switch_debug:
96         # Plot QP envelope:
97         print(c_opt)
98         p1 = Process(target=plts.plot_qp_envelope,
99                     ↪ args=(x_terrain,z_terrain,x_obstacle,z_obstacle,
100                     ↪ spline_y,self.a,self.delta_meas,self.delta_dem))
101         p1.start()
102
103         # Solve again:
104         #sol = self.opti.solve()
105
106     return spline_y
107
108 if __name__ == '__main__':
109
110     # Read Terrain Data:
111     filename = 'terrain_data.pkl'
112     with open('codes/data/'+filename, 'rb') as f:
113         data = pickle.load(f)
114     d_path = np.array(data['d_path'])
115     z_path = np.array(data['z_path'])
116
117     # We define a certain amount of steps upon we can introduce updated
118     ↪ terrain:
119     dist = 100 # updated distance knowledge
120     lenT = int(dist/np.mean(np.diff(d_path)))
121     # Length of Control points (reccomend =< lenT)
122     lenC = lenT
123     # Length of obstacle points (with a large buffer space)
124     lenO = 100
125
126     # Define Terrain Points (DEM):
127     x_terrain = data['d_path'][:lenT]
128     z_terrain = data['z_path'][:lenT]
129     # Additional points above terrain (fill obstacle buffer with last
130     ↪ terrain knowledge)
131     x_meas,z_meas = np.array([2,2.5,1.5,-1,0]) +
132     ↪ 10,np.array([3,2.2,4,3,1]) + 110
133     x_obstacle,z_obstacle = np.full(lenO, x_terrain[-1]),np.full(lenO,
134     ↪ z_terrain[-1])
135     x_obstacle[:len(x_meas)] = x_meas
136     z_obstacle[:len(z_meas)] = z_meas
137     # Prepare vertical standard deviation of measurements (non-completed
138     ↪ == end DEM have 0 std)
139     sigma_meas = np.full(lenO, 0.0)
140     sigma_meas[:len(z_meas)] = 0.5

```

```

135     # Initialise MPC class:
136     qp = QPenelope(lenT,len0,lenC,aggressiveness=0.1,delta_meas=0.01,
137         ↪ delta_dem=0.9,sigma_dem=0.2)
138
139     # QP envelope test to check functionality:
140     qp.run_qp_solver(x_terrain,z_terrain,x_obstacle,z_obstacle,
141         ↪ sigma_meas,switch_debug=True)

```

B.2 UAS Longitudinal Dynamics

```

1  #!/usr/bin/env python3
2
3  """\
4  Function: dynamic implementation of longitudinal dynamic equations in
5  ↪ CasADi
6  Author: Pol Padilla Moure
7  """
8
9  import casadi as ca
10 import numpy as np
11 from multiprocessing import Process
12
13 from utils.plot_tools import PlotTools
14 plts = PlotTools()
15
16 # --- PARAMETERS AND COEFFICIENTS --- #
17 """ HYPOTHESIS:
18     Vertical Flight - All lateral effects/derivatives negligible
19     No wind
20     No altitude effects
21
22     AIRCRAFT: Skywalker X8 -
23     ↪ (https://link.springer.com/article/10.1007/s13272-025-00816-3)
24 """
25 # Aircraft Parameters:
26 m = 3.364          # [kg] mass
27 I_y = 0.140        # [kg m^2] inertia
28 cmean = 0.36       # [m] mean chord
29 b = 2.10           # [m] wingspan
30 S = 0.75           # [m^2] surface
31
32 # Atm parameters:
33 rho = 1.225        # [kg/m^3] air density
34 g = 9.81           # [m/s^2] acceleration
35
36 # Motor/Thrust:
37 k_E = 0.0157       # [V/(rad/s)] Back-EMF constant

```

```

37 R = 0.017          # [Ohm] motor resistance
38 D = 0.3556         # [m] prop. diameter
39 I_p = 3.46*10**(-4) # [kg m^2] prop. inertia
40 U_b = 14.9          # [V] motor/battery voltage (manufacturer data)
41 I_m = 49            # [A] motor current (manufacturer data)
42
43 # Thrust Coeff:
44 C_T0, C_T1, C_T2, C_T3 = 0.1400, -0.0300, -0.2370, 0.0847
45
46 # Drag Coeff:
47 C_D0, C_Dq, C_DCT, C_Dk1, C_Dk2 = 0.058, 0.480, -0.217, -0.034, 0.225
48
49 # Lift Coeff:
50 C_L0, C_Lalpha, C_Lq, C_Ldeltae = -0.077, 2.573, 17.119, 1.369
51
52 # Pitch Coeff:
53 C_m0, C_malpha, C_mq, C_mdeltae = 0.027, -0.274, -1.608, -0.276
54
55
56 # --- DYNAMICS CLASS --- #
57 class DynamicsUAV:
58     def __init__(self, switch_sxm=0):
59
60         """
61         % ODE System
62         % - - - - -
63         %     Variable order
64         %         u - body speed
65         %         w - body speed
66         %         q - pitch rate
67         %         theta - pitch angle
68         %         x - inertial position
69         %         z - inertial position (upwards)
70         %     Control order
71         %         delta_e - elevator command (rad)
72         %         delta_t - throttle command
73         % - - - - -
74         """
75
76         # CasADi variable declaration
77         if switch_sxm == 0: # SX implementation (for computational
78             ↪ efficiency)
79             # State variables:
80             self.X = ca.SX.sym('X', 6) # 6 state variables
81             u, w, q, theta, x, z = self.X[0], self.X[1], self.X[2],
82             ↪ self.X[3], self.X[4], self.X[5]
83             # Control variables:
84             self.U = ca.SX.sym('U', 2) # delta_e, delta_t
85             delta_e, delta_t = self.U[0], self.U[1]

```

```

84
85 elif switch_sxm == 1: # MX implementation (for more complex
    ↪ system)
86     # State variables:
87     u, w, q, theta, x, z = ca.MX.sym('u'), ca.MX.sym('w'),
    ↪ ca.MX.sym('q'), ca.MX.sym('theta'), ca.MX.sym('x'),
    ↪ ca.MX.sym('z')
88     # Control variables:
89     delta_e, delta_t = ca.MX.sym('delta_e'), ca.MX.sym('delta_t')
90     self.X = ca.vertcat(u, w, q, theta, x, z)
91     self.U = ca.vertcat(delta_e, delta_t)
92
93     # --- EQUATION CONSTRUCTION --- #
94     eps = 1e-3 # for numerical error disipation
95     # Change of variables:
96     V_a = ca.sqrt(u**2 + w**2 + eps) # aerodynamic speed (=
    ↪ ground as no wind)
97     q_ast = q * cmean / (2 * V_a + eps) # nondim angular rate
98     alpha = ca.atan2(w + eps, u + eps) # angle of attack
    ↪ (atan(w/u))
99     q_bar = 0.5 * rho * V_a**2 # dynamic pressure
100
101     ## COEFFICIENTS ##
102     # Thrust coefficient:
103     Omega_p = (delta_t * U_b - I_m * R) / k_E
104     J = 2 * ca.pi * V_a / (Omega_p * D)
105     C_T = C_T0 + C_T1 * J + C_T2 * J**2 + C_T3 * J**3
106
107     # Aero Coeff Equations:
108     C_L = C_L0 + C_Lalpha*alpha + C_Lq*q_ast + C_Ldeltae*delta_e
109     C_D = C_D0 + C_Dq*ca.fabs(q_ast) + C_DCT*ca.fabs(C_T) +
    ↪ C_Dk1*ca.fabs(C_L) + C_Dk2*ca.fabs(C_L)**2
110     C_m = C_m0 + C_malpha*alpha + C_mq*q_ast + C_mdeltae*delta_e
111
112     ## FORCES (body axis) ##
113     # Prop force:
114     T = rho * D**4 * C_T * Omega_p**2 / (4 * ca.pi**2)
115     Fb_prop = ca.vertcat(T, 0)
116
117     # Aero Forces:
118     R_sb = ca.vertcat(
119         ca.horzcat(ca.cos(alpha), ca.sin(alpha)),
120         ca.horzcat(-ca.sin(alpha), ca.cos(alpha))
121     )
122     Fb_aero = q_bar * S * ca.mtimes(R_sb.T, ca.vertcat(-C_D, -C_L))
123
124     # Gravity Forces:
125     R_nb = ca.vertcat(
126         ca.horzcat(ca.cos(theta), ca.sin(theta)),

```

```

127         ca.horzcat(-ca.sin(theta), ca.cos(theta))
128     )
129     Fb_grav = ca.mtimes(R_nb.T, ca.vertcat(0, m * g))
130
131     # Total Force:
132     Fb = Fb_prop + Fb_aero + Fb_grav
133
134     ## MOMENT (body axis) ##
135     Mb = q_bar * S * cmean * C_m
136
137     ## DYNAMIC EQUATIONS ##
138     # ---
139     dudt = Fb[0] / m - q * w
140     dwdt = Fb[1] / m + q * u
141     dqdt = Mb / I_y
142     dthetadt = q
143     dxdt = u * ca.cos(theta) + w * ca.sin(theta)
144     dzdt = u * ca.sin(theta) - w * ca.cos(theta)
145     # ---
146
147     # --- CASADI ODE SYSTEM --- #
148     # Define CasADi function:
149     self.f = ca.vertcat(dudt, dwdt, dqdt, dthetadt, dxdt, dzdt)
150     self.f_dyn = ca.Function('f_dyn', [self.X, self.U], [self.f])
151
152     # Define ODE for integrator
153     self.ode = {'x': self.X, 'p': self.U, 'ode': self.f_dyn(self.X,
154         ↪ self.U)}
155
156     def test_integrator(self, dt= 0.01, fin=20):
157         # Initial conditions:
158         x0 = [16.8053+1, 2.5652, 0, 0.1515, 0, 100]
159         u0 = [-0.0526, 0.5835]
160         N = int(fin/dt)
161         t = np.linspace(0, dt*N, N+1)
162
163         # Create integrator handle // 'rk' is runge kutta 4 // for a
164         ↪ single step, use 'dt' instead of 't'
165         intg = ca.integrator('intg', 'rk', self.ode, 0, t)
166         # Call integrator from a starting point
167         res = intg(x0=x0, p=u0)
168         print(res)
169
170         # Plot Dynamic Response:
171         p1 = Process(target=plts.plot_dynamic_response, args=(t,res))
172         p1.start()
173
174     if __name__ == '__main__':
175         dyn = DynamicsUAV(switch_sxm=0)

```

```

174
175     # Integrator test function (for debugging):
176     dyn.test_integrator(dt=0.01, fin=20)

```

B.3 Terrain Following NMPC

```

1  #!/usr/bin/env python3
2
3  """\
4  Function: Model Predictive Control of UAV for Terrain Following in CasADi
   ↳ (multiple shooting, intermediate control and state, sparsity
   ↳ structure)
5  Author: Pol Padilla Moure
6  """
7
8  import pickle
9  import casadi as ca
10 import numpy as np
11 from multiprocessing import Process
12
13 from utils.bcolors_tools import bcolors as bcs
14 from utils.plot_tools import PlotTools
15 plts = PlotTools()
16
17 from dynamics_casadi import DynamicsUAV
18
19
20 class MPCsolver:
21     def __init__(self, x_data, z_data, lenT, x_min, x_max, u_min, u_max,
22                 minh=3.0, href=6.0, maxh=50.0, vref=17.0, sdt=0.2,
23                 ↳ N=20, max_iter=1e3):
24
25         # Horizon parameter class variables (Hp = sdt * N)
26         self.dt = sdt # solver timestep
27         self.N = N # prediction timespteps
28         self.x_min, self.x_max, self.u_min, self.u_max =
29         ↳ x_min, x_max, u_min, u_max
30         """
31         self.x_min = ca.vertcat(x_min[0], x_min[1], x_min[2], x_min[3],
32         ↳ x_min[4], x_min[5])
33         self.x_max = ca.vertcat(x_max[0], x_max[1], x_max[2], x_max[3],
34         ↳ x_max[4], x_max[5])
35         self.u_min = ca.vertcat(u_min[0], u_min[1])
36         self.u_max = ca.vertcat(u_max[0], u_max[1])
37         self.opti.subject_to(self.x_min[0] <= (self.X[k][0] <=
38         ↳ self.x_max[0]))
39         self.opti.subject_to(self.x_min[1] <= (self.X[k][1] <=
40         ↳ self.x_max[1]))

```

```

35         self.opti.subject_to(self.x_min[2] <= (self.X[k][2] <=
36             ↪ self.x_max[2]))
37         self.opti.subject_to(self.x_min[3] <= (self.X[k][3] <=
38             ↪ self.x_max[3]))
39         self.opti.subject_to(self.x_min[4] <= (self.X[k][4] <=
40             ↪ self.x_max[4]))
41         self.opti.subject_to(self.x_min[5] <= (self.X[k][5] <=
42             ↪ self.x_max[5]))
43         self.opti.subject_to(self.u_min[0] <= (self.U[k][0] <=
44             ↪ self.u_max[0]))
45         self.opti.subject_to(self.u_min[1] <= (self.U[k][1] <=
46             ↪ self.u_max[1]))
47     """
48
49     # --- CASADI ODE SYSTEM --- #
50     # Initilaise dynamic equations system:
51     # ('0' for SX variables // '1' for MX variables)
52     switch_sxmx = 1
53     dyn = DynamicsUAV(switch_sxmx)
54     """
55
56     % ODE System
57     % - - - - -
58     %     Variable order
59     %         u - body speed
60     %         w - body speed
61     %         q - pitch rate
62     %         theta - pitch angle
63     %         x - inertial position
64     %         z - inertial position (upwards)
65     %     Control order
66     %         delta_e - elevator command (rad)
67     %         delta_t - throttle command
68     % - - - - -
69     """
70
71     # Discretize system and build explicit integrator function
72     nx = dyn.X.numel()      # number of states
73     nu = dyn.U.numel()      # number of controls
74     dt = ca.MX.sym("dt")
75     sys = {}                # Discretized ode system definition
76     sys["x"] = dyn.X
77     sys["u"] = dyn.U
78     sys["p"] = dt
79     sys["ode"] = dyn.f*dt   # Explicit ODE discretization and time
80                             ↪ scaling (fastness)
81     # Integrator setup: (solvers are 'cvmex', 'idas', 'rk',
82                             ↪ 'collocation') || "simplify" enables JIT shell compilation
83     self.intg = ca.integrator('intg', 'rk', sys, 0, 1, {"simplify": True,
84                             ↪ "number_of_finite_elements": 4})

```

```

75 self.F = ca.Function('F', [dyn.X, dyn.U, dt], [self.intg(x0=dyn.X,
76   ↪ u=dyn.U, p=dt) ["xf"]], ["x", "u", "dt"], ["xnext"])
77
78 # CasADi variable declaration for Terrain Interpolation
79 if switch_sxm == 0: # SX implementation (for computational
80   ↪ efficiency)
81     x_pos = ca.SX.sym('x_pos') # UAV position over
82     ↪ terrain
83     x_terr_data = ca.SX.sym('x_terr_data', lenT) # Terrain
84     ↪ position
85     z_terr_data = ca.SX.sym('z_terr_data', lenT) # Terrain
86     ↪ elevation
87
88 elif switch_sxm == 1: # MX implementation (for more complex
89   ↪ system)
90     x_pos = ca.MX.sym('x_pos') # UAV position over
91     ↪ terrain
92     x_terr_data = ca.MX.sym('x_terr_data', lenT) # Terrain
93     ↪ position
94     z_terr_data = ca.MX.sym('z_terr_data', lenT) # Terrain
95     ↪ elevation
96
97 # --- CASADI OPTI SYSTEM --- #
98 # Create Opti class for NMPC solver, and horizon states
99 ↪ declaration:
100 # Multiple Shooting - intermediate states for CONTROL and STATE
101 self.opti = ca.Opti()
102 self.x0 = self.opti.parameter(nx) # Initial state
103 self.X = [] # Horizon states (do
104   ↪ not mix with dyn.X)
105 self.U = [] # Horizon control (do
106   ↪ not mix with dyn.U)
107 for k in range(N): # Fatrop requires a
108   ↪ specific sparse structure
109     self.X.append(self.opti.variable(nx))
110     self.U.append(self.opti.variable(nu))
111 self.X.append(self.opti.variable(nx)) # X is size [N+1], U
112   ↪ is size [N]
113
114 # Terrain Interpolator Setup ('0' for fixed terrain knowledge //
115   ↪ '1' for terrain profile update):
116 switch_interp = 1
117 self.z_terr = self.opti.parameter(lenT)
118 self.x_terr = self.opti.parameter(lenT)
119 if switch_interp == 0: # Case: use CasADi's interpolant
120   # ca.interpolant allows for 'bspline' or 'linear' interpolation
121   z_terrain_interp = ca.interpolant('z_terrain', 'linear',
122     ↪ [x_data], z_data.tolist())

```

```

107         z_terrain_fun = ca.Function('z_terrain', [x_pos],
108                                     ↪ [z_terrain_interp(x_pos)])
109     elif switch_interp == 1: # Case: custom interp function
110         z = 0 # terrain elevation
111         for i in range(x_terr_data.size1() - 1):
112             cond = ca.logic_and(x_pos >= x_terr_data[i], x_pos <
113                                 ↪ x_terr_data[i+1])
114             z += ca.if_else(cond,
115                             z_terr_data[i] +
116                             ↪ (x_pos-x_terr_data[i])/(x_terr_data[i+1]-x_terr_data[i])
117                             ↪ * (z_terr_data[i+1]-z_terr_data[i]),
118                             0)
119         z += ca.if_else(x_pos >= x_terr_data[-1], z_terr_data[-1], 0)
120         z += ca.if_else(x_pos < x_terr_data[0], z_terr_data[0], 0)
121         z_terrain_fun = ca.Function('z_terrain', [x_pos,
122                                             ↪ x_terr_data, z_terr_data], [z])
123
124     # --- OCP PROBLEM DEFINITION --- #
125     cost = 0 # Cost function
126     wa,wv,wq = 1,1,1 # State cost weights
127     wje,wjt = 3,0.2 # Control rate Weights
128     norm_x, norm_ud = 1,1 # Normalisation weights
129     #norm_x, norm_ud = (1/(self.N+1)), (1/(self.N-1))
130
131     for k in range(self.N):
132
133         # 1> Add dynamics as equality constraint (for next state,
134         ↪ multiple shooting method)
135         self.opti.subject_to(self.X[k+1]==self.F(self.X[k],self
136         ↪ .U[k],self.dt))
137
138         # 2> Initial condition (first state equality constraint)
139         if k == 0:
140             self.opti.subject_to(self.X[0] == self.x0)
141
142         # 3> Boundary and control conditions (inequality constraints)
143         self.opti.subject_to(self.x_min <= (self.X[k] <= self.x_max))
144         self.opti.subject_to(self.u_min <= (self.U[k] <= self.u_max))
145
146         # 4> Add constraints on state/control at time k (crash for
147         ↪ AGL < 1 m):
148         # Terrain elevation knowledge:
149         if switch_interp == 0:
150             zt_k = z_terrain_fun(self.X[k][4])
151             ↪ # DEM terrain elevation at uav position
152         elif switch_interp == 1:
153             zt_k =
154             ↪ z_terrain_fun(self.X[k][4],self.x_terr,self.z_terr)
155             ↪ # ENV terrain elevation at uav position

```

```

145     self.opti.subject_to(minh <= (self.X[k][5]-zt_k <= maxh))
146
147     # 5> Add stage cost (process, normalised for horizon length):
148     #cost += ca.sumsqr(self.X[5,k] - zt_k)
149     Ja = wa * norm_x * (self.X[k][5] - zt_k - href)**2
150     ↪ # Altitude cost (try to fly at ref. h)
151     Jv = wv * norm_x * (ca.hypot(self.X[k][0], self.X[k][1]) -
152     ↪ vref)**2 # Velocity cost (try to fly at ref. V)
153     Jq = wq * norm_x * (self.X[k][2])**2
154     ↪ # Pitch rate cost (minimise pitch rate)
155     Jje, Jjt = 0, 0
156     ↪ # Jerk cost (control rate)
157     if k > 0:
158         Jje += wje * norm_ud * ((self.U[k][0] -
159         ↪ self.U[k-1][0])/self.dt)**2
160         Jjt += wjt * norm_ud * ((self.U[k][1] -
161         ↪ self.U[k-1][1])/self.dt)**2
162     cost += Ja + Jv + Jq + Jje + Jjt
163
164     # 6> Terminal conditions (N+1 (k=N) remained to be added for
165     ↪ state, not controls)
166     if k==self.N-1:
167         if switch_interp == 0:
168             zt_k = z_terrain_fun(self.X[-1][4])
169         elif switch_interp == 1:
170             zt_k = z_terrain_fun(self.X[-1][4], self.x_terr, self
171             ↪ .z_terr)
172         # Final Boundary constraints
173         self.opti.subject_to(self.x_min <= (self.X[-1] <=
174         ↪ self.x_max))
175         # Terminal AGL constraints
176         self.opti.subject_to(minh <= (self.X[-1][5]-zt_k <=
177         ↪ maxh))
178         # Terminal cost
179         Ja = wa * norm_x * (self.X[-1][5] - zt_k - href)**2
180         Jv = wv * norm_x * (ca.hypot(self.X[-1][0],
181         ↪ self.X[-1][1]) - vref)**2
182         Jq = wq * norm_x * (self.X[-1][2])**2
183         cost += Ja + Jv + Jq
184
185     self.X = ca.hcat(self.X)
186     self.U = ca.hcat(self.U)
187
188     # > Cost optimization
189     self.opti.minimize(cost)
190
191     # --- SOLVER DEFINITION --- #
192     jit_options = {"flags": ["-O3"], "verbose": True}
193     solver = "ipopt"

```

```

183     if solver == "ipopt":
184         # IPOPT solver (t=0.277s)
185         # https://web.casadi.org/api/internal/d7/d2f/classcasadi\_1\_1IpoptInterface.html
186         # https://web.casadi.org/python-api/
187         opts_setting = {'print_time': 0,
188                        'ipopt.print_level': 0,          # 0 = quiet // 3 =
189                        ↪ typical // 5{7 = detailed // 12 = very verbose
190                        'ipopt.max_iter': max_iter,
191                        'ipopt.acceptable_tol': 1e-1,
192                        'ipopt.acceptable_obj_change_tol': 1e-2,
193                        'jit': True, 'jit_temp_suffix': True, 'compiler':
194                        ↪ 'shell', 'jit_options': jit_options}
195     elif solver == "fatrop":
196         # Fatrop solver (t=0.212s): https://arxiv.org/pdf/2303.16746
197         # https://github.com/jgillis/fatrop\_demo?tab=readme-ov-file
198         # https://web.casadi.org/api/internal/d0/dc2/classcasadi\_1\_1FatropInterface.html
199         # https://web.casadi.org/api/internal/d0/dc2/classcasadi\_1\_1FatropInterface.html
200         opts_setting = {'print_time': 0,
201                        'verbose': False,
202                        'debug': False,                  # Problem information
203                        'expand': False,                 # False to keep MX //
204                        ↪ True to transform to SX
205                        'structure_detection': 'auto',   # none / auto / manual
206                        ↪ (fatrop)
207                        'fatrop.print_level': 0,
208                        'fatrop.max_iter': max_iter,
209                        'fatrop.mu_init': 1e-1,
210                        'fatrop.acceptable_tol': 1e-1,
211                        'jit': True, 'jit_temp_suffix': True, 'compiler':
212                        ↪ 'shell', 'jit_options': jit_options}
213     self.opti.solver(solver, opts_setting)
214
215     # Solver status vector
216     self.status=[] # 0 - for nominal resolution || 1 - for max iter
217     ↪ and fallback
218
219     def start_mpc_solver(self, x_cstart, u_cstart, x_adata, z_adata,
220                        ↪ switch_debug=False):
221         """ Cold start of NMPC, we use a trimming state as prediction """
222         # Set initial conditions parameters:
223         self.opti.set_value(self.x_terr, x_adata.tolist())
224         self.opti.set_value(self.z_terr, z_adata.tolist())
225         self.opti.set_value(self.x0, x_cstart)
226         # Set initial guess (horizontal forward movement):
227         V = np.sqrt(x_cstart[0]**2 + x_cstart[1]**2) # sqrt(u^2 + w^2)

```

```

221     XG = np.tile(x_cstart.reshape(-1, 1), (1, self.N+1))
222     for k in range(1, self.N+1):
223         XG[4, k] = XG[4, k-1] + V * self.dt
224     UG = np.tile(u_cstart.reshape(-1,1), (1,self.N))
225     self.opti.set_initial(self.X, XG)
226     self.opti.set_initial(self.U, UG)
227
228     # Execute solver
229     sol = self.opti.solve()
230     sol_X = sol.value(self.X) # length N+1
231     sol_U = sol.value(self.U) # length N
232     self.status.append(0) # nominal status / start must succeed
233     ↪ to proceed
234
235     # Apply integrator to current start for real next uav step:
236     #rx_next =
237     ↪ self.integrator_dynamic_solver(x_cstart,sol_U[:,0],self.dt)
238
239     # Debugging:
240     if switch_debug:
241         print(XG.shape)
242         print(sol.stats())
243         # --- EVALUATE SOLUTION --- #
244         t = np.linspace(0, self.dt*self.N, self.N+1)
245         sol_T = np.vstack([ sol_X, np.hstack(( sol_U,
246         ↪ np.array([[ -99],[ -99]]) )) ]) # add a meaningless row
247
248         # Plot NMPC response:
249         p1 = Process(target=plts.plot_nmpc_response,
250         ↪ args=(t,sol_T,x_udata,z_udata,self.x_min,self.x_max,self
251         ↪ .u_min,self.u_max))
252         p1.start()
253
254         # Run again:
255         #sol = self.opti.solve()
256         #print(sol.stats())
257
258     return sol_X, sol_U
259
260 def warm_mpc_solver(self,x_cstart,prev_sol_X,prev_sol_U,x_udata,
261 ↪ z_udata,sim_idx):
262     """ Cold start of NMPC, we use a trimming state as prediction
263     """
264     # Set initial conditions parameters:
265     self.opti.set_value(self.x_terr, x_udata.tolist())
266     self.opti.set_value(self.z_terr, z_udata.tolist())
267     self.opti.set_value(self.x0, x_cstart)
268     # Set initial guess (move forward/receding, and use last control
269     ↪ input again on integrator):

```

```

263     x_guess = self.integrator_dynamic_solver(prev_sol_X[:, -1],
264         ↪ prev_sol_U[:, -1], self.dt)
265     pred_sol_X = np.hstack((prev_sol_X[:, 1:], prev_sol_X[:, -1][:,
266         ↪ np.newaxis]))
267     pred_sol_X[:, -1] = x_guess
268     pred_sol_U = np.hstack((prev_sol_U[:, 1:], prev_sol_U[:, -1][:,
269         ↪ np.newaxis]))
270     self.opti.set_initial(self.X, pred_sol_X)
271     self.opti.set_initial(self.U, pred_sol_U)
272
273     # Execute solver (if no convergence, follow with last prediction)
274     try:
275         sol = self.opti.solve()
276         sol_X = sol.value(self.X) # length N+1
277         sol_U = sol.value(self.U) # length N
278         self.status.append(0) # nominal status
279     except RuntimeError:
280         print(bcs.WARNING + f"[Sim - {sim_idx}] max. iterations -
281             ↪ fallback to previous solution" + bcs.ENDC)
282         sol_X, sol_U = pred_sol_X, pred_sol_U
283         self.status.append(1) # fallback status
284
285     # Apply integrator to current start for real next uav step:
286     #rx_next =
287     ↪ self.integrator_dynamic_solver(x_cstart, sol_U[:, 0], self.dt)
288
289     return sol_X, sol_U
290
291 def integrator_dynamic_solver(self, x_current, u_applied,
292     ↪ dt_timestep):
293     # Apply integrator to current position and applied control for
294     ↪ next uav step:
295     x_next = self.intg(x0=x_current, u=u_applied, p=dt_timestep)["xf"]
296
297     return np.array(x_next).flatten()
298
299 if __name__ == '__main__':
300
301     # Read Terrain Data:
302     filename = 'terrain_data.pkl'
303     with open('codes/data/'+filename, 'rb') as f:
304         data = pickle.load(f)
305         d_path = np.array(data['d_path'])
306         z_path = np.array(data['z_path'])
307
308     # We define a certain amount of steps upon we can introduce updated
309     ↪ terrain:
310     dist = 100 # updated distance knowledge
311     lenT = int(dist/np.mean(np.diff(d_path)))

```

```

304     print(lenT)
305
306     # Define updated envelope terrain (with the lenT section):
307     x_adata = d_path[:lenT]
308     z_adata = z_path[:lenT]
309
310     # Define state and control limits (for the UAV)
311     x_min = ca.vertcat(10, -10, -ca.pi/2, -ca.pi/3, 0, 0)
312     x_max = ca.vertcat(30, 10, ca.pi/2, ca.pi/3, 5e3, 1e3)
313     u_min = ca.vertcat(-ca.pi/8, 0.1)
314     u_max = ca.vertcat(ca.pi/8, 1.0)
315
316     # Define initial states:
317     x_cstart = np.array([16.8053, 2.5652, 0, 0.1515, 0, 109+3]) # x0 =
    ↪ [16.8053+1, 2.5652, 0, 0.1515, 0, 100]
318     u_cstart = np.array([-0.0526, 0.5835])
319
320     # Initialise MPC class:
321     mpc = MPCsolver(data['d_path'], data['z_path'], lenT, x_min, x_max,
    ↪ u_min, u_max)
322
323     # NMPC solver test to check functionality:
324     mpc.start_mpc_solver(x_cstart, u_cstart, x_adata, z_adata,
    ↪ switch_debug=True)

```