Guidance, Navigation and Control for Interception of Non-Cooperative UAV

Marina Gouveia Moreira

Thesis to obtain the Master of Science Degree in Aerospace Engineering

Supervisor(s): Prof. Rodrigo Martins de Matos Ventura
M.Sc. Endre-Aladár Papp (DLR)

Examination Committee
Chairperson: Prof. José Fernando Alves da Silva
Supervisor: Prof. Rodrigo Martins de Matos Ventura
Member of the Committee: Prof. Alexandra Bento Moutinho

May 2019
Acknowledgments

I first want to thank my advisers for the support and hours spent reviewing the work developed. To Prof. Rodrigo Ventura, for the guidance and expert knowledge, and to Endre Papp for the opportunity of developing the current dissertation at DLR, which was a truly an enriching experience.

I would also like to acknowledge the Intelligent Robots and Systems Group at ISR for the opportunity to use the equipment, especially Prasad and Rute for the patience and for showing me the way around.

To all the friends that were in this journey with me, I would not be the person I am today without all of you. Thank you for the provided distractions from the studies.

Finally, to my family and to João Paulo, thank you for the unconditional love and support throughout this long journey, I hope to continue making you proud.
Resumo

O uso difundido de Drones não licenciados tem levantado preocupações sérias de segurança, como por exemplo, o risco de sobrevoar espaços aéreos restritos, como aeroportos. Portanto, há uma necessidade de neutralização destes veículos que violem restrições de espaço aéreo. Esta tese propõe uma solução para a neutralização de Drones não-cooperativos baseados em interceptação de alta velocidade usando outro Drone. A tarefa de interceptar um alvo não cooperativo, minimizando o tempo de interceptação, é um desafio, já que exige o seguimento do Alvo, ao mesmo tempo que, o envelope de atuação do interceptador é levado aos seus limites. Abordamos esse problema usando uma arquitetura de navegação, guiamento e controlo.

Para navegação, a fim de estimar os estados relevantes do Alvo e do Interceptador, usamos um Filtro de Kalman Extendido como uma solução base e um Filtro de Multiplos Modelos Interactivos (Interactive Multiple Model), melhorando a estimativa do Alvo sob uma incerteza do modelo.

Para guiamento adotamos um método de Navegação Proporcional, expandindo a literatura atual formulada para guiagem de mísseis, para incorporar Drones.

Para o controlo usamos um método de Controlo Vectorial da Propulsão, que incorpora a dinâmica específica do Drone. De modo a considerar o envelope total de atuação, o algoritmo de Navegação Proporcional e Controlo Vectorial é expandido.

A solução proposta foi validada através de simulação usando ROS e Gazebo, e através de testes experimentais, usando um Drone real interceptando alvos virtuais e avaliados com um Sistema de Captura de Movimento preciso.

Keywords: Drone, Intercepção, Drone não-cooperativo, Guiação, Navegação, Controlo.
Abstract

The widespread use of unlicensed, UAVs, has been posing serious safety and security concerns, e.g., the risk of overflying restricted airspaces, such as airports. Therefore, there is a need for the neutralisation of rogue UAVs that are found violating airspace restrictions.

This thesis proposes a solution for neutralising non-cooperative UAVs based on high-speed interception using another UAV. The task of intercepting a non-cooperative Target while minimising the time to interception is challenging, as it demands both Target tracking while pushing the actuation envelope of the Interceptor to its limits. We approach this problem using a navigation, guidance and control architecture.

For navigation, in order to estimate both the Target and Interceptor’s relevant states, we use a Kalman Filter as a baseline solution, and an Interactive Multiple Model Filter which improves Target estimation under model uncertainty. For guidance, we adopt a Proportional Navigation control, expanding the current literature regarding the implementation of classic missile guidance system to UAVs. For Control, we use a Thrust Vectoring Controller method, which incorporates the specific dynamics of the Interceptor. To consider the full thrust envelope of the UAV, the Proportional Navigation and Thrust Vector controller algorithm are expanded.

The proposed solution was validated through simulation using ROS and Gazebo, and through experimental testing, using a real UAV intercepting virtual Targets and evaluated with a precise Motion Capture System.

Keywords: UAV, Interception, Non-Cooperative UAV, Guidance, Navigation, Control.
# Contents

Acknowledgments ................................................................. i
Resumo ................................................................. iii
Abstract ............................................................... v
List of Tables .............................................................. xi
List of Figures .............................................................. xiii
Symbols ................................................................. xx
Acronyms .............................................................. xxi

1 Introduction ................................................................. 1
   1.1 Motivation ............................................................ 1
      1.1.1 Counter UAV Systems ........................................ 1
      1.1.2 Interception of Non-Cooperative Aircrafts .......... 3
   1.2 Objectives .......................................................... 3
      1.2.1 Architecture of the Interceptor UAV ................. 4
      1.2.2 Navigation, Guidance and Control ................. 4
      1.2.3 Simulation ................................................... 5
      1.2.4 Flight Tests Verification and Validation .......... 6
   1.3 Contributions ..................................................... 6
   1.4 Outline ............................................................ 7

2 Literature Review ............................................................ 9
   2.1 Navigation, Guidance and Control ......................... 9
   2.2 Navigation ........................................................ 10
      2.2.1 Particle Filter ............................................. 10
      2.2.2 Extended Kalman Filter ................................... 10
      2.2.3 Interacting Multiple Model Filter ............... 11
   2.3 Guidance .......................................................... 11
      2.3.1 Proportional Navigation ................................... 11
      2.3.2 Augmented Proportional Navigation ............... 11
      2.3.3 PN and APN Comparison ................................... 12
      2.3.4 Modern Solutions .......................................... 12
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.5 Game Theory</td>
</tr>
<tr>
<td>2.4 Control</td>
</tr>
<tr>
<td>2.4.1 PID Control</td>
</tr>
<tr>
<td>2.4.2 Feedback Linearization and Integral-Backstepping Control</td>
</tr>
<tr>
<td>2.4.3 Model Predictive Control</td>
</tr>
<tr>
<td>2.4.4 Thrust Vectoring Control</td>
</tr>
<tr>
<td>2.5 Simulation Tools</td>
</tr>
<tr>
<td>2.6 Model Parameters Identification</td>
</tr>
<tr>
<td>3 Background Theory</td>
</tr>
<tr>
<td>3.1 UAV Modelling</td>
</tr>
<tr>
<td>3.1.1 Interceptor UAV Model</td>
</tr>
<tr>
<td>3.1.2 Target UAV Model</td>
</tr>
<tr>
<td>3.1.3 Reference Frames</td>
</tr>
<tr>
<td>3.2 Extended Kalman Filter</td>
</tr>
<tr>
<td>3.2.1 Prediction Cycle</td>
</tr>
<tr>
<td>3.2.2 Update Cycle</td>
</tr>
<tr>
<td>3.3 Interacting Multiple Model Filter</td>
</tr>
<tr>
<td>3.3.1 State Interaction</td>
</tr>
<tr>
<td>3.3.2 State Estimation</td>
</tr>
<tr>
<td>3.3.3 Model Probability Update</td>
</tr>
<tr>
<td>3.3.4 Estimated States Combination</td>
</tr>
<tr>
<td>3.3.5 IMM Filter Representation</td>
</tr>
<tr>
<td>3.4 Proportional Navigation</td>
</tr>
<tr>
<td>3.5 Thrust Vectoring Control</td>
</tr>
<tr>
<td>3.5.1 TVC Formulation</td>
</tr>
<tr>
<td>3.5.2 TVC Orientation</td>
</tr>
<tr>
<td>3.5.3 TVC Outcome</td>
</tr>
<tr>
<td>4 Approach Formulation</td>
</tr>
<tr>
<td>4.1 Extended Kalman Filter Formulation</td>
</tr>
<tr>
<td>4.1.1 Motion Model</td>
</tr>
<tr>
<td>4.1.2 Measurement Model</td>
</tr>
<tr>
<td>4.2 Interactive Multiple Mode Filter Formulation</td>
</tr>
<tr>
<td>4.2.1 Switching Matrix</td>
</tr>
<tr>
<td>4.2.2 Target Kinematic Models</td>
</tr>
<tr>
<td>4.3 Proportional Navigation</td>
</tr>
<tr>
<td>4.3.1 PN Velocity</td>
</tr>
<tr>
<td>4.4 Thrust Vectoring Control</td>
</tr>
<tr>
<td>4.4.1 Removal of the Position command</td>
</tr>
</tbody>
</table>
# Implementation

5.1 System Architecture
5.1.1 Supporting Blocks
5.1.2 NGC Architecture

5.2 System Mission Control
5.2.1 Mission Modes
5.2.2 Mission States
5.2.3 Mission Events

5.3 Testing Environments
5.3.1 Simulation Environment
5.3.2 Simulation Dynamics
5.3.3 Real Environment
5.3.4 Simulation and Real Environment Comparison

# Identification

6.1 Sensors
6.1.1 Onboard Sensors
6.1.2 Motion Capture System
6.1.3 Noise Tracked Position

6.2 Moments of Inertia

6.3 Static Thrust and Torque Tests
6.3.1 Static Thrust
6.3.2 Static Torque

6.4 Drag
6.4.1 Equilibrium of Forces
6.4.2 Least Square Error Estimator
6.4.3 Drag Estimation Results

6.5 Flight Controller

# Results and Discussion

7.1 EKF & IMM Filter
7.1.1 Target UAV Trajectories
7.1.2 Considerations about the EKF and IMM results
7.1.3 Defensive Manoeuvre
7.1.4 Circular Trajectory
List of Tables

5.1 Mission Modes to NGC algorithms allocation ...................................................... 38
6.1 Onboard Sensor characterization from Hover Flight Test ................................. 44
6.2 Motion Capture System characterization from Hover Flight Test ...................... 44
6.3 Relative position Noise artificially added ............................................................. 45
6.4 Results of the identification of the drag coefficients ........................................... 51
7.1 Experimental trials for PN and TVC testing ......................................................... 60
7.2 Experimental trials for Complete Mission testing ................................................. 66
A.1 Simulators comparison Table .............................................................................. A.82
A.2 Dynamics Comparison ....................................................................................... A.83
B.1 Summary of trials performed for x-Drag Estimation ......................................... B.85
B.2 Summary of trials performed for y-Drag Estimation ......................................... B.88
# List of Figures

1.1 Examples of Counter UAV Systems ................................. 2
1.2 Interception manoeuvre ........................................... 3
1.3 Architecture of the Interceptor UAV ............................... 4
1.4 Morpheus Interceptor UAV ......................................... 6

3.1 Static Forces and Torques that affect the UAV .................... 16
3.2 IMM Filter Algorithm Scheme ..................................... 21
3.3 Proportional navigation diagram in 2D ............................... 22
3.4 Thrust Vector controller illustration ................................. 23

5.1 System Architecture .............................................. 34
5.2 NGC Block Architecture ............................................ 36
5.3 NGC Algorithm Interfaces .......................................... 36
5.4 State Machine of the Mission Control .............................. 37
5.5 Gazebo Simulation Environment .................................... 39
5.6 Experimental Testbed ............................................. 41
5.7 Symmetry between the Experimental Setup and the Simulation Setup 42

6.1 Gazebo screenshot of the UAV 3D Model ............................ 45
6.2 Experimental setup for static thrust constant measurement .......... 46
6.3 PWM output vs. generated Thrust ................................ 47
6.4 Time Contant Up Estimation ...................................... 48
6.5 Time Contant Down Estimation .................................... 48
6.6 Generated Thrust vs. Torque ..................................... 49
6.7 Force equilibrium in constant height flight ......................... 50
6.8 Drag vs Velocity & Linear fit .................................... 51
6.9 Roll angle Flight Controller ..................................... 52

7.1 Defensive Manoeuvre - Trajectory Path ............................ 54
7.2 Defensive Manoeuvre - Model Probability .......................... 54
7.3 Defensive Manoeuvre - Position Estimation and Error ............... 55
7.4 Defensive Manoeuvre - Velocity Estimation and Error ............... 55
7.5 Defensive Manoeuvre - Acceleration Estimation and Error ......................... 55
7.6 Circular Trajectory - Trajectory Path ................................................. 56
7.7 Circular Trajectory - Model Probability .............................................. 56
7.8 Circular Trajectory - Position Estimation and Error ........................... 56
7.9 Circular Trajectory - Velocity Estimation and Error .......................... 57
7.10 Circular Trajectory - Acceleration Estimation and Error ..................... 57
7.11 Random Trajectory - Trajectory Path .............................................. 58
7.12 Random Trajectory - Model Probability ........................................... 58
7.13 Random Trajectory - Position Estimation and Error .......................... 58
7.14 Random Trajectory - Velocity Estimation and Error ......................... 59
7.15 Random Trajectory - Acceleration Estimation and Error .................... 59
7.16 Interception phases in the Experimental Setup .................................. 60
7.17 PN - Line Trajectory - Interception Paths ....................................... 61
7.18 PN - Line Trajectory - Interception Errors ..................................... 61
7.19 PN - Line Trajectory - Distance to Target ....................................... 61
7.20 PN - Line Trajectory - Interceptor Velocity .................................... 62
7.21 PN - Line Trajectory - Interceptor Acceleration ................................ 62
7.22 PN - Circle Trajectory - Interception Paths .................................... 63
7.23 PN - Circle Trajectory - Interception Errors ................................... 63
7.24 PN - Circle Trajectory - Distance to Target ..................................... 64
7.25 PN - Circle Trajectory - Interceptor Velocity .................................. 64
7.26 PN - Circle Trajectory - Interceptor Acceleration ................................ 64
7.27 Modified PN - Interception Path ...................................................... 65
7.28 Modified PN - Distance to Target .................................................... 65
7.29 Modified PN - Interceptor Velocity ................................................. 65
7.30 Modified PN - Interceptor Acceleration ........................................... 65
7.31 Complete Mission - Line Trajectory - Interception Paths ................... 66
7.32 Complete Mission - Line Trajectory - Interception Errors .................. 66
7.33 Complete Mission - Line Trajectory - Distance to Target ................... 67
7.34 Complete Mission - Line Trajectory - Interceptor Velocity .................. 67
7.35 Complete Mission - Line Trajectory - Interceptor Acceleration .............. 67
7.36 Complete Mission - Circle Trajectory - Interception Errors .................. 68
7.37 Complete Mission - Circle Trajectory - Interception Errors .................. 68
7.38 Complete Mission - Circle Trajectory - Distance to Target .................. 69
7.39 Complete Mission - Circle Trajectory - Interceptor Velocity .................. 69
7.40 Complete Mission - Circle Trajectory - Interceptor Acceleration .............. 69
7.41 Complete Mission - Random trajectory - Interception Paths ................ 70
7.42 Complete Mission - Random trajectory - Interception Errors ................ 70
7.43 Complete Mission - RT - Distance to target, Interceptor Velocity and Acceleration 71
7.44 Yaw Angle Control for Circle Trajectory .............................................. 72

A.1 Scheme on Simulation target ................................................................. A.79
A.2 Structure of the Simulation and interaction with other blocks ...................... A.82

B.1 Drag Test $v_x = 0.5 \text{ [m/s]}$ ................................................................. B.86
B.2 Drag Test $v_x = 1 \text{ [m/s]}$ ................................................................. B.86
B.3 Drag Test $v_x = 1.5 \text{ [m/s]}$ ................................................................. B.86
B.4 Drag Test $v_x = 2 \text{ [m/s]}$ ................................................................. B.87
B.5 Drag Test $v_x = 2.5 \text{ [m/s]}$ ................................................................. B.87
B.6 Drag Test $v_x = 3 \text{ [m/s]}$ ................................................................. B.87
B.7 Drag Test $v_x = 3.5 \text{ [m/s]}$ ................................................................. B.88
B.8 Drag Test $v_y = 1 \text{ [m/s]}$ ................................................................. B.88
B.9 Drag Test $v_y = 1.5 \text{ [m/s]}$ ................................................................. B.89
B.10 Drag Test $v_y = 2 \text{ [m/s]}$ ................................................................. B.89
B.11 Drag Test $v_y = 2.5 \text{ [m/s]}$ ................................................................. B.89
B.12 Drag Test $v_y = 3 \text{ [m/s]}$ ................................................................. B.90
Symbols

Greek Symbols
\( \alpha \)  Line of Sight angle
\( \Delta \)  Interval or Variation
\( \Lambda \)  Likelihood Function
\( \mu \)  Mode Probability
\( \eta \)  Gain
\( \omega \)  Angular velocity vector
\( \psi \)  Roll angle
\( \psi \)  Yaw angle
\( \tau \)  Time Constant
\( \theta \)  Pitch angle
\( \Theta \)  Attitude vector

Mathematical Operators
\( \circ \)  Hadamard Product
\( \otimes \)  Hamilton Product
\( \times \)  Cross Product

Roman Symbols
\( 0 \)  Zero matrix
\( a \)  Acceleration vector
\( A_{bla}, a_{bla} \)  Blade Flapping Coefficient Matrix, Blade Flapping coefficient
\( b \)  Constant
\( C_M \)  Adapted Moment constant
\( C_T \)  Adapted Thrust constant
\( e \)  Error vector
\( f \)  State function
\( F \)  Force vector
\( F \)  Transition Model Matrix
\( g \)  Gravity acceleration vector
\( h \)  Observation function
\( H \) Observation model matrix
\( I \) Identity matrix
\( J \) Inertia matrix
\( k \) Discretized time instance
\( k \) Gain vector
\( k_{par} \) Parasitic drag coefficient
\( K \) Kalman Gain matrix
\( l \) Motor Distance to center of Body reference frame
\( m \) Mass
\( M, M \) Torque vector and Torque magnitude
\( N \) Number of Samples
\( N_m \) Number of Models
\( N_{PN} \) PN Constant
\( p \) Position vector
\( P \) Error covariance matrix
\( q \) Covariance Gain
\( q \) Quaternions vector
\( Q \) Covariance of the process noise
\( R \) Covariance of the observation noise, Rotation matrix
\( R_{AB} \) Rotation matrix from A to B
\( S \) Innovation Covariance matrix
\( t \) Time
\( T, T \) Thrust vector and Thrust magnitude
\( u \) Input vector
\( v \) Velocity vector
\( v, w \) State and Observations independent random process vector
\( x \) State vector
\( x, y, z \) With respect to the \( x, y \) and \( z \) axis
\( y \) Observations vector

**Subscripts and Superscripts**

\( B, W \) With respect to the Body and World reference frame
\( cmd \) Commanded reference
\( CA, CV, CT, TA \) Constant Acceleration, Velocity, Turn and Thrust Acceleration Models
\( D \) Drag
\( I \) Interceptor
\( LOS \) Line of sight
pwm In PWM units

PN Relative to Proportional Navigation

r Relative

T Target

' Modified

T Transpose
Acronyms

**APN**  Augmented Proportional Navigation
**CAD**  Computer-Aided Design
**DLR**  German Aerospace Center - Deutsches Zentrum für Luft- und Raumfahrt e.V.
**EKF**  Extended Kalman Filter
**GNC**  Guidance, Navigation and Control
**GNSS**  Global Navigation Satellite System
**GPS**  Global Positioning System
**IMM**  Interacting Multiple Model Filter
**IMU**  Inertial Measuring Unit
**ISR**  Intelligent Robots and System
**IST**  Instituto Superior Técnico
**LOS**  Line of Sight
**MCS**  Motion Capture System
**MPC**  Model Predictive Control
**MPSE**  Multiple-Pursuer-Single-Evader
**MSP**  Multiwii Serial Protocol
**NGC**  Navigation, Guidance and Control
**PF**  Particle Filter
**PID**  Proportional, Integral and Derivative
**PN**  Proportional Navigation
**PWM**  Pulse-Width Modulation
**RC**  Radio Controlled
**ROS**  Robot Operating System
**SPSE**  Single-Pursuer-Single-Evader
**TVC**  Thrust Vectoring Control
**UAV**  Unmanned Aerial Vehicle
Chapter 1

Introduction

This introductory Chapter will explain the main goals of this thesis and shed light on how they will be achieved. We will start by mentioning the motivations behind this work, in Section 1.1, and then enumerate the objectives in Section 1.2. In Section 1.3 we will enumerate the contributions of this work so as to achieve the mentioned objectives. Finally, in Section 1.4 we will lay out the outline of the thesis, explaining how it is structured.

1.1 Motivation

Due to the increasing number of Unmanned Aerial Vehicles (UAV) users, these are gradually becoming an increasing threat to safety and security. Some past occurrences can be found in [1], where it is shown additionally that the occurrences are escalating. Just last year, an incident in Gatwick airport\(^1\) affected 110,000 passengers. Threats range from an unharmful uninformed amateur operator breaching airspaces, to deliberately planned breaches with the intent of causing harm. Learning to operate a small scale UAV only takes a few hours but this ease of use, however, imposes the risk that beginner operators are not sufficiently qualified with respect to the airspace regulations.

1.1.1 Counter UAV Systems

To overcome the mentioned threats, a few counter UAV Systems already exist, which attempt to neutralize the unauthorized UAV. These Systems can be categorised into two groups according to the containment method: electromagnetic, meaning non invasive solutions, or mechanical, meaning evasive solutions.

Electromagnetic solutions attempt to either jam the communication between the unauthorized UAV and the remote operator, or to spoof its GPS receiver. One example of a jamming System is the IDS’s Black Knight Radar\(^2\). This System incorporates optical and thermal sensors to assist in the identification of an unauthorized UAV in flight. Once it is detected and identified by the operator, the jammers can be

---

\(^1\)https://www.bbc.com/news/uk-england-sussex-46623754
\(^2\)https://www.idscorporation.com/pf/black-knight
used to either disrupt the unauthorized UAV's navigation system or data-link connection to the remote pilot, resulting in a forced landing or even a crash. Jamming signals can be transmitted for GPS signals and RC/video-link data at 2.4 or 5.8 [GHz]. When placed in an Airport or its vicinity, the main drawback of the electromagnetic solution is the risk of collateral damage that can disrupt the navigation or data-link connection of authorized participants as well.

Mechanical solutions attempt to impede the progress of the unauthorized UAV with the help of objects, such as nets. These can be attached to an interceptor UAV or launched from the ground. The Tokio Police Department\(^3\) has developed an UAV with the capability to carry a large net. A remote operator then attempts to manually operate the interceptor UAV such that the unauthorized UAV gets entangled in the net. Another example is the SkyWall 100\(^4\) developed by OpenWorks Engineering. SkyWall 100 is a ground based solution with a 10 [kg] shoulder-mounted pressurised gas-powered net launcher that, according to the manufacturer, provides a range of 100 [m] with a recharge time of 8 [s].

**Shortcomings of available Counter UAV Solutions**

Although all solutions mentioned earlier intend to avoid the intruding threats, they present nevertheless some shortcomings. A major one is the necessity of a human operator, relying on human skill to neutralise the intruder UAV, being susceptible to human error. Another shortcoming is the limited range of these counter measures. Considering all the possible areas that one might want to protect, such as airports, open-air events or other restricted areas, large investment would have to be made.

---

\(^3\)https://www.telegraph.co.uk/technology/2016/01/21/tokyo-police-are-using-drones-with-nets-to-catch-other-drones

\(^4\)https://openworksengineering.com/skywall
1.1.2 Interception of Non-Cooperative Aircrafts

To overcome the limitations of the available technologies, and to develop a System that combines detection and neutralisation in order to provide greater flexibility and efficiency, the DLR Institute of Flight Systems is working on a project named “Interception of Non-Cooperative Aircrafts”, or “Drohnenabwehr” in German. The Project goal is to develop a System consisting of an UAV, the Interceptor, capable of detecting, intercepting and neutralising the threat of an unauthorized UAV, the Target. The main requirement is that the Interceptor should have the capability to detect, track and intercept a Intruder with the use of its onboard Flight Controller and sensors in real-time. The interception of the Intruder, in this stage of the Project, should occur at the maximum speed possible, in order to inflict maximum damage to the Intruder during a head on collision with the Intruder. Such an interception manoeuvre is illustrated in Figure 1.2.

![Figure 1.2: Interception manoeuvre](image)

1.2 Objectives

The main objective present is to design, build, and validate a working prototype experimentally given all the performance constraints. This thesis main goal is thus to develop the algorithms for the interception manoeuvre between the Interceptor and Intruder UAVs, as well as all the supporting systems to the main interception Mission.

This means that not only the Interception algorithms need to be investigated, implemented and tested, but also testing workbenches need to be specified, and the Interceptor UAV Model parameters need to be identified.

We will now go though the main components of the system so as to and identify the objectives for each stage. One must consider that no previous work within this project has been previously performed toward UAV interception. Therefore all proposed solutions will need to be developed from scratch. Due to the fact that an interception is expected, the Intruder will be further referred as Target.

---

5https://www.dlr.de/ft/en/desktopdefault.aspx/tabid-1336/1837_read-32330/
1.2.1 Architecture of the Interceptor UAV

The simplified Architecture of the Interceptor UAV can be seen in Figure 1.3. Only the GNC Algorithm block is this thesis’s responsibility. The Tracking Algorithm block addresses the problem of acquiring the Target’s relative position to the Interceptor. This algorithm is being developed separately, and throughout this thesis, all Target position acquisition will be manufactured, so as to override this block. The Flight Controller is defined \textit{a priori}, with open-source firmware, and new development is not this thesis' responsibility. However, some modification to the firmware, regarding interface protocol and parameter tuning responsibilities were done as to allow experimental testing. The UAV, Actuators and Sensors were also chosen \textit{a priori}, and this work approaches them only in the user’s perspective.

![Figure 1.3: Architecture of the Interceptor UAV](image)

As per project requirements, of the GNC algorithms will be implemented on a commercial on-board computer, a Raspberry Pi 3 B+ \footnote{https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/}, running the algorithms in a ROS framework \footnote{https://www.ros.org/}. To do the aggressive interception path, the manoeuvre needs to be very responsive and fast. Thus, considering the hardware, the algorithms selected need to be lightweight, allowing for a high update frequency. This limits the development because not only the GNC algorithms need to be implemented in the selected hardware, but also the Tracking Algorithm, with no available information on processing/memory usage.

1.2.2 Navigation, Guidance and Control

The three topics of Guidance, Navigation and Control are the most important ones in order to operate an UAV, either manually or automatically for non-interception or interception manoeuvres. Because this is an interception problem, firstly the Navigation algorithm is performed, then, according to its results, a Guidance strategy is defined, sending the commands to the Control. Therefore, throughout this thesis, instead of referring to Navigation, Guidance and Control as GNC, when the order is relevant we refer to it as NGC. This work will be mainly focused on these three topics, not necessarily with the same emphasis on each one.

Navigation

The Navigation algorithm is responsible for converting the information from the sensors and tracking algorithm, into an estimation of the state of the Interceptor and Intruder UAVs. The available Interceptor
sensors are a GPS and Magnetometer module, and an IMU equipped with a Gyroscope, an Accelerometer and Magnetometer. The attitude values are also available, as processed by the Flight Controller, and these are the values that will be used for the attitude estimation instead of the Magnetometer, Gyroscope and Accelerometer. The GPS will provide position and velocities estimates. The Tracking algorithm will provide relative coordinates of the Intruder with respect to the Interceptor.

The Navigation algorithm should output estimates of the attitude, position, velocity and acceleration of the Interceptor, as well as the the position, velocity and acceleration of the Target. The algorithm should be precise enough to allow for a successful interception.

**Guidance**

The Guidance algorithm is in charge of guiding the Interceptor on a collision path with the Intruder. To do so, it uses both the Interceptor and Intruder UAV information available from the Navigation. The Interceptor and Target's relative position, velocity and acceleration are used, so as to create the intercepting commands.

**Control**

The Controller algorithm, using the Guidance algorithm's commands and the Navigation algorithm's state estimate, is responsible for the interception path execution.

In order to fulfil the interception path commands, the Control algorithm will be responsible for sending the correct instructions to the Flight Controller regarding Thrust and Attitude. The Control algorithm will translate the output of the Guidance and Navigation algorithm into autopilot instructions, making sure the reference is followed through a feedback loop. The command references input to the Control algorithm can be the desired attitude, position, velocity and acceleration.

The Controller's performance should be good enough to perform the high acceleration manoeuvres. This means that it will need to estimate and counteract the aerodynamic effects that emerge.

**1.2.3 Simulation**

The Simulation of the mission plays an important role in this Thesis for two reasons. Firstly, a Simulation environment will help to achieve a sustainable development and validation workbench for the algorithms in an early stage before hardware implementation. Due to the nature of the Mission, it is not feasible to destroy the UAV testbed every time there is a successful test. Secondly, due to limited test resources namely: no Motion Capture System available throughout development; outside test flights depend on personnel and weather.

For these reasons, investing on creating a comprehensive simulation of the overall System shown in Figure 1.3 is the best step for the development process to run smoothly, given these resource restrictions. The Simulation should be able to thoroughly mimic the actual UAV setup, due to the expected aggressive manoeuvres.
Because the GNC implementation uses the ROS framework, the Simulations should too, enabling easy interfacing, where all the simultaneous progress are incorporated with minimal effort. Given that the Simulation is implemented in ROS, the widely used Gazebo simulator\(^8\) will be the physics engine.

**Model Parameters Identification**

Throughout this thesis development, the Interceptor UAV chose, is based on a Racing Quadcopter frame. The reasons behind choosing a smaller frame are to have a nimble, lightweight and cost effective platform which can be used to perfect the different software components and flight behaviour. The Interceptor UAV is named Morpheus and can be seen in Figure 1.4. It has a 270 [mm] carbon fibre frame, weighs approximately 0.7 [kg], reaches a top speed of 138 [km/h], and has a maximum flight time of 6 [min] depending on flight conditions.

The Model Identification of the UAV and sensors parameters shall be performed, in order to obtain an accurate representation of the vehicle in the simulation environment. The Identification will approach all the aerodynamic characteristics of the UAV, as well as sensor noise.

![Figure 1.4: Morpheus Interceptor UAV](image)

**1.2.4 Flight Tests Verification and Validation**

Finally, the last topic of the objectives is that the algorithms should be tested through both Simulation and Flight Tests. Firstly, verification of the algorithms shall be done through a comprehensive Simulation, testing several flight scenarios. Moreover, experimental Flight Tests should be performed as to assess whether the Simulation is accurate as well as to assess the success of the executed Mission. All experimental tests will be performed at the Intelligent Robots and System Lab, Instituto Superior Técnico\(^9\).

**1.3 Contributions**

In order to fulfil the objectives and requirements mentioned in the previous Section 1.2, we now introduce an enumeration of methods, that will be implemented and analysed within this thesis. As we will see later, these methods are the result of the conclusions taken from the Literature Review done in Chapter 2, and they can be grouped as follows:

\(^8\)http://gazebosim.org/

\(^9\)www.isr.tecnico.ulisboa.pt
• **Guidance**: Extension of the classical interception algorithms Proportional Navigation; Implementation of a PID Yaw Control.

• **Navigation**: Implementation of an Extended Kalman Filter for Estimation, with joint Interceptor and Target states; Implementation of an IMM Algorithm, taking into consideration the uncertainty on the Target model.

• **Control**: Implementation of a Thrust Vectoring Controller that considers external disturbances and UAV dynamics; Extension of the Thrust Vectoring Controller method, to use maximum thrust envelope.

• **Simulation**: Implementation of a Simulation in Gazebo based on RotorS Simulation structure.

• **Identification**: Identification of the Interceptor UAV model parameters such as the Inertia, Static Thrust and Momentum, and Drag parameters.

• **Experiments**: Performance of Flight Experiments, in a Simulated environment and an Experimental environment where the algorithms were validated.

### 1.4 Outline

The remainder of the thesis is structured as follows:

Firstly, Chapter 2, gives a brief explanation of other studies that have been conducted in the same research field, when it comes to all the methods implemented herein. The decision on which algorithms were implemented are based on this review.

Following, the Background Theory related with the chosen GNC algorithms is shown in Chapter 3. This Chapter acts as a base of knowledge obtained from the Literature Review.

With the Background Theory established, in Chapter 4 we will explore deeper how the selected GNC algorithms were tailored to this project, modified and extended with respect to the original definition.

Building the bridge between theory and practice, in Chapter 5 it is explored how the entire System developed throughout is constructed and how the subsystems interact. The NGC Algorithm architecture will be described in detail, as well as its Interfaces, the Simulation, and Experimental setup.

In Chapter 6, an Identification of the entire system is performed, compiling theory and practice. The parameters determined in this Chapter will be useful to personalize the Simulation and the Controller implemented.

Towards the end, the Results and Discussion in Chapter 7, will summarize all the Navigation, Guidance and Control results, comparing the Simulation with the Experimental data obtained.

Finally, the Conclusions in Chapter 8, evaluates the thesis as a whole, summarising the goals achieved, and discussing what could be improved as future work, in next steps to the Project.
Chapter 2

Literature Review

In this Chapter, we will present a Literature Review with the purpose of revealing the possible methods to solve the problem at hand. In Section 2.1, we will create a base of knowledge regarding State-of-the-Art for NGC algorithms, which will be further discussed in Sections 2.2, 2.3 and 2.4, respectively. In Section 2.5, we will investigate what are the available Simulation Tools that allow us to obtain artificial results and compute an approximation of the real environment. Finally, in Section 2.6 we will discuss the model of the Interceptor UAV to be used in simulation and the necessary parameters to be identified, as well as identification methods.

To the best of authors knowledge, the problem of Interception of non-cooperative Targets using an autonomous UAV is a novelty.

2.1 Navigation, Guidance and Control

The interception problem can be separated into three different modules: Guidance, Navigation and Control (GNC). These three stages are usually separated in such a way that the selected algorithms can be implemented and tested separately as part of a modular system. This will add robustness for future implementation and upgrades, and also add manageability in terms of implementation.

In simple terms, according to [2]:

1. **Guidance** is the function that defines the current or future desired states. It is the establishment of the desired path for the system to follow.

2. **Navigation** is the function that determines the current or future estimated states from the measured data. It is knowing where the system is with respect to its environment.

3. **Control** is the function that derives control commands to match the current or future estimated state with the desired state. It is about applying changes to the system to follow the desired path in the current environment.
In the specific case of the interception problem, the Navigation algorithm will be firstly executed in order to estimate the current state of the Interceptor. Based on the current state, the Guidance algorithm will define the trajectory/commands, in order to intercept the Target. Finally, the control algorithm will make sure the guidance references are being followed. As previously stated, this differs from a normal GNC problem where the trajectory is already defined and the execution order is Guidance, Navigation, Control. We will refer to the problem as NGC (Navigation, Guidance and Control), as to clearly differentiate the two cases, and state that this is an Interception problem.

Choosing the best algorithm combination to achieve the requirements is then crucial. In the following Sections 2.2, 2.3 and 2.4, each stage will be analysed individually, having into consideration factors as real-time capabilities and accuracy of the solutions. Based in these criteria, different solutions will be defined.

2.2 Navigation

In this thesis, the Navigation algorithm will be responsible for the estimation of both the Interceptor and Target states. The Navigation will solves a filtering problem, which consists of estimating the internal states of a dynamic system, using partial observations, subjected to perturbations such as sensor noise, as well as uncertainty in the dynamic models. Based on the problem, different approaches emerge.

2.2.1 Particle Filter

For the state estimation of the UAV, there are several possibilities for non-linear systems. A Particle Filter (PF) method is one of the solutions, as shown in [3]. Particle Filter methods are a set of Monte Carlo algorithms used to solve filtering problems mainly used in signal processing.

Its main benefit is that no linearisation of the non-linear system is needed. However, due to the non-linearisation, the cost of the required computational power is increased.

2.2.2 Extended Kalman Filter

For systems in which computational power is a scarce resource, one solution widely used is the Extended Kalman Filter (EKF), which works by combining the sensors measurements with the predicted state. The predicted state is based on the previous state and a system model. Both solutions are combined through a weight function based on the uncertainty. EKF solutions are already being used commercially \(^8\), with new methods arising such as [4]. One must also consider that due to the asynchronous nature of the sensors, the EKF can be adapted in order to improve results as in [5].

Nevertheless, the EKF has the constraint that it is not optimal for non-linear systems and there is no stability guarantee. If the initial state estimate is not correct, the filter can diverge by underestimating the covariance matrices [6]. Having stated this, the extended Kalman filter can give reasonable performance.

\(^8\)http://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html
and is arguably the *de facto* standard in navigation systems. To minimize these effects, a matching step in the EKF can be added, filtering destabilizing measurements, making the filter more stable [7].

### 2.2.3 Interacting Multiple Model Filter

The trajectory of the Target UAV is very uncertain, thus complicating its model definition. Since the EKF relies on the model accuracy, to improve robustness, an aggregation of several EKFs can be used, using an Interacting Multiple Model Filter (IMM) [3, 8–10]. The IMM works by considering several approximated system models, and fusing the estimation results in a weighted Markov Chain. Although the IMM algorithm can also be implemented with PFs as described in [3], it is still computationally demanding and not suitable for real-time estimation.

This method as proven to provide robust results for high manoeuvrable targets with high uncertainty such as missiles and aircrafts [8, 9]. As this method was only tested on missiles and aircrafts with limited movement, the question arises on whether the additional models will be useful in improving the estimation for the Target state in a broader flight envelope.

### 2.3 Guidance

To approach this stage, solutions for the Guidance of Interception problems were considered.

#### 2.3.1 Proportional Navigation

Air-to-air interception problems were first considered when deriving methods for Homing Missile Systems. Homing Missile Systems are self-propelled missiles that can be guided while on flight. In order to guide these missiles, several techniques have been developed over the years. One of the most widely implemented is Proportional Navigation (PN) [11].

The PN method is still used nowadays for such missile applications [12], as it is reliable and robust. However, the method only considers constant speed Targets, and when intercepting manoeuvring Targets with variable speed, interception is not guaranteed.

#### 2.3.2 Augmented Proportional Navigation

Augmented Proportional Navigation (APN), a derivative of PN, includes the target acceleration information, in order to allow interception of manoeuvring targets [13]. To obtain the acceleration from the position measurements, an acceleration estimator needs to be added. This estimation will induce delay, because it relies on previous measurements to predict the current state. It will also amplify measurement noise as it will be differentiating the measurement. For these reasons, sensor quality is important. If there is high delay or noise present in the System, it will lead to a drop in the accuracy of the output, thus, potentially performing worse than PN [14].
2.3.3 PN and APN Comparison

In [15], simulation experiments are conducted where it is concluded that APN should perform better than PN even with delay present for manoeuvring Targets, however the influence of noise in this application is not discussed. In [16], PN and APN were used to solve a path following problem. The goal was to follow a road using an on-board camera. APN showed better path following capabilities due to static state error rejection. Nevertheless, results from this research can not be extrapolated due to the fact that, the requirements of the system were vastly different from the one at hand.

With the increasing number of robotic vehicles, these algorithms have also been adapted and implemented for a range of other applications besides missiles. In the field of Robotics, [17] uses PN algorithms to grasp objects. In the special case of UAV interception a research was conducted in which PN was successfully implemented and tested in a virtual environment [18]. The virtual environment was 2D in which the evader moved in a straight line.

It is still unclear if these algorithms can perform well in a real environment with an unpredictable moving Target. Therefore, implementing PN and APN in a real UAV setup is still a novelty.

2.3.4 Modern Solutions

Modern Guidance techniques for intercepting have been developed focusing on the limitations and capabilities of small scale multi rotor unmanned aircraft, by using a physical model. These algorithms run optimisation equations considering flight dynamics, flight time, final state and energy to find the optimal path. Implementation of such techniques were performed in [19] using Pontryagin’s minimum principle for efficient trajectory generation. In [20] a polynomial trajectory with a certain degree is generated such that the final conditions match the goal. These experiments address the challenge of juggling/hitting a tennis ball. These studies are similar to the problem at hand, because they solve the problem of getting to a certain predicted position with a specific final State. Therefore, these solutions can be modified in such way that the final position is the Target, and the final State includes a high impact velocity.

Nonetheless, the obtained results in [19] and [20] are precise and computationally lightweight, because motion capture systems are used, making all dynamic values accessible with millimetre precision, rising questions on whether it is possible to perform these algorithms in an outside environment. Another constrain of the proposed methods is that the physical model of the target is well defined and relatively deterministic (e.g. physical model of a ball which has no actuation). This allows for reliable predictions on where the target will be in the interception point, increasing performance. These techniques were never implemented and tested for evading Targets.

2.3.5 Game Theory

Taking a different approach to the problem and considering that the Target is trying to evade the Interceptor by performing the optimal action to avoid impact, one must consider the field of Game Theory. More specifically for our application, the Single-Pursuer Single-Evader problem (SPSE). This problem was first formulated by [21] also known as the “homicidal chauffeur problem”. This problem is a classic
example of a differential game played in continuous time in a continuous state space, with its solution consisting on a set of optimal responses based on the relative pose between the target and the evader.

Such strategies can be adapted to UAVs such as performed in [22] where the Multiple-Pursuer Single-Evader (MPSE) problem was solved in a simulation. These type of approaches show promising results for Multi-Player pursuit, allowing for more efficiency and coordination.

In the research conducted by [23], it has been shown that if there is a constant flow of Target position data, the solution to the SPSE problem is approximately the output of PN in the conditions where: there are linear dynamics; a defined final time for interception; small deviations from zero state and unbounded actuation, with quadratic cost.

2.4 Control

Having references from the Guidance stage, it becomes necessary for the Interceptor UAV to fulfil them, with the aid of a controller in the Control Stage.

2.4.1 PID Control

A PID controller is a loop feedback mechanism, where the error, difference between the desired and measured set-point, goes through a correction based on proportional, integral, and derivative terms.

These type of controllers are typically implemented by standard PID controllers [24–26]. In these approaches the dynamic characteristics of the quadrotor are disregarded, with the underlining thought that the controller will be able reject automatically the disturbances. However, as the gains are fixed, it lacks robustness against external disturbances and nonlinear dynamics [27].

2.4.2 Feedback Linearization and Integral-Backstepping Control

The Feedback Linearization and Integral-Backstepping techniques can also be implemented for UAV controllers [28, 29]. These technique can take in factors such as the dynamics effects because, although the controller is linear, controller output is subjected to a non-linear transformation. Moreover, the stability of the controller can be proven through Lyapunov methods [30]. However, in the considered works, this is applied in a simulation environment. It also has the constraint that linearized systems are only valid for conditions where only low velocities are considered.

2.4.3 Model Predictive Control

Model predictive control (MPC) is an advanced method used to control a process while satisfying a set of constraints. Model Predictive Control Algorithms have been implemented in [31–33]. As applying Model Predictive Control is computationally intractable on the embedded hardware typically available on a quadrotor aerial robot, the referred methods use simplifications such as linearization [31], and offline simulation computation [32, 33].
2.4.4 Thrust Vectoring Control

Finally, another interesting approach to the problem is depicted in [34–36], where Thrust Vectoring Control (TVC) is applied to UAVs. This type of controller can take into account the non-linearity of the UAV dynamic effects, and other disturbances such as wind, while following an aggressive trajectory. Moreover, with this method, the quality of the images taken with an onboard camera improves [27]. In [36], it is proven that this control strategy results in a faster response than a classic PID controller, as it considers the effects directly instead of rejecting them slowly through an integral term.

2.5 Simulation Tools

For the Simulation of UAV Quadcopters, two main options arise, the Hector Quadcopter [37] and RotorS [38].

The Simulators Hector Quadcopter [37] and RotorS [38], both contain packages related to modeling, control and simulation of quadrotor UAV systems. Both are open-source, powerful and flexible, implemented in the ROS environment with the Gazebo simulator. Moreover, both UAV simulators are conceptually very similar, providing multi-rotor examples with the possibility of customisation, with simulation of onboard sensors such as an IMU and GPS, with some example controllers, and basic worlds.

Because of the similarities, a deeper analysis comparing both Simulations is made in Appendix A, where evaluation topics are defined and a score is calculated. The simulation environment with the upper hand was the RotorS because, even though it does not have as many features, its architecture allows for easy modifications and faster adaptations.

2.6 Model Parameters Identification

To define the system dynamics, in [39] and [24], equations of motion are defined alongside the aerodynamic effects, and procedures for capturing static thrust and momentum measurements are defined. In [40] a Nonlinear Dynamic Model for a High Performance UAV is presented. It is taken into account the non-linear equations of motion considering the motor-rotor system and the drag-like effects.

The motor-rotor system parameters are determined through experiments as explained in [41]. The drag-like effects are such as Blade Flapping, Induced Drag, Translational Drag, Profile Drag and Parasitic Drag. It is seen that the Parasitic Drag is only important at speeds higher than \(10 \, [\text{m/s}]\). The drag-like effects described in [40] are measured in [36] with explicit procedures and results.

In [42], different testbeds for momentum inertia, static thrust and static momentum are defined and measured.
Chapter 3

Background Theory

In this Chapter the Background Theory regarding the concepts necessary to understand the remainder of this thesis are defined. In Section 3.1 the Model of both the Interceptor and Target UAVs will be derived. In Section 3.2, the Extended Kalman Filter (EKF) is described, in order to handle Navigation stage. The EFK is then expanded in Section 3.3 through a Interactive Multiple Model which uses different EKFs with different models for the Target UAV in a Markov Chain. In Section 3.4 the concepts of Proportional Navigation and Augmented Proportional Navigation, that are implemented as the Guidance method, are introduced. Finally, in Section 3.5 the Thrust Vectoring Controller method is discussed, as the Control method for the Interceptor UAV.

3.1 UAV Modelling

In this Section the Model of the UAV will be defined. It is important to do so early on because the concepts and definitions used herein will be used throughout the thesis.

3.1.1 Interceptor UAV Model

To model the Interceptor UAV we will use a Non-Linear Dynamics Model based on the model described in [43]. This approach is preferred because the usage of quaternions eliminate undesired effects such as gimball-lock or discontinuities, common problems when using traditional approaches with Euler angles. In [43] the state vector $x$ is given by

$$x = \begin{bmatrix} p_i^T & v_i^T & q_i^T & \omega_i^T \end{bmatrix}^T,$$

(3.1)

where $p_i$, $v_i$ and $q_i$ are the position, velocity and attitude vectors of the Interceptor w.r.t. the inertial frame, and $\omega_i$ denotes the Interceptor’s angular velocity w.r.t. the body-fixed frame.
Taking \[43\] and adding the drag Force, the state function \( f \) becomes

\[
\dot{x} = f(x, u) = \begin{bmatrix} v_I \\ q_I \otimes \frac{(T/m)}{\otimes}q_I^* + \mathbf{g} + \left( \frac{F_D}{m} \right) \\ \frac{1}{2}(q_I \otimes \omega_I) \\ J^{-1}(\tau - \omega_I \times J\omega_I) \end{bmatrix},
\]

(3.2)

where \( \otimes \) is the Hamilton product, \( \times \) is the cross product, \( q^* \) is the conjugate quaternion, \( T \) and \( \tau \) are the Thrust and Torque generated by the motors applied in the body-fixed frame, \( m \) and \( J \) are the Interceptor’s mass and inertia matrix, \( \mathbf{g} \) is the gravity vector and \( F_D \) is the drag Force.

The static Thrust and Torque used in the previous Model are displayed in Figure 3.1.

![Figure 3.1: Static Forces and Torques that affect the UAV](image)

According to the Figure 3.1, the Force and Torque, where \( \tau = (\tau_x, \tau_y, \tau_z) \), are defined as

\[
\begin{bmatrix} T \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 \\ -l_yT_1 & l_yT_2 & l_yT_3 & -l_yT_4 \\ -l_xT_1 & -l_xT_2 & l_xT_3 & l_xT_4 \\ -M_1 & M_2 & -M_3 & M_4 \end{bmatrix},
\]

(3.3)

where \( l_x \) and \( l_y \) are the motor’s distance to the center of the body-fixed-frame.

Because the Thrust and Momentum generated are not calculated by the Flight Controller, this definition was modified to use the Pulse Width Modulation (PWM) instead, since it is available as a measurement. Using the relationship equation in \[44\], we have

\[
T_i = C_T u_{pwm_i}, \quad M_i = C_M u_{pwm_i},
\]

(3.4)

with \( C_T \) and \( C_M \) the adapted thrust and momentum constants, which map the input to the generated force and momentum respectively. The input \( u_{pwm_i} \) is the PWM value sent to the ESC \( i \) that controls the Motor \( i \). This correlation was validated experimentally for the referred UAV. With this modification,
the Force and Torque become
\[
\begin{bmatrix}
    T_x \\
    M_x \\
    M_y \\
    M_z
\end{bmatrix}
= 
\begin{bmatrix}
    C_T & C_T & C_T & C_T \\
    -l_yC_T & -l_yC_T & l_yC_T & l_yC_T \\
    l_zC_T & l_zC_T & -l_zC_T & -l_zC_T \\
    C_M & -C_M & -C_M & C_M
\end{bmatrix}
\begin{bmatrix}
    u_{PWM_1} \\
    u_{PWM_2} \\
    u_{PWM_3} \\
    u_{PWM_4}
\end{bmatrix},
\tag{3.5}
\]
where the input vector \( u \) is given by \( u = [u_{PWM_1} \ u_{PWM_2} \ u_{PWM_3} \ u_{PWM_4}]^T \).

### 3.1.2 Target UAV Model

The Target UAV model is more difficult to model, as we do not know how it will behave. However, one must consider that, according to the specific manoeuvre being executed, a different model might be better well suited in a certain period of time. For the Target UAV kinematic model, the question arising is which model to use for the state estimation, given the uncertainty. In Section 7.1, a general constant acceleration model will be considered in an EKF and compared with an IMM filter implementation, where other possible models are incorporated.

### 3.1.3 Reference Frames

In this thesis, the body axis are defined as in Figure 3.1, where the Euler angles are defined as: \( \phi \) is the positive rotation around the \( x \) axis; \( \theta \) is the positive rotation around the \( y \) axis; and \( \psi \) is the positive rotation around the \( z \) axis.

The Rotation Matrix from the Body reference frame to the World reference frame is
\[
R_W^B = 
\begin{bmatrix}
    \cos \theta \cos \psi & \sin \theta \sin \phi \cos \psi - \cos \sin \psi & \sin \theta \cos \phi \cos \psi + \sin \phi \sin \psi \\
    \cos \theta \sin \psi & \sin \theta \sin \phi \sin \psi + \cos \phi \cos \psi & \sin \theta \cos \phi \sin \psi - \sin \phi \cos \psi \\
    -\sin \theta & \cos \theta \sin \phi & \cos \theta \cos \phi
\end{bmatrix},
\tag{3.6}
\]

### 3.2 Extended Kalman Filter

In this Section, we will explore the EKF formulation. The EKF is used when either the System State or the System Dynamics is not linear. The EKF is considered non optimal because it assumes that the noise is Gaussian. However, optimal filters are not guaranteed and are computationally demanding. The Extended Kalman filter (EKF) is a sub-optimal solution, but computationally fast. The EKF linearises the original non-linear filter dynamics around the previous state estimate [45].

Considering the non-linear dynamics
\[
x(k) = f(x(k-1), u(k)) + v(k), \quad v(k) \sim \mathcal{N}(0, Q),
\]
\[
y(k) = h(x(k)) + w(k), \quad w(k) \sim \mathcal{N}(0, R),
\tag{3.7}
\]
where \( x \) is the state, \( y \) is the observations and \( f \) and \( h \) are the state and observation functions, which
can be non-linear. The independent random processes $v$ and $w$ are white Gaussian, with zero mean and covariance matrix defined as

$$E[v(k)v^\top(k)] = Q, \quad E[w(k)w^\top(k)] = R.$$  \hspace{1cm} (3.8)

The EKF comprises a Prediction Cycle Step and an Update Step done consecutively. The Prediction Step uses the last state and the model to Predict what the current state is. The update step fuses the prediction with the measurements from the sensors.

### 3.2.1 Prediction Cycle

The Prediction Cycle Step predicts the (a priori) state estimation using

$$\hat{x}(k|k-1) = f(\hat{x}(k-1|k-1), u(k)), \hspace{1cm} (3.9)$$

and generates the Predicted (a priori) error covariance matrix

$$P(k|k-1) = F(k)P(k-1|k-1)F^\top(k) + Q \hspace{1cm} (3.10)$$

where the state Transition Model $F(k)$ is

$$F(k) = \frac{\partial f}{\partial x} \bigg|_{x=\hat{x}(k|k-1), u=u(k)} \hspace{1cm} (3.11)$$

### 3.2.2 Update Cycle

The Update Cycle Step firstly calculates the Innovation (or pre-fit residual) covariance

$$S(k) = H(k)P(k|k-1)H^\top(k) + R \hspace{1cm} (3.12)$$

where $H(k)$ is the observation model mapping the true state space into the observed space as

$$H(k) = \frac{\partial h}{\partial x} \bigg|_{x=\hat{x}(k|k-1)} \hspace{1cm} (3.13)$$

After, the optimal Kalman gain is calculated as

$$K(k) = P(k|k-1)H(k)^\top S^{-1}(k), \hspace{1cm} (3.14)$$

which is used to obtain the updated (a posteriori) state estimate as

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k) (y(k) - H\hat{x}(k|k-1)), \hspace{1cm} (3.15)$$
and the updated (a posteriori) estimate covariance with

\[ P(k|k) = P(k|k-1) - K(k)S(k)K^T(k). \]  (3.16)

The EKF uses a set of approximations in the matrices \( P \) and depends on previous estimates for the matrices \( F \) and \( H \). Depending on previous estimates means that it also depends on measurements, and therefore it is important to state that it is not an optimal filter. If the measurements are incorrect at some point or if the initial state is far from reality, the EKF may diverge. This happens because consecutive linearisations are not a good approximation of the true model in all the associated uncertainty domain.

### 3.3 Interacting Multiple Model Filter

The Interactive Multiple Model (IMM) Filter is a Navigation method that combines the hypothesis from multiple filter models in order to get a better estimate of the state space under model uncertainty. The IMM implements a Markov Chain of different EKFs and is composed by 4 main phases.

1. **State Interaction:** In the State interaction, the results from the last cycle are fused based on the switch covariance, outputting the state and covariance to the EKF’s.

2. **State Estimation (EKF):** This phase is a normal EKF algorithm with the addition of the model likelihood calculation.

3. **Model Probability Update:** The results from the EKF’s likelihoods are used together with the last model probabilities to obtain the current model probabilities.

4. **Estimated States Combination:** This phase combines the model probabilities with the model results such that the output combined state is obtained.

In addition to the EKF matrices, the transition probability matrix of the Markov chain matrix \( p \) needs to be defined a priori. The matrix \( p \), commonly named state switching matrix, defines the probability of transitioning from one model \( i \) to the other model \( j \), with \( i, j \in [1, N_m] \) and \( N_m \) the number of EKF models.

#### 3.3.1 State Interaction

Prior to the filter update, the model state estimates, \( \hat{x}_{0i}(k-1) \), and model covariances, \( P_{0i}(k-1) \), are mixed using computed conditional model probabilities as in

\[ \bar{\mu}_i(k) = \sum_{j=1}^{N_m} p_{ji} \mu_j(k-1) \]  (3.17)

\[ \mu_{ij}(k) = \frac{p_{ji} \mu_j(k-1)}{\bar{\mu}_i(k)} \]  (3.18)

where \( \mu_i \) is the mode probability and \( \mu_{ij} \) the mixing probability.
The mixed state and covariance for model $i$ at time $k$ is computed as

$$\hat{x}_{0i}(k-1) = \sum_{j=1}^{N_m} \mu_{ij}(k) \hat{x}_j(k-1) \quad (3.19)$$

$$P_{0i}(k-1) = \sum_{j=1}^{N_m} \mu_{ij}(k) \left\{ P_j(k-1) + [\hat{x}_j(k-1) - \hat{x}_{0i}(k-1)][\hat{x}_j(k-1) - \hat{x}_{0i}(k-1)]^T \right\} \quad (3.20)$$

One can observe that when leading with time-sensitive operations, one can perform this step at the end of the last cycle after outputting the result, as it does not depend on the current measurements.

### 3.3.2 State Estimation

The state estimation uses $N_m$ EKFs, as described in Section 3.2, with each model likelihood, $\Lambda_i(k)$, as

$$\Lambda_i(k) = \mathcal{N}(y(k)-\hat{y}(k|k-1), S_i(k)) = \frac{1}{\sqrt{2\pi \det S_i(k)}} \exp \left\{ -\frac{1}{2} [y(k) - \hat{y}(k|k-1)] S_i^{-1}(k) [y(k) - \hat{y}(k|k-1)] \right\},$$

with the innovation covariance matrix $S(k)$ computed in the State Estimation Step of each EKF.

To accelerate the computing process these computations can be parallelised, as the result of one EKF does not affect the other.

### 3.3.3 Model Probability Update

The probability of each model is computed using the likelihood $\Lambda_i$. This step is done after state prediction of each mixed state estimate. If Gaussian statistics are assumed, the likelihood of model $i$ is given by

$$\mu_i(k) = \frac{\Lambda_i(k) \sum_{j=1}^{N_m} p_{ij} \mu_j(k-1)}{\sum_{l=1}^{N_m} \Lambda_l(k) \sum_{j=1}^{N_m} p_{lj} \mu_j(k-1)} \quad (3.22)$$

### 3.3.4 Estimated States Combination

The combined state estimate and covariance is computed from the updated filtered states from each model weighted by the updated model probabilities.

$$\hat{x}(k|k) = \sum_{i=1}^{N_m} \mu_i(k) \hat{x}_i(k|k) \quad (3.23)$$

$$P(k|k) = \sum_{i=1}^{N_m} \mu_i(k) \left\{ P_i(k|k) + [\hat{x}_i(k|k) - \hat{x}(k|k)][\hat{x}_i(k|k) - \hat{x}(k|k)]^T \right\} \quad (3.24)$$
3.3.5 IMM Filter Representation

A representation of the algorithm flow and the relationship between phases can be seen in Figure 3.2. The inputs to the algorithm are individual states and covariances for each model as well as previous model probability.

![Figure 3.2: IMM Filter Algorithm Scheme [8]](image)

3.4 Proportional Navigation

Regarding the specific definition of Proportional Navigation, there are several versions and some discussion on which one is one better [46]. In this thesis we will use the Pure Proportional Navigation (PPN), as it is the classic definition, and despite surging new definitions [47], it still is the one performing better or comparable to the others [48].

Proportional Navigation is a Guidance law in which the Interceptor is commanded to turn at a rate proportional to the angular velocity of the line-of-sight (LOS) line. In Figure 3.3, a diagram illustrating the geometry of the solution is depicted. The notation present here differs from the Literature, as it uses a generalized version, so as to accommodate a three-dimension result in the inertial reference frame.

The acceleration of the Interceptor is defined such that it is proportional and perpendicular to its velocity and angular rate as

\[
a_{PN} = N_{PN} v_t \times \omega_{LOS},
\]  

(3.25)

where \( N_{PN} \) is a constant. The constant \( N_{PN} \) for missile Systems usually ranges from 2 to 6 [11], but
always greater than 1. When it is greater than 1, it means that the Interceptor is turning faster than the LOS, building up a lead angle with respect to the LOS.

\[ \mathbf{r}_{LOS} = \mathbf{p}_T - \mathbf{p}_I, \]

connecting the Interceptor to the Target. Having this into account, the LOS angular rate \( \omega_{LOS} \) is defined as

\[ \omega_{LOS} = \frac{\mathbf{r}_{LOS} \times \mathbf{v}_r}{\mathbf{r}_{LOS} \cdot \mathbf{r}_{LOS}}, \quad (3.26) \]

where \( \mathbf{v}_r = \mathbf{v}_T - \mathbf{v}_I \) is the relative velocity of the Target with respect to the Interceptor in the Inertial reference frame.

### 3.5 Thrust Vectoring Control

The Thrust Vectoring Control (TVC) is a type of controller, in which the desired thrust vector is firstly calculated, and then the control outputs are defined according to it. In the TVC, outside forces and perturbations can be directly taken into account such as drag, gravity and wind. As these forces are included directly in the model, the vehicle rejects the perturbations faster as they are feed forward. Thus, the controller is as good as the estimate of the outside forces and perturbations are.

In the specific case of the UAV it has four degrees of freedom, namely attitude angles yaw, pitch, roll, and thrust. Since the TVC defines the 3 dimensional thrust vector, only 3 degrees of freedom are needed which are the roll, pitch and thrust magnitude. The yaw angle can then be separately controlled, which could be used to aid the tracking of the Target.

#### 3.5.1 TVC Formulation

In the TVC, the commanded thrust vector is defined by [36] as

\[ \mathbf{T}_{cmd} = k_p \circ \mathbf{e}_p + k_{pu} \circ \int_0^\Delta t \mathbf{e}_p dt + k_w \circ \mathbf{e}_w + k_a \circ \mathbf{e}_a + m g + m \mathbf{a}_{cmd} + \mathbf{F}_D, \quad (3.27) \]
where \( k_p, k_{pi}, k_v \) and \( k_a \) represent vector gains associated with the position, velocity and acceleration errors defined by \( e_p, e_v, \) and \( e_a \). The element-wise multiplication \( \circ \) is necessary because, due to the difference in dynamics between the z axis and the x and y axis, the gains should be different for each axis. The errors are defined as

\[
e_p = p_{cmd} - p, \quad e_v = v_{cmd} - v, \quad e_a = a_{cmd} - a,
\]

where \( p_{cmd}, v_{cmd} \) and \( a_{cmd} \) are the commanded position, velocity and acceleration that come from the Guidance algorithm. In Equation (3.27) all terms are in the inertial reference frame, with the first 4 terms being feedback related and the others being feed-forward related.

When the wind velocity is ignored, the Drag Force \( F_D \) is defined as

\[
F_D = T R A_{bla} R^T v + k_{par} \circ \|v\|v,
\]

where \( A_{bla} \) refers to the Blade Flapping Coefficients matrix and \( k_{par} \) refers to the Parasitic Drag Coefficients vector. The first term is the most widely used as it is relevant for low speeds, and the second term is only relevant for velocities above \( 10 \text{ [m/s]} \), which is the case in the current mission.

### 3.5.2 TVC Orientation

In order to translate the commanded thrust vector into roll, pitch and thrust magnitude, we must firstly identify the axis of the desired orientation of the UAV, as depicted in Figure 3.4.

![Figure 3.4: Thrust Vector controller illustration](image)

For this we need to take into account the independent commanded yaw angle \( \psi_{cmd} \). We identify the intermediate \( \mathbf{x}^{\psi}_{cmd} \) orientation of the first reference frame rotation as

\[
\mathbf{x}^{\psi}_{cmd} = \begin{bmatrix} \cos(\psi_{cmd}) & \sin(\psi_{cmd}) & 0 \end{bmatrix}^T,
\]

(3.30)
and knowing that commanded UAV orientation in the z axis $z_{cmd}^B$ is given in the body-fixed frame by

$$z_{cmd}^B = \frac{T_{cmd}}{\|T_{cmd}\|}, \quad (3.31)$$

we can use the orthogonal property of orthogonal reference frames to identify the desired $y_{cmd}^B$ axis as

$$y_{cmd}^B = \frac{z_{cmd}^B \times x_{cmd}^B}{\|z_{cmd}^B \times x_{cmd}^B\|} \quad (3.32)$$

and afterwards the $x_{cmd}^B$ vector as

$$x_{cmd}^B = y_{cmd}^B \times z_{cmd}^B \quad (3.33)$$

With these three vectors, we can obtain the commanded Rotation Matrix as

$$R_{cmd} = \begin{bmatrix} x_{cmd}^B & y_{cmd}^B & z_{cmd}^B \end{bmatrix}, \quad (3.34)$$

which is equivalent to the one defined in 3.6.

### 3.5.3 TVC Outcome

Resorting to Equation (3.34), and the Rotation Matrix in 3.6, by equalling the indexes, we can obtain the desired roll and pitch angles as

$$\begin{bmatrix} \phi_{cmd} \\ \theta_{cmd} \end{bmatrix} = \begin{bmatrix} \tan^{-1} \left( \frac{R_{32}}{R_{33}} \right) \\ \tan^{-1} \left( \frac{-R_{31}}{\sqrt{R_{32}^2 + R_{33}^2}} \right) \end{bmatrix} \quad (3.35)$$

Finally to retrieve the thrust magnitude $T_{cmd}$ one must project the thrust vector $T_{cmd}^\ast$ into the commanded body vector $z_{cmd}^B$ as

$$T_{cmd} = T_{cmd}^\ast \cdot z_{cmd}^B \quad (3.36)$$

Given the outcomes of the TVC, $\phi_{cmd}$, $\theta_{cmd}$ and $T_{cmd}$, its values can be sent to the Flight Controller, such that the UAV follows the given references.
Chapter 4

Approach Formulation

In this Chapter details regarding the detailed formulation of the previous Chapter Background Theory will be explored. The goal of this chapter, is such that the reader will understand the parts of the algorithms that are unique to this thesis application, such as defining the specific models used and the assumptions.

Firstly we will formulate the two Navigation algorithm implementations. In Section 4.1 the specifics regarding the EKF for the assumed Model will be characterised, and in Section 4.2 the IMM Filter will be explored, where several Model possibilities will be defined as well as other relevant parameters. Then, in Section 4.3, the modification on the original PN Algorithm will be explained. Finally, in Section 4.5, to use the entire thrust envelope, the TVC algorithm is modified.

4.1 Extended Kalman Filter Formulation

To formulate the EKF, we need to define the Motion Model and the Measurement Model. The Motion Model relates the previous state and inputs to the current state. The Measurement Model relates the measurements with the States.

4.1.1 Motion Model

In the Motion Model we firstly define the state vector with the Interceptor and Target UAVs states

\[ x = \begin{bmatrix} x_I^T & x_T^T \end{bmatrix}^T = \begin{bmatrix} p_I^T & v_I^T & q_I^T & \omega_I^T & p_T^T & v_T^T & a_T^T \end{bmatrix}^T. \]  

(4.1)

For the Interceptor UAV Model, the state function was already defined in Section 3.1, and we will use the model in Equation (3.1) with the simplification that the drag force is not considered for simplicity. For the Target UAV Model, we have to make an assumption on how it behaves. The Dynamic Model chosen
was based on kinematics with constant acceleration. Thus, the overall state function becomes

$$
\dot{x} = f(x, u) = \begin{bmatrix}
\dot{p}_T \\
\dot{v}_T \\
\dot{\omega}_T \\
\dot{p}_T \\
\dot{v}_T \\
\dot{a}_T
\end{bmatrix}
= \begin{bmatrix}
v_I \\
q_I \otimes (T/m) \otimes q_I + g \\
(1/2) (q_I \otimes \omega_I) \\
J^{-1} (\tau - \omega_I \times J \omega_I)
\end{bmatrix}
\cdot
$$

(4.2)

The linearised State transition matrix, used in the covariance calculation is defined as

$$
F(k) = \frac{\partial f}{\partial x}|_{x=x(k|k-1), u=u(k)},
$$

(4.3)

becoming

$$
F = \begin{bmatrix}
F_T & 0_{13 \times 9} \\
0_{9 \times 13} & F_T
\end{bmatrix}
$$

(4.4)

with

$$
F_T = \begin{bmatrix}
I_3 & \Delta t I_3 & 0_{4 \times 4} & 0_3 \\
0_3 & I_3 & \frac{\partial \dot{v}_I}{\partial q} & 0_3 \\
0_{4 \times 3} & 0_{4 \times 3} & \frac{\partial \dot{q}_I}{\partial q} & \frac{\partial \dot{q}_I}{\partial \omega} \\
0_3 & 0_3 & 0_{4 \times 4} & \frac{\partial \dot{\omega}_I}{\partial \omega}
\end{bmatrix}
$$

and

$$
\frac{\partial \dot{v}_I}{\partial q} = 2\Delta t \begin{bmatrix}
-2q_2 T_x - q_1 T_y + q_0 T_z \\
-2q_3 T_y - q_0 T_y + q_1 T_z \\
-q_3 T_y + q_2 T_x \\
-q_0 T_x + q_3 T_y - 2q_2 T_z \\
q_1 T_x + q_2 T_y \\
q_2 T_x + q_1 T_y \\
q_3 T_x + q_0 T_y - 2q_1 T_z
\end{bmatrix}
$$

(4.6)

$$
\frac{\partial \dot{q}_I}{\partial q} = \begin{bmatrix}
1 & -\frac{\Delta t}{2} \omega_1 & -\frac{\Delta t}{2} \omega_2 & -\frac{\Delta t}{2} \omega_3 \\
\frac{\Delta t}{2} \omega_1 & 1 & -\frac{\Delta t}{2} \omega_3 & \frac{\Delta t}{2} \omega_2 \\
\frac{\Delta t}{2} \omega_2 & \frac{\Delta t}{2} \omega_3 & 1 & -\frac{\Delta t}{2} \omega_1 \\
\frac{\Delta t}{2} \omega_3 & -\frac{\Delta t}{2} \omega_2 & \frac{\Delta t}{2} \omega_1 & 1
\end{bmatrix}
$$

(4.7)

$$
\frac{\partial \dot{q}_I}{\partial \omega} = \frac{\Delta t}{2} \begin{bmatrix}
-q_1 & -q_2 & -q_3 \\
q_0 & q_3 & -q_2 \\
-q_3 & q_0 & q_1 \\
q_2 & -q_1 & q_0
\end{bmatrix}
$$

(4.8)
\[
\frac{\partial \omega}{\partial t} = \frac{\Delta t}{2} \begin{bmatrix}
0 & \omega_3 (J_{zz} - J_{yy}) & \omega_2 (J_{zz} - J_{yy}) \\
\omega_3 (J_{xx} - J_{zz}) & 0 & \omega_1 (J_{xx} - J_{zz}) \\
\omega_2 (J_{yy} - J_{zz}) & \omega_1 (J_{yy} - J_{zz}) & 0
\end{bmatrix} .
\] (4.9)

The matrix \( I_N \) represents an identity matrix with dimension \( N \times N \), and \( 0_N \) represents a matrix of zeros with dimension \( N \times N \).

### 4.1.2 Measurement Model

For the Measurement Model, we need to look at the measurements available, and how they relate to the state. The available measurements that will be used are: Interceptor position \( \hat{p}_I \), from the GPS, Compass and IMU, filtered through the Flight Controller; Interceptor velocity \( \hat{v}_I \), also filtered through the Flight Controller; Interceptor attitude \( \hat{\Theta}_I \), in Euler angles from Flight Controller’s Sensor Fusion; Angular Velocity \( \hat{\omega}_I \), from Gyroscope, filtered through a low-pass filter in the Flight Controller; and Target Relative Position \( \hat{p}_r \), from the Tracking Algorithm. Note that we are not explicitly using the Gyroscope, Accelerometer and Magnetometer available, given that the Flight Controller already fuses the results when estimating the Attitude. In Equation (4.10), the used observations are depicted, and its relation with the state.

\[
y = \begin{bmatrix}
\hat{\Theta}_I \\
\hat{p}_I \\
\hat{\psi}_I \\
\hat{\omega}_I \\
\hat{p}_r
\end{bmatrix} = \begin{bmatrix}
\phi_I \\
\hat{\phi}_I \\
\hat{\psi}_I \\
\hat{\omega}_I \\
\hat{p}_r
\end{bmatrix} = h(x) = \begin{bmatrix}
\tan^{-1} \left( \frac{2q_w q_x + q_y q_z}{1 - 2(q_x^2 + q_z^2)} \right) \\
\sin^{-1} \left( \frac{2q_w q_y - 2q_z q_x}{1 - 2(q_y^2 + q_z^2)} \right) \\
\tan^{-1} \left( \frac{2q_w q_z + q_y q_y}{1 - 2(q_y^2 + q_z^2)} \right) \\
\hat{p}_I \\
\hat{v}_I \\
\hat{\omega}_I \\
q \otimes (p_T - p_I) \otimes q^*,
\end{bmatrix}
\] (4.10)

During implementation, provisions were taken so as to avoid discontinuities.

In order to evaluate the EKF observation transition matrix

\[
H(k) = \frac{\partial h}{\partial x} \bigg|_{x = \hat{x}(k|k-1)}
\]

we differentiate the observation function and we can arrive at the observation transition matrix as

\[
H = \begin{bmatrix}
H_1 & 0_{12 \times 9} \\
H_2 & H_3
\end{bmatrix}
\] (4.12)
with

\[
H_1 = \begin{bmatrix}
0_{1\times3} & 0_{1\times3} & \frac{\partial \phi}{\partial q_I^r} & 0_{1\times3} \\
0_{1\times3} & 0_{1\times3} & \frac{\partial \theta}{\partial q_I^r} & 0_{1\times3} \\
0_{1\times3} & 0_{1\times3} & \frac{\partial \psi}{\partial q_I^r} & 0_{1\times3} \\
I_3 & 0_3 & 0_{3\times4} & 0 \\
0_3 & I_3 & 0_{3\times4} & 0_3 \\
0_3 & 0_3 & 0_{3\times4} & I_3
\end{bmatrix}, \quad H_2 = \begin{bmatrix}
\frac{\partial p_r}{\partial p_I^r} & 0_3 & \frac{\partial p_r}{\partial q_I^r} & 0_3 \\
\end{bmatrix}, \quad H_3 = \begin{bmatrix}
\frac{\partial p_r}{\partial p_T^r} & 0_3 & 0_3
\end{bmatrix}
\]

and where

\[
\frac{\partial \phi}{\partial q_I^r} = \tan^{-1} \left( \frac{2q_0 q_1 + q_2 q_3}{1 - 2(q_1^2 + q_2^2)} \right), \quad \frac{\partial \theta}{\partial q_I^r} = \tan^{-1} \left( \frac{x_1}{\sqrt{1 + x_1^2}} \right), \quad x_1 = \frac{2q_0 q_1 + q_2 q_3}{1 - 2(q_1^2 + q_2^2)}
\]

\[
\frac{\partial \psi}{\partial q_I^r} = \tan^{-1} \left( \frac{2q_0 q_2 + q_3 q_1}{1 - 2(q_1^2 + q_2^2)} \right), \quad \frac{\partial \psi}{\partial q_I^r} = \tan^{-1} \left( \frac{x_1}{x_1^2 + 1} \right), \quad x_2 = \frac{2q_0 q_2 + q_3 q_1}{1 - 2(q_1^2 + q_2^2)}
\]

\[
\frac{\partial p_r}{\partial p_I^r} = \frac{\partial q^* \otimes (p_T^r - p_I^r) \otimes q}{\partial p_I^r} = \frac{\partial R^{-1}(p_T^r - p_I^r)}{\partial p_I^r} = -R^{-1}
\]

\[
\frac{\partial p_r}{\partial p_T^r} = \frac{\partial q^* \otimes (p_T^r - p_T^r) \otimes q}{\partial p_T^r} = \frac{\partial R^{-1}(p_T^r - p_T^r)}{\partial p_T^r} = R^{-1}
\]

\[
\frac{\partial p_r}{\partial q_I^r} = \begin{bmatrix}
-q_3(p_T^r - p_T^r)_y + q_2(p_T^r - p_T^r)_z & q_2(p_T^r - p_T^r)_y + q_3(p_T^r - p_T^r)_z \\
q_3(p_T^r - p_T^r)_z - q_1(p_T^r - p_T^r)_y & 2q_1(p_T^r - p_T^r)_y - q_0(p_T^r - p_T^r)_z \\
-q_2(p_T^r - p_T^r)_y - q_1(p_T^r - p_T^r)_y & q_3(p_T^r - p_T^r)_z + q_0(p_T^r - p_T^r)_y - 2q_1(p_T^r - p_T^r)_z \\
-2q_2(p_T^r - p_T^r)_y - q_1(p_T^r - p_T^r)_y & q_3(p_T^r - p_T^r)_z + q_0(p_T^r - p_T^r)_y - 2q_1(p_T^r - p_T^r)_z \\
q_0(p_T^r - p_T^r)_y + q_3(p_T^r - p_T^r)_y & q_3(p_T^r - p_T^r)_z + 2q_1(p_T^r - p_T^r)_y + q_2(p_T^r - p_T^r)_z \\
-q_0(p_T^r - p_T^r)_y + q_3(p_T^r - p_T^r)_y & q_3(p_T^r - p_T^r)_z + 2q_1(p_T^r - p_T^r)_y + q_2(p_T^r - p_T^r)_z
\end{bmatrix}
\]

4.2 Interactive Multiple Mode Filter Formulation

The Interactive Multiple Mode Filter (IMM) has the capacity of merging different Target models so as to correctly predict a wider range of behaviour.

For the IMM Filter formulation, one should define:
1. Switching Matrix $p$: is a $N \times N$ matrix defines the a priori probability for switching from one model to another;

2. Interceptor Dynamic Model: already defined in Section 4.1;

3. Target Kinematic Models and respective Covariances: defined in the remainder of this Section, based on four models in [9], we consider $N = 4$ models with constant velocity, constant acceleration, constant turning rate, and thrust accelerated.

4.2.1 Switching Matrix

The four models used are Constant Velocity (CV), Constant Acceleration (CA), Constant Turn (CT) and Thrust Acceleration (TA). The Switch matrix is defined as the transition probability from one state to the other, usually defined with high values in the diagonal as,

$$
\begin{bmatrix}
    p_{CV \rightarrow CV} & p_{CV \rightarrow CA} & p_{CV \rightarrow CT} & p_{CV \rightarrow AT} \\
    p_{CA \rightarrow CV} & p_{CA \rightarrow CA} & p_{CA \rightarrow CT} & p_{CA \rightarrow AT} \\
    p_{CT \rightarrow CV} & p_{CT \rightarrow CA} & p_{CT \rightarrow CT} & p_{CT \rightarrow AT} \\
    p_{AT \rightarrow CV} & p_{AT \rightarrow CA} & p_{AT \rightarrow CT} & p_{AT \rightarrow AT}
\end{bmatrix}
$$

4.2.2 Target Kinematic Models

Regarding the Target UAV kinematic Models, we use its state defined by

$$
x_T = \begin{bmatrix} p_T^T & v_T^T & a_T^T \end{bmatrix}^T
$$

The Target transition matrix $F_T$ and the covariance matrix for the Target process noise $Q_T$ are stated as follows, where the value $q_T$ is a noise spectral density parameter, and has units of $[m^2/s^5]$ and $\Delta t$ represents time difference between actual and past measurement. The models are assumed to have acceleration noise.

Target with Constant Velocity

The Constant Velocity Model (CV) assumes that the Target moves with constant velocity. It is useful for the state estimation of non-manoeuvring targets. The matrices are defined as

$$
F_{T_{CV}} = \begin{bmatrix} I_3 & I_3 \Delta t & 0 \\ 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & I_3 \end{bmatrix}, \quad Q_{T_{CV}} = q_{T_{CV}} \Delta t \begin{bmatrix} 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 \end{bmatrix}.
$$
Target with Constant Acceleration

The Constant Acceleration Model (CA) assumes that the Target moves at constant acceleration. It has been used for state estimation of targets with complex manoeuvres. The matrices are defined as

\[
F_{T,CA} = \begin{bmatrix}
1 & 1 & t/2 & t^2/2 \\
0 & 1 & t & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
Q_{T,CA} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

(4.23)

Target Turning Constantly

The Constant Turn Model (CT) assumes the Target moves according to a constant turn with constant angular velocity \(\omega\). The matrices are defined as

\[
F_{T,CT} = \begin{bmatrix}
1 & 1 & \omega^{-1} \sin(\omega \Delta t) & \omega^{-2} (1 - \cos(\omega \Delta t)) \\
0 & 1 & \cos(\omega \Delta t) / \omega & \omega^{-1} \sin(\omega \Delta t) \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
Q_{T,CT} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

(4.24)

where we use an estimation of the angular velocity calculated by

\[
\omega = \frac{\| \ddot{\mathbf{a}} \|}{\| \mathbf{v} \|}
\]

(4.25)

every time the filter runs with the state previously estimated. A Filter can be implemented as in [49], which might provide better results, however, this solution was not adopted as to keep simplicity.

Target with Thrust Acceleration

The Thrust Acceleration Model (TA), as defined in [9], considers an exponentially growing velocity, used mainly in terminal phases. High-speed manoeuvring Targets usually present this behaviour because of the propulsion mechanism. Contrarily to what has been done in [9], we will not assume constant acceleration because if the velocity grows exponentially, so does the acceleration. This strategy was adopted in Literature because only tracking quality was considered, therefore, the acceleration estimation was not important. This is not the case for this thesis and if acceleration could be rightfully estimated, it should be incorporated in the Guidance. Since acceleration is not available we defined it as the velocity difference over time, as

\[
a_{TA}(k) = \frac{\mathbf{v}_{TA}(k) - \mathbf{v}_{TA}(k - 1)}{\Delta t} = \frac{e^{\Delta t} - 1}{\Delta t} \mathbf{v}_{TA}(k - 1).\]

(4.26)

Therefore, the matrices are defined as

\[
F_{T,TA} = \begin{bmatrix}
1 & 1 & (e^{\Delta t} - 1) & 0 \\
0 & 1 & e^{\Delta t} & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
Q_{T,TA} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

(4.27)
4.3 Proportional Navigation

Regarding the algorithm presented in Section 3.4, one must consider that it was initially thought with the purpose of being adapted to missiles. Because of this, in Literature, only acceleration commands are promptly available. Considering this fact, there is an important factor that must be considered: the velocity. In the original PN formulation, the velocity is assumed constant, not necessarily pointing at a specific direction. It is assumed that the PN algorithm will adjust the velocity direction automatically, but that only happens due to the missile dynamics (nozzle at the back). As it is obvious, this does not happen with UAVs, and the velocity direction will need to be commanded. This is the main concern of this Section: to define the velocity vector, such that it resembles the dynamics of the UAV.

Considering the Position, this will not be considered, and there will not be a Position command.

4.3.1 PN Velocity

Therefore, the velocity definition adopted to maintain consistency between a missile and an UAV Model is defined in Equation (4.28). This definition is adopted so as to mimic a missile behaviour where the commanded acceleration changes the velocity vector direction.

\[ v_{PN}(k) = V_{PN} \frac{v_{PN}(k-1) + \Delta t a_{PN}(k-1)}{\|v_{PN}(k-1) + \Delta t a_{PN}(k-1)\|} \]  

(4.28)

where \( V_{PN} \) is fixed, and should be higher than \( \|v_T\| \). The initial velocity is defined using

\[ V_{PN} = \left\| b \frac{r_{LOS}}{\|r_{LOS}\|} + v_T(0) \right\| \Rightarrow v_{PN}(0) = b \frac{r_{LOS}}{\|r_{LOS}\|} + v_T(0), \]  

(4.29)

where the constant \( b \) can be easily calculated, using the solution for a second degree equation.

4.4 Thrust Vectoring Control

Regarding the TVC algorithm, two changes were performed with respect to the definition in Literature.

4.4.1 Removal of the Position command

The first change is the removal of the Position command, due to the non-existence of such command from the PN algorithm, becoming

\[ T_T = k_{pi} \int_0^{t_{hover}} e_p dt + k_a \circ e_a + mgz^W + ma_{cmd} + TRA_{bla} R^T \mathbf{v} + k_{par} \|v\| \mathbf{v}. \]  

(4.30)

4.4.2 Modification of the Thrust Magnitude computation

The second is a change in the conversion of the Trust Vector to the Thrust command, so as to maintain altitude and comply with the Thrust command in the \( z \)-axis the commanded Thrust becomes

\[ T_{cmd} = \frac{T_T}{\cos \phi \cos \theta}. \]  

(4.31)
4.5 Modified TVC with PN

Having into account the PN, one can easily observe that the formulation for the velocity as in Equation (4.28) is not optimal due to the constant velocity requirement. As missiles travel constantly at maximum speed (nozzle burning fuel), this is not a concern, and is not important for the solution. With UAVs, this is not true. Taking always constant speed as a reference, then if velocity is set too high, and a high PN command is presented, the overall controls can saturate; if it is too low, it will not reach the maximum Thrust capabilities.

4.5.1 Adapted Thrust vector

In cases where the Interceptor UAV has low correction commands, it should use its maximum capabilities for accelerating against the Target. When it has many corrections to perform, these should be prioritised so that it maximises the hit percentage. It is based on this thinking track that emerges the necessity of modifying the initial terms of the PN. Therefore, the total TVC vector shall be corrected using

\[
T'_T = T_T + T_{vel},
\]

(4.32)

where \(T_{vel}\) is the remaining thrust vector that depends on velocity vector that comes from the remaining Thrust magnitude. After obtaining the final desired Thrust vector, one can follow the same procedure as in the previous Section 4.4 and go through Equations (3.30)-(3.36), in order to obtain the yaw, roll, pitch and Thrust magnitude commands to send to the Flight controller.

4.5.2 Maximum Thrust magnitude

Taking into account the knowledge regarding the maximum possible Thrust magnitude that the motors can output, we can calculate the maximum velocity that scales the velocity coming from the Guidance algorithm so that it provides the maximum possible velocity in the correct direction. This velocity is computed as

\[
T_{vel} = \eta (T_T, v_{cmd}) v_{cmd},
\]

(4.33)

where \(\eta\) is a value that needs to be computed every time.

As defined in the previous section, the velocity direction is given by Equation (4.28). Therefore, defining the maximum allowed thrust magnitude \(T_{max}\) one can state that

\[
\| T_T + T_{vel} \| = T_{max},
\]

(4.34)

and solve for \(\eta\) using Equation (4.33). Equation (4.34) is a quadratic equation with two possible solutions, where the highest positive \(\eta\) is obviously the solution.
Chapter 5

Implementation

This thesis handles multiple problems such as Algorithm development, Simulation testing and Experimental testing. Therefore, in this Chapter a clear Architecture is define so as to keep tractability, organization and flexibility between the different Testing Environments.

Firstly, in Section 5.2, we will show a System Mission Control that will be responsible for achieving the Target UAV interception, with all supporting trajectory infrastructure. In Section 5.1 an Architecture of the System is defined and its subsystems are explored, where it will be depicted how all the algorithms interact with each other. In Section 5.3, we will describe how the Testing Environments, and show which dynamics were considered and all other models such as the Sensors’s models and respective noise.

5.1 System Architecture

The System Architecture is depicted in Figure 5.1. This gives a more detailed description that the Figure 1.3 shown in the Introduction Chapter, and we can clearly identify the two major subsystems:

- The Race Quadcopter System, a RoboCat 270 UAV model that can be easily purchased and is available out-of-the-box for RC Pilots.

- The Onboard Computer, which is added to the standalone System, and runs the high-level algorithms, including the ones developed in this thesis.

Within the System Architecture, we developed for this thesis the NGC Algorithm Block, which will be described in Section 5.1.2. Nevertheless, this Block interacts with most of the other ones, and Section 5.1.1 gives more detail on them.
5.1.1 Supporting Blocks

Sensors

The Sensors onboard the UAV are an IMU Sensor integrated in the Flight Controller, containing a Gyroscope, Accelerometer and Magnetometer. Connected to the Flight computer, there is a GPS with a compass module. The combination of these sensors allow the position $p_I$, velocity $v_I$, angular velocity $\omega_I$ and attitude $\Theta_I$ of the Interceptor UAV to be obtained.

Actuators

The actuators available are the four Brushless DC Motors, ESCs and Propellers. The Flight Controller gives the PWM signals to the ESCs, which transform them into 3 phase electrical signals that rotate the motors. Consequently, the attached propellers rotate as well producing the Thrust that moves the UAV.

Tracking

The Tracking Block ideally would make use only of the onboard flight Sensors, and would with, with the help of tracking algorithms, capture the Target UAV position $p_T$. After obtaining it in real-time, the position is sent to the NGC Algorithm for Interception. At the current state of the Project, this is not yet available.

Flight Controller

The Flight Controller’s main task is to make sure the RC or onboard computer commands are fulfilled. Moreover, it performs sensor fusion and filtering, and runs at $1 \text{ kHz}$. As mentioned earlier, it also fuses estimates of the early mentioned $p_I$, $v_I$, $\omega_I$ and $\Theta_I$ UAV states.
The inputs to the Flight Controller are interchangeable as the ones it would obtain from the RC Controller or from the NGC algorithms. As the Flight Controller is in Angle Mode, the inputs to it are:

- $\phi_{cmd}$ and $\theta_{cmd}$, is the roll and pitch angle, respectively, commanded to the Flight Controller;
- $\dot{\psi}_{cmd}$, is yaw rate commanded to the Flight Controller;
- $T_{cmd}$, is the total Thrust output to the motors.

The outputs of the Flight Controller are:

- $p_I$ and $v_I$, are the Interceptor's position and velocity. The data is obtained by fusing the information from the Onboard sensors - GPS, Accelerometer and Compass(x and y-axis), and GPS Accelerometer and Barometer (z-axis).
- $\Theta_I$, is the Interceptor's Euler angles. The data is obtained by fusing the information from the Onboard sensors in the Flight Controller - Gyroscope, Accelerometer and Magnetometer;
- $\omega_T$, is the Interceptor's angular velocity. The data is obtained from the Flight controller, which filters the Gyroscope's data;
- $u_I$, is the Interceptor UAV output to the motors, given in Pulse Width Modulation (PWM) units.

### 5.1.2 NGC Architecture

In the NGC Architecture, the main algorithms developed in this thesis are contained. Additionally to these algorithms, when considering the entire Mission, some support algorithms are added, in order to integrate the remaining States other than Target interception, $S_{TI}$.

#### NGC Algorithm Block

The NGC Algorithm Block is divided in the three components, first Navigation, then Guidance and finally Control, and is depicted in Figure 5.2. The decisions N, G and C depicted in Figure 5.2, refer to the chosen Navigation, Guidance and Control Algorithm. The main Interception algorithms included are:

1. **EKF**: Extended Kalman Filter, see Section 4.1;
2. **IMM**: Interactive Multiple Model Filter, see Section 4.2;
3. **PN**: Proportional Navigation, see Section 4.3;
4. **TVC**: Thrust Vector Controller, see Section 4.4.
5. **MTVC**: Modified Thrust Vector Controller see Section 4.5;

The supporting algorithms are the following:

1. **EKF**: Modified Extended Kalman Filter, with only the Interceptor States. This is important, because there are states where the Target UAV position is not captured or needed. Using a fixed value would deteriorate the results for the Interceptor UAV.
2. **RT**: The Reference Trajectory Guidance, which simply outputs to the TVC the references from a predefined desired trajectory. The trajectory is given in terms of position, velocity, acceleration and yaw angle commands.

![NGC Block Architecture](image)

**Figure 5.2: NGC Block Architecture**

**NGC Interfaces**

All the interfaces are constructed in such a way that regardless of the Mission (Interception or Trajectory), the Interfaces are the same. Having said that, in this subsection we will explore the interfaces in a functional level with Figures 5.3a, 5.3b and 5.3c.

![NGC Algorithm Interfaces](image)

**Figure 5.3: NGC Algorithm Interfaces**

In the Navigation, the only difference is that, when the Target's position is not available, will not be received and estimated. In the Guidance, when an interception is occurring, the values from the Navigation are received so as to determine the reference. However, when following a Reference Trajectory, the trajectory is received beforehand, therefore, no inputs are needed from other algorithms. For Control, the inputs and outputs are always the same.

### 5.2 System Mission Control

In order to achieve this thesis objectives, we designed a System Mission Control. The System Mission Control in which the developed Interceptor UAV is embedded is designed in such a way that is able to support different Mission Modes, States and Events, shown in Sections 5.2.1 to 5.2.3. Overall, using the
Modes, States and Events, a Mission State Machine used for the Mission Control can be achieved. This State Machine can be seen in Figure 5.4.

![State Machine of the Mission Control](image)

**Figure 5.4: State Machine of the Mission Control**

### 5.2.1 Mission Modes

The System Mission Control supports the following Mission Modes:

- **Target Interception Mission** (M\textsubscript{TI}): The Target Interception Mission is the main purpose of this thesis, in order to achieve the objective of intercepting the Target UAV. The Mission incorporates the process of the Interceptor UAV taking-off, acquiring the Target UAV in LOS, and intercepting it.

- **Reference Trajectory Mission** (M\textsubscript{RT}): The Reference Trajectory Mission is a secondary mission, created to achieve the first one, and handles practical cases where a reference Trajectory is given. This is mainly designed for experimental feasibility in which typical trajectories are analysed, or fail-safe Mission for when an unexpected Mission Event occurs during Target Interception Mission.

### 5.2.2 Mission States

The System Mission Control supports the following Mission States:

- **Wait on Ground** (S\textsubscript{WG}): This is the initial State of the System Mission Control. At this State, the System is waiting on ground for a Mission request, which is signalled by a Take-Off Event.

- **Take-Off** (S\textsubscript{TO}): In this State, when any Mission starts, the Interceptor UAV will take-off and reach the desired hover position. The desired hover position is specified to be a certain height above the ground for the Target Interception Mode and the trajectory's initial position for the Reference Trajectory Mode.

- **Yaw Acquisition** (S\textsubscript{YA}): After hover position is obtained, the Interceptor UAV proceeds to adjust its Yaw angle. For the Target Interception Mode it will point at the Target UAV and start the Estimation Filters. For the Reference Trajectory Mode, it will acquire the initial Yaw angle reference.

- **Target Interception** (S\textsubscript{TI}): This State will begin the interception manoeuvre and carry it out until an interception is detected.
- **Reference Trajectory** ($S_{RT}$): This State will follow a reference trajectory defined by the user. The Reference Trajectory already contains a landing.

- **Land Trajectory** ($S_{LT}$): In this State, the Target UAV will follow a landing trajectory. The landing trajectory is defined based on the current position and urgency to land.

### NGC Algorithms and Mission States

As seen before, several NGC algorithms exist that can be swapped with each other. For that effect, there is a control function that chooses which one to execute, according to the current Mission State. In Table 5.1 this allocation can be seen.

<table>
<thead>
<tr>
<th></th>
<th>Navigation</th>
<th>Guidance</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{WG}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$S_{TO}$</td>
<td>EKF’</td>
<td>RT</td>
<td>TVC</td>
</tr>
<tr>
<td>$S_{YA}$</td>
<td>EKF/EKF’</td>
<td>RT</td>
<td>RVC</td>
</tr>
<tr>
<td>$S_{TI}$</td>
<td>IMM/EKF</td>
<td>PN</td>
<td>TVC/MTVC</td>
</tr>
<tr>
<td>$S_{RT}$</td>
<td>EKF’</td>
<td>RT</td>
<td>TVC</td>
</tr>
<tr>
<td>$S_{LT}$</td>
<td>EKF’</td>
<td>RT</td>
<td>TVC</td>
</tr>
</tbody>
</table>

Table 5.1: Mission Modes to NGC algorithms allocation

#### 5.2.3 Mission Events

The System Mission Control supports the following Mission Events:

- **Take-Off** ($E_{TO}$): This Event occurs upon starting of any given Mission. When this happens, the Interceptor UAV goes into a Take-Off State.

- **Hover Achieved** ($E_{HA}$): This Event occurs once the hover position is achieved. Afterwards, the Interceptor UAV proceeds to the Yaw Acquisition State.

- **Yaw Acquired** ($E_{YA}$): This Event occurs once the Yaw angle is acquired. Afterwards, the Interceptor UAV proceeds to Target Interception or Reference Trajectory State, based on the Mode.

- **Breach of Geo-Fence** ($E_{BG}$): This Event occurs whenever the Interceptor UAV reaches the vicinity of walls or ceilings. When this happens, an aggressive landing trajectory is created. The Geo-Fencing parameters can be defined beforehand.

- **Land after interception** ($E_{LI}$): This Event occurs used whenever interception is detected. The interception is achieved when the Interceptor and Target UAVs are within a small acceptable distance. When this happens, a landing trajectory is created.

- **Land on user command** ($E_{LU}$): This Event occurs whenever there is a land command issued by the remote control operator. When this happens, a smooth landing trajectory is created.
5.3 Testing Environments

5.3.1 Simulation Environment

In order to test the algorithms, a Simulation was created. The Simulation is based on RotorS [38] which uses a Gazebo simulation environment, together with ROS (Robot Operating System). The original Simulation was adapted, so as to match the mission and UAV settings. An example of the view in the Gazebo simulation is depicted in Figure 5.5. A deeper analysis about the Simulation Tools chose can be seen in Appendix A.

![Gazebo Simulation Environment](image)

(a) Far view  
(b) Close view

Figure 5.5: Gazebo Simulation Environment

5.3.2 Simulation Dynamics

Because the Forces and respective models used in the RotorS Environment [38] are well established, therefore, will not be changed, so as to maintain integrity with other UAV Simulations. Having said this, we will describe here the dynamics used.

For the core formulation, the Newton law and Euler equation can be written as

\[ F = ma \] \hspace{1cm} (5.1)
\[ M = J\dot{\omega} + \omega \times J\omega, \] \hspace{1cm} (5.2)

with \( m \) being the mass of the UAV, \( a \) the acceleration, \( J \) the inertia matrix, and \( \omega \) the angular velocity.

For the Morpheus UAV it becomes

\[ \sum_{i=0}^{3} (R_{i}^{w}(T_{i} + D_{i})) + mg = ma \] \hspace{1cm} (5.3)
\[ \sum_{i=0}^{3} (M_{R,i} + M_{D,i} + T_{i} \times r_{i}) = J\dot{\omega} + \omega \times J\omega. \] \hspace{1cm} (5.4)

The core formulation, for every Simulator should be the same, what usually changes is which Forces are considered, and the correspondent model. For the RotorS Simulator, four main effects are considered: Thrust Force, Drag Force, Rolling Moment and Drag Moment.
Thrust Force:
The Thrust force is the vertical force produced by the propellers. It is defined as
\[ T_i = \omega_{ROT}^2 C_T e_{zB}, \]  
(5.5)
where \( \omega_{ROT} \) is the angular velocity of the propeller, \( C_T \) is the motor thrust constant, and \( e_{zB} \) is the unit vector pointing in the z-direction in the rotor’s body frame.

Drag Force:
The Drag force is the vertical force produced by the propellers. It is defined as
\[ F_D = -\omega_{ROT} C_D v_{A}^\perp, \]  
(5.6)
where \( C_D \) is the drag constant, including the Induced, Flapping and Profile drag into one factor (as they are proportional). Even though this differs from the Model considered in Section 3.1, using different approximations, for consistency with other available work, the dynamics will be maintained as RotorS standards. Even with different dynamics, the effect should not be great, because, even with different models, it approximates the same quantity.

Rolling Moment:
The rolling moment is the angular force produced by the propellers. It is defined as
\[ M_R = \omega_{ROT}^2 C_R v_{A}^\perp, \]  
(5.7)
where \( C_R \) is the motor thrust constant, and \( v_{A}^\perp \) is the velocity projected to the \( x - y \) plane.

Drag Moment:
The drag moment if the moment used in the yaw control, defined as
\[ M_D = -\epsilon C_M \cdot T, \]  
(5.8)
where \( C_M \) is the rotor moment constant, and \( \epsilon \) denotes the turning direction of the rotor, more specifically, for clockwise motion \( \epsilon = 1 \), and for counterclockwise motion \( \epsilon = -1 \).

Parameter Estimation
The Parameters for the Dynamics will be, for the most part estimated in Section 6. The parameters that are not estimated will be taken from similar UAVs. The quality of the Simulation Dynamics will be validated in the results Section 7.
5.3.3 Real Environment

The System is based on a Robocat 270 Racing Quadcopter UAV frame; a Matek F722-STD Flight Controller performing angle and thrust Control; a Raspberry Pi 3 B+ running ROS, executing the interfaces and algorithms. Details can be seen in Figure 5.6, where the interfaces and algorithms were thoroughly profiled.

![Diagram](image)

**Figure 5.6: Experimental Testbed**

Due to the fact that the Real testing Environment is indoors, the GPS and Compass/Magnetometer’s measurements cannot be used. To counteract this shortcoming, a Motion Capture System (MCS) OptiTrack is used, as the source for pose measurements.

Moreover, since the available Motion Capture System used does not have any velocity and acceleration output, where only pose (position and attitude) was available, the algorithm in [50] was reproduced, and considered herein as Ground Truth. The position is fed to the algorithm at 200 [Hz], the frequency used by the Motion Capture System, and the frequency where acceleration is considered constant is [50] Hz, the frequency of the NGC algorithm.

The Flight Controller provides the remaining measurements, from the onboard flight Sensors. The Flight Controller uses an Multiwii Serial Protocol (MSP) protocol, and the exchanged message and firmware were customized such that only one message type needs to be exchanged between the Flight Controller and Onboard Computer. With this change, all the variables can be transmitted at a frequency of 50 [Hz].

The dummy Tracking algorithm uses also the Motion Capture System position of the Interceptor, so as to define the relative position between the Target trajectory and the Interceptor. Noise is later added. This is done at 50 [Hz].

---


1[^1]: [https://github.com/iNavFlight/inav/wiki/MSP-V2](https://github.com/iNavFlight/inav/wiki/MSP-V2)
5.3.4 Simulation and Real Environment Comparison

The main algorithms when in a simulation environment should have the same interfaces as the ones implemented the UAV hardware. The symmetry can be seen in Figure 5.7, where every component in the Experimental Setup has an equivalent in Simulation. For the effect, the Flight Controller firmware present in the UAV was adapted so as to reproduce the functions in the ROS and Gazebo Simulation environment.

The Dynamics are Simulated in the Gazebo Environment, where the experimental elements are simulated, namely sensor characteristics, and actuation effects.

Figure 5.7: Symmetry between the Experimental Setup and the Simulation Setup
Chapter 6

Identification

In this Chapter, the Model parameters that describe the dynamics of the UAV and the Sensors will be identified. Because this is the first time this hardware setup flies autonomously using external commands, no previous work has been made towards identification of these parameters. The goal of the identification, is to make sure the Simulation will highly resemble the actual behaviour of the UAV, which is very important when dealing with potentially destructive missions as this one. Moreover, the drag parameters identification is also necessary for the TVC controller.

For the identification, we will start in Section 6.1 where we will see how the sensors were modelled, with corresponding noise, bias and frequency characterization. In Section 6.2, an estimation of the Inertia matrix will be computed. In Section 6.3, a precise Static Thrust and Static Momentum evaluation, characterizing the motor and propeller response is made. Finally, in Section, 6.4 Drag tests will be made, so as to characterize the aerodynamic UAV response.

6.1 Sensors

Instead of considering only the sensors that are self-contained within the Interceptor UAV, we will additionally characterize the measurements from the Motion Capture System, as well as the artificial added noise to the Tracked position.

To characterize the data, a hover test was performed, where, the UAV was commanded to stay at a fixed position. This way, as the UAV is in flight, the vibrations should be consistent with a real scenario.

6.1.1 Onboard Sensors

The Onboard Sensors used in the experiments were the Accelerometer, Gyroscope and Magnetometer. The raw measurements are fused in the Flight Controller, where a Kalman filter is used to obtain the attitude estimate and a low pass filter is used to reduce the noise on the Gyroscope measurements. The sensor characterization performed herein is of the filtered data from the Flight Controller, and not the sensors themselves. As the Flight Controller runs at a higher frequency, these measurements will be
used directly, as these results will always be better than anything calculated in the Onboard Computer due to the lower frequency. The measurement results can be seen in Table 6.1.

<table>
<thead>
<tr>
<th>Angular Velocity</th>
<th>Attitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_x$</td>
<td>$\omega_y$</td>
</tr>
<tr>
<td>Noise spectral density</td>
<td>0.1326</td>
</tr>
<tr>
<td>Bias</td>
<td>–0.0152</td>
</tr>
<tr>
<td>Frequency</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 6.1: Onboard Sensor characterization from Hover Flight Test

### 6.1.2 Motion Capture System

The MCS was used to obtain:

1. Yaw angle $\psi$, because indoors, the Onboard Compass/Magnetometer Sensor could not function properly due to the magnetic field disruption of the building.

2. Position $p_I$ and velocity $v_I$, which in the future of the Project is predicted to be obtained through the Onboard GNSS System.

The MCS can be considered as the ground truth due to its milimetric precision. Therefore, a position characterization depicts more the UAV vibration evaluation rather than the actual measurement noise. The velocity is not directly measured by the Motion Capture System, using the filter defined in [50], therefore, no identification is performed.

For the yaw angle $\psi$, due to a protection net installed between the cameras and the test area, sometimes it was observed that the estimation oscillated according to marker visibility, affecting the angle measured. Special care should be taken in these circumstances regarding lighting. The measurement results obtain in Table 6.2 are obtained when minimising these factors.

<table>
<thead>
<tr>
<th>Attitude</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi$</td>
<td>$p_x$</td>
</tr>
<tr>
<td>Noise spectral density</td>
<td>0.0252</td>
</tr>
<tr>
<td>Frequency</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 6.2: Motion Capture System characterization from Hover Flight Test

### 6.1.3 Noise Tracked Position

As the Position Tracking algorithm is not currently available in the Project, and it is not within the scope of this Thesis, there was an artificial Gaussian zero mean noise added to the System with the characteristics as in Table 6.3
### 6.2 Moments of Inertia

The Moments of Inertia were estimated from the 3D Model of the UAV frame, motors and battery. Instead of making experimental tests, it was decided to use the 3D Models, given that it would be an undertaking without important benefits when testing the algorithms experimentally. Moreover, as it is predicted that the results will be an understatement of the actual Inertia Matrix values, the estimate obtained will be slightly changed to fit within similar UAV’s and higher than the calculated.

For the computation, the center of gravity is considered the center of rotation, in more detail, the considered elements are:

- **Carbon fiber frame**: is a widely used race quadcopter frame, therefore, 3D Models are available online. An example of the 3D Model is in [https://www.thingiverse.com/thing:2609479](https://www.thingiverse.com/thing:2609479), which was the one adopted herein. Having the 3D model available, the Inertia matrix for this part was calculated using a CAD program. The Frame weighs 216 $[g]$.

- **Motors**: a single motor was modelled as a cylinder. The Inertia was calculated as transposed to the center of rotation of the UAV. Each motor weighs approximately 20 $[g]$, for a total of 80 $[g]$.

- **Battery**: the battery cells are bundled together, as they are contained in the same area. These elements are modelled as a parallelepiped, fitting inside the UAV frame. The joint weight is considered to be 394 $[g]$.

All the remaining elements, such as the propellers, were disregarded for simplicity. A 3D Model of the UAV can be seen in Figure 6.1.

![Gazebo screenshot of the UAV 3D Model](attachment:image.png)

Figure 6.1: Gazebo screenshot of the UAV 3D Model

---

### Table 6.3: Relative position Noise artificially added

<table>
<thead>
<tr>
<th>Noise Covariance</th>
<th>$P_{rx}$</th>
<th>$P_{ry}$</th>
<th>$P_{rz}$</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td></td>
<td>[m]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Frequency</th>
<th>50</th>
<th>[Hz]</th>
</tr>
</thead>
</table>
Based on similar Models from Literature and adding a symmetry simplification to facilitate the calculations done in the State Model considered in Section 3.1, the computed Inertia Matrix is

\[ J = \begin{bmatrix}
0.008 & 0 & 0 \\
0 & 0.008 & 0 \\
0 & 0 & 0.012
\end{bmatrix} \text{[kg \cdot m}^2\text{]} \]  

(6.1)

6.3 Static Thrust and Torque Tests

Because a Static Thrust analysis is not only relevant for the Simulation, but also for the GNC Algorithm adopted, detailed testing was made. The test were performed with an RCBenchmark 1580 Series\(^\text{11}\), which is capable of measuring, thrust and torque, as well as recording the time response. These tests usually separate the motor and propeller from the UAV. However, because this is a multicopter, and the power going through the motor depends on the battery, the entire UAV was attached to the platform for fidelity. Moreover, by dividing the total thrust/torque by four, we further obtain a better estimate than testing only one motor.

![Experimental setup for static thrust constant measurement](image)

6.3.1 Static Thrust

The motor thrust magnitude is considered to have the dynamic response of a First Order System. A First Order system has a differential equation of the form

\[ \tau \frac{dx(t)}{dt} + x(t) = k(t)u(t) \]  

(6.2)

where \(x\) is the System state, \(u\) is the system input and \(\tau\) is the time constant. The function \(k(t)\) represents a gain that can be time variable. In the Simulation, even the up \(\tau_U\) and down \(\tau_D\) time constants are differentiated, so we will do the same in this analysis. Moreover, due to the fact that the inputs to the system motor are in PWM, and a motor only starts rotating after a given PWM value, we also need to add a offset constant to have the correct value in Thrust magnitude. Therefore, the motor

\(^{11}\text{https://www.rcbenchmark.com/pages/series-1580-thrust-stand-dynamometer}\)
thrust magnitude is modelled as:

\[ T(t) = x(t) + K_{\text{pwm}} \]  (6.3)

**Gain and offset constants**

In order to obtain the gain and offsets shown in Equations (6.2) and (6.3), the steady state response to a step can be evaluated, as in this condition we have

\[
\begin{align*}
\left. \frac{dx}{dt} \right|_{t \to \infty} &= 0 \\
u(t) &= U_{\text{pwm}} \\
T|_{t \to \infty} &= k|_{t \to \infty}U_{\text{pwm}} + K_{\text{pwm}},
\end{align*}
\]  (6.4)

where \(U_{\text{pwm}}\) is a constant PWM input. The measurement results for several trials can be seen in Figure 6.3, and the fitting equation was computed as

\[ T = 0.0357u_{\text{pwm}} - 39.1655 \, [N], \]  (6.5)

with correlation coefficient \( R = 0.9929 \), showing a very high linear correlation.

![Figure 6.3: PWM output vs. generated Thrust](image)

It is also possible to observe that the Thrust produced changes according to the Battery level. The higher the battery level, the higher the Thrust magnitude for the same input. This proves that the gain function \( k(t) \) indeed varies with time, as the battery loses power over time. In the future, this data could be used to compensate for this factor, allowing for a consistent performance throughout the battery charge. Nevertheless, we modelled in the Simulation all gains as constants, with \( k(t) = 0.0357 \frac{N}{\mu s} \) and \( K_{\text{pwm}} = -39.1655 \, [N] \). Moreover, for \( u_{\text{pwm}} < 1100 \) we modelled the Thrust magnitude as \( T = 0 \).
Motor Time Constant

To measure the time constants, we compiled all the trials made as shown in Figure 6.3, and observed the time evolution of the response to a PWM input step. The data was fit into a First Order System, where the distribution was also analysed.

The measurement results can be seen in Figures 6.4a to 6.5b, with the parameters identified with $\tau_U = 0.0739 \text{ [s]}$ and $\tau_D = 0.0676 \text{ [s]}$, showing that the motors accelerate faster than decelerate. Considering the range of the constants obtained and the uncertainty associated with the measurement, we consider the differences unimportant. Better results might be possible using a higher frequency measuring equipment.

![Figure 6.4: Time Contant Up Estimation](image)

(a) Time Response

(b) Constant Distribution

Figure 6.4: Time Contant Up Estimation

![Figure 6.5: Time Contant Down Estimation](image)

(a) Time Response

(b) Constant Distribution

Figure 6.5: Time Contant Down Estimation

6.3.2 Static Torque

The produced torque is expected to be proportional to the produced thrust, as in [42]. Therefore, the Torque equation is as follows

$$M = k_T T,$$

(6.6)
where $k\tau$ is the Torque constant. Again, several trials were taken and the response was evaluated, with the resulting relation shown in Figure 6.6. A fitting equation was computed as

$$M = 0.007T + 0.00027 \approx 0.007T \ [N \cdot m],$$  \hspace{1cm} (6.7)

with correlation coefficient of $R = 0.997$, which again show a very high linear correlation. As seen from both Equation (6.7) and Figure 6.6, the relation is mostly linear and intercepts zero. Thus, we modelled the simulation as $k\tau = 0.007 \ [m]$.

![Graph showing thrust vs. torque](image)

**Figure 6.6: Generated Thrust vs. Torque**

### 6.4 Drag

The final Identification step in this Thesis is to perform Drag tests. We will start by defining the estimator with which the data collected will be compiled, and the coefficients obtained. Following are the experimental results for the $x$ and $y$ axis.

#### 6.4.1 Equilibrium of Forces

According to a force equilibrium balance, represented in Figure 6.7, if Thrust compensates the gravity force in the $x$ axis, then the Drag can be determined as

$$D_x = T \sin \theta - m a_x = m(\bar{g} \tan \theta - a_x),$$ \hspace{1cm} (6.8)

where the angle $\theta$ can be measured and $a_x$ estimated. To minimise the uncertainty from the $a_x$ estimation, the experimental data will be obtain on a constant velocity flight.
6.4.2 Least Square Error Estimator

The Least Square Error Estimator is used in regression analysis, as a method that returns a solution for an equation, where the squares of the residuals is minimized [51]. The method is similar for the estimation in the $x$ and $y$ axis, and therefore we will only define it for the $x$ axis.

Considering the model defined in Equation (3.29), assuming that $A_{bla}$ is diagonal and the UAV is oriented as in Figure 6.7, then the estimate of the Drag is

$$\hat{D}_x = T\hat{a}_{bla}v_x + \hat{k}_{par}\|v_x\|v_x,$$

where the coefficients $a_{bla}$ and $k_{par}$ are the ones we are trying to estimate. Having into account a sample with $N$ measurements, the optimization function is therefore

$$J = \sum_{i=1}^{N} \left( D_{x,i} - \hat{D}_{x,i} \right)^2 = \sum_{i=1}^{N} \left( D_{x,i} - T_i\hat{a}_{bla}v_{x,i} - \hat{k}_{par}\|v_{x,i}\|v_{x,i} \right)^2.$$

Following the Least Squares method, the optimization equation becomes

$$\begin{align*}
\frac{\partial J}{\partial \hat{a}_{bla}} &= \sum_{i=1}^{N} \left( D_{x,i} - T_i\hat{a}_{bla}v_{x,i} - \hat{k}_{par}\|v_{x,i}\|v_{x,i} \right) (-T_i v_{x,i}) = 0 \\
\frac{\partial J}{\partial \hat{k}_{par}} &= \sum_{i=1}^{N} \left( D_{x,i} - T_i\hat{a}_{bla}v_{x,i} - \hat{k}_{par}\|v_{x,i}\|v_{x,i} \right) (-\|v_{x,i}\|v_{x,i}) = 0
\end{align*}$$

where the variables can be reorganized in the solvable matrix form, and the parameters estimated, as in

$$\begin{bmatrix}
\hat{a}_{bla} \\
\hat{k}_{par}
\end{bmatrix} = \left[ \sum_{i=1}^{N} T_i^2 v_{x,i}^2 \quad \sum_{i=1}^{N} T_i \|v_{x,i}\|v_{x,i}^2 \right]^{-1} \left[ \sum_{i=1}^{N} D_{x,i} T_i v_{x,i} \quad \sum_{i=1}^{N} D_{x,i} \|v_{x,i}\|v_{x,i} \right]$$

6.4.3 Drag Estimation Results

The profile of the velocity, acceleration and angle for the experimental data acquired is depicted in Appendix B. The velocities where it was possible to achieve constant speed or near that event range from $0.5 \ [m/s]$ to $3.5 \ [m/s]$. Given that the test-bed available length was $8 \ [m]$, including the accelerating and decelerating length, the achieved velocities are satisfactory. However, the velocities achieved are

Figure 6.7: Force equilibrium in constant height flight
not close to the 10 [m/s] mark, where the parasitic Drag becomes significant.

In Figures 6.8a and 6.8b the Drag fit obtained through the estimator is depicted. Due to the low velocities achieved, estimating the parasitic drag coefficient can be seen as overfitting the data, and the proof to it is that the parameter obtained for both the $x$ and $y$ axis are physically impossible, because the coefficient is positive, and drag cannot be in the same direction as the velocity. The computed results of the identification of the drag coefficients can be seen in Table 6.4.

![Figure 6.8: Drag vs Velocity & Linear fit](image)

![Figure 6.8: Drag vs Velocity & Linear fit](image)

For the data obtained in the $y$ axis, where the maximum speed where constant velocity was achieved is lower than in the $x$ axis, the evidence that the parasitic drag coefficient cannot be estimated is even greater. Moreover, by looking at the root mean square error for the fit with and without the parasitic drag coefficient, they are very similar. Even more, the error decreases if we do not account for the parasitic drag, which validates that the value computed does not make sense, and is only overfitting. The dispersion in the drag results is within the same values as in the Literature for similar UAVs [36].

<table>
<thead>
<tr>
<th></th>
<th>$a_{bla}$ [s/m]</th>
<th>$k_{par}$ [kg/m]</th>
<th>$RMSE$ [N]</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-axis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with Parasitic Drag</td>
<td>-0.0236</td>
<td>0.0038</td>
<td>0.14816</td>
</tr>
<tr>
<td>without Parasitic Drag</td>
<td>-0.0224</td>
<td>-</td>
<td>0.14822</td>
</tr>
<tr>
<td>y-axis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with Parasitic Drag</td>
<td>-0.0249</td>
<td>0.023</td>
<td>0.23697</td>
</tr>
<tr>
<td>without Parasitic Drag</td>
<td>-0.0191</td>
<td>-</td>
<td>0.20087</td>
</tr>
</tbody>
</table>

Table 6.4: Results of the identification of the drag coefficients

where the root mean square error $RMSE$ is defined using the approximation that $T \approx m \ddot{a}$ as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left( D - \left( m \ddot{a}_{bla} v + \dot{k}_{par} \|v\| v \right) \right)^2}{N}}.$$  (6.13)
6.5 Flight Controller

The Flight Controller firmware chosen is the INAV, running on a MatekF722. Because it uses open-source software, the exact controller was adapted to a ROS node, simulating the PID controller implemented. The Identification of this component is not aimed at obtaining the parameters, as they are available, but at comparing how the Simulated Flight Controller performs in comparison to the actual Flight Controller, using experimental and simulated data.

![Figure 6.9: Roll angle Flight Controller](image)

The results are depicted in Figure 6.9, where Simulated and Experimental response to similar commands during the same flight trajectory are shown. Based on the results, one can conclude that the Flight Controller has a delay of approximately 0.5 [s] until it reaches the setpoint, and the Simulation behaves similarly to the actual UAV.
Chapter 7

Results and Discussion

In this Chapter, we will display and analyse the Simulation and Experimental results from the NGC Algorithm tests. The Algorithms developed will firstly be evaluated separately, evaluating individual performance, and then together, evaluating the Complete Mission performance.

In Section 7.1, the EKF and IMM Navigation algorithms will be compared. It will be discussed on which circumstances it is beneficial to add extra complexity with the IMM filter. In Section 7.2, the PN algorithms will be evaluated experimentally and in Simulation, to determine whether or not the algorithm converges quickly to an intercepting path, and whether it can perform well under uncertainty. In Section 7.3, the adopted TVC will be analysed regarding the differences to the original application and whether it follows successfully the desired commands. Finally, in Section 7.4, the entire Target Interception Mission will be evaluated experimentally and in Simulation, where all the separated algorithms work together, intercepting virtual UAV Targets.

7.1 EKF & IMM Filter

The Navigation data regarding the Interceptor UAV itself does not pose any challenge or is a novelty, because the model used is already explored and verified throughout the Literature. Therefore, only the Target estimation will be the focus of this Section, and whether a simple EKF solution is good enough to perform high-speed interceptions, where Target velocity estimation and potentially Target acceleration estimation are required. Contrarily to the Literature, where the focus is mainly on position estimation, here velocity and acceleration estimation are equally important.

7.1.1 Target UAV Trajectories

As virtual UAV Targets were used, the Target UAV trajectory is determined beforehand, adding artificial noise, which makes the experimental data exactly the same as the Simulation data. Therefore as only one analysis is necessary, we used the Simulation data.

We will evaluate the estimation using three different trajectories. The first one is a classic example of an IMM showcase, as appears in [9], going through different trajectory types at high speed. The
second one is a circular trajectory that is used later on in Section 7.4. The third and last trajectory analysed is a random trajectory in which at constant speed the target turns randomly. The goal is to test both algorithms on high uncertainty trajectories, checking if IMM actually has an increased performance, comparing to the flexible EKF model.

7.1.2 Considerations about the EKF and IMM results

With regard to the IMM Models, one should be aware that the covariances were not optimized, considering $q_{CA} = q_{CA} = q_{CT} = q_{TA} = 1$, which might deteriorate the results. Nevertheless, the concept that compares an IMM with an EKF should is still valid.

For depiction simplicity, all results are only evaluated in the x-axis, even though the estimation is three dimensional. The results on the y-axis are similar.

7.1.3 Defensive Manoeuvre

In the Defensive Manoeuvre (DM), the Target moves at constant velocity for $5 \text{ [s]}$, performs a full circle path maintaining velocity norm, and uses the thrust acceleration model as in [9] for $2 \text{ [s]}$. This serves to prove not only if the IMM works properly, but also how it compares to the baseline EKF solution.

Trajectory Path

The trajectory is depicted in Figure 7.1, where the first stage contains a portion with constant velocity, followed by a constant turn, and finally a Thrust acceleration manoeuvre.

![Figure 7.1: DM - Trajectory Path](image)

IMM Model Probability

The probability estimation from the IMM algorithm is depicted in Figure 7.2, where one can clearly see that the models were correctly estimated. During the transition phases, the versatile Constant Acceleration model, the one used in the EKF, has the highest probability.

![Figure 7.2: DM - Model Probability](image)

Position and Position Error

The estimated position and position error are depicted in Figure 7.3, where it appears that the IMM Filter performs worse than the EKF, taking longer to converge in the beginning, and similar or higher error.
Velocity and Velocity Error

The estimated velocity and velocity error are depicted in Figure 7.4. Contrarily to the position estimates, the EKF takes now longer to stabilize on an initial velocity, has equivalent error on the circular phase, and worst performance on the thrust accelerated manoeuvre.

Acceleration and Acceleration Error

The estimated acceleration and acceleration error are depicted in Figure 7.5. Looking at the results it is clear why the IMM performed better than the EKF. At such high circular phase speeds, because the IMM correctly detects a circular trajectory, the velocity and acceleration contain no delay. Considering that the Ground Truth algorithm has induced delay, here the IMM actually leads it.

Discussion

From this trajectory, even though the position estimates are equivalent, from the velocity and acceleration estimates we can conclude that the IMM clearly outperforms the EKF. However, questions arise to the physical reality of this trajectory, especially the final thrust acceleration trajectory.
7.1.4 Circular Trajectory

This manoeuvre is a Circular Trajectory (CT), where the radius is $0.5 \, [m]$ and the Target velocity is $0.5 \, [m/s]$. This Interceptor response to the Target trajectory is seen in Section 7.4. This trajectory is the most demanding due to the high Target accelerations, which constantly change the path direction.

Trajectory Path

The trajectory is depicted in Figure 7.6. Because the last trajectory had the high position amplitudes, the added noise could not be clearly seen. However, here the noise is clearly shown.

IMM Model Probability

The probability estimation from the IMM algorithm is depicted in Figure 7.7, where one can clearly see that the IMM constant turn model is correctly estimated quickly. During the transition phase, the constant acceleration model surges as the most likely. This happens because, in order for the constant turn model to be successfully calculated, firstly a velocity and acceleration should be correctly estimated, so as to obtain the angular velocity estimate.

Position and Position Error

The estimated position and position error are depicted in Figure 7.8. Both position estimates are very similar, however, the EKF takes longer to converge. The EKF estimation is also less susceptible to the added noise, with a smoother estimate. This is expected due to the nature of the constant acceleration model, acting as a low-pass filter.
Velocity and velocity Error

The estimated velocity and velocity error are depicted in Figure 7.9. Again, the EKF takes longer to converge. Contrarily to the position estimate, here the IMM presents lower velocity errors than the EKF throughout the entire trajectory.

![Figure 7.9: CT - Velocity Estimation and Error](image)

Acceleration and Acceleration Error

The estimated acceleration and acceleration error are depicted in Figure 7.10. The results are very similar to the velocity results, but the error difference is even bigger, and we can clearly see a outperform in terms of estimation delay.

![Figure 7.10: CT - Acceleration Estimation and Error](image)

Discussion

From the obtained results, it is clear that, even though the tracking position estimate does not increase greatly from the introduction of the IMM Filter, the velocity and acceleration do, in the case that the model is correctly estimated.

Given these results, one can deduce that results using an IMM estimator would be somewhat in the middle between using an EKF estimator and the Ground Truth. For this reason, when the entire Mission is assessed as a whole the EKF can be used to show the worst case scenario, and the IMM performance is expected to be somewhat between the obtained results and the Ground Truth data. This will be shown in Section 7.3, and the entire mission, in Section 7.4, using precisely this circular trajectory,
7.1.5 Random Trajectory

In the previously analysed cases, it was always used a model already incorporated in the IMM formulation. The Random Trajectory (RT) is therefore aimed at comparing both possibilities, in the case where both the EKF and IMM Filter, do not incorporate the model.

To generate this trajectory, it was considered that the Target UAV would move at a constant speed of $0.5 \text{ m/s}$, while every $1 \text{ s}$ the heading would be randomly generated. In between heading references, the Target UAV changes heading linearly until reaching the reference. This way, the trajectory is continuous and physically feasible. Because this trajectory was tested in a experimental setup in the Laboratory, the Target’s velocity needed to be lowered, to minimize the risk of breaching the Laboratory limits.

Trajectory Path

The trajectory is depicted in Figure 7.11. Due to the fast changing heading and the incorporated noise amplitude, higher that the actual trajectory detail, it is clear that any correct estimation is difficult.

IMM Model Probability

The probability estimation from the IMM algorithm is depicted in 7.12, where the model guessed is the Constant Turn, which is not far from the reality, where the turn direction changes every $1 \text{ s}$.

Position and Position Error

The estimated position and position error are depicted in Figure 7.13. The EKF definitely takes more time to converge, however, afterwards, it is equivalent to the IMM in terms of position error.
**Velocity and Velocity Error**

The estimated velocity and velocity error are depicted in Figure 7.14. The velocity details are clearly not completely captured, where both estimators behave similar to average filters.

![Figure 7.14: RT - Velocity Estimation and Error](image)

**Acceleration and Acceleration Error**

The estimated acceleration and acceleration error are depicted in Figure 7.15. For the Acceleration, the characteristics are even more dim, comparing to the Ground Truth signal.

![Figure 7.15: RT - Acceleration Estimation and Error](image)

**Discussion**

Given the added noise and the nature of the random trajectory, one should not expect any filter to correctly estimate the Target’s state, and that is the goal here. To evaluate how both Filters behave given this uncertainty. From the results, both have similar estimations, acting as low-pass filters, where the position characteristics are roughly estimates, but the velocity and acceleration are greatly attenuated.

All in all, when comparing the EKF with the IMM Filter, it becomes clear from the results that, in the case where the models are correctly estimated, the IMM outperforms the EKF. In the case where the model cannot be estimated, the results are comparable. Therefore, one can conclude that the IMM Filter is capable if performing well for UAV targets, improving estimation, mainly for velocity and acceleration, being preferred to a standalone generalised kinematics EKF in the case where there is model uncertainty.
7.2 PN and TVC

In this Section, the Guidance and Control algorithms will be analysed independently from the Navigation algorithm, so as to establish an independent analysis, proving the PN and TVC Interception concept. For that effect, all the results depicted herein use the Ground Truth obtained from the MCS, and therefore no influence from the uncertainty associated with the Navigation Algorithm exist.

The main challenges to the PN algorithm are: to understand if the collision trajectory can be promptly achieved; and if it is able to follow UAV Targets subjected to high accelerations, given that it was not built for it as it only considers constant velocity Targets.

The main challenge to the TVC algorithm is that the PN solution does not provide position commands. The basis on which TVC was tested in Literature, was Trajectory Following, therefore, proving efficiency in this case is uncertain.

It is also important to prove that the Yaw Controller can be considered independent, such that the Target could eventually be followed by a camera during interception.

In Table 7.1, a summary of the experimental trials performed for each trajectory can be found. The results presented are analysed from these experimental trials.

<table>
<thead>
<tr>
<th>Line Trajectory</th>
<th>Circle Trajectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trials</td>
<td></td>
</tr>
<tr>
<td>Set A</td>
<td>9</td>
</tr>
<tr>
<td>Set B</td>
<td>5</td>
</tr>
<tr>
<td>Set C</td>
<td>9</td>
</tr>
<tr>
<td>Set D</td>
<td>10</td>
</tr>
<tr>
<td>Set A</td>
<td>10</td>
</tr>
<tr>
<td>Set B</td>
<td>10</td>
</tr>
<tr>
<td>Set C</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 7.1: Experimental trials for PN and TVC testing

In Figure 7.16, the process associated with the results collection is illustrated. The UAV takes-off, acquires the Target on LOS and starts the interception manoeuvre. When the Target is intercepted, it lands. This sequence is the same for all the interceptions considered throughout the result analysis.

![Figure 7.16: Interception phases in the Experimental Setup](a) Start Take-off  (b) Start Target Acquisition  (c) Target Acquired  (d) Start Interception  (e) Interception, Start Land  (f) Landed
7.2.1 Line Trajectory

For the Line Trajectory (LT), the Target moves at \(0.5 \text{ m/s}\) velocity, and there are four different data sets, which correspond to four different Interceptor relative position. As the PN can predict successfully constant speed trajectories it is expected that if chooses the shortest path to Target interception.

**Interception Path**

Looking at the Trajectories performed in Figure 7.31 to Target interception it can be seen that the deviations from the optimal interception path (a straight line) are small. These deviations happens because the Interceptor starts from a hovering position. The time that it takes to stabilize on the commanded velocity is enough such that the Interceptor must change the initial course. Overall, the Interceptor successfully predict where the Target is heading.

**Interception Errors**

In Figure 7.18, obtained Interception Errors can be seen. These are very small, when comparing to the size of the Interceptor and a Target would have. With maximum errors around \(10 \text{ cm}\), it can be considered that Interception would always occur.
Distance to Target

The Distance to the Target norm can be seen in Figure 7.19. As expected, in the first moments it is still on a transitory state, where the distance does not change. After that, the distance in all sets lowers at a constant slope. The slopes are different between sets because, when subjected to different relative positions, the relative velocity to the Target is different.

Interceptor Velocity

If no great acceleration commands are performed, and the TVC works correctly the velocity is expected to stabilize around $1 \text{ [m/s]}$, which is the commanded PN velocity. This can be seen in Figure 7.20, where in all cases the velocity stabilizes around $1 \text{ [m/s]}$, showing the concept that the TVC is able to perform well, even in the absence of position commands.

![Figure 7.20: PN - LT - Interceptor Velocity](image)

Interceptor Acceleration

The acceleration over time during interception can be seen in Figure 7.21. The initial peak is such that it achieves the commanded velocity define by the PN. Afterwards, acceleration commands are due to the PN adjustment commands, such that the Interceptor is in an interception path with the Target. Because Set A and D start form a higher distance than C, they take longer to stabilize the velocity before the interception. Set C therefore has higher acceleration commands than the rest.

![Figure 7.21: PN - LT - Interceptor Acceleration](image)

Discussion

The PN algorithm can intercept successfully a constant, when the Interceptor. The closer the Interceptor is to the Target, the higher the PN corrections are. This happens because the commanded acceleration is proportional to the distance in the PN formulation. When the Interceptor is closer, the commands are more aggressive so as to make sure interception happens.
7.2.2 Circle Trajectory

The Circle Trajectory (CT) is aimed at testing how the PN algorithm behaves when subjected to an accelerating Target. For this purpose, a circular Target trajectory is considered, where the Target moves with an angular velocity of $0.2 \text{ [rad/s]}$ on a circle with radius of $0.5 \text{ [m]}$.

Interception Path

Looking at the Trajectory Paths in Figure 7.22, one can see that the Interceptor slowly changes the initial estimated velocity so as to accommodate the Target path. Therefore it is expected that, if the Target would move faster, the Interceptor would have to be able to change its path faster so as to intercept the Target successfully. We will see this to the end of this Chapter. The interception paths do not differ greatly than the ones in the Line Trajectory, being very close to a straight line which would be the shortest optimal path to the Target.

![Figure 7.22: PN - CT - Interception Paths](image)

In Figure 7.23, obtained Interception Errors can be seen. The results obtained are surprising, because, even though the Target is manoeuvring, because the relative velocity is higher, the interception precision actually improves. The acceleration added to the Target is therefore not sufficient to deteriorate the interception results.

![Figure 7.23: PN - CT - Interception Errors](image)

Distance to Target

The distance to Target can be seen in Figure 7.24, and are overall very similar to the line trajectory. Even though the Target has changing acceleration is is not enough to prevent a smooth interception.
Interceptor Velocity

The velocity of the Interceptor, depicted in Figure 7.25 follows similar behaviour to the Line trajectory, stabilizing at the commanded $1 \text{ [m/s]}$. Through these plots, because the acceleration commands are low, one can see that the velocity reference is successfully being followed, even without position commands.

Interceptor Acceleration

The acceleration of the Interceptor, depicted in Figure 7.26 follows a behaviour very similar to the Line Trajectory. However, one can see that the reference towards the end is higher in order to accommodate the last gap between an accelerating Target and the Interceptor.

Discussion

For this Circle Trajectory interception, even when subjected to some Target acceleration, the interception results are better than the ones previously considered. This is due to the velocity difference between Interceptor and Target, where the Interceptor moves at a much higher velocity. The question arises on whether this is an effect spanning in the entire velocity envelope.
7.3 Modified PN and TVC

Because this algorithm uses the full thrust envelope of the UAV, testing this algorithm in an experimental setup means achieving high-speeds very quickly, which clearly is not possible considering the available experimental setup. All results will be therefore simulated.

To test this modification to the original algorithm, the considered Target moves at $1\, [m/s]$ in the x-axis, and the Interceptor was placed at different distances. The results can be seen in Figures 7.27 to 7.30. Although it reaches high speeds and accelerations in such a short span of space, and the relative velocity is very high, the Interceptor now lacks manoeuvrability, with increased interception errors.

**Discussion**

The results obtained for this modification, prove that an increased interception velocity, even if it inflicts more damage, there is a loss in interception precision due to loss of reaction time. Instead of using the entire flight envelope, modifying the PN algorithm, a better hypothesis is to optimize the PN velocity and controller gains, so as to obtain a balance between precision and collision effectiveness.
7.4 Complete Mission

Having into account all the methods for Navigation, Guidance and Control combined, in this section we will analyse the Complete Mission (CM), using the EKF, PN and TVC. It is expected that, for the same trajectories using the IMM, the results should be somewhat between the one in Section 7.2 and the current Section.

<table>
<thead>
<tr>
<th>Line Trajectory</th>
<th>Circle Trajectory</th>
<th>Random Trajectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set A</td>
<td>Set B</td>
<td>Set C</td>
</tr>
<tr>
<td>Trials</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 7.2: Experimental trials for Complete Mission testing

7.4.1 Line Trajectory

This analysis will start with the same Line trajectory (LT) as depicted previously. The goal is to identify that the Trajectory deteriorates on a Target estimation subjected to noise, given that the Navigation algorithm should perform well, estimating correctly a zero acceleration Target.

Trajectory Path

Looking at the trajectories performed in Figure 7.31 to Target interception it can be seen that the deviations from the optimal interception path continue to be small for this Line Trajectory, even when the estimation is subjected to noise. The simulation behaviour seems to match the experimental data.

![Figure 7.31: CM - LT - Interception Paths](image1)

Interception Errors

The interception errors associated with this trajectory are depicted in Figure 7.32. The error, even if higher than before, is still small, considering that the total error is comparable to the added noise.

![Figure 7.32: CM - LT - Interception Errors](image2)
Distance to Target

The distance to the Target over time can be seen in Figure 7.33, where the behaviour is very similar to the Line trajectory in the absence of Navigation, because a constant velocity Target is correctly estimated.

Figure 7.33: CM - LT - Distance to Target

Interceptor Velocity

The Interceptor velocity depicted in Figure 7.34, is again very similar to the previously obtained values.

Figure 7.34: CM - LT - Interceptor Velocity

Interceptor Acceleration

The Interceptor acceleration depicted in Figure 7.35 is not as smooth as previously, with higher acceleration commands greatly due to the added noise. Even with a correctly predicted interception path, a small error near Target can produce an acceleration of $4\,[m/s^2]$.

Figure 7.35: CM - LT - Interceptor Acceleration

Discussion

For the performed test, in all cases the Interceptor would have successfully hit the Target, making the loss in interception precision acceptable, showing that the main objective of this thesis is accomplished.
7.4.2 Circle Trajectory

Because the Circular Trajectory (CT) evaluated previously would not evaluate the algorithm to all its potentially due to the low accelerations, here the Target velocity was incremented five fold to 0.5 [m/s] or an angular velocity of 1 [rad/s]. This is thus a more difficult trajectory, and the success of the interception will now lie on the Interceptor’s capability of changing interception direction quickly.

Trajectory Paths

The Trajectory Paths are depicted in Figure 7.36. The Paths are very similar to the ones produced during experimental tests. This is a big validation proof for the Simulation, because the Interceptor follows aggressive manoeuvres. Nevertheless, the Interceptor, clearly does not perform the shortest path, which would be straight line, due to the Target’s acceleration, which changes heading direction constantly.

Interception Errors

Due to the aggressive manoeuvre, the interception errors, depicted in Figure 7.37, are much higher than the previous ones. A maximum error can reach 30 [cm] where there is no certainty that the Target would be hit. The error in the z-axis is always the smallest, as the Target height does not change.

Distance to Target

In the Distance to Target depicted in Figure 7.38, due to aggressive Target manoeuvres, there are periods where the distance actually increases. Moreover, due to a failing in Interception detection to land, one can note that for Set C, there is a trial with a second interception, more successful that the remaining ones. This possibility is not considered during this thesis, but shows that cases where a first interception is not successfully performed could be further explored.
Interceptor Velocity

The Interceptor velocity, depicted in Figure 7.39, oscillates according to the Target direction change. Instead of the smooth curve stabilizing in the $1 \text{ [m/s]}$ setpoint, now it oscillates on that value. The oscillation occurs due to the PN commands, so as to change the Interceptor’s heading. Impact velocities are usually higher than the specified $1 \text{ [m/s]}$.

Interceptor Acceleration

The Interceptor acceleration is depicted in Figure 7.40. The accelerations can be as high as $6 \text{ [m/s]}$, which is extremely aggressive for our Laboratory reduced dimensions. The PN algorithm created high commands even with reduced velocities. This means that with an increase in velocity, the algorithm could lose response capacity.

Discussion

This Circle trajectory interception has shown that, even though the PN algorithm does not consider manoeuvring Targets, it is capable of adjusting to diversion manoeuvres. Considering the accelerations that are considered, even with increased errors, the results are satisfactory.
7.4.3 Random Trajectory

Here, the Random trajectory (RT) depicted in Section 7.1.5 will now be evaluated in what regards the interception capacity. It was seen previously that, even tough the position is somewhat correctly estimated, the velocity estimated does not capture the actual dynamics, acting almost as a low-pass filter.

Interception Path

The Trajectory Path is depicted in Figure 7.41, where one can observe that, contrarily to what happens in the Circle trajectory, because the velocity is not correctly estimated, there are not many oscillations in the trajectory.

![Figure 7.41: CM - RT - Interceptor Path](image)

![Figure 7.42: CM - RT - Interception Errors](image)

Interception Errors

The interception errors, depicted in Figure 7.42 are very small, lower than the ones in the Circle Trajectory. This is greatly due to the fact that in this Trajectory, the Target's velocity is lower. However, if comparing to the line trajectory they are higher, even though the Target's velocity is much lower.

Distance to Target

Regarding the Distance to Target in Figure 7.43, during the beginning of the interception the results look similar to a line interception. However, towards the end, because the Target changes direction abruptly there is a halt on distance decrease and a peak in the velocity.

Interceptor Velocity

In the interception velocity Figure in 7.43, the same behaviour is verified, where the beginning of the trajectory resembles the line trajectory. Close to interception, the velocity rises abruptly reaching almost double the commanded speed.
Interceptor Acceleration

The acceleration that the Interceptor performs is depicted in Figure 7.43. Towards the end, the accelerations are very aggressive so as that the Target is intercepted. Even though the Target’s velocity is underestimated, the PN commands are still very high due to distance error when passing by the Target.

![Figure 7.43: CM - RT - Distance to target, Interceptor Velocity and Acceleration](image)

Discussion

In this Random trajectory cases, loss of estimation detail might actually improve the interception, when considering the PN algorithm. If the velocity direction was completely captured, the Interceptor would be constantly trying to change path.

The complete proposed solution is hereby tested on a range of trajectory scenarios, successfully verifying its interception capabilities.

7.4.4 Yaw Control

In Figure 7.44, one can observe the Yaw control for the worst case tested, the Circle trajectory. The yaw gains were set very low, which creates high delay, because safe landing after interception was a concern. These results are within acceptable Field of View range, even though the design parameters were hand-tuned.

Discussion

These results prove that the Yaw Control can be successfully considered independent from the TVC. This opens the possibility of tracking the Target UAV using an Interceptor UAV body-fixed camera while performing aggressive interception manoeuvres.
Figure 7.44: Yaw Angle Control for Circle Trajectory. Colors represent a Field of View of $\pm 30^\circ$ and $\pm 45^\circ$. 
Chapter 8

Conclusions

This thesis proposes a solution for the interception of a non-cooperative Target UAV by an Interceptor UAV, where the relative position of the Target UAV is fed with noise to the Interceptor UAV. To solve the problem, different algorithms were proposed and implemented. The approach was bottom up, starting from a commercially available hardware, performing identification tests, producing a realistic Simulation, and finally performing Experimental testing where the created software was tested. We concluded that interception, even when subjected to an aggressive manoeuvring or a randomly escaping virtual Target, can be accomplished using the proposed solutions.

8.1 Achievements

8.1.1 Interception Supporting Tasks

UAV Identification

Firstly, a thorough Identification of the UAV system was performed. The Sensors were characterized through a hover test. Static Thrust and Moment tests were performed, where the relationship between the PWM sent to the ESCs that control the motors and the output is established. Additionally, the motor time characteristics were derived from the time-response of the motors subjected to a step reference. To characterize the flight dynamics of the UAV, drag tests were performed, where the UAV performed constant velocity trajectories. From these tests, it was possible to establish a relationship between the UAV velocity and the produced drag. Finally, so as to identify and reproduce the response of the commercially available Flight Controller board, the open-source firmware was converted into a ROS node for Simulation analysis. The resulting dynamics was analysed in flight tests and found to be similar.

Simulation and Experimental environment

Based on the identification, a simulation was developed, where the Interceptor UAV, its dynamics and Sensors, and perturbations are present. The algorithms tested in simulation were implemented in the real UAV, and flown in a laboratory, using a Motion Capture System as tracking algorithm replacement.
8.1.2 Interception NGC Algorithm

Navigation

To estimate the Interceptor and Target's states, an Extended Kalman Filter was developed where it was shown that the position, velocity and acceleration states can be successfully estimated from the Target's relative position. However, the results show that the velocity and acceleration estimation highly depends on the chosen Target's Model accuracy. To fight this shortcoming, an Interactive Multiple Model Filter was developed, showing better results for the same sets of data. Even though the position estimate does not improve greatly, for velocity and acceleration, the results obtained with the IMM are better.

Guidance

It was shown through simulation and experimental tests that a classic PN Missile Guidance System method can be successfully adapted to UAVs using small changes to the Literature solutions, mimicking a Missile Model. The Interceptor UAV can therefore correct its path to enter a collision course, minimizing interception time according to the current information.

Control

Regarding the Control algorithm, a Thrust Vectoring Controller was implemented, with some modifications regarding the Literature. It has been displayed experimentally and through simulation that in the absence of position commands during the interception stage, the implemented adapted TVC is still capable of following references correctly.

It was also shown that the yaw control can be considered independent, leaving it as an extra degree of freedom. This extra degree of freedom could eventually be used to capture image data, pointing at the Target, autonomously obtaining the Target's position.

In addition to the Classic PN and TVC method combination, a modification to the TVC was performed so as to use the full flight envelope of the UAV, based on the maximum thrust. Due to the experimental limitations, it was tested only in simulation, achieving high speeds in short spans of space, with acceptable interception precision.

8.2 Future Work

In terms of future work, there are several lines of research that could be pursued. The algorithms presented in this thesis should be tested further to understand the minimum requirements that a Tracking strategy would impose, by including delay and increasing the noise. Also, an optimization strategy could be developed to derive optimal PN velocity and gains. Moreover, the solution presented should be tested outdoors, subjected to wind perturbation, using GNSS methods, instead of the a Motion Capture System. For validation purposes, an RTK GPS solution is advised. Finally, a likely improvement to the proposed solutions is a Model Predictive Controller, combining the Guidance and Control Tasks.
References


Appendix A

Simulator Comparison

In this Appendix, a report produced to aid the choice of which simulator to use as a base for the simulation of the interception manoeuvre.

A.1 Introduction

Field tests on a real hardware environment are time consuming and not always accessible due to factors such as: weather, personnel availability, hardware availability, etc. When errors do occur in such environments, they are hard to reproduce, require post analysis and often result in damaging the Unmanned Aerial System (UAV). Due to these reasons, a realistic simulator is necessary where the algorithms can be promptly tested and corrected in a realistic environment, preparing for real-life testing.

The goal is to make Software in the Loop (SIL) testing accessible and realistic. Figure A.1 illustrates the potential software architecture of such a simulation environment, where components outside the companion computer, (e.g. the physical environment, flight controller and sensors) could be simulated. Once the simulation is implemented, the GNC algorithms under development can be iteratively and rapidly tested and validated.

Figure A.1: Scheme on Simulation target
Currently, there are two main open source options that we will consider for adaptation: Hector Quadcopter [37] and RotorS. [38] Both are based on the Gazebo open source simulator and the Robot Operating System (ROS), that has become a de facto standard in robotics research and facilitates integration of contributions by other researchers [37]. The Gazebo simulation will interface with ROS, where the required information can be published, making it easily adaptable to the working environment. All the simulation Plug-ins (code that customises simulator behaviour, implementing functions such as dynamics models and sensor publishing) are coded in C++.

Both simulators contain tasks such as: quadrotor dynamics simulation, sensor simulation, surrounding environment simulation and ros topic publishing.

To choose the simulator for adaptation, a set of characteristics will be considered according to the project requirements, these characteristics will be attributed a total number of points according to importance. Each parameter will be scored, within the total points, according to completion of such requirement. The score will later be added such that the simulator with more points is the chosen one. The characteristics we are looking for in a simulation for adaptation are:

1. **Documentation**: Whether documentation is available describing the algorithms used in the simulation (e.g. dynamics, control strategies, sensor models). Ideally, there should be enough documentation, that the user, in order to understand how the simulation is structured, is not required to read the code. Importance: 10 points.

2. **System Tractability**: Meaning easily managed or controlled, evaluates whether the software structure is logically organised for adaptation. Ideally, the development would be modular, the folder directory organisation would be in accordance with Gazebo repository structure and the dependencies would be direct. This point is important for adaptation and development of new algorithms. Importance: 10 points.

3. **Sensor availability**: Sensors will need to be added to the UAV. Ideally these would already be implemented in a comprehensive Gazebo sensor/publisher plug-in defined for each necessary sensor, with the possibility to change parameters such as bias and noise to match the desired system. Importance: 10 points.

4. **Dynamics Definition** This criteria evaluates whether the dynamics model is documented, easily accessible, understandable and modifiable. Importance: 10 points.

5. **Example models/environments**: The examples should be plug-and-play (without need for adjustments/modifications). Ideally there would already be a model with all the components necessary to the simulation. Importance: 5 points.

These characteristics will be analysed and scored one by one, in order to arrive to a final decision in section A.2.
A.1.1 Hector Quadcopter

The Hector Quadrotor simulation was developed by the Department of Mechanical Engineering in TU Darmstadt, Germany and released in 2012.

The documented code structure is ¹

1. `hector_quadrotor_description` provides a generic quadrotor URDF model as well as variants with various sensors.

2. `hector_quadrotor_gazebo` contains the necessary launch files and dependency information for simulation of the quadrotor model in gazebo.

3. `hector_quadrotor_teleop` contains a node that permits control of the quadrotor using a gamepad.

4. `hector_quadrotor_gazebo_plugins` provides plugins that are specific to the simulation of quadrotor UAVs in gazebo simulation.

The system was tested with the help of the provided launch files and exhibits the same characteristics as described in [37]. The quadrotor dynamics are realistic.

A.1.2 RotorS

The RotorS simulation was developed by the Autonomous Systems Lab in ETH Zurich, Switzerland and released in 2015.

The available code structure schematics is depicted in Figure A.2.

The system was tested with the example launch files provided and it works as described in the documentation provided by the authors. The quadrotor dynamics are realistic.

¹http://wiki.ros.org/hector_quadrotor
A.2 Comparison

The comparison results can be seen in Table A.1. The results are very close and the System Tractability is the decisive characteristic. The Publishing Date and the Maintainability of the repository were not taken into account.

<table>
<thead>
<tr>
<th></th>
<th>Hector Quadcopter</th>
<th>RotorS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publishing Date</td>
<td>2012</td>
<td>2015</td>
</tr>
<tr>
<td>Maintained</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Documentation</td>
<td>7/10</td>
<td>7/10</td>
</tr>
<tr>
<td>System Tractability</td>
<td>5/10</td>
<td>9/10</td>
</tr>
<tr>
<td>Sensor Availability</td>
<td>10/10</td>
<td>9/10</td>
</tr>
<tr>
<td>Physical Model</td>
<td>8/10</td>
<td>8/10</td>
</tr>
<tr>
<td>Example environments</td>
<td>5/5</td>
<td>4/5</td>
</tr>
<tr>
<td><strong>Final Score</strong></td>
<td><strong>43/55</strong></td>
<td><strong>45/55</strong></td>
</tr>
</tbody>
</table>

Table A.1: Simulators comparison Table
A.2.1 Documentation

Hector Quadrotor: [37] outlines the project offering a detailed description of the dynamics model, sensor modelling and control algorithms. 7/10

RotorS: [38] outlines the project offering a detailed description of the dynamics model, sensor modelling and control algorithms. 7/10

A.2.2 System Tractability

Hector Quadrotor: The software architecture is not modular in nature (e.g. repeated code in some occasions). This system rigidity, will lead to complications in the tailoring the existing system to our needs. In order to easily add functionality the software architecture would first need to be standardised. 5/10

RotorS: Modular system, organised directory structure and clear separation of functions. 9/10

A.2.3 Sensor availability

Necessary sensors: Monocular camera, IMU, GPS, compass.

Hector Quadrotor: This simulation has already all the sensors available and published with noise/bias customisation. The outputs of these sensors are being published as individual ROS Topics. 10/10

RotorS: This simulation has already all the sensors available with noise/bias customisation. The source code for the sensors is available however, these are not attached to the UAV in the examples provided. 9/10

A.2.4 Physical Model

A summary of the physical dynamics of both simulations can be found in table A.2. One can observe that they both depart from the same Dynamics Definitions.

<table>
<thead>
<tr>
<th></th>
<th>Hector Quadrotor</th>
<th>RotorS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics Definition</td>
<td>$a = C^b_n T/m$</td>
<td>$F = m \cdot a^h$</td>
</tr>
<tr>
<td></td>
<td>$M^b = J^{-1} M$</td>
<td>$M = J \cdot \omega^b + \omega^b \times J \cdot \omega^b$</td>
</tr>
<tr>
<td>Thrust Force</td>
<td>$T = C_{T,0}\omega^a_m + C_{T,1}\nu_1 \omega_M \pm C_{T,2}\nu_1^2$</td>
<td>$T = \omega^b_{ROT}C_T \cdot z^b$</td>
</tr>
<tr>
<td>Rolling Moment</td>
<td>$M_D = -C_{d,M} \cdot</td>
<td>\omega</td>
</tr>
<tr>
<td>Drag Force</td>
<td>$F_D = -C_{d,F} \cdot C_n^b \cdot</td>
<td>\nu</td>
</tr>
<tr>
<td>Drag Moment</td>
<td>$M_{M^b}^i = \begin{pmatrix} (F_4 - F_2) \cdot l_M \ (F_1 - F_3) \cdot l_M \ -M_1 + M_2 - M_3 + M_4 \end{pmatrix}$</td>
<td>$M_{D,i} = -eC_M \cdot T_i$</td>
</tr>
</tbody>
</table>

Table A.2: Dynamics Comparison

It is not the scope of this documentation to explain all the terms in the equations, detailed description of the terms can be found in [37] and [38]. However we can observe that both models are based on the
same Newton Laws (Dynamics Definition in A.2), with shared complexity in the term definitions. Only the accuracy/complexity of the physical models we analysed, and not the specific parameters. One must also observe that an added complexity is only beneficial, if parameters can be accurately estimated.

**Hector Quadrotor:** Well documented in [37], comprehensive definition. 8/10

**RotorS:** Well documented in [38], comprehensive definition. 8/10

Full score was not given to any of the simulators, because neither achieved full complexity.

In addition, one must consider, that defining a perfect dynamics model of the system is something unattainable.

### A.2.5 Example environments

**Hector Quadrotor:** A large variety of comprehensive examples exists, using different models/sensors/worlds 5/5

**RotorS:** Some examples are provided. These examples however need a few modifications, for instance, the throttle control needs to be remapped to a different button of the input device. 4/5

### A.3 Conclusion

Based on the overall score, we have chosen to use the RotorS simulation environment. Instead of just replacing the model, world and dynamics, we need to assure that the resulting system is adaptable and has a modular architecture. In the case of the Hector Quadrotor, no extension is necessary, every sensor is readily available, all the necessary topics are created.

Therefore, in the next phase, the RotorS simulation will be adapted to the desired framework.

Future tasks include:

1. **File directory documentation:** document all the directories and draw a detailed structure diagram.

2. **Gazebo-ROS Interface:** Customise topic publications that will interact with future ROS nodes.

3. **Model/World Replacement:** Replace existing UAV models with Morpheus and the surrounding environment.

4. **Edit drone dynamics:** Investigate similar drone dynamics, modify existent dynamics and define parameters.

5. **Target drone:** Add a target drone to the simulation capable of performing pre defined flight manoeuvres.
Appendix B

Drag Experimental Results

In this Appendix, we will put to display a summary of all the Experimental Tests involved in the Drag estimation.

To estimate the Drag, the UAV was set to perform constant velocity circuits. When the velocity and height is constant, one can measure the Drag Force, with the turning angle.

B.1 Drag Estimation Results X-axis

The output of these experiments culminated in producing a drag estimate in the $x$-axis, illustrated in Figure 6.8a.

In Table B.1, there is a summary of all the trial performed. Because sometimes there were flight perturbations, as insufficient lighting or protection net disturbance, damaging the Motion Capture System’s estimation, from all the successful flight tests, only a fraction were selected.

<table>
<thead>
<tr>
<th>Velocity $v_x$</th>
<th>0.5</th>
<th>1.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trials Performed</td>
<td>17</td>
<td>15</td>
<td>14</td>
<td>17</td>
<td>20</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Trials Used</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>11</td>
<td>12</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Table B.1: Summary of trials performed for $x$-Drag Estimation

The Figures B.1 to B.7, contain the used data for the Drag estimation.
Figure B.1: Drag Test $v_x = 0.5 \ [m/s]$

Figure B.2: Drag Test $v_x = 1 \ [m/s]$

Figure B.3: Drag Test $v_x = 1.5 \ [m/s]$

B.86
Figure B.4: Drag Test $v_x = 2 \text{ [m/s]}$

Figure B.5: Drag Test $v_x = 2.5 \text{ [m/s]}$

Figure B.6: Drag Test $v_x = 3 \text{ [m/s]}$
B.2 Drag Estimation Results Y-axis

The output of these experiments culminated in producing a drag estimate in the \(x\)-axis, illustrated in Figure 6.8b.

The summary of experiments performed are depicted in Table B.2. For some velocities, due to time constrains, it was not possible to obtain representative data. Therefore, these trials were not used in the Drag estimation.

<table>
<thead>
<tr>
<th>Velocity (v_y)</th>
<th>0.5</th>
<th>1.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trials Performed</td>
<td>4</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Trials Used</td>
<td>0</td>
<td>8</td>
<td>13</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Table B.2: Summary of trials performed for \(y\)-Drag Estimation

The Figures B.8 to B.12 contain the used data for the Drag estimation.

Figure B.8: Drag Test \(v_y = 1 \, [m/s]\)
Figure B.9: Drag Test $v_y = 1.5 \ [m/s]$

Figure B.10: Drag Test $v_y = 2 \ [m/s]$

Figure B.11: Drag Test $v_y = 2.5 \ [m/s]$

B.89
Figure B.12: Drag Test $v_y = 3 \ [m/s]$