UNIVERSITÀ DEGLI STUDI DI PALERMO

Information, Communication and Technologies
Department of Engineering

AN ADVANCED HEXACOPTER FOR AUTONOMOUS EXPLORATION OF MARS:

ATTITUDE CONTROL AND NAVIGATION STRATEGIES

AUTHOR
Laura Sopegno

ADVISOR
Prof. Patrizia Livreri
Prof. Kimon P. Valavanis
Prof. Adriano Fagiolini

XXXVII Cycle
ACADEMIC YEAR 2024/2025

Contents

Chapter 1

Introduction

1.1 Motivation and Rationale

Unmanned missions for space exploration have emerged during the last two decades. The *NASA Mars 2020* [54] project shows that such mission contests dominate the scientific interest. In the latest *Mars Sample Return* (*MSR*) mission, the overall system includes a robotic rover, *Perseverance*, and a small Mars Helicopter (*MH*), *Ingenuity* [4, 60] currently used to explore the Red Planet and look for signs of past life. The successful landing on Mars (2021) of the NASA helicopter *Ingenuity* and its flights on the Martian surface has opened the door to exploring Mars without facing the limitations of Unmanned Ground Vehicles (*UGVs*), underlying the importance of UAVs when it comes to extraterrestrial exploration and support the use of rotorcrafts in complex scenarios such as the second stage of the *MSR* [25, 35, 56] where two helicopters may contribute to the transportation of terrain samples in collaboration with the rover's activity. Moreover, the completion of such significant missions paves the way for future manned and unmanned exploration missions [14].

In the wide frame of space exploration, the use of UAVs can allow for accurate monitoring and collection of detailed images of the Martian surface while navigating above different areas at different heights, not always feasible when using traditional rovers that require to move on variable and rugged terrains which, in turn, may cause the vehicle to move at a slow pace and limited range. Granted, orbiters have always provided aerial images of Mars, but with insufficient spatial resolution and near-surface capability when compared to data collected by UAVs.

1.2 Problem Statement

As stated, the MH has started a new era of planetary exploration as a technological demonstrator, proving for the first time the feasibility of flight on Mars. However, the current MH configuration neither carries a specific scientific payload (P/L) nor enhanced sensors onboard [39], limiting its capabilities to short flights over rock-free

and relatively flat terrains [4]. The consideration of this aspect serves as the primary motivation that has driven the investigation of the problem, considering the previous study on the MSH as a starting point for the configuration design [24].
The aim of this research is twofold:

The first part provides an in-depth analysis of an enhanced Mars Hexacopter (*MHex*) configuration, which includes the shape and size of the main subsystems as well as autonomous navigation and control strategies together. Starting with the mission analysis, the choice of the exploration site is based on both the scientific interest of the areas and previous data collected from the HiRISE satellite and rovers analysis, which underlies the evidence of morphological behavior on the surface. Particularly, the *Belva* crater is considered as the primary site for the exploration. Then, the flight trajectory is based on geological waypoints sent by the Ground Control Station (*GCS*) to the MHex, simulating a realistic mission case. The MHex shall autonomously navigate and map the surrounding terrain, proving the robustness of the GNC system without external communication aids. To this end, different navigation strategies are presented along with the implementation of model-based controllers for better trajectory tracking.

The second part targets the implementation of Deep Reinforcement Learning (DRL) algorithms for the GNC system of UAVs in the frame of space autonomy exploration. The analysis, validation and verification for the autonomous navigation based on a novel technique of the policy-based algorithms is implemented for the hexacopter to optimize waypoint navigation without prior knowledge of the surrounding environment. The DRL algorithms are improved by incorporating both Long Short-Term Memory (LSTM) and the Transformer architecture. Then, the design of a reward function tailored for continuous action spaces incorporating two main components: a position error term and a physics-informed component representing the minimization of the least action principle (*LAP*). Finally, a comprehensive simulation environment to evaluate the hexacopter's performance in completing mission tasks is developed. In the first phase of the simulation, the policy-based algorithms, Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG) are implemented in Python to be trained and then tested for guiding the UAV to a final destination in the continuous action space, while the 3D environment and hexacopter's physics are modeled using Gazebo and ROS plugins. The integration of the Python-based agent with the 3D environment allows for flexible control over the problem's physics and dynamics, as well as the ability to test different algorithms for different mission scenarios. Then, based on the superior performance of the PPO algorithm compared to its DDPG counterpart, a new environment, control and navigation setup is fully developed in Python and implemented alongside the mathematical modeling of a quadcopter. The quadcopter is chosen over the hexarotor because the final Python implementation is intended for HIL and flight phases, leveraging pre-existing quadrotor-like setups for real flight sce-

nario testing. Finally, according to the flight performance, the equivalent algorithm can be trained and tested on different rotor configurations, including adaptation for the hexacopter.

1.3 Proposed Solution

The selection of a six-rotor rather than a quadrotor or helicopter configuration is preferred because hexacopters demonstrate better stability, redundancy, payload capacity, and increased lifting power, representing a robust solution already addressed by NASA as a potential candidate for future missions on Mars. In the analysis of the GNC strategies, a new proposed approach deals with the implementation of the Unknown Input Observer (UIO) instead of the well-known Extended Kalman Filter (EKF), addressing the limitations of Kalman filters in the estimation of the MHex states. The UIO [45] is adopted to estimate both the MHex states and unknown disturbances that act on the system [45, 30, 69, 51, 8, 68, 36] during different flight phases. The communication delay between Earth and Mars cannot guarantee a continuous direct link for real-time data transfer, commands, and operations, as well as relying on GPS data, as stated above. Given such limitations, the presented research is based on some assumptions:

1. the MHex receives limited (transmitted - encoded) commands from the GCS;
2. the MHex shall be able to autonomously navigate and execute specific tasks in a GPS-denied environment along different mission profiles;
3. the MHex shall maintain a stable attitude along its trajectory, only relying on the onboard GNC system, optimized for different flight conditions.

1.4 Summary of Contributions

Aerial mobility is a promising solution for space exploration of inaccessible areas to surface vehicles, offering a comprehensive view of the Martian geological features together with the study of atmospheric conditions.
The objectives and aims of this research are summarized as follows:
First Part Contributions

1. Defining a specific mission profile for which Simultaneous Localization and Mapping ($SLAM$) and autonomous navigation strategies are developed to map the surface of the *Belva* crater and its surroundings;
2. Providing an accurate design of the proposed MHex following a comprehensive approach, starting with the aerodynamic analysis derived from the *Ingenuity* blade's profile partially based on available data in literature from [24, 5], together

with the sizing and choice of the onboard sensors for the implementation of the navigation strategies;

3. Developing a simulator for representing a realistic Martian environment and experiments simulation by integrating ROS, Gazebo and Ardupilot, which allows for implementation and testing of SLAM, navigation, and control algorithms;

4. Presenting an observer-based navigation algorithm to handle system nonlinearities and addressing the limitations of traditional approaches based on Kalman filters.

Second Part Contributions

1. Implementing a novel DRL-based autonomous navigation system using an improved DRL algorithms that integrates LSTM and transformer architectures, enhancing performance in high-dimensional, partially observable environments;

2. Designing a reward function tailored for continuous action spaces by incorporating a position error term and a physics-informed component based on LAP;

3. Developing a comprehensive simulation environment by integrating Python, Gazebo, and ROS, enabling the training and testing of the DRL agent for UAV navigation.

4. Implementing a fully-integrated setup in Python as a test bench for DRL algorithms to align with HIL and flight testing.

Chapter 2

Unmanned Exploration of Mars: Past, Present and Future Developments

2.1 Literature Survey

This section presents the main scientific contributions available in the literature on the exploration of Mars along with remarkable achievements and insights gained through the deployment of rovers, probes, and satellites, also providing an in-depth overview regarding the innovative approaches and technologies that promise to boost the future of Mars exploration.

2.1.1 History of Exploration on Mars

Over the past decades, robotic explorers such as the rovers *Spirit, Opportunity*, and *Curiosity*, as well as orbiters like the *Mars Reconnaissance Orbiter*, have significantly expanded our understanding of the Martian geology and climate, in addition to advancing research on the potential for human habitability. As a focal point, these missions provided crucial data for contemplating the possibility of past life on Mars as rovers traversed the Martian terrain. 2.1 summarizes the main explorer contributions to the Martian exploration. Rovers such as Curiosity and the current Perseverance have meticulously analyzed Martian rocks and soil, seeking biological markers and morphology features that could prove the existence of past, even present, life.
The discovery of ancient lakebeds, rivers and mineral formations has underscored the planet's watery history, enhancing our understanding of the Martian geological evolution.
After the first successful landing on Mars of the probes *Viking 1* and *Viking 2* in 1976 to search for signs of life [40] initial mapping experiments were performed, but with the *Mars Global Surveyor* launched in 1997 the topography and study of the Martian

Table 2.1: Mars Exploration Missions

Mission	Nation	Launch Date	Description
Mariner 4	US	1964	Flyby, first close-up images
Mariner 6	US	1969	Flyby, imaging data
Mariner 7	US	1969	Flyby, imaging data
Mariner 9	US	1971	Orbiter, mapping data
Mars 5	USSR	1973	Orbiter, imaging data
Viking 1	US	1975	Orbiter, mapping
Viking 2	US	1975	Orbiter, mapping
Phobos 2	USSR	1988	Probe, remote sensing
Pathfinder	US	1996	Lander and Rover, scientific data
Global Surveyor	US	1996	Orbiter, imaging data
Mars Odyssey	US	2001	Orbiter, imaging and remote sensing
Spirit	US	2003	Rover, scientific data
Opportunity	US	2003	Rover, scientific data
Mars Express	EU	2003	imaging, remote sensing
Mars Reconnaissance Orbiter	US	2005	Orbiter, imaging and remote sensing
Phoenix	US	2007	Lander, soil and ice experiments
Curiosity	US	2011	Rover, scientific data
InSight	US	2018	Lander, scientific data
Tianwen-1	China	2020	Orbiter and Rover, scientific data
Perseverance	US	2020	Rover, scientific data

atmosphere provided crucial data on Mars'climate and geology. Signs of past water activity and sedimentary rocks were identified for the first time by the *Spirit* and *Opportunity* rovers (2004 and 2010), indicative of a once wet and potentially habitable environment on Mars. In 2005, the *MRO* provided the first high-resolution features of the Martian surface, discovering evidence of recent liquid water flows. The still-active *Mars Odyssey* (2001) discovered vast amounts of ice below the Martian surface, also serving as a critical communication relay for other Mars missions. Still, the *Curiosity* rover made a big contribution in 2012 by detecting organic compounds in rocks as a potential support to past life. Seasonal variations in the atmosphere methane levels were identified too. Information about the interior structure of the planet came with the *InSight* lander in 2018 and still working on the surface, while the most recent Perseverance rover, which landed on Mars in 2021, is currently exploring the Jazero region searching for signs of past microbial life and collecting samples of regolith as a crucial part in the more articulated *MSR* mission.

2.1.2 Robotic Explorers: Advantages and Limitations

Robotic explorers played a crucial role in the history of space exploration. Particularly, rovers excelled in conducting detailed analyses of the Martian surface, providing critical insights into the planet's geological and mineralogical composition. However, their mobility and, in turn, the covered range, are strongly constrained and susceptible to the terrain composition, as demonstrated by *Spirit* and *Opportunity* which faced challenges with their wheels in soft sand. Orbiters still offer global perspectives in

site selection, but the imaging resolution does not provide an in-depth analysis of the ground, with limitations imposed by the Martian atmosphere [41, 13]. UAVs emerge as a promising alternative, surmounting some of the challenges faced by both satellites and ground explorers.

Previously, studies and concepts for designing drones were investigated in many literature works [8, 9, 17]. A summary of the main proposed rotorcraft architectures in the recent past and present studies is given in 2.2. Drones can access hard-to-reach locations, map wide portions of the explored site in a shorter amount of time, and play an active role in sample transportation thus complementing the strengths of a rover. The atmosphere of Mars represents a crucial aspect of flight feasibility, posing several challenges for any type of vehicle. This constraint becomes particularly

Table 2.2: Mars Rotorcrafts: Summary of past configurations

Year	Institution	Rotorcraft Type	Description
2002	University of Maryland	*MARV*, coaxial	50Kg, 4.26m rotor diameter, 25Km range
2002	Georgia Institute of Technology	*GTMARS*, quadrotor	10Kg, 1.84m rotor diameter
2006	University of Surrey	*VTLOL*, tailsitter	15Kg, 7 square meter wing area, 1.4m rotor diameter, 100 - 450Km range
2018	Georgia Institute of Technology	*UAV*, quadrotor	—-
2018	Tohoku University	*JMH*, coaxial quadrotor	1.05m diameter
2018	Delft University of Technology	*VITAS*, tailsitter	14Kg
2020	NASA	*Ingenuity, MH*, coaxial helicopter	1.8Kg, 1.21m rotor diameter
2020	NASA (JPL-ARC)	*MSH*, helicopter	4.6Kg, 1.21m rotor diameter 20Kg, 2.5m rotor diameter 20Kg, 1.28m rotor diameter

pronounced for drones, where the reduced air density makes it challenging to generate the necessary lift for sustained flight, requiring a rigorous design in the blade and rotor configuration. Due to the recent development of the MH, a significant amount of data is now available on coaxial rotors in the Mars atmosphere, enhancing our understanding of their operational dynamics in the study of the next generation of rotorcrafts.

2.1.3 Autonomous Exploration Strategies

As shown in 2.2, all configurations exhibit transversal aspects influenced by the flight conditions on Mars. These include a significant adjustment in motor rpm compared to terrestrial flight, to augment thrust in the thinner Martian atmosphere. Another

critical aspect is the ability to execute fully autonomous flights, eliminating dependence on communication signals from Earth due to the considerable signal travel time to Mars and delay. This autonomy is pivotal for efficient obstacle avoidance and path planning, considering the limitations in real-time responsiveness. Additionally, the dimensions of the rotorcraft system must adhere to constraints imposed by the aeroshell of the spacecraft, a crucial factor in ensuring the successful landing of the vehicle on the Martian surface.

Sensors: Review and Analysis in Planetary Exploration

In the robotic sector, terrain modeling and reconstruction techniques have grown in the field of autonomous navigation and Simultaneous Localization and Mapping ($SLAM$) for both short and long distances. Such techniques result particularly challenging in autonomous planetary exploration, where mobile robots must autonomously perceive and model the planet's surface with minimal or no intervention from ground stations.

During past exploration, the majority of rovers used cameras for terrain perception as onboard vision systems. The feasibility of stereo cameras as a 3D perception technology has been verified and tested by decades of planetary rover missions, also showing that Martian terrain has enough features for stereo vision to be applied across nearly all areas. The accuracy of the estimated depth with the increasing distance from the camera reduces significantly, compromising the resolution of the terrain perception. Other methods based on monocular vision may offer great potential for terrain modeling, with a lower computational demand as compared to stereo vision processing as well as detecting objects at a much further distance. Nevertheless, achieving accurate 3D reconstruction of the terrain from monocular images remains a highly challenging problem. Also, previous missions did not consider loop closure techniques, limiting the navigation to a one-way traverse. However, future missions such as MSR would require revisiting previously known landmarks multiple times while traveling longer distances at higher speeds. Therefore, the use of sensors and techniques beyond standard cameras is needed for more accurate scene reconstruction covering greater surface area with higher fidelity and feature complexity.

Sensors such as Radio Detection and Ranging ($RADAR$) and Light Detection And Ranging ($LIDAR$) have been extensively exploited for target detection, identification, and depth estimation in terrestrial robotics. If compared to LIDARs, RADARs leak precision in providing a detailed 3D mapping of the surroundings, posing difficulties and limitations in accurate exploration. In contrast, using LIDARs for 3D mapping with laser scanners is preferred due to their simplicity and high accuracy, achieving low-drift motion estimation while maintaining an acceptable computational complexity. LIDARs utilize time-of-flight (ToF) principles, obtaining depth measurements of the surrounding environment employed for 3D surface mapping and scene reconstruction. In space applications, LIDARs serve spacecraft by aiding in rendezvous and

docking.

Considering autonomous navigation strategies, while cameras capture visual information to perform SLAM, LIDARs provide point clouds for geometric data. However, regardless of the SLAM approach (vision-based or LIDAR-based), a single sensor system has limitations, resulting in a fragile and uncertain system. Indeed, while vision methods rely heavily on initialization and variations in illumination, LIDARs, on the other hand, offer sparse information that quickly degrades positioning accuracy in unstructured scenes, introducing instability into the system. The IMU perceives slight changes in the system in a relatively short period, but long-term drift is inevitable. In addition, the rapid motion mode and long-term error accumulation further invalidate the odometer process.

Based on the purpose of this research, *Camera-LIDAR* fusion technique together with the IMU is used to overcome the limitations of individual sensors with potential for application in UAV planetary exploration. Indeed, the accurate mapping of visually perceived information from a camera onto LIDAR data for terrain modeling and reconstruction optimizes the benefits of both sensors. This approach mainly integrates saturated texture features with highly accurate depth information, spanning hundreds of meters.

Multi-Sensor Fusion SLAM Review

Over the past few decades, the SLAM problem has evolved from two independent modules, localization and mapping, into a more complex and integrated system. The SLAM technique creates a map of an unknown environment while simultaneously determining the system's location within that environment. Both mapping and localization are performed simultaneously, allowing the system to update its understanding of the environment and its position when moving. SLAM systems can rely on either single sensors or integrated sensor data, as well as computational methods for efficient real-time operations.

The selection of SLAM as the primary technique for autonomous exploration on Mars is driven by the fact that it results crucial when a pre-existing map is unavailable.

In the SLAM problem, the concept of modularity is frequently employed to introduce independence between sensor data processing and the SLAM algorithm. This architecture enables the system to accommodate diverse sensor configurations, providing flexibility and facilitating the integration of new sensors and techniques. Numerous solutions have been proposed so far considering modularity. In [37], sensors are classified into *idiothetic*, *absolute allothetic*, and *relative allothetic* categories. The distinction between absolute and relative allothetic lies in the coordinate frame. Idiothetic sensors rely on intrinsic data from the platform, such as IMUs, to incrementally estimate the robot's position. In contrast, relative allothetic sensors such as lasers, cameras and LIDARs use external features or landmarks for position estimation in a local frame,

while absolute allothetic sensors rely on a global coordinate frame (GPS, GNSS). This categorization facilitates uniform treatment of similar sensors, enhancing reusability. However, when the modularity approach considers the type of sensor, the primary drawback arises when a sensor or technique deviates from the defined standards, rendering it incompatible with the framework. Nevertheless, modularity in SLAM can be applied in various other dimensions, beyond sensor types, to address specific challenges and enhance system flexibility. One notable modularity approach in modern SLAM **systems** involves two main components: the *front-end* and the *back-end*. The *front-end* focuses on real-time estimation of the current frame pose and the storage of corresponding map information, while the *back-end* is responsible for large-scale pose and scene optimization, including loop closure detection, as shown in 2.1. Considering

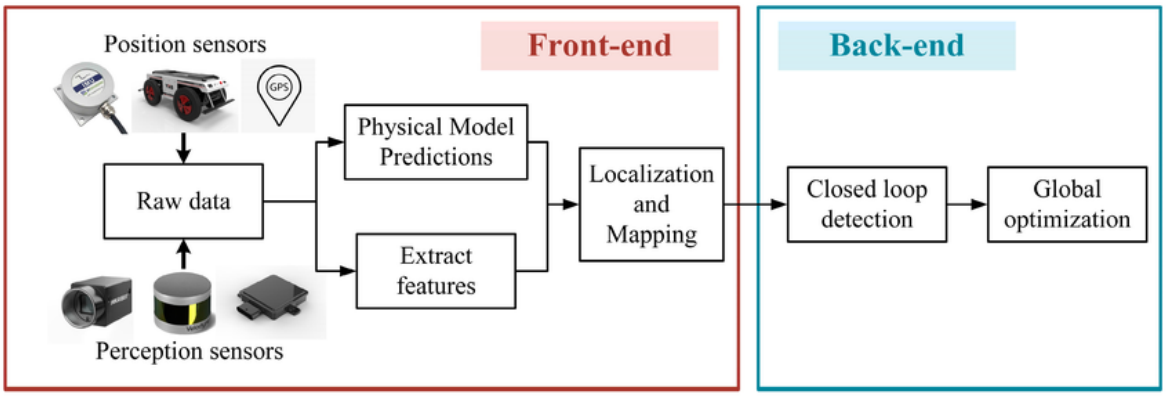


Figure 2.1: SLAM system: main components, [68]

the back-end stage, the different combinations of LIDAR, camera and IMU lead to two main subcategories of SLAM modes: *loosely coupled* and *tightly coupled*. In a loosely coupled system, measurement data from individual sensors is processed independently, and the system then merges this data in a filter that marginalizes the current frame's information to yield the most recent state estimation results. On the other hand, a tightly coupled system jointly optimizes measurement data from all sensors, integrating observation characteristics and physical models from each sensor to achieve a more robust pose estimation. The loosely coupled system deals with minimal computational load, a simple system structure, and easier implementation. However, its positioning accuracy tends to be limited. In contrast, the tightly coupled system, while computationally demanding and challenging to implement, excels in providing a more accurate state estimation, particularly in complex and/or dynamic environments.
Since the autonomous exploration of Mars requires a coherent and consistent estimation of the surroundings, the choice of a tightly coupled method as a back-end strategy is preferred to achieve better overall accuracy in the estimation process.
The back-end module is thus responsible for implementing the SLAM algorithms. Over the years, a wide range of SLAM algorithms have been developed, and the three most popular SLAM approaches can be divided into: EKF, Particle Filter (*PF*), and Graph-based SLAM.

Considering the purpose of this research the **graph-based SLAM** approach is chosen as it offers advantages over filtering methods due to its inherent flexibility and ability to handle complex scenarios. Indeed, unlike EKF and PF which rely on specific assumptions and may struggle with non-linearities, noise type and high-dimensional state spaces and matrices, graph-based SLAM represents the environment as a graph, where nodes correspond to robot poses and edges capture the relationships between them. This graph structure allows more accurate and robust modeling of uncertainties, loop closures, and sensor measurements. Furthermore, graph-based SLAM facilitates the incorporation of various sensor modalities and heterogeneous data sources. In the last decades, the graph-based SLAM approach became popular and several improvements in its performance have been made. Two different SLAM algorithms are implemented: Lidar Odometry and Mapping *LOAM*, and Real-Time Appearance-Based Mapping *RTAB-Map*. LOAM is tested at SOftware-in -the-Loop *(SIL)* level, while RTAB-Map is considered as a reliable solution for the Hardware-in-the-Lop *(HIL)* setup. Indeed, while LOAM provides odometry and point cloud maps as its primary outputs to achieve precise 3D mapping of the environment, RTAB-Map, a specific graph-based SLAM approach that can be categorized as a tightly coupled system, also incorporates loop closure detection techniques to recognize previously visited locations (thus closed loop in the generated maps). An in-depth description of such techniques is given in 3.5.

2.2 Conclusions

This chapter covered the main contributions to the exploration of Mars over the last decades, underlying the strengths and limitations of unmanned robots exploring the Red Planet. UGVs are limited in coverage of long paths as they deal with ground imperfections, while probes can span wide areas gathering images of the surroundings, but leaking in-depth data information. UAVs are a promising solution as they overcome both limitations, despite the challenges associated with the complexity of their design and operational flight conditions. So far, the Ingenuity demonstrator has proved the feasibility of the flight on Mars, and atmospheric data are being collected to improve the future design of the next prototypes [61]. Because of its demonstrative nature, the system carried onboard Ingenuity is limited in terms of autonomous exploration, motivating the investigation of different autonomous navigation strategies considering both Components Off The Shelf (*COTS)* and techniques currently being applied for terrestrial flights only. SLAM emerges as a viable solution for autonomous localization and mapping through complex environments. Particularly, the potentials of graph-based SLAM is underlined and considered for the autonomous navigation of the MHex.

Chapter 3

Mission Profile and System Design

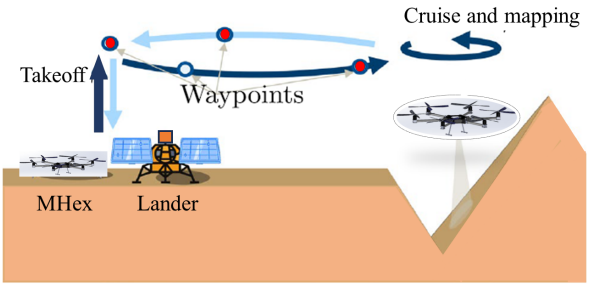
3.1 Problem Statement

This work presents the conceptual design for a Mars Hexacopter to perform an autonomous exploration mission for mapping a partial area in the Jezero region. The MHex must achieve the mission goal by autonomously exploring a GPS-denied environment and without relying on external communications from the terrestrial GCS. The system configuration is described along with the main onboard subsystems, mainly focusing on the GNC system for the implementation of observer-based navigation and control.

3.2 Mission Profile and System Design

The exploration mission consists of five main segments: takeoff, climb at the required altitude, cruise, descent and landing. Geological waypoints are sent from the GCS to the MHex after previous analysis of the flying site, according to satellite images. After completing the mapping task, the MHex recharges for 1sol and then continues the mission with a target scanning range of about 3 Km^2 each flight. This represents the operative area including slight deviations from a direct course [55]. Upon completing the mapping process in the designated area, the MHex returns to the lander to offload the accumulated scientific data, facilitating the optimization of onboard processor weight and power. In the context of the current mission, a 34 Km^2 region encompassing the Belva crater in the Jezero region and its vicinity has been selected for autonomous exploration. This area holds significant scientific relevance [57], [54] as is deemed suitable for surficial deposits of ichnofossils such as morphological evidence of past biological behavior. 3.1 illustrates the mission concept.

(a) Belva crater site. HiRISE database [38]



(b) MHex mission profile

Figure 3.1: MHex mission scenario for autonomous mapping of the Belva crater

3.2.1 Mission Requirements

3.1 summarizes the technical requirements of the mission. The definition is given according to *Functional, Interface, Operational* and *Implementation* requirements.

3.2.2 Mars Atmosphere

The Martian atmosphere directly impacts the MHex flight conditions and, in turn, its configuration. Furthermore, because of such atmospheric composition, power sources such as oxidizing cannot be used, and the gravitational acceleration g and atmospheric density ρ become driving factors in lift generation. The low-density that reflects low Reynolds numbers, *Re*, in the range between *10000* to *25000*, significantly impacts the airfoil design and aerodynamic performance, and suggests the use of lightweight materials for the hexacopter structure. Furthermore, the lower sound speed limits the rotors' maximum rotational speed. 3.2 compares the Earth and Mars atmosphere, also considering the atmospheric conditions of the Jezero region, which impact the design and analysis of the MHex. The speed of sound on Mars refers to a Mach number (M) of *0.7*.

3.3 System Architecture

The design of the MHex is done considering both environmental and sizing constraints. After completing one flight, it does either return to a lander to download the data collected during the mapping process, or continue the mission until the onboard gained data are consistent, thus remain one night on the Martian surface and autonomously recharge with onboard solar cells. The analysis of the MHex design does consider the spring season in the Jezero region, and the operational flight conditions refer to an average density ρ of 0.015 Kg/m^3 and temperature T of -50° C.

Table 3.1: Technical Requirements Definition

Number	Requirement
Functional and Performance Requirements	
R1	The MHex shall be able to autonomously depart from a lander/rover, navigate along geological waypoints, map the surrounding, and land at the last waypoint.
R2	The MHex shall be equipped with a power system capable to support the whole mission.
R3	The MHex shall be equipped with a propulsion system capable to support the whole mission.
R4	The MHex shall be able to 3D map the site, store the collected data on-board and transmit them to the rover/lander at the end of (TBD) mission segments.
R5	The MHex P/L shall not exceed 2 Kg.
R6	The MHex gross mass shall be less than 20 Kg.
R7	The MHex shall be able to cover an area of 3 Km^2 (TBC) each flight.
R8	The MHex shall be able to reach a cruise altitude up to 10 m.
R9	The MHex shall be able to reach a cruise speed up to 8 m/s.
Interface Requirements	
R10	The MHex thermal subsytem shall withstand day and night cycles on Mars without performance degradation.
R11	The MHex shall be able to withstand the launch and transport loads solicitation when stowed on the launcher.
Operational Requirements	
R12	The MHex shall be able to navigate and map the Mars environment according to onboard sensors.
R13	The MHex shall be able to fly autonomously by means of geological waypoints sent from the GCS.
R14	The MHex shall be able to navigate and map the Mars environment according to onboard sensors.
Implementation Requirements	
R15	The MHex SIL shall be able to simulate the vehicle physical architecture and environmental conditions with high fedelty.
R16	The MHex HIL shall be able to reproduce the SLAM technique simulated at SIL level.
R17	The MHex prototype shall be able to reproduce the vehicle physical scaled architecture.

Table 3.2: Atmosphere comparison on Earth and Mars (Jezero region)

	Unit	Earth	Mars
Grav. acceleration, g	m/s^2	9.81	3.71
Temperature, T	K	288	223
Density, ρ	kg/m^3	1.225	0.015
Viscosity, μ	Ns/m^2	1.75x10−5	1.13x10−5
Sound speed, a	m/s	340.3	233.1
Tip speed, V_{tip}	m/s	238	163

The structure of the MHex accounts the volumetric implications of fitting different folding configurations in the launcher aeroshell. The most efficient utilization of the volume shall be identify, considering the limitations of the latter on the blade design. This study considers a reference rotor radius of about 0.64 m, as derived in [24]. Details are given in 3.3.1.

3.3.1 MHex Sizing: Subsystems and Sensors

This section covers the main subsystems of the MHex, providing a description of the main components analyzed in the design phase. The whole design includes a 20% contingency margin of the aircraft empty weight to account for modeling uncertainties. 3.3 summarized the parameters and weights of the MHex.

Table 3.3: MHex design. Geometry and weights

Parameters		Value	Unit
Airframe		**15.5**	Kg
	Structure	5.9	Kg
	Propulsion	4.3	Kg
	Avionics and equipment	2.7	Kg
	Contingency (20%)	2.58	–
P/L		**2.0**	Kg
Gross weight		**17.5**	Kg
Disk radius		0.64	m
Rotor numbers		6	–
Blades per rotor		4	–
Rotor average speed range		2483	rpm
Airframe size	*length*	3.15	m
	width	3.15	m
	legs height	0.50	m

Payload

The MHex P/L is intended as a sensing payload, containing the 3D LIDAR sensor (plus miscellaneous) for mapping the environment. A 2.0 Kg P/L is identified for the MHex. The scientific device draws inspiration from the Overmap 3D LIDAR [20] for its scanning capabilities and the reliability it offers in complex scenarios. Technical specification of the P/L are provided in 3.4 and 3.2.

Structure

The preliminary design concept considered the MH weight and power to calibrate the initial MSH airframe. According to optimization considerations derived in this study, such as a better compensation of the drag effects acting on the rotors when rotating, the MHex frame size is increased from a baseline of 3 m to 3.15 m. The materials considered

Table 3.4: MHex P/L Technical Specification

Sensor	
VelodyneVLP-16 Lite LIDAR	- 100 meter range - 16 channel - dual return - 300,000 point per second - Rotation, 360°x360° FoV
Physical	
Size	393x155x150 mm (L x W x H)
Weight	1.8 Kg
Power	12V-54V, Max 90W

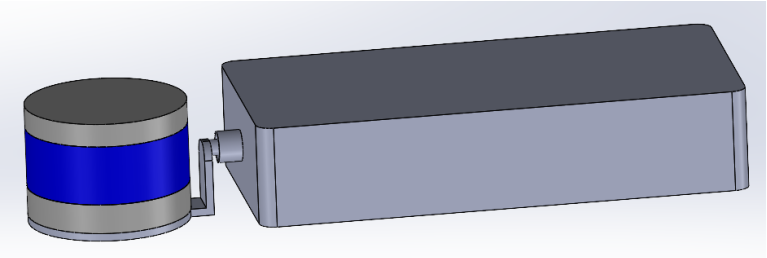


Figure 3.2: *Solidworks* design of MHex scientific P/L, inspired by [20]

for the body and legs are carbon fiber and composite, as both offer stiffness and a high strength-to-weight ratio, as well as thermal stability and corrosion resistance.

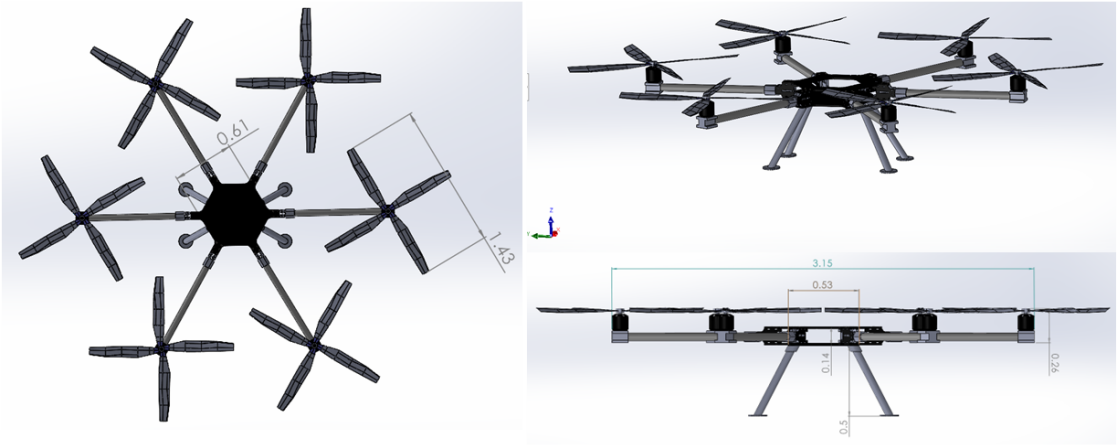


Figure 3.3: MHex frame

Propulsion and blade profile

The six motors are sized with a 150% control margin hover power, accomodating a motor sped up to 3008 rpm. The MHex blade is derived from the fluidic analysis in [24], considering the same diamond-shaped airfoils for the blade inboard sections with an 8% thickness-to-chord ratio and flight (hover and cruise) performance for M 0.55 to 0.8. In the aforementioned MSH conceptual design, the reference airfoil sections are

already optimized for better stall behavior and lower power during flight, leading to the CAD (*Solidworks*) implementation of the same design (3.4). As noticed in the figure below, the blade presents a -18 *deg* linear (root to tip) twist, resulting in an optimized hover and low speed rotor performance.

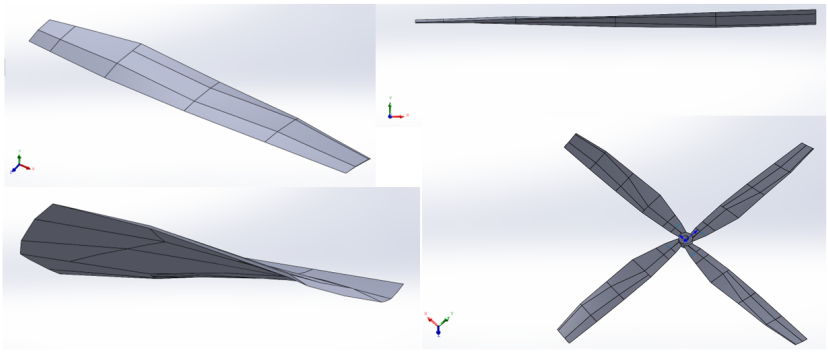


Figure 3.4: MHex blade profile and rotor configuration

3.4 Observer-based Navigation Strategy: Unknown Input Observer

In complex scenarios such as the Martian environment, stability and exploration go along with the robustness of the MHex external and unknown disturbances acting on the system. As a matter of fact, external disturbances and system uncertainties, such as sensor noise, directly impact both navigation accuracy and control integrity. To address this, Kalman filters have emerged in recent decades as a prominent category of filters to tackle various challenges in different operative scenarios, providing a more stable and reliable estimation of the UAV' states. By integrating information from multiple sensors and compensating for measurement errors, Kalman filters enhance UAV navigation accuracy. Among these, the Extended Kalman Filter (*EKF*) has gained widespread application thanks to its simplicity and adaptability across a wide range of operational contexts, establishing a standard approach for nonlinear system estimation [50], [17]. However, they are constrained by the requirement of linearization of the system dynamics, choice of initial conditions and, above all, Gaussian noise distribution, requiring complex tuning of parameters and covariance matrices especially with high-dimensional systems. Non-Gaussian distributions lead to biased estimates and, thus, reduced filter performance. The Unscented Kalman Filter (*UKF*) [66] offers an alternative solution to handle nonlinearities more effectively compared to the EKF. By employing a deterministic sampling technique to propagate a set of sigma points through the nonlinear functions, the UKF provides a more accurate estimate of the

states, even though this advantage comes at a cost of higher computational complexity and accurate parameter tuning. Other nonlinear filters, like Particle Filters (*PF*) [49], leverage a set of discrete representative samples to approximate the probability density function (*Monte Carlo*) of the system states, resulting in effective handling of non-Gaussian and nonlinear systems as well as multimodal distributions. Nonetheless, for high-dimensional state spaces, a significant increase in the number of particles may be necessary.

Compared to traditional filters, state observers deal with systems with noisy inputs, providing accurate state estimates even in the presence of significant external disturbances or measurement noise. In particular, the Unknown Input Observer (*UIO*) estimates and compensates both the state variables and unknown input affecting the system without requiring any assumption or description of the disturbance as they are assumed to be unknown or not measurable. There exist applications implementing the UIO observer [2], [3], [52],[42],[46]. Considering the simplicity and efficiency of the observer as a robust solution for estimating system states and compensating for disturbances in relation to traditional filters (as the EKF), performance evaluation and comparison of EKF and UIO is conducted with the main focus on estimating the states of the MHex in the context of the autonomous exploration mission. Since the theory and implementation approach of EKF is well known, the reader is referred to [50] for further details. The next section provides the analytical description of the UIO, while 4.1 provides the full implementation and performance comparison of EKF and UIO according to the MHex mathematical model derived in 3.4.

MHex Mathematical model

The hexacopter under consideration is depicted in Figure 1. It has a symmetric and

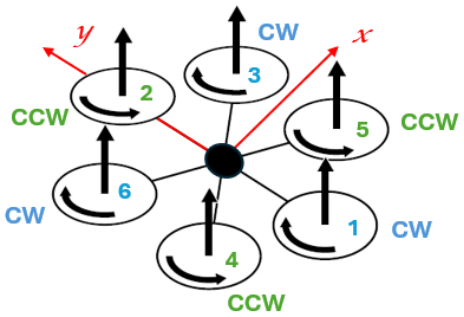


Figure 3.5: MHex configuration

rigid structure with a diagonal and symmetric body-fixed inertia matrix of the form

$$I_B = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

Position and attitude in Euler angles are expressed in the inertial frame with the vectors η and ζ, respectively, and then grouped in $q \in \mathbb{R}^6$

$$\zeta = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad q = \begin{bmatrix} \zeta \\ \eta \end{bmatrix} \tag{3.1}$$

The conventional hexacopter model is defined and adapted according to the dynamic in [31], [33]

$$\ddot{\zeta} = \frac{1}{m} TRe_3 - ge_3, \tag{3.2}$$

$$\ddot{\eta} = J_r^{-1}(W^T\tau - C\eta) \tag{3.3}$$

$$B\ddot{q} + C\dot{q} + g + G_{gyro}\dot{q} + D_{par}\dot{q} = \Lambda\Omega \tag{3.4}$$

with m the hexacopter total mass, $\xi = \Lambda\Omega$ and $R \in R^{3\times3}$ the rotation matrix of the form

$$R = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\theta)s(\psi) \\ s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & -s(\theta) & s(\phi)c(\theta) \\ c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) & c(\phi)c(\theta) \end{bmatrix}$$

where *s(x) = sin(x), c(x) = cos(x)*, and *t(x) = tan(x).*
Vector $\mathbf{e}_3 = [0,\ 0,\ 1]^T$ represents the inertial z-axis. g denotes gravitational acceleration, and $J_R = W^T JW$ is the Jacobian matrix, while the transformation matrix W relates angular velocities ω to Euler rates $\dot{\eta}$, allowing for the kinematic inversion from the inertial frame to the body-fixed frame. J is the symmetric constant inertia matrix. Matrix C incorporates centrifugal and Coriolis effects. To manage hexacopter stability, a *PID* controller is implemented as a feedback control algorithm. The controller design is based on the hexacopter configuration (3.5), given the matrix form Λ

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ k & k & k & k & k & k \\ -k & k & \sqrt{3}/2 & -\sqrt{3}/2 & -\sqrt{3}/2 & \sqrt{3}/2 \\ 0 & 0 & -\frac{1}{2}k & \frac{1}{2}k & -\frac{1}{2}k & \frac{1}{2}k \\ b & -b & b & -b & -b & b \end{bmatrix}$$

Therefore, the relation between the total thrust $T_t = [0,0,T]$ and the torque vector $\tau_v = [\tau_\phi,\ \tau_\theta,\ \tau_\psi]$ and the rotor angular velocities Ω is given by

$$\begin{bmatrix} T_t \\ \tau_v \end{bmatrix}^{6\times1} = \Lambda \times \Omega \tag{3.5}$$

Vector $\mathbf{\Omega} = [\omega_1^2\ \omega_2^2\ \omega_3^2\ \omega_4^2\ \omega_5^2\ \omega_6^2]^T$ is the controller input corresponding to the rotor angular velocity square.

Unknown Input Observer

Consider the continuous-time nonlinear model (3.4); following the reasoning made in [44], [47], with the state vector $x = [q^\top, \dot{q}^\top]^\top$, output $y = q$ and, finally, by extracting the linear and time-invariant components from the state equation (3.4), the hexacopter dynamic can also be modeled according to the state space representation

$$\begin{aligned} \dot{x} &= A\,x + B\,u + W_d\,\delta, \\ y &= C\,x + D\,u + H\,\delta \end{aligned} \tag{3.6}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^p$ and $\delta \in \mathbb{R}^v$ are the known and unknown input vectors respectively, $y \in \mathbb{R}^c$ is the output vector, and A, B, C, D, W and H are matrices of suitable dimensions. Without loss of generality, matrices D and H in (3.6) are assumed null, thus the output matrix only depends on the state. Considering the L-th Taylor series vectors U^L and Γ^L of the known and unknown inputs, i.e. $u(t)$ and $\delta(t)$, the L-th system's response vector is given by

$$Y^k = CA^k\,x + \textstyle\sum_{i=0}^{k-1} CA^{k-1-i}\left(B\,u^{(i)} + W_d\,\delta^{(i)}\right) \tag{3.7}$$

Or, in a compact form

$$Y^k = O^k\,x + J_u^k\,U^k + J_\gamma^k\,\Gamma^k \tag{3.8}$$

where O^k, J_u^k and J_γ^k are the k-th order observability, known and unknown input invertibility matrices, respectively. U^k and Γ^k are the k-th Taylor series vectors of the known and unknown inputs, i.e., u and δ. In this setting, according to the analytical reasoning described in [58], the following

$$\dot{\hat{x}} = E\,\hat{x} + F\left(Y^k - J_u^k\,U^k\right) + B\,u \tag{3.9}$$

represents the dynamics of the UIO for the system in (3.6), with E and F being the two design matrices. Substituting (3.8) in (3.9), the state estimation error dynamics, i.e., $\dot{\tilde{x}} = \dot{\hat{x}} - \dot{x}$, leads to

$$\begin{aligned} \dot{\tilde{x}} &= E\,\hat{x} + F\left(Y^L - J_u^L\,U^L\right) + B\,u - A\,x - B\,u - W\,\delta = \\ &= E\,\tilde{x} + F\left(Y^L - J_u^L\,U^L\right) + (E - A)\,x - W\delta = \\ &= E\,\tilde{x} + F\,O^L\,x + F\,J_\gamma^L\,\Gamma^L + (E - A)\,x - W\delta = \\ &= E\,\tilde{x} + \left(F\,O^L + E - A\right)x + F\,J_\gamma^L\,\Gamma^L - W\,\delta \end{aligned} \tag{3.10}$$

Finally, if the following conditions are simultaneously satisfied

$$\begin{aligned} F\, J_\gamma^k &= [W_d, 0_{n\times v}] \\ E &= A - F\, O^L \end{aligned} \tag{3.11}$$

The solvability of which is guaranteed if and only if (3.6) is invertible and strongly observable, i.e., for some $k \leq n$ must hold

$$rank(J_\gamma^L) = v + rank(J_\gamma^{L-1}), \tag{3.12}$$

$$n = rank([O^L, J_\gamma^L]) - rank(J_\gamma^L) \tag{3.13}$$

Then, the estimation error dynamics is

$$\dot{\tilde{x}} = E\, \tilde{x} \tag{3.14}$$

and through appropriate choice of the gain matrix E, it converges to zero.

3.5 GNC System

As mentioned before, the MHex can explore harsh environments and remote areas otherwise denied to ground vehicles. For this reason, a robust and reliable GNC system is required to perform both advanced control stability and autonomous GPS-denied navigation.

3.5.1 Sensors

The VLP-16 3D LIDAR [20] has been selected as the primary sensor for mapping the Belva crater. Although cameras are effective in SLAM systems, LIDARs have enabled advances in the knowledge of celestial bodies such as the Moon, Mercury, Mars, and various asteroids [11], promising a pivotal role in future planetary science endeavors. A notable study presented at the *50th Lunar and Planetary Science Conference* in 2019 [27] focused on geological investigations at the Lofthellir site (Iceland), proposing the exploration of lava tubes on the Moon and Mars using a 3D LIDAR-equipped drone. This study showcased promising initial results, including successful cave mapping. Additionally, NASA studies [1] aim to developing advanced precision landing technologies and enhancing terrain scanning accuracy through LIDAR-based navigation to optimize the descent and landing phase of landers. 3D LIDARs offer rich structural information about the environment and operate effectively in low-light or varying illumination conditions, without relying on external light sources (e.g. LEDs), which increase onboard power consumption. The resulting point cloud provides depth information directly from the environment's geometry, eliminating the need for stereo matching or depth inference, thereby reducing reliance on complex feature extraction algorithms required

by camera-based systems.

The MHex LIDAR incorporates a tilting mechanism to explore both wide areas, such as craters, and caves, enabling obstacle detection and collision avoidance by scanning the surrounding ground and vertical range at a rotation rate of 1 Hz (3.6). However, single-sensor systems often exhibit fragility and uncertainty during localization. To address this, the MHex carries three different onboard sensors to ensure precise SLAM navigation. A VLP16 LIDAR is chosen as the main sensor for the 3D mapping of the environment, while odometry data are integrated among the tracking camera and altimeter fusion. The Garmin Lite v3 altimeter, currently used onboard the Ingenuity helicopter [16], is a light, optical distance measurement sensor that uses infrared pulses of light projected down to the ground to measure the flight altitude of the MHex. 3.5 provides a summary of the sensor features [65, 23] used onboard the MHex.

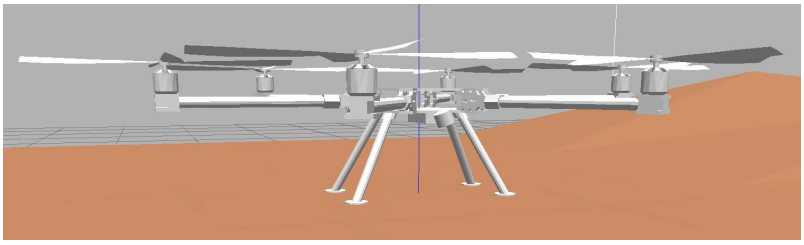


Figure 3.6: MHex LIDAR tilting mechanism, Gazebo simulation

Table 3.5: MHex Sensor Specifications

Sensor	Frequency	Power	Range	Weight
VLP-16 **3D LIDAR**	5-20 Hz	8 W	<100 m	590 g
t265 **Tracking camera**	30-200 Hz	1,5 W	173°	55 g
Lite v3 **Altimeter**	10-30 Hz	1,3 W	<40 m	22 g

3.5.2 GNC architecture

3.7 provides the GNC framework for the MHex. In the proposed system, the *Perception and Estimation* block collects the measurements coming from a 3D LIDAR, a downward laser altimeter, a tracking camera and the onboard IMU to obtain partial information about the MHex's state through the sensor fusion of different odometry estimations (LIDAR Inertial Odometry, *LIO*, and Visual Inertial Odometry, *VIO*). Then, the UIO estimates both the unknown inputs of the dynamic system, which are not directly measured by the onboard sensors, and the full state of the MHex. The *Planning* block performs the optimal trajectory given a series of waypoints sent from

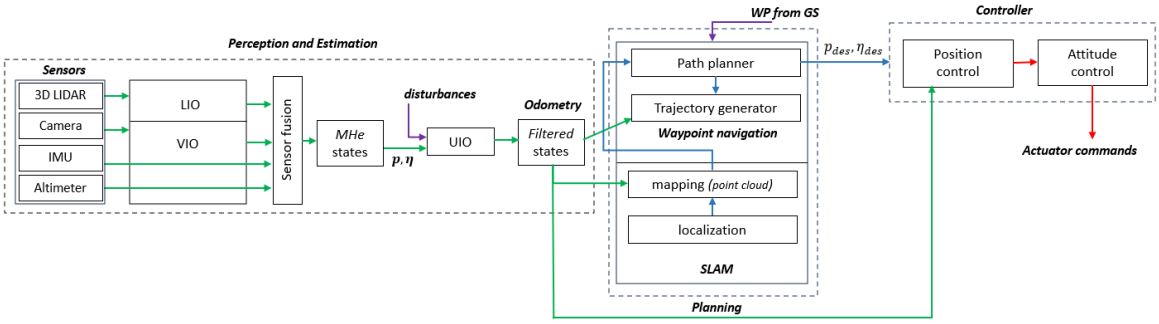


Figure 3.7: GNC subsystem: Perception and Estimation, Planning and Control frameworks

the ground station, and the SLAM algorithm allows for the autonomous mapping of the surrounding environment, in turn sending feedback to the path planner for trajectory optimization. The *Controller* block computes the control action required to steer the MHex along the desired trajectory. Given the underactuated property of multirotors, the control structure is divided in outer (*position*) and inner (*attitude*) loop. The stock position controller within Ardupilot computes the attitude trajectory to reach the mission waypoints while the custom deployed inner loop controller provides the control action for attitude tracking.

3.5.3 LOAM SLAM

As described in the GNC framework of 3.7, the *Planning* block is in charge of implementing the SLAM process along with the path planning algorithm. 3D LIDARs permit to capturing of fine details of wide environments without being affected by variable illumination conditions. The SLAM method is also preferred in the context of the mission because of the complexity of autonomous navigation in a GPS-denied environment, thus a dedicated localization and mapping technique is required from the MHex to perform the mission tasks. The SLAM framework implemented onboard the MHex is shown in 3.8.

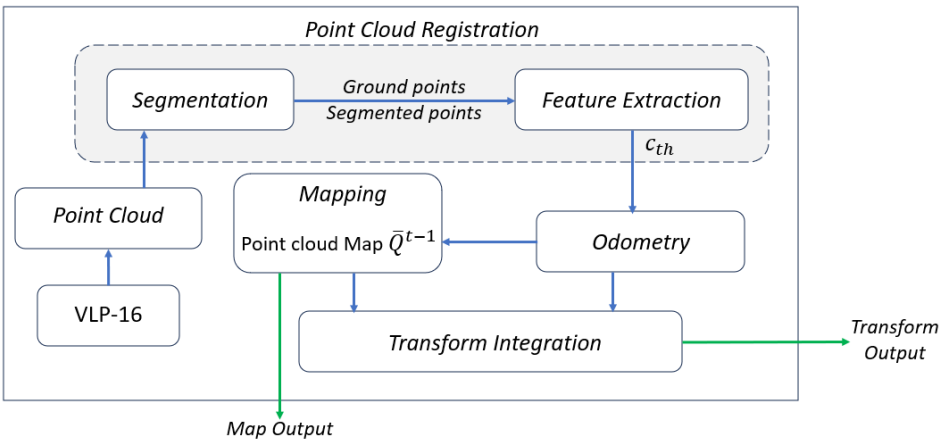


Figure 3.8: 3D LIDAR Odometry and Mapping system overview

The SLAM system receives data input from the VLP-16 LIDAR and then generates outputs regarding the computed map (*Map Output*) and the pose with respect to the generated map (*Transform Output*). The overall SLAM system is divided into four modules: In the *Point Cloud Registration* module, the *Segmentation* takes the scan's point cloud $P_t = \{p_1, p_2, ..., p_n\}$ acquired at time t and, based on the VLP-16 horizontal and vertical resolution (0.2° and 2°) projects it onto a range image of resolution 1800 by 16. The points p_i in the point cloud are then represented in the range image by a unique pixel with a corresponding Euclidean distance r_i from the sensor. A second level of clustering is applied to the range image and the selected points are grouped into different clusters of 50 points each. A final selection only keeps and saves features of characteristic points (ground points and segmented points) in the range image allowing for an efficient computation of the entire data processing. In the *Feature Extraction* module the selected points are processed and sorted based on their roughness c with respect to a threshold c_{th}

$$c = \frac{1}{|S|.||r_i||} || \sum_{j \in S, j \neq i} (r_j - r_i) || \tag{3.15}$$

with S the points set. Points for which $c > c_{th}$ are classified as *edge features*, while *planar features* for c smaller than c_{th} and then saved in two different sets F_e and F_p. In the *Odometry* module, the sensor motion is estimated between two consecutive scans by finding the corresponding features between points in the feature sets and the ones collected in the previous scan. The odometry algorithm performs at a high frequency (but low fidelity) to estimate the LIDAR's velocity, while the *Mapping* algorithm runs at a lower frequency for fine matching of features in the set $\{F_e, F_p\}$ to a surrounding wider point cloud map Q^{t-1} and further registration of the point cloud. For a better computation of data, each feature set is stored in Q^{t-1} instead of a single point. Each feature set is associated with the pose of the LIDAR when the scan is taken, considering the sensor FoV, and the selected feature seta are then transformed and fused into Q^{t-1}.

3.5.4 Graph-bsed RTAB-Map SLAM Architecture

As mentioned earlier, the graph-based SLAM utilizes a pose-graph to highlight the spatial structure. The pose-graph is formed by two main components: Nodes and Edges. A node represents the 2D or 3D robot *pose* in space, as well as *landmarks* which identify features in the environment. A landmark differs from a node as it stores spatial information and not orientation data. On the other hand, edges represent the spatial constraint between two nodes, storing information in 3DoF or 6DoF, based on how the space is represented. An edge also considers the uncertainty introduced by sensors. Generally, edges are established under two conditions: Odometry-based or Observation-based. In the Odometry-based, data come from the odometry measurement, while the Observation-based is formed when the robot revisits a location already

stored in the graph, adding an edge between two nonsequential nodes, commonly referred to as a loop-closure. Edges are described using homogeneous coordinates z and an information matrix $\mathbf{\Omega}$ which takes into account the system uncertainty, providing a trust level for each constraint.3.9 shows the basic structure of a 2D graph SLAM representation, where nodes n are given in terms of position and orientation while edges by z and $\mathbf{\Omega}$. In the graph SLAM, the SLAM problem is decoupled into two main tasks:

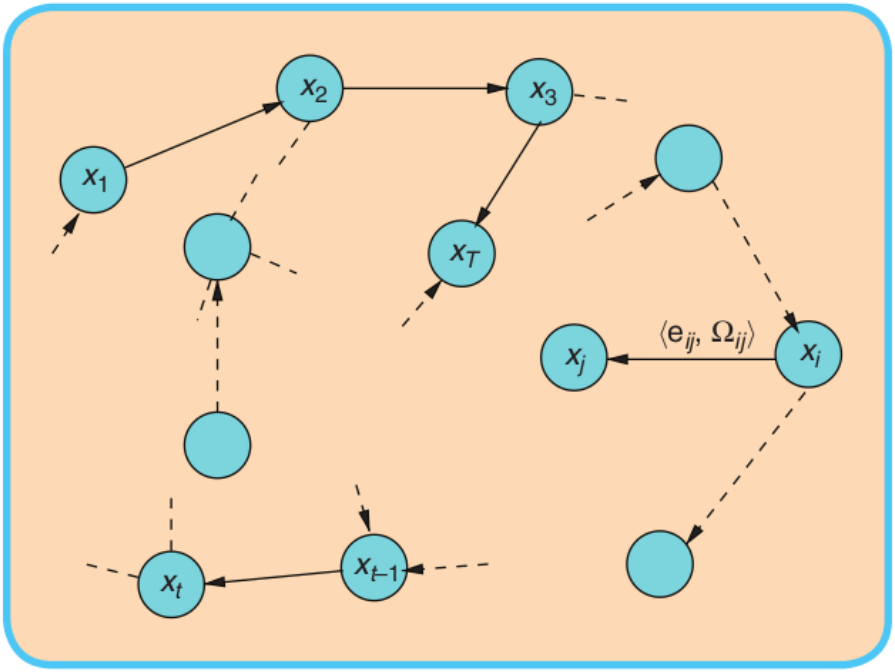


Figure 3.9: Graph-based SLAM: Pose Node Estimate and Edges, [22]

graph construction (front-end) and graph optimization. The first task is in charge of the front-end, and relevant information such as feature extraction, feature matching, data association, and local optimization is extracted from the sensor data. In contrast, graph optimization depends on the system's back-end, where data are represented abstractly, being the back-end sensor-agnostic.

As mentioned above, RTAB-Map is a graph-based SLAM approach that recreates the environment using nodes (signatures) and edges, also known as *link*. The frequency of the synchronized sensor input data coming from the *Synchronization* module determines the frequency of creating new nodes. Following sensor synchronization, the Short-Term Memory (*STM*) module generates a node that stores the odometry pose, raw sensor data, and additional information relevant to subsequent modules. According to how the links are created, the RTAB-Map convention defines three types of links: Neighbor, Loop Closure, and Proximity links. These links are established within the STM. Neighbor links connect consecutive nodes using odometry transformations, while Loop Closure and Proximity links are incorporated through loop closure detection or proximity detection, respectively. All the links serve as constraints for graph optimization. When a new loop closure or proximity link is incorporated into the graph, graph optimization disseminates the calculated error throughout the entire graph to

mitigate odometry drift. Once the graph is optimized, the outputs (e.g. Point Cloud) are compiled and shared with external modules. One crucial point of RTAB-Map is its compatibility with ROS, facilitating extensive support from the scientific community and flexible user use. Another main contribution is the memory management that runs on top of the graph modules. It is used to limit the size of the graph, allowing for long-term online SLAM to be achieved in large environments without exceeding real-time constraints. This aspect directly impacts the feasibility of SLAM exploration in wide Martian habitats.

The RTAB-Map memory system is divided into three main memories: STM, Working Memory (*WM*), and Long Term Memory (*LTM*). The STM works as a fixed-size buffer that processes and stores the most recent nodes and does not affect loop closure detection. The WM contains all the nodes candidates for loop closure detection, typically stored in the RAM. Then, when a node is transferred to the LTM, it does not remain available anymore for the modules inside the WM. In cases where RTAB-Map's update time exceeds a fixed *time threshold*, nodes in the WM are selectively transferred to LTM, aiming to constrain the WM size and thus reduce update time. Similar to the fixed time threshold, a *memory threshold* exists to cap the maximum number of nodes that the WM can accommodate. To identify nodes for transfer to the LTM, a weighting mechanism prioritizes locations based on heuristic methods, considering factors like the duration a location has been observed—the longer, how significant it is, and hence if it should remain in the WM. To implement this, when a new node is created, the STM initializes the node's weight to 0 and visually compares it with the last node in the graph, deriving a percentage of corresponding visual words. If a similarity is detected (thus with the percentage of corresponding visual words surpassing the *similarity threshold*), the weight of the new node increases by 1, plus the weight of the last node. The weight of the last node resets to 0 and the last node is discarded if the robot is stationary, to prevent unnecessary graph expansion. Upon reaching either the time or memory thresholds, the oldest among the smallest weighted nodes are prioritized for transfer to the LTM first. Oldest nodes are those that have been stored in the STM for the longest duration which represent locations or observations the system has encountered over time. This process helps prioritize the retention of significant environmental information for a more accurate loop closure decision, while potentially "archiving" in the LTM less critical (or less frequently) encountered features.

RTAB-Map pipelines

The RTAB-Map algorithm can implement different odometry approaches, from wheel odometry to visual odometry using cameras and 2D/3D LIDARs. The Odometry node is deepened in 3.5.5, according to the odometry strategy implemented for the exploration of the Beva crater through a tracking camera and a 3D LIDAR, while the main modules reported in 3.10 are explained below.

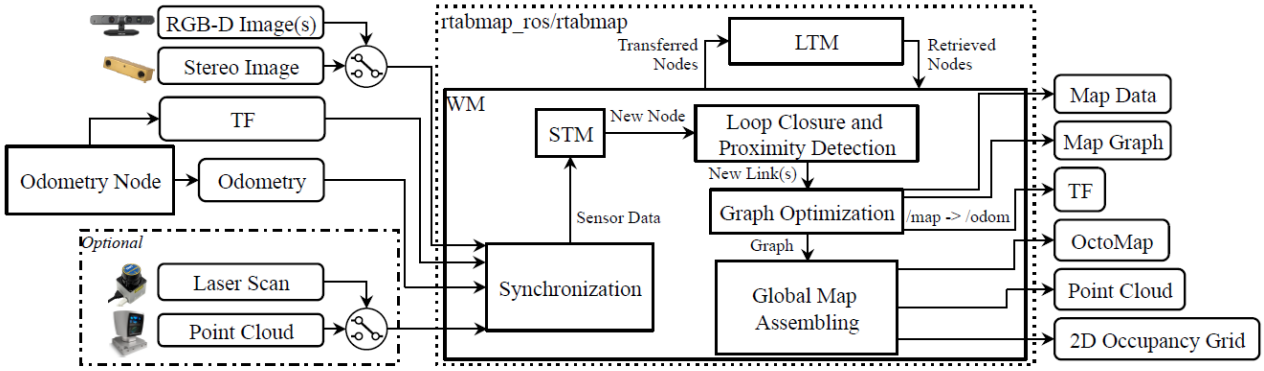


Figure 3.10: RTAB-Map Network, [8]

- **Synchronization** According to the odometry inputs, the next step is the synchronization of the heterogeneous data, as sensors do not always publish data at the same rate and time. Good synchronization is thus important to avoid bad data registration. ROS supports two types of synchronization: *exact* and *approximate*. Exact synchronization requires that input topics have precisely the same timestamp, typically used for topics originating from the same sensor (e.g. left and right images from a stereo camera). On the other hand, approximate synchronization compares timestamps of incoming topics and attempts to synchronize all topics with minimal delay error, commonly used for topics from different sensors. Synchronization is usually challenging, particularly when a subset of input topics needs exact time policy synchronization while being approximately synchronized with other sensors.

- **STM** 3.11 illustrates the workflow for the STM's local occupancy grid creation.

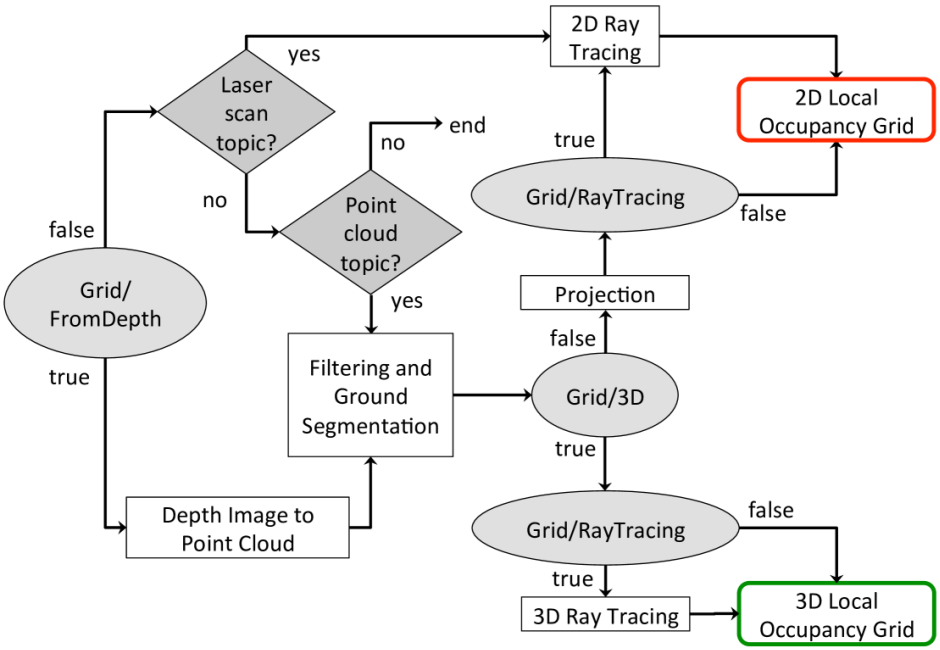


Figure 3.11: Local Occupancy Grid creation

As previously mentioned in 3.5.4, the STM is responsible for the creation of new nodes. Then, a local occupancy grid map is computed from the input source (e.g. point cloud). The size of a local occupancy grid is determined by the sensor's range and Field of View (FoV). Transforming these local grids into the map's reference frame using the graph's poses allows for the creation of a global occupancy grid. Although pre-computing the local occupancy grids increases memory usage for each node, it significantly reduces the time needed to regenerate the global occupancy grid in the LTM after graph optimization. It is important to specify that 2D occupancy grid maps are less memory-consuming as only two dimensions (e.g. x,y) are saved, and obstacles are usually reduced to only one obstacle cell. While 3D occupancy grid maps can also be used to generate 2D maps, the opposite is not feasible, thus the generation of a 3D point cloud map necessarily needs 3D occupancy grids. The processing pipeline begins with converting the input depth image into a 3D point cloud within the sensor frame and camera calibration, which is then transformed into the robot base frame. Following this, the point cloud undergoes downsampling through a voxel grid filter, and ground segmentation is performed by identifying points with normals parallel to the z-axis as ground, labeling others as obstacles. If configured for 2D mapping, the 3D ground and obstacle point clouds are projected onto the ground plane, and a voxel grid filter is applied again. 2D ray tracing may be employed to fill the space between obstacles and the camera. In the absence of 2D ray tracing and if there are no points segmented as ground, no empty cells are set in the occupancy grid between the sensor and obstacles. The process also involves creating an OctoMap from the single local occupancy grid, utilizing 3D ray tracing to detect empty cells between the camera and occupied cells. The OctoMap is then converted back to the local occupancy grid format with empty, ground, and obstacle cells.

- **Loop Closure and Proximity Detection** Proximity detection is used to localize nodes close to the current position by identifying instances when the robot revisits a location already been stored in the graph. This is essential for recognizing nearby places and ensuring the accuracy and consistency of the map. Differently, Loop closure refers to the concept of recognizing a previously visited location, thus forming a closed "loop" in its trajectory. This is a crucial step for correcting accumulated odometry errors and improving the overall consistency of the map. In RTAB-Map, the loop closure involves the comparison of visual features or point clouds between the current observation and the stored locations in the map. When a loop closure is detected, it helps refine the map and corrects any drift that may have occurred during the robot's movement. Both proximity and loop closure detection are performed in the STM.

- **Graph Optimization** As loop closure is detected, the error in the graph map is

minimized through a graph optimization process. There are different approaches to optimization, such as g2o, GTSAM and TORO. g2o is a widely used optimization library for solving nonlinear optimization problems, while the GTSAM library combines factor graphs and Bayes networks for efficient optimization. TORO is more robust over the previous two, specifically designed for optimizing pose graphs. GTSAM is currently employed on RTAB-Map, being slightly more robust than g2o. During the graph optimization process, the system refines and adjusts the poses of the robot at different keyframes to minimize the accumulated errors in the map. This adjustment improves the overall consistency of the map and ensures that the robot's trajectory aligns with the observed sensor data, reducing the accumulated error. The choice of optimization method depends mainly on factors such as computational efficiency and accuracy requirements.

- **Global Map Assembling** The global map assembly involves combining information from the three memories to generate a comprehensive and accurate representation of the environment. The new local occupancy grid is then combined with the existing global occupancy grid. This combination involves updating the global grid by incorporating information from the new local grid. Areas in the global grid that were observed as obstacles in the new local grid are marked as obstacles in the global grid and areas that are clear in the new local grid contribute to clearing those areas in the global grid. This process ensures that the global occupancy grid is continuously refined and updated as the robot explores and gathers new sensor data.

3.5.5 RTAB-Map Odometry Node

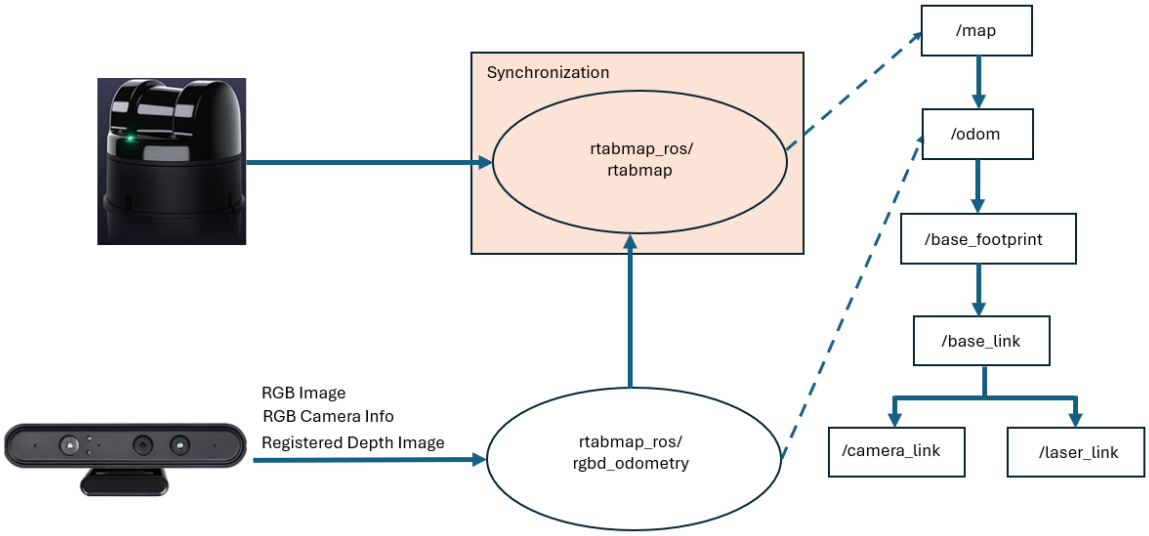


Figure 3.12: MHex SLAM configuration

For the MHex setup, the T265 camera provides odometry information, and the 3D LIDAR sensor provides point cloud data, which is used for mapping in conjunction

with the T265 odometry to create a 3D map of the environment. The RTAB-Map algorithm integrates both sensor inputs to perform SLAM.

3.6 Conclusions

This chapter provided a description of the mission requirements to autonomously explore the Martian Belva crater, together with the definition of both the MHex architecture along the main subsystems, and the GNC strategy to autonomously navigate the unknown environment to 3D map the surroundings. Compared to traditional filters, such as the EKF, the description and analysis of the UIO is presented and chosen as potential candidate to implement the navigation algorithm and overcome external disturbances (e.g. wind) and sensor noise.
Within numerous SLAM techniques that consider different types of sensors and approaches for both data fusion and map optimization, LOAM is chosen at SIL level because of its semplicity and versatility, while RTAB-Map can result a better solution when working with the hardware thanks to its high level of modularity and efficiency in graph-based optimization, expecially when dealing with data fusion of various sensor sources. Additionally, being an open-source algorithm, it provides a practical solution for users to get custom architectures to fit multiple and complex scenarios.

Chapter 4

Simulated Experiments

4.1 SIL: Validation and Verification

The following section presents the process of implementing and verifying the MHex system through SIL simulations. This involves utilizing Matlab/Simulink modeling to develop control strategies as well as EKF and UIO observer. This first simulation precedes the validation of the MHex within the ROS and Gazebo environment.

4.2 Proposed Solution

To address the feasibility of the MHex autonomous exploration of the Martian crater, the proposed solution integrates the mission scenario within ROS and Gazebo simulator, allowing for robust SIL. Firstly, validation and verification of the EKF, and UIO observer is presented according to Matlab/Simulink analysis. Then, the environmental representation of the crater, as well as the MHex real-scale system, are imported into Gazebo and the MHex is tested according to the navigation and control algorithms developed in ROS.

4.2.1 Matlab/Simulink modeling

To assess the performance of both different control strategies, a comprehensive mathematical model of the MHex is firstly developed using Matlab and Simulink. This model enables an initial analysis of the GNC subsystem. Then, based on the gained results, the filter and observers performances are evaluated, and only the component demonstrating superior performance is integrated into ROS. This integration involved exporting the code through C++ code generation and incorporating it as a ROS plugin. Finally, the modeling of the MHex in ROS and Gazebo provides a platform for creating a realistic simulation of the Martian environment, including atmospheric conditions.
The Euler-Lagrange model representation of the MHex is implemented in Simulink along with the inner and outer loop control and filter/observer block, as given in 4.1.

For a clear understanding, the compact description of the simulation parameters and results are given in 5.

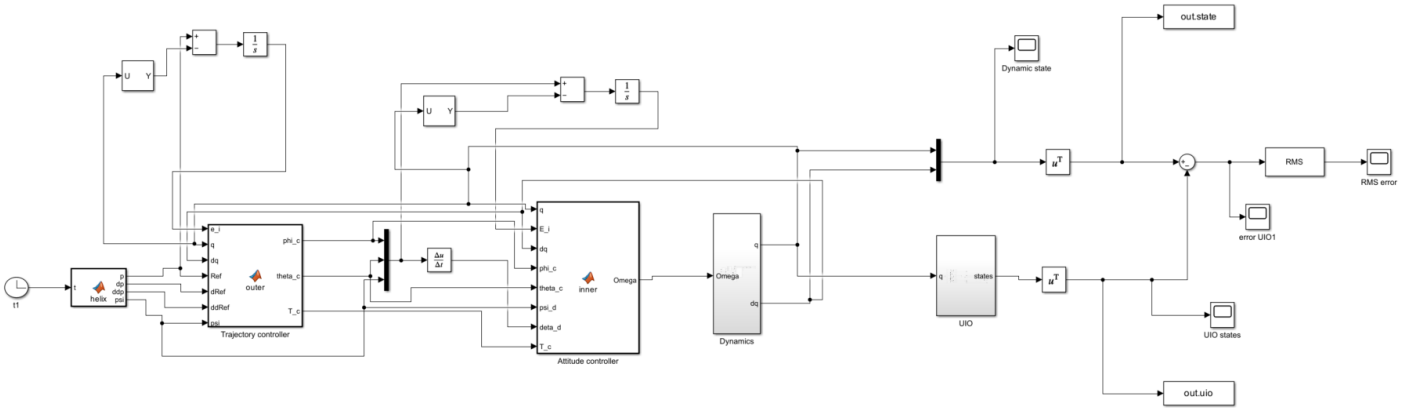


Figure 4.1: MHex Simulink model, *PID* controller

4.2.2 Controllers

Three controllers are tested and integrated in the model: The *Ardupilot A-PID*, the *PID Feedback Linearization* and the *Backstepping BC* controller.

PID Feedback Linearization

The *PID* is one of the most common controller in literature and industry due to its straightforward implementation and ease of tuning, including model based dynamic compensation into the loop that has been shown to further improve tracking performance [31]. To this end, feedback linearization is used to exactly linearize the MHex nonlinear dynamics. As such, the commanded torque vector is written as

$$\tau_c = J(\eta)v + C_\tau(\eta, \dot{\eta})\dot{\eta}, \tag{4.1}$$

so that the closed loop system becomes

$$\ddot{\eta} = v \tag{4.2}$$

which correspond to the following linear system

$$\dot{x} = Ax + Bv \tag{4.3}$$

where

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix}, B = \begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix}, x = \begin{bmatrix} \eta \\ \dot{\eta} \end{bmatrix} \tag{4.4}$$

hence, the virtual control vector v becomes the control action to be designed [53]. To show the performance improvement, the feedback linearization is placed in cascade to

the Ardupilot attitude *PID* controller and, as such, the *A-PID* output value is assigned to v.

Backstepping

BC model based control has been shown to be very effective in controlling nonlinear system in presence of noise and external disturbances [32]. The following *BC* controller is presented in [12], for brevity, only the commanded torque is listed here

$$\hat{F_d} = M\ddot{\xi}_d + M\Lambda_1^2 e_1 + g + K_{r_1} r_1 + K_{i_1} \int_0^t r_1 dt \tag{4.5}$$

with $e_1 = \xi_d - \xi$ position tracking error, and $r_1 = \dot{e}_1 + \Lambda_1 e_1$ sliding mode error. This virtual control input allows the computation of the Euler angles reference trajectory and commanded thrust as

$$\theta_{c,BC} = \arctan\left(\frac{\hat{F}_{d_1}\cos\psi + \hat{F}_{d_2}\sin\psi}{\hat{F}_{d_3}}\right), \tag{4.6}$$

$$\phi_{c,BC} = \arcsin\left(\frac{\hat{F}_{d_1}\sin\psi - \hat{F}_{d_2}\cos\psi}{\sqrt{\hat{F}_{d_1}^2 + \hat{F}_{d_2}^2 + \hat{F}_{d_3}^2}}\right), \tag{4.7}$$

$$\begin{aligned} T_{c,BC} = \quad & \hat{F}_{d_1}(\sin\theta\cos\psi\cos\phi + \sin\psi\sin\phi) + \\ & + \hat{F}_{d_2}(\sin\theta\sin\psi\cos\phi - \cos\psi\sin\phi) + \\ & + \hat{F}_{d_3}\cos\theta\cos\phi \end{aligned} \tag{4.8}$$

Finally, the commanded torque vector is computed as

$$\begin{aligned} \tau_{c,BC} = \quad & J(\ddot{\eta}_d - \Lambda^2 e) + C_\tau(\dot{\eta}_d + \Lambda e) + \\ & + K_r r + K_i \textstyle\int_0^t r dt \end{aligned} \tag{4.9}$$

with orientation tracking error $e = \eta_d - \eta$, sliding mode error $r = \dot{e} + \Lambda e$, and Λ, K_r, K_i gain matrices to be tuned. Let's define the position tracking error $e_1 = \xi_d - \xi$ and the sliding mode error

$$r_1 = \dot{e}_1 + \Lambda_1 e_1 \tag{4.10}$$

The error dynamics is

$$\begin{aligned} M\dot{r}_1 \quad & = M\ddot{e}_1 + M\Lambda_1\dot{e}_1 \\ & = M\ddot{\xi}_d - M\ddot{\xi} + M\Lambda_1(r_1 - \Lambda_1 e_1) \\ & = M\ddot{\xi}_d - R_{321}T_B + g + M\Lambda_1(r_1 - \Lambda_1 e_1) \\ & = M\ddot{\xi}_d + M\Lambda_1 r_1 - M\Lambda_1^2 e_1 - R_{321}T_B + g \end{aligned} \tag{4.11}$$

with $R_{321}T_B = F_d$. Given the virtual force as

$$\hat{F}_d = M\ddot{\xi}_d + M\Lambda_1^2 e_1 + g + K_{r_1} r_1 + K_{i_1} \int_0^t r_1 dt \tag{4.12}$$

the closed loop error dynamics becomes

$$M\dot{r}_1 = -(K_{r_1} - M\Lambda_1) r_1 - K_{i_1} \int_0^t r_1 dt \tag{4.13}$$

it follows

$$\theta_{c,BC} = \arctan\left(\frac{\hat{F}_{d_1}\cos\psi + \hat{F}_{d_2}\sin\psi}{\hat{F}_{d_3}}\right), \tag{4.14}$$

$$\phi_{c,BC} = \arcsin\left(\frac{\hat{F}_{d_1}\sin\psi - \hat{F}_{d_2}\cos\psi}{\sqrt{\hat{F}_{d_1}^2 + \hat{F}_{d_2}^2 + \hat{F}_{d_3}^2}}\right), \tag{4.15}$$

$$\begin{aligned} T_{c,BC} = \; & \hat{F}_{d_1}(\sin\theta\cos\psi\cos\phi + \sin\psi\sin\phi) + \\ & + \hat{F}_{d_2}(\sin\theta\sin\psi\cos\phi - \cos\psi\sin\phi) + \\ & + \hat{F}_{d_3}\cos\theta\cos\phi \end{aligned} \tag{4.16}$$

which give the Euler angles reference signal. Let's define now the orientation tracking error $e_2 = \eta_d - \eta$ and the sliding mode error

$$r_2 = \dot{e}_2 + \Lambda_2 e_2 \tag{4.17}$$

The error dynamics is

$$\begin{aligned} J\dot{r}_2 &= J\ddot{e}_2 + J\Lambda_2\dot{e}_2 \\ &= J\ddot{\eta}_d - J\ddot{\eta} + J\Lambda_2(r_2 - \Lambda_2 e_2) \\ &= J\ddot{\eta}_d - (\tau - C_\tau\dot{\eta}) + J\Lambda_2(r_2 - \Lambda_2 e_2) \\ &= J\ddot{\eta}_d + J\Lambda_2 r_2 - J\Lambda_2^2 e_2 - \tau - C_\tau r_2 + \\ &\quad + C_\tau(\dot{\eta}_d + \Lambda_2 e_2) \end{aligned} \tag{4.18}$$

leading to the commanded torque vector

$$\begin{aligned} \tau_{c,BC} = \; & J(\ddot{\eta}_d - \Lambda_2^2 e_2) + C_\tau(\dot{\eta}_d + \Lambda_2 e_2) + \\ & + K_{r_2} r_2 + K_{i_2} \textstyle\int_0^t r_2 dt \end{aligned} \tag{4.19}$$

and the following closed loop error dynamic

$$J\dot{r}_2 = -(K_{r_2} + C_\tau - J\Lambda_2) r_2 - K_{i_2} \int_0^t r_2 dt \tag{4.20}$$

4.3 Simulator Architecture in ROS and Gazebo

4.2 displays an overview of the resulting workflow that the simulation environment allows. The Simulation environment relies on the integration of Ardupilot, a widespread

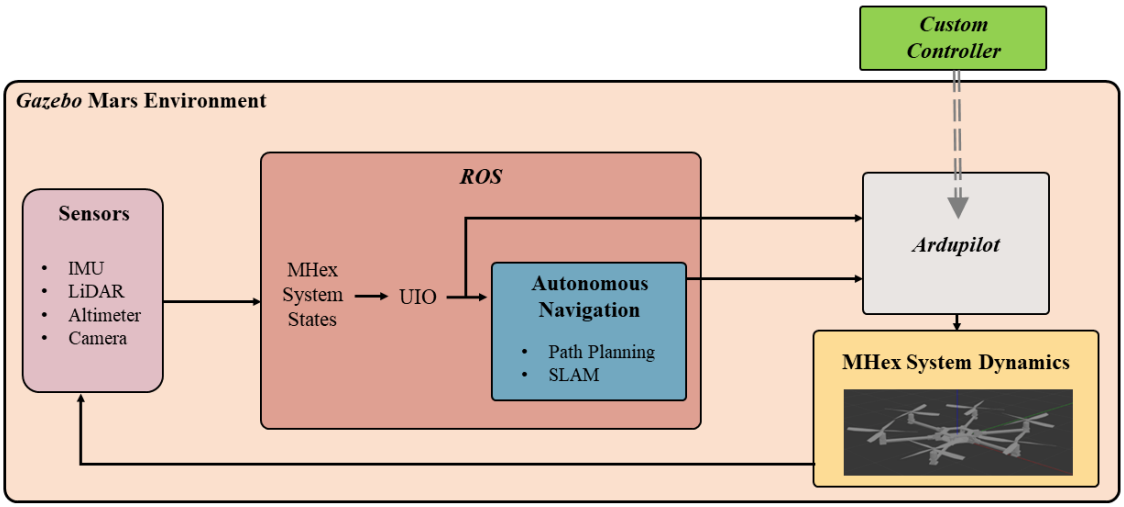


Figure 4.2: Simulator Architecture

open-source autopilot system, along with the Gazebo simulator and ROS. This combination of free and open-source tools allows for simulating the dynamic behavior of the MHex model in a Mars-like environment, accounting for high accuracy of the numerous MHex attributes such as mass, inertia, wind gusts and noise thus allowing it to behave realistically when simulating. Because of the complexity of the simulated mission, extensive design and some software modification are required to achieve a realistic simulation. Particularly, continuous efforts in implementing advanced control strategies and integrating control techniques into the Ardupilot system not only highlight the significant adaptability of the software, but also allow researchers to develop in-depth analysis and get realistic results in complex simulation environments. Recent works considering the implementation of control strategies, such as the Adaptive Integral Sliding Mode Control ($AISMC$) as well as the adaptive control integration into the Ardupilot software can be found in [28, 29]. Experiments on the adaptive method for the Ardupilot-based controller can also be found in [6] for fixed-wing UAVs. For this research, the simulator runs on Ubuntu 20.04, with ROS Noetic, Gazebo 11.11.0, and Ardupilot Copter 4.3.

MHex model and Crater representation

The Belva crater is modeled according to the data obtained from the HiRISE dataset, representing the real morphology of different Martian sites. By extrapolating the ge-

ological map from [62] and based on the elevation data of the area, the model of the Belva crater is processed discretized in *Blender*, [7], as shown in 4.3.

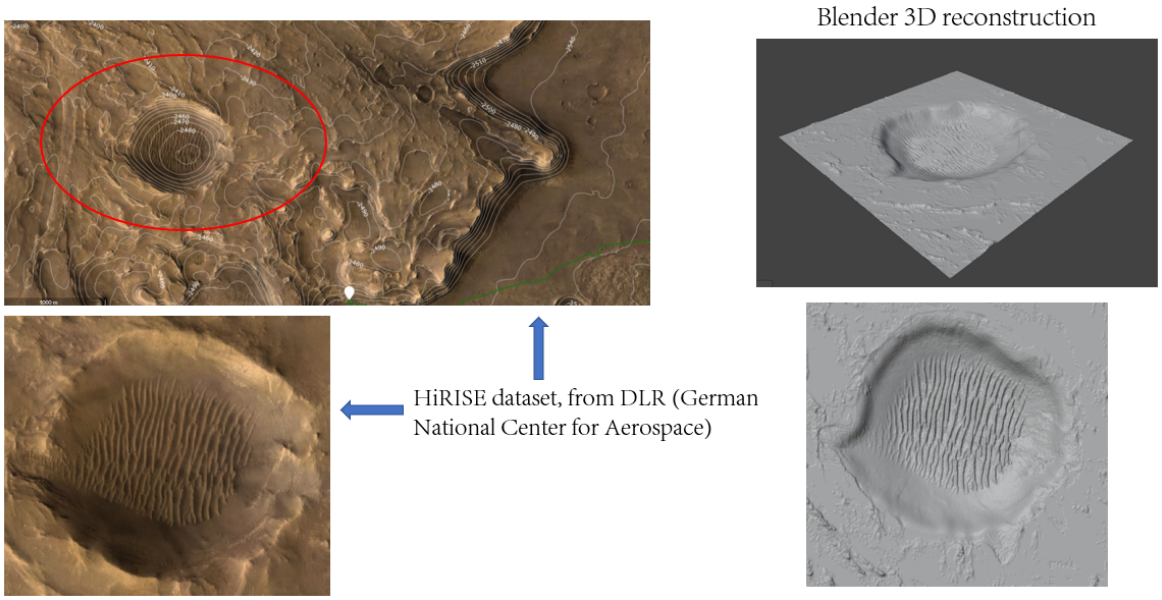


Figure 4.3: Modeling of Belva crater, *Blender* software

As noticed in the previous chapter, the 1.5 Km radius, 80 m depth crater is of scientific interest because of its morphological composition. Also, its proximity to the landing area of the Perseverance rover along with favorable atmospheric conditions of the Jezero region enabling flights, further accentuates its scientific importance.
The MHex model is exported from Solidworks to *urdf* file format, which is then converted to Gazebos's *sdf* format maintaining all inertial characteristics and meshes (3.6). The aerodynamic profile of the propellers is implemented using the *LiftDrag Plugin*, and the motor actuation from Ardupilot flight controller is managed through the *Ardupilot Gazebo Plugin*. To simulate the sensor fusion output, the odometry information is taken from an *odometry sensor* publishing up to 200 Hz. Indeed, as mentioned, given the Martian conditions, the MHex system cannot rely on either the the GNSS, barometer, or magnetometer. To this end, these sensors are disabled and the alternative selection of sensors from 3.5, along with the IMU, is implemented to guarantee navigation, control, and mapping functionalities. Considering the use of a rotating VLP-16 LIDAR, the spinning sensor is modeled and attached to the MHex body to account for its additional dynamics.

4.4 Conclusions

This chapter covered the development of the SIL for validating the MHex autonomous exploration of the Belva crater. The MHex model was firstly represented in Matlab and Simulink together with the *A-PID, PID-FL* and *BC* control approaches and the filter and observer blocks. This allowed to recreate the main structure of the GNC

subsystem. Then, the advanced simulator architecture in ROS and Gazebo allows for a realistic implementation of the same system in the recreated Martian environment.

Chapter 5

Observer-based navigation: Results

5.1 Comparative Analysis of Proposed Approaches

This section presents the results derived from simulations. A first analysis is conducted within the Matlab/Simulink environment, focusing on the evaluation of the EKF and UIO. Firstly, the filter and observer performances are compared, and the UIO is selected based on the results. Then, the UIO is imported in ROS and Gazebo, and a comparative examination is conducted among the three distinct controllers (*A-PID, PID-FL* and *BC*) according to the MHex mission profile.

5.1.1 Filter and Obsever-based Navigation

As a first validation step, the comparison between the EKF and UIO is implemented in Matlab/Simulink along a helix-based trajectory. As the observer's inputs, the simulation considers only the pose q as known measured input coming from the MHex onboard sensors and affected by the bias, while the velocities $\dot{\boldsymbol{q}}$ are estimated by the filter/observers. Furthermore, to obtain a fair comparison between the two estimation methods, the eigenvalues of the closed-loop matrices $\boldsymbol{E}$ and $\boldsymbol{A}_{\tilde{z}}$ were placed in the same positions. The simulation parameters are given in 5.1. Wind acting on the system is simulated along the horizontal and vertical directions. The modeling of the wind is also applied in the second validation step with ROS and Gazebo, as shown below.

Table 5.1: Simulation parameters

Parameters	**Description**	**Value**	**Unit**
b	Rotor drag coefficient	2.690×10^{-8}	–
k	Thrust coefficient	1.9018×10^{-4}	–
I_x	Inertia moment along x_b	10.19	Kgm^2
I_y	Inertia moment along y_b	18.67	Kgm^2
I_z	Inertia moment along z_b	10.19	Kgm^2
g	Acceleration of gravity	3.72	m/s^2

EKF

The EKF acts along a different process split into two phases: the *state prediction*, based on knowledge of the process before the arrival of the *k-th* measurement, and the *state update* which incorporates the new measures in the prior (a-priori) estimate to obtain the improved (a-posteriori) estimation. Considering the MHex system model, the general form can be written as

$$\dot{x} = f(x(t), u(t), w(t)) \tag{5.1}$$

$$z(t) = h(x(t), v(t)) \tag{5.2}$$

with *x(t)* state vector, u(t) known inputs, z(t) output vector, and w(t) and v(t) the process and measurement noise affecting the process, while f and h are nonlinear functions describing the system dynamics. After receiving measurements at discrete times, the EKF performs a measurement update based on the measurement residual and the predicted measurement on the current state estimate. The EKF operates through a two-phase algorithm wherein an a-priori estimate $\hat{\mathbf{Z}}_k^-$ of the system state $\mathbf{Z}_k$ and an a-posteriori estimate $\hat{\mathbf{Z}}_k$ are computed [43]. Specifically, $\hat{\mathbf{Z}}_k^-$ is determined during the state prediction phase based on the process knowledge prior to the arrival of the k-th measurement $\mathbf{Y}_k$. Subsequently, the state update phase implements a feedback mechanism that incorporates the new measurements into the a-priori estimate $\hat{\mathbf{Z}}_k^-$, resulting in the refined a-posteriori estimate $\hat{\mathbf{Z}}_k$. The a-priori and a-posteriori estimation processes are characterized by the following covariance matrices:

$$\mathbf{P}_k^- = \mathbb{E}\left[(\mathbf{Z}_k - \hat{\mathbf{Z}}_k^-)(\mathbf{Z}_k - \hat{\mathbf{Z}}_k^-)^\top\right],$$

$$\mathbf{P}_k = \mathbb{E}\left[(\mathbf{Z}_k - \hat{\mathbf{Z}}_k)(\mathbf{Z}_k - \hat{\mathbf{Z}}_k)^\top\right],$$

where $\mathbb{E}[\cdot]$ denotes the expectation value operator. It is also worth noting that, compared to a linear Kalman filter, an EKF differs in that the matrices involved in the Riccati equations are obtained by linearizing the nonlinear state equations around the state estimate from the previous step.

UIO

Similarly, the UIO is then implemented in the MHex system. To reach the form of 3.6, analytical calculations lead to the matrices

$$A = \begin{bmatrix} 0_{6\times6} & I_{6\times6} \\ 0_{6\times6} & 0_{6\times6} \end{bmatrix} \quad B = \begin{bmatrix} M & 0_{3\times3} \\ 0_{3\times3} & J \end{bmatrix} \tag{5.3}$$

$$C = \begin{bmatrix} I_{6\times6} & 0_{6\times6} \end{bmatrix} \quad W = \begin{bmatrix} 0_{6\times6} & I_{6\times6} \end{bmatrix} \tag{5.4}$$

The state vector x includes the inertial position p, orientation η, and their respective derivatives $\dot{p}, \dot{\eta}$. Only the pose (position and orientation) is taken as the system's output, while $\dot{p}$ and $\dot{\eta}$ are estimated through the observer. According to 3.12 and 3.13, the smallest integer satisfying the above condition is $L = 2$, with the invertibility and observability matrices assuming the form

$$J^2 = \begin{bmatrix} 0 & 0 & 0 \\ CW & 0 & 0 \\ CAW & CW & 0 \end{bmatrix}, \quad O^2 = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} \tag{5.5}$$

for which it holds, based on 5.3, 5.4

$$rank(J) = 6 \tag{5.6}$$

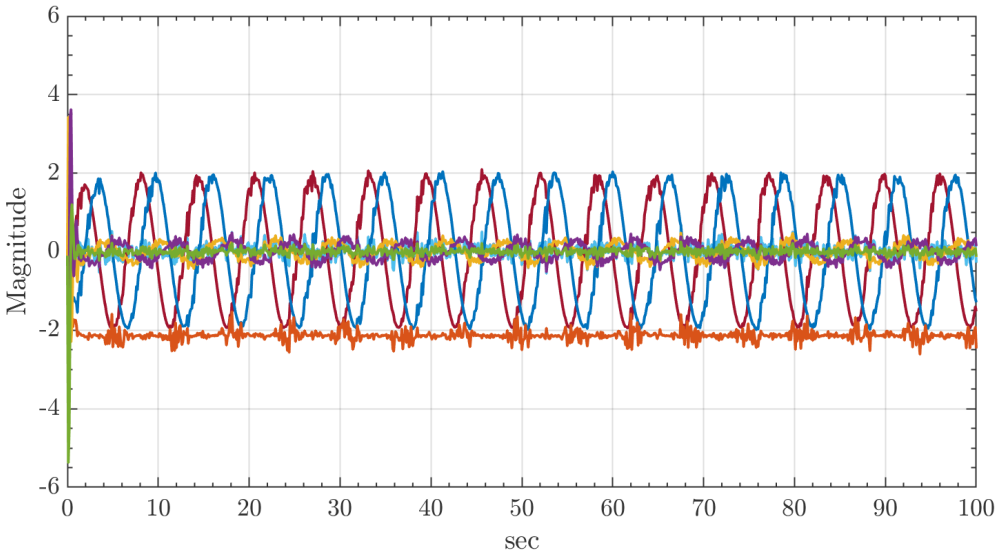
$$rank(O) = 12 \tag{5.7}$$

thus satisfying the conditions of solvability.

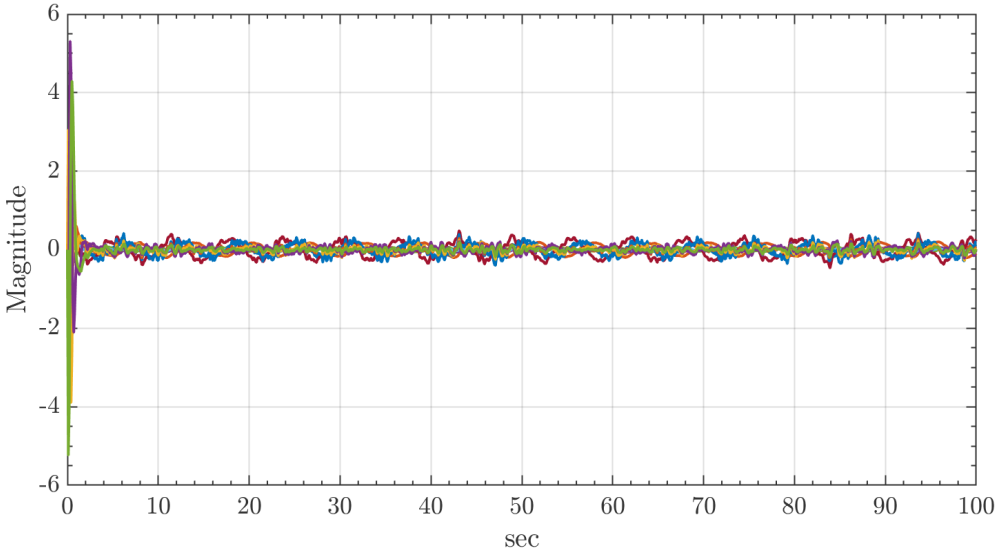
5.2 Discussion of Results

Matlab/Simulink analysis

The Simulink model includes disturbance phenomena, such as wind along x and y directions, and sensor bias to simulate the real effects impacting the MHex dynamic during flight. For the EKF, the equivalent noise considered in the simulation follows a Gaussian distribution with $\mu = 0$ and variance $\sigma^2 = 0.01$, corresponding to the realistic sensor noise for both the t265 camera and the VLP-16 3D LIDAR. Both EKF and UIO receives only the pose as input state, and estimate the linear and angular velocities according to the simulated flight dynamic.



(a) EKF state estimation error



(b) UIO state estimation error

Figure 5.1: UIO and EKF comparison

The state estimation error for the EKF and UIO is given in 5.1. As reported in the plots, the UIO tracks the states even in the presence of external disturbances and more effectively than the EKF. Even in the presence of sensor bias, the UIO demonstrates robustness to the noise maintaining an accurate estimate of the states and a corresponding smaller error magnitude, also not requiring any information about the unknown inputs, here in the form of external disturbance. The computed Root Mean Square Error ($RMSE$) for both the EKF and the UIO is given in 5.2. The results clearly demonstrate the superior performance of the UIO in all the state estimations. Only the angular velocity estimation closely approaches the accuracy of the EKF estimate, while the remaining states are better defined . These simulation outcomes highlights the UIO superiority over the EKF in state estimation together with its simpler implementation,

Table 5.2: UIO and EKF RMSE, *Simulink* Comparison

RMSE	*Position*	*Orientation*	*Linear vel*	*Ang vel*
UIO	0.18	0.09	0.25	0.47
EKF	0.31	0.30	2.87	0.48

emphasizing its practical advantage.

Simulator analysis

In the simulated mission scenario, the MHex performs the autonomous navigation along with SLAM given a series of waypoints sent from the ground control station. The takeoff is performed in the proximity of the crater region, with the main focus of mapping both the entire crater and its surroundings. The MHex is required to perform collision-free flights during the mapping process, thus without prior information, and landing at the goal point when the last waypoint is reached.

5.3 gives the aerodynamic coefficients used to simulate the lift and drag experienced by the MHex during flight.

Table 5.3: MHex aerodynamic parameters

Parameters	α_0	α_{st}	CL_α	CD_α	*VLP-16 spin*
Value	0.0175	0.1920	7.791	0.55	1 Hz

Based on the previous results gained from the analysis in Matlab/Simulink, the UIO is tested in the simulator architecture to both test and assess the algorithm's robustness to external disturbances and measurement uncertainties.
Two different simulated experiments are performed in the simulator architecture: in the first, only the sensor noises are taken into account without other external disturbances, testing smooth flight conditions. In the second one, both wind along horizontal ($\bar{v}_w$) and vertical ($\bar{v}_{w,z}$) directions and sensor bias are considered. 5.4 gives the comparison between the RMSE of the attitude with respect to the desired attitude trajectory along with the different control approaches implemented.

Table 5.4: MHex controllers RMSE, *Simulator*

RMSE [deg]	*A-PID*	*PID FL*	*BC*
No Wind	4.23	1.82	3.78
Wind	9.36	1.37	3.96

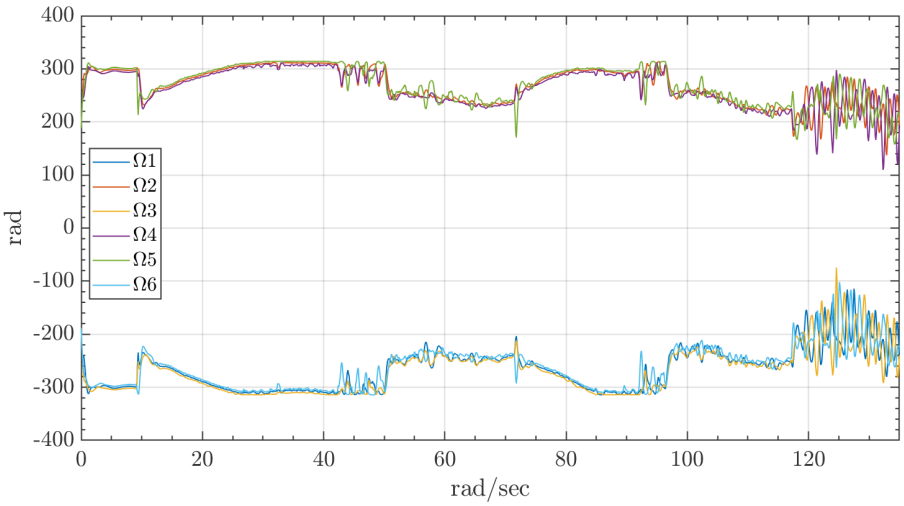
As shown, the model-based control approaches are able to achieve smaller RMSE compared to the *A-PID* controller. The propeller angular velocity during cruise remains in the range of M = 0.55 to 0.8 at tip, corresponding to 1910 to 2780 rpm (200 rad/sec to 291 rad/sec respectively, with max peaks of 315 rad/sec (3008 rpm, M = 0.86)).

Considering the flight speed as ground speed, the limit of 8 m/s guarantees that the linear velocity at the propeller tip is less than M = 0.89. The propeller angular velocity has an average of 260 rad/sec (2483 rpm, M = 0.71), and a median of 265 rad/sec (2531 rpm, M = 0.73), resulting close to the blade optimum operating point of M = 0.75, based on the data available in [24], as shown in plots in 5.5. The resulting control

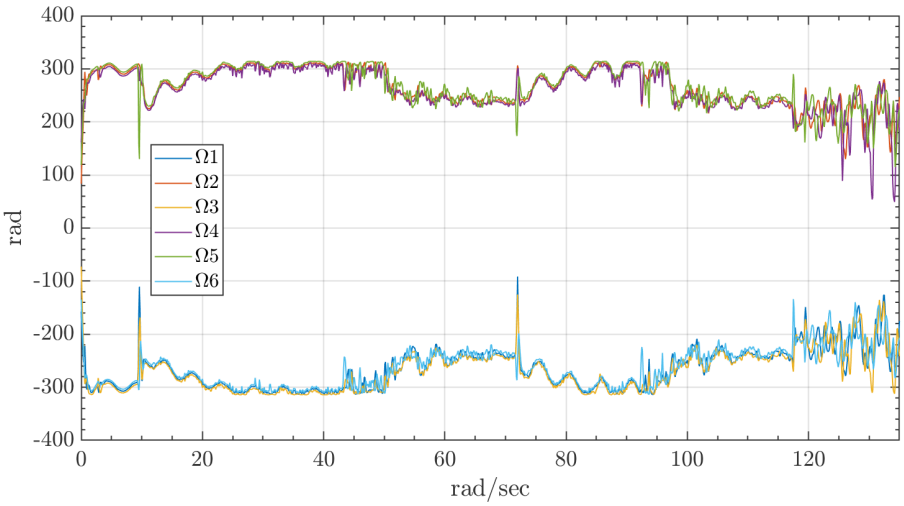
Table 5.5: Controller Gains

Gains *PID*	$K_{p,prefilter}$	K_p	K_i	K_d
PID-AP	$4.5 \times I_{3\times3}$	diag(0.135, 0.135, 0.180)	diag(0.135, 0.135, 0.018)	diag(0.0036, 0.0036, 0)
PID-FL	$9 \times I_{3\times3}$	diag(1.25, 1.25, 0.50)	$0.1 \times I_{3\times3}$	diag(0.01, 0.01, 0)
Gains *BC*	K_r	K_i	Λ	$--$
BC	$1 \times I_{3\times3}$	$30 \times I_{3\times3}$	$1 \times I_{3\times3}$	$--$

gains for each controller show the ability of the tuning process to achieve the best performances while avoiding saturation of the control action, while the propeller rates, displayed in 5.2a 5.2b remain bounded and stable for all controller strategies.



(a) *A-PID*



(b) *PID-FL*

Figure 5.2: MHex propeller angular velocity, simulated experiment including wind

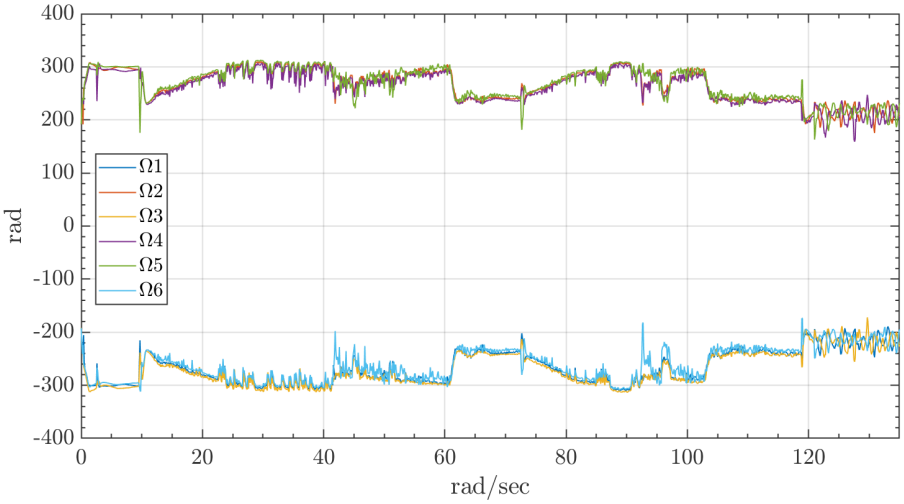
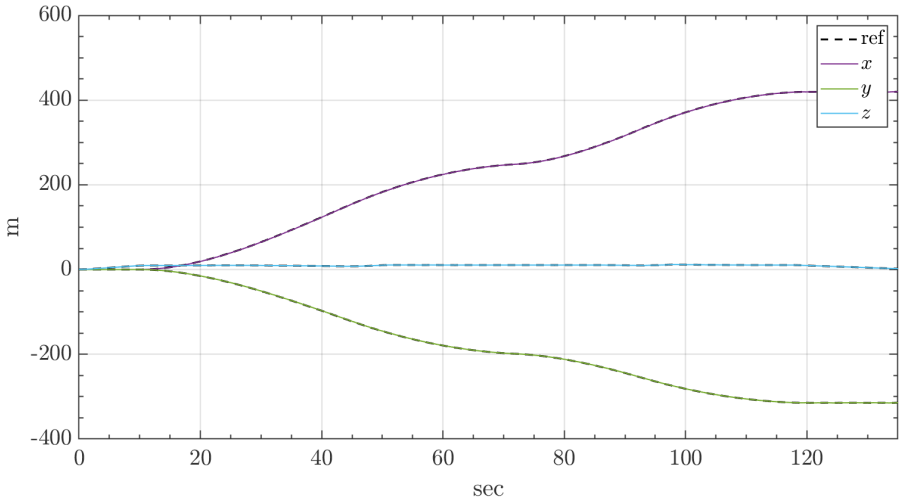
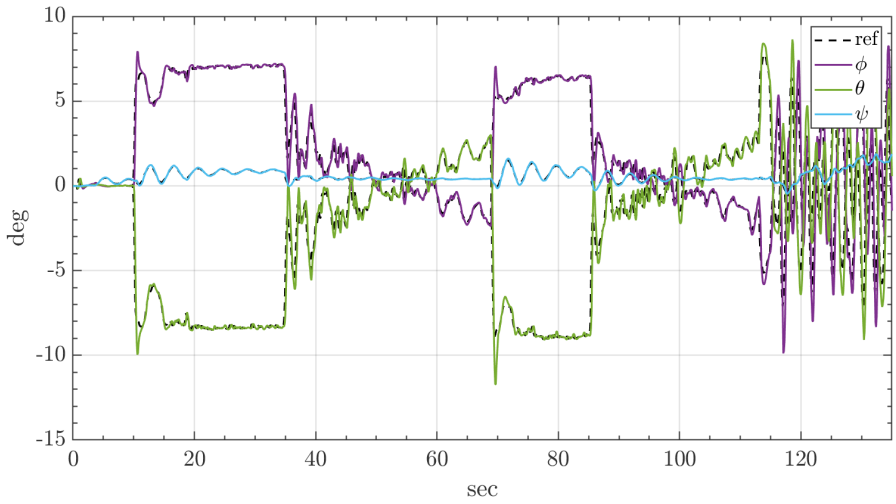


Figure 5.3: *BC*

Finally, given the overall better performance, even during wind simulations, the model-based approaches are preferable for mapping and navigation tasks.
The simulation results carried out in the simulator for the state estimation are presented in 5.6, 5.4 and 5.5.
The wind is applied along the horizontal direction ($\bar{v}_w$) with a maximum intensity of 4m/sec, and vertically up to 1 m/sec ($\bar{v}_{w,z}$).
As shown, the UIO is able to estimate the unknown states of the system (*velocities*), having only the pose as input.

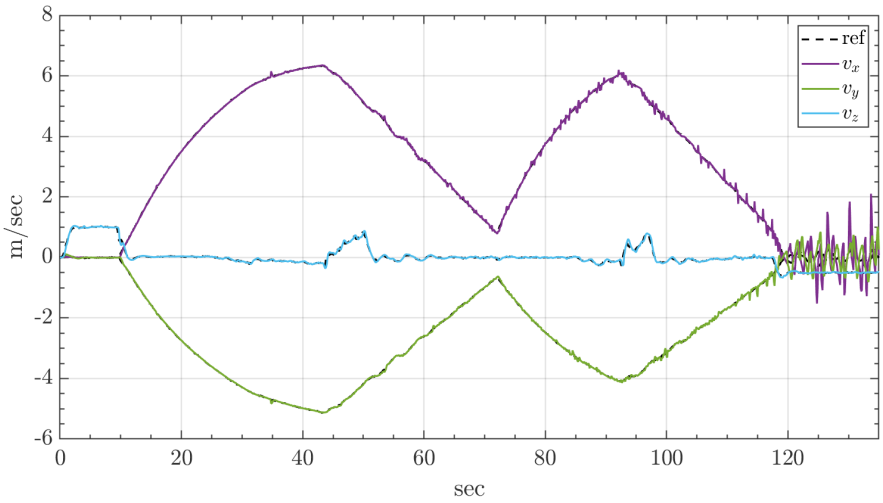


(a) Position

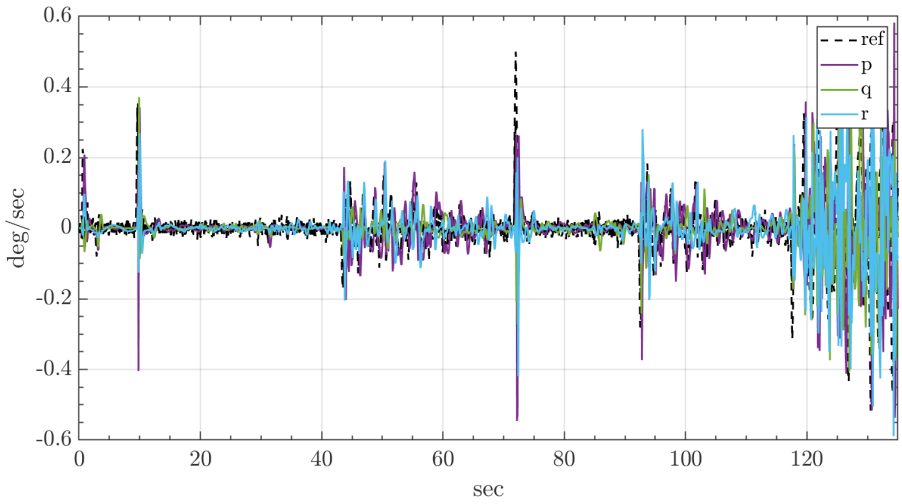


(b) Orientation

Figure 5.4: UIO state estimation, *position*



(a) Linear velocity



(b) Angular velocity

Figure 5.5: UIO state estimation, *orientation*

Table 5.6: UIO RMSE, *Simulator*

UIO estimate	*Position*	*Orientation*	*Linear vel*	*Ang vel*
RMSE	0.22	0.72	0.23	0.11

The estimated states closely track the original states without significant drift or deviation from the reference path, with the most significant impact during the landing phase. The overall accuracy of the proposed method is validated by the RMSE measurements.

5.2.1 3D Mapping

Figure 5.6 shows the mapping process in the explored site based on the point cloud feature extraction.

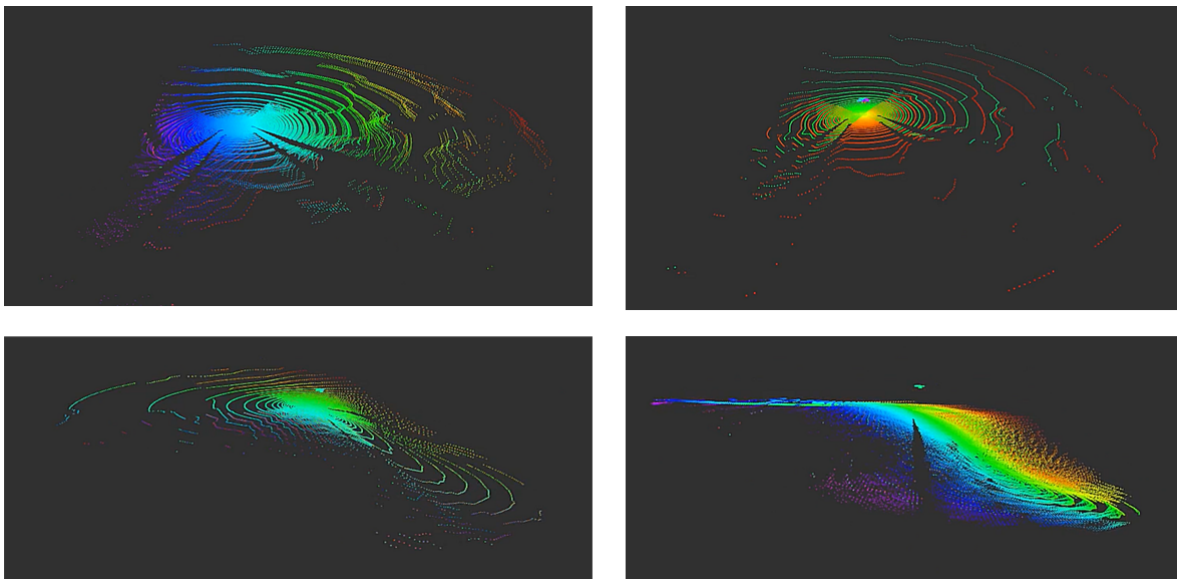


Figure 5.6: MHex mapping framework: The MHex approaches the crater and performs the point cloud map. Edge and planar features are progressively extracted, and selected subsets of points are then stored in the final map

As reported in Section 3.5.3, when performing SLAM both the edge and planar features are extracted from the same lidar scan, and the corresponding feature subset is first optimized through point cloud segmentation and then processed by the mapping module. This procedure allows for a reduction of the number of points stored in the map, also filtering out unstable features based on the c_{th} threshold. The SLAM algorithm is able to track morphological features of the crater while performing real-time pose estimation. The VLP-16 provides precise and direct depth measurements in the environment, resulting in accurate 3D point cloud data even in a wide and feature-poor scenario such as Martian terrain. Moreover, since the entire mapping process results optimized in terms of ground point cloud segmentation, the lidar-mapping accuracy is improved, reducing the computational complexity when compared to the vision-based counterpart. Last but not least, the lidar FoV results in a wide range of collecting data thus particularly valuable when a thorough comprehension of the environment is essential during long-range exploration.

5.3 Conclusions

This chapter presented the results obtained by the simulation of the autonomous navigation mission. To validate the entire GNC analysis, a simulator architecture based on free and open-source ROS, Gazebo, and Ardupilot software was proposed, and the simulated experiments considering both state estimation and model-based control were performed with promising results. The SITL implemented in ROS and Gazebo accounted for several realistic features of the Martian environment, such as aerodynamic and wind effects, VLP-16 Lidar rotating dynamics, GPS-denied position estimates, sensor noises, and conservative sampling rates. Firstly, the UIO is presented

and compared to the well-known EKF for filtering sensor output, showing robustness against disturbances and sensor bias and guaranteeing accurate estimates of the unknown states, overcoming the limitations of different classes of filters more sensitive to external disturbances. Secondly, the comparison between the EKF and UIO observer on the same MHex model showed how the UIO tracks the trajectory by estimating the system states even in the presence of both aerodynamic disturbances and system uncertainties, as well as sensor noise. In the UIO, the lumped disturbances are observed and mitigated without prior knowledge of either the boundary of the disturbance or its derivative, proving that the UIO can rapidly and accurately respond to sudden changes in noise. As a final validation step, the UIO is imported in ROS according to the better performances between the analyzed filter and observer, and a comparison between model-based control strategies (*PID-FL*, *BC*, and the stock Ardupilot attitude PID controller *A-PID*) showed the advantage of considering the MHex nonlinear dynamics for achieving a more precise control and compensating for the tight flight envelope and environmental conditions characterizing the flying on Mars. The simulated autonomous exploration mission shows different behaviors of the control strategies, with the PID-FL demonstrating the highest performances. SLAM and 3D mapping showed the capability of accurate mapping and stratigraphic investigation of the surrounding environment, addressing the numerous challenges of Martian flight together with the comparison of different control strategies.

Chapter 6

Reinforcement Learning Strategies: PPO and DDPG algorithms

6.1 Use of Reinforcement Learning in Autonomous Navigation

The growing need for autonomy in various mission scenarios has led to the development of robots capable of performing a wide range of real-world tasks including, but not limited to, industrial applications, search and rescue, security surveillance, and human-like activities. Traditional approaches often face limitations depending on the specific task to be accomplished. In particular, navigation requires a high level of autonomy to adapt to both static and dynamic factors, such as changing environmental conditions. Software development for autonomous systems demands rigorous experimental testing and code validation. In this context, *Reinforcement Learning (RL)* techniques play a crucial role in enhancing decision-making capabilities, allowing robots to operate effectively across diverse application scenarios.

As mentioned in the previous sections, one of the main goals of UAV autonomy is the navigation of unknown environments and the achievement of different tasks without human assistance, particularly suited for space missions. The GNC system plays a crucial role in guaranteeing UAV stability and autonomy in complex scenarios, such as GPS-denied environments, and tracking optimal trajectories. The UIO-based navigation presented an efficient approach when dealing with partially observable and unstructured settings. On the other hand, the traditional techniques discussed in 3.5.4 and 5.1.1 exhibit various limitations when compared to recent advancements in Reinforcement Learning, as detailed in 6.2. From this perspective, RL techniques are becoming crucial in supporting the development of decision-making tasks in complex operational scenarios, overcoming the limitations of previous methods [59], [26], [63].

6.2 Deep Reinforcement Learning: Classification of Algorithms

RL techniques represent a methodology of learning how to implement actions in the environment by maximizing a numerical reward signal. The main key components that define how a RL agent learns to make decisions in an environment can be classified as: the *Agent*, as the entity that learns and makes decisions by interacting with the environment; the *Environment* in which the agent operates; the *State, s*, representing the environment at a given time; the *Action, a*, as the set of possible choices the agent can make in a given state; the *Policy, n*, as the strategy (or mapping) from states to actions that the agent follows to maximize rewards; the *Reward, r*, a numerical feedback signal from the environment that tells the agent the quality of a performed action; the *Value function, V*, that estimates the expected long-term reward helping the agent predict future rewards.
6.1 reports the basic diagram of a RL logic

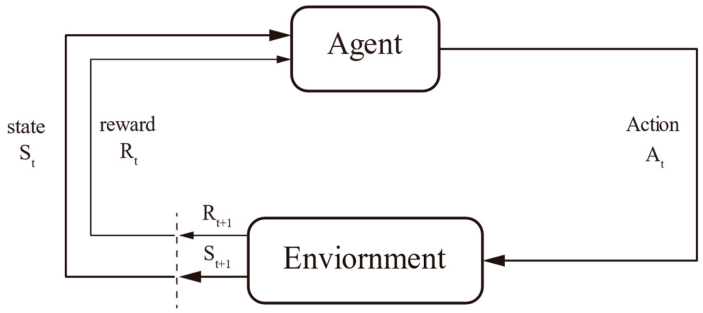


Figure 6.1: RL conceptual architecture. The agent takes the action a_t, resulting in a change in the state s_t that makes the agent closer (or not) to the target. The reward r_t quantifies how good the implemented action was, acting as a system's feedback, [10]

Deep Reinforcement Learning (DRL) extends traditional RL capabilities by incorporating deep neural networks (DNNs) to handle more complex problems, particularly for continuous state representation.
DRL approaches can be broadly classified into three main categories: value-based, policy-based, and model-based methods [19]. In value-based RL, the agent's primary goal is to determine a policy that maximizes the value function over a sequence of actions in the long run. The focus is on estimating how good it is to be in a particular state and selecting actions accordingly. In policy-based RL, the agent aims to directly learn the optimal policy that maximizes the objective function, and this approach can be further categorized into *deterministic* methods, where a specific action is chosen for each state, and *stochastic* methods, where actions are selected based on a probability distribution. In model-based RL, the agent relies on a predefined model of the environment. This model helps predict future states and rewards, enabling the agent to make informed decisions about how to act within such environment.

In recent years, significant progress has been made in utilizing advanced DRL algorithms, such as Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Deterministic Policy Gradient (DDPG) [26], [34], [21]. Particularly, PPO is designed to enhance stability and reliability by leveraging Trust Region Policy Optimization (TRPO), and is commonly used in UAV applications as it strikes a balance between stability, efficiency, and adaptability, making it well-suited for UAV autonomous navigation. Indeed, compared to DQN, which struggles with continuous actions, and DDPG which is less stable and sample-efficient, PPO offers robust performance in real-world UAV applications, including autonomous exploration, obstacle avoidance, and real-time trajectory optimization [67], [15], [48]. 6.2 provides a taxonomy of the main RL algorithms.

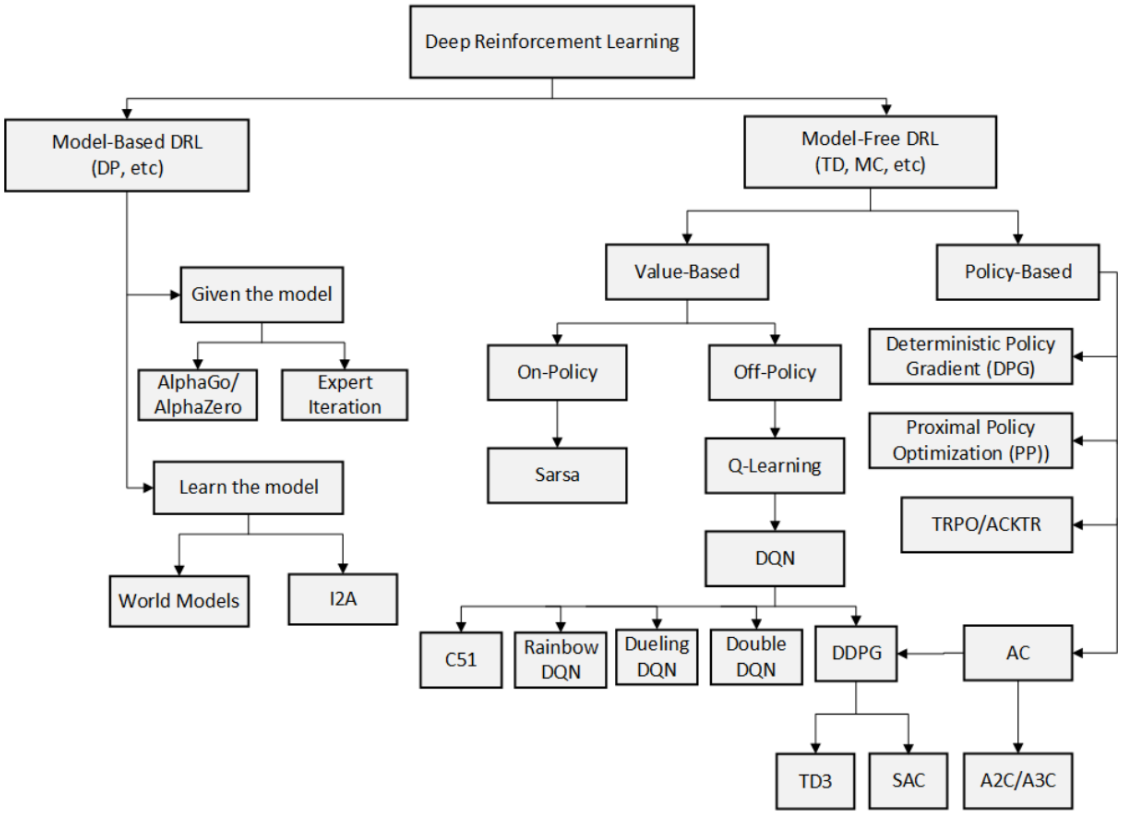


Figure 6.2: RL Taxonomy. PPO and DDPG belong to the *policy-based* technique, [34]

DRL techniques are also distinguished as *positive* or *negative*, based on the agent rewarding. Positive RL involves rewarding an agent for desirable actions, encouraging it to repeat those behaviors in the future. By reinforcing beneficial actions with rewards, the agent learns to maximize long-term gains. This approach is commonly used in scenarios where exploration and gradual improvement are crucial, as it helps the agent discover optimal strategies through trial and error. On the other hand, negative RL involves penalizing the agent for undesirable actions. The agent learns by avoiding penalties, steering its behavior toward actions that minimize negative outcomes. This technique is particularly useful in environments where preventing mistakes is crucial. Sometimes, both positive and negative approaches can be used in combination,

depending on the nature of the problem and the desired learning behavior of the agent.

6.2.1 Transformer and LSTM

Waypoint navigation involves autonomously guiding an agent through a sequence of waypoints to reach a destination. This often occurs in partially observable environments, where the agent does not have access to a full map and must rely on past observations. Without memory, DRL algorithms may lack a way to recall past observations, which causes limitations. Adding Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and Long Short-Term Memory (LSTM) to DRL algorithms expands their capabilities in several ways. These recurrent architectures allow DRL agents to handle partially observable environments and learn from sequences of observations.

Recurrent models process computations sequentially along the symbol positions of input and output sequences. By aligning these positions with computation steps, they generate a sequence of hidden states h_t, where each state depends on the previous hidden state h_{t-1} and the current input at position t. This sequential nature limits parallelization during training, making it increasingly challenging to handle long sequences due to memory constraints that restrict batching across examples.

To effectively manage sequential data and address the vanishing gradient issue in traditional RNNs, LSTM networks are commonly incorporated into DRL algorithms. This integration enhances policy learning by improving decision-making in uncertain and dynamic environments while efficiently capturing temporal dependencies in partially observable settings. In this regard, the integration of LSTMs enhances the UAV's ability to make timely maneuvers and access the risk of future collisions.

In particular, attention mechanisms have become crucial in achieving sequence modeling and transduction tasks, enabling the modeling of dependencies regardless of their distance within the input or output sequence.

The Transformer, introduced by Vaswani et al. [64] is a model architecture that avoids recurrence and instead relies entirely on self-attention mechanism to capture global dependencies between input and output, without using sequence-aligned RNNs or convolution. 6.3 shows the Transformer architecture

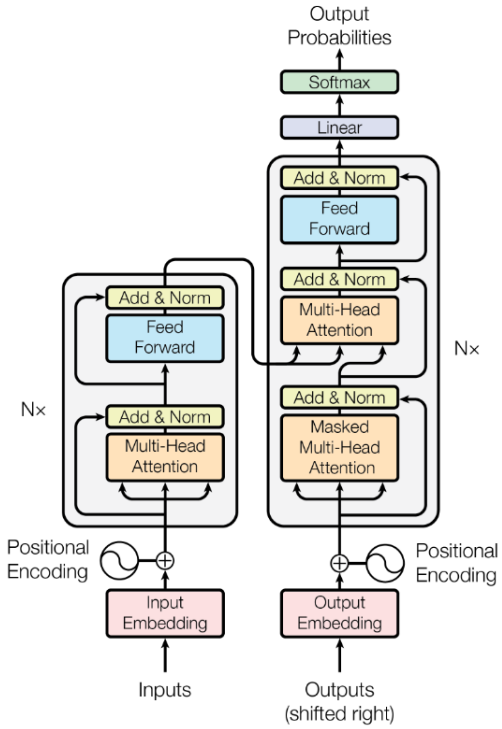


Figure 6.3: Transformer Architecture, *Encoder* and *Decoder* stacks.

In the Transformer model, two main elements can be defined: the *Encoder* and the *Decoder*. The Encoder consists of a stack of N_x identical layers, each of them containing two sub-layers: a *multi-head self-attention* mechanism, and a *position-wise fully connected feed-forward* network. A residual connection around each sub-layer followed by layer normalization is applied to enhance learning. This ensures that the output of each sub-layer is computed as

$$LayerNorm(x + Sublayer(x)) \tag{6.1}$$

where Sublayer(x) represents the function implemented by the respective sub-layer. Then, to maintain consistency across layers, both the sub-layers and embedding layers generate outputs of a fixed dimension d_{model}.

The decoder also comprises identical layers N_x. It retains the two sub-layers of the encoder, with an additional third sub-layer, which performs multi-head attention over the encoder's output. Like the encoder, the decoder applies residual connections around each sub-layer, followed by layer normalization. To ensure the predictions for position i depend only on the known outputs at positions $< i$, the self-attention sub-layer is modified by applying masking to prevent positions from attending to future positions. The *Attention* function can be described as mapping a *query (Q)* and a set of *key-value (K, V)* pairs to an output, all vectors. The output is computed as the weighted sum of the values, where each weight is determined by a compatibility function that measures the relevance of Q to its corresponding K. This component enhances the model's ability to capture complex patterns and relationships by transforming the attended represen-

tations into richer feature embeddings. Additionally, residual connections and layer normalization are applied to both sub-layers to facilitate stable training and efficient gradient flow.

As discussed, in a transformer architecture, the encoder processes the input sequence, generating context-aware representations, while the decoder leverages these representations to generate the output sequence. In the transformer-encoder, like the BERT, the input sequence is processed using a series of self-attention and feedforward layers, enabling it to capture complex dependencies within the data. The main architecture of the transformer-encoder is given in 6.4. Considering the application under analysis of

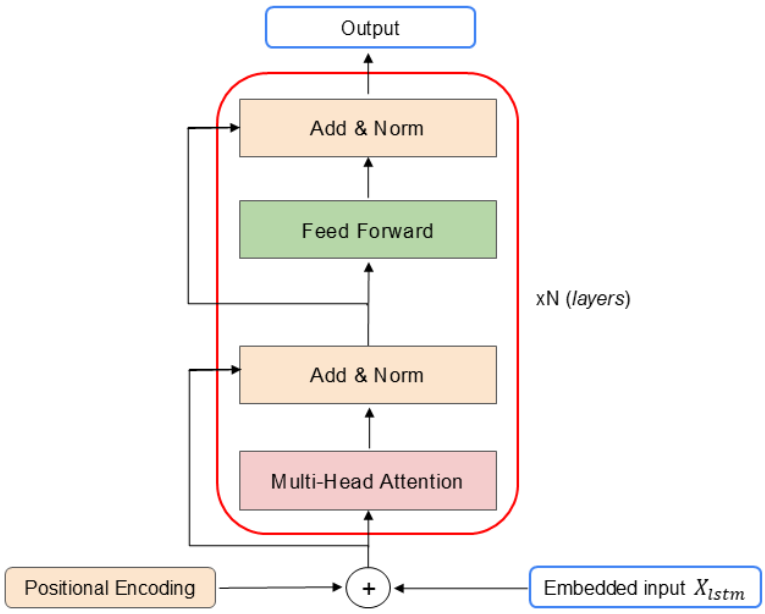


Figure 6.4: transformer-encoder Architecture

a flying quadcopter with limited onboard power, the choice of using a Transformer encoder over the full Transformer architecture is justified by its key advantages: reduced computational complexity, lower power consumption, and faster inference time. Indeed, the Transformer-encoder architecture omits the decoder stack, which has additional layers; this, in turn, cuts computational cost and memory roughly by half compared to the full model. In addition, the decoder in the Transformer uses encoder-decoder cross attention mechanism, which adds another expensive attention computation. In 6.2.4 the description of the Transformer-encoder integration along with DDPG and PPO algorithms is presented.

6.2.2 Proximal Policy Optimization

PPO belongs to the class of *on-policy* algorithms, as it learns and updates based on experiences generated by the current policy. Instead of using the gradient of the log-probability of the taken action to estimate the policy gradients, PPO performs the

update using the ratio between the new and old policy probabilities, scaled by the advantage function:

$$J_\theta = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t\right] \tag{6.2}$$

Here, $\mathbb{E}_t$ denotes the *expectation* over time steps t, and $A_t = Q(s_t, a_t) - V(s_t)$ is the *advantage function*, estimated using Generalized Advantage Estimation (GAE). However, directly maximizing (6.2) can lead to excessively large policy updates. To mitigate this, PPO introduces a clipped surrogate objective:

$$J^{\text{clip}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{6.3}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, and ϵ is a small constant (commonly 0.1 or 0.2). This clipping restricts the policy update to the range $[1-\epsilon, 1+\epsilon]$, preventing drastic changes and improving training stability.

PPO uses an *actor-critic* framework, where the *actor* selects actions according to the policy, and the *critic* evaluates the actions via a value function. The total loss typically combines three components:

- The clipped policy loss $J^{\text{clip}}(\theta)$,
- A value function loss, usually the mean-squared error between predicted and target state values,
- An entropy bonus to encourage exploration by discouraging premature convergence to deterministic policies.

The final PPO objective can be expressed as:

$$\mathcal{L}^{\text{PPO}} = \mathbb{E}_t\left[J^{\text{clip}}(\theta) - c_1 \cdot \mathcal{L}^{\text{VF}}(\theta) + c_2 \cdot \mathcal{H}[\pi_\theta](s_t)\right], \tag{6.4}$$

where $\mathcal{L}^{\text{VF}}(\theta)$ is the value function loss, $\mathcal{H}$ is the entropy of the policy, and c_1, c_2 are coefficients weighting the importance of each term.

6.2.3 Deep Deterministic Policy Gradient

The *Deep Deterministic Policy Gradient (DDPG)* algorithm is an off-policy, model-free reinforcement learning method designed for environments with *continuous action spaces*. It combines principles from *Deterministic Policy Gradient (DPG)* and *Deep Q-Networks (DQN)*, leveraging deep neural networks to approximate both the *actor* (policy) and the *critic* (Q-function). The actor network $\mu(s|\theta^\mu)$ deterministically maps each state to a specific action. In continuous action domains, where each action is represented by a real-valued number, the actor takes a state as input and outputs N continuous values—one for each action dimension. The *critic network* $Q(s, a|\theta^Q)$ evaluates the quality of state-action pairs by estimating the *Q-value*, which represents

the expected cumulative (discounted) reward of taking action a in state s. It receives both the state and the action as inputs and outputs a single scalar value $Q(s, a)$. The critic is trained by minimizing the following loss function:

$$L(\theta^Q) = \mathbb{E}\left[\left(Q(s_t, a_t|\theta^Q) - y_t\right)^2\right],$$

where the target $y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'})$, and Q', μ' are slowly updated *target networks* used to stabilize training. The actor is updated using the sampled *policy gradient*:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s_t}\left[\nabla_a Q(s, a|\theta^Q)\big|_{a=\mu(s)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)\right].$$

Since the policy is deterministic, the actor directly depends on the gradients of the critic's Q-function, making the entire system *fully differentiable* and trainable end-to-end via *stochastic gradient descent (SGD)*. One of the key strengths of DDPG is its *off-policy nature*, which allows experience to be stored in a large *replay buffer*, improving sample efficiency by enabling repeated training on past experiences. However, because the policy is deterministic, *exploration* must be explicitly introduced. This is typically done by adding *noise* (e.g., Ornstein-Uhlenbeck or Gaussian) to the actor's actions during training, encouraging the agent to explore the environment more effectively.

6.2.4 Network Integration

Based on the same setup of the first experimental part given in 4, the observation vector consists of odometric data coming from a simulated sensor, including the hexacopter pose, as well as linear and angular velocities.

$$\mathbf{X_{obs}} = \left[\ x, y, z, q_w, q_x, q_y, q_z, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z\ \right] \tag{6.5}$$

The LSTM and transformer-encoder framework are integrated in the PPO actor architecture, and X_{obs} represents the input state vector processed by the LSTM layer

$$X_{lstm} = LSTM(X_{obs}) \tag{6.6}$$

X_{lstm} has shape *(k, T, n)*, where k is the number of mini-batches per episode, T the sequence length as the number of time steps for each episode in the mini-batch, and n the embedding dimension, or the number of features per time step in each sequence. The embedded input vector X_{lstm} is then processed by each transformer layer in the MHA through the computation of the three separate learned weight matrices W^Q, W^K, and W^V to transform the input into Q, K and V

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{6.7}$$

Each weight matrix W^Q, W^K, W^V has a shape of

$$W^Q, W^K, W^V \in \mathbb{R}^n \times d_k$$

where d_k is the attention dimension per head.
After obtaining Q, K and V, the attention scores are computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6.8}$$

The softmax function normalizes the scores sum by determining how much each value V contributes to the output, and the result is a weighted sum of V that focuses on the most relevant information of the input sequence. By repeating the computation across multiple heads, the network attends to different aspects of the input simultaneously. The output of self-attention is passed through the FFN to refine feature extraction, and the final encoded representation is used to compute the action distribution (μ and Σ), which determines the agent's continuous actions for the PPO policy given in terms of the hexacopter's control velocities.

$$\mathbf{a}_t = \left[v_x, v_y, v_z, \omega_z\right] \in \mathbb{R}^4 \tag{6.9}$$

6.5 shows the architecture of the proposed network. The agent receives the observation vector from the environment, which are first processed through the LSTM layer to capture temporal dependencies. The LSTM output is then enriched with positional encoding and passed into the transformer-encoder, allowing the model to capture long-range dependencies and semantic history. This encoded representation is then used in the PPO loop to compute the advantage and update the policy.

6.2.5 Reward Structure

As given in 6.9, the learning agent's action represents the UAV's control velocity in the Cartesian space (x,y,z). Having both actions and observations defined, the reward function has to ensure that the agent understands the impact of its actions. The reward function is continuous and depends on two main parameters:

$$R = R_{SE} + R_{LAP} \tag{6.10}$$

the minimization of the norm error R_{SE}, and the minimization of the Lagrangian formulation, R_{LAP} as outlined in 6.1. The reward in e_p is cumulative as it minimizes the distance to the goal. For each episode, the final waypoint is clipped with a tolerance of ±0.20m, and the goal reward provides a positive reinforcement ($+100$) when the agent reaches the waypoint within the margin. This approach ensures that the agent learns to navigate in a 3D space while minimizing the relative distance to the goal, thereby

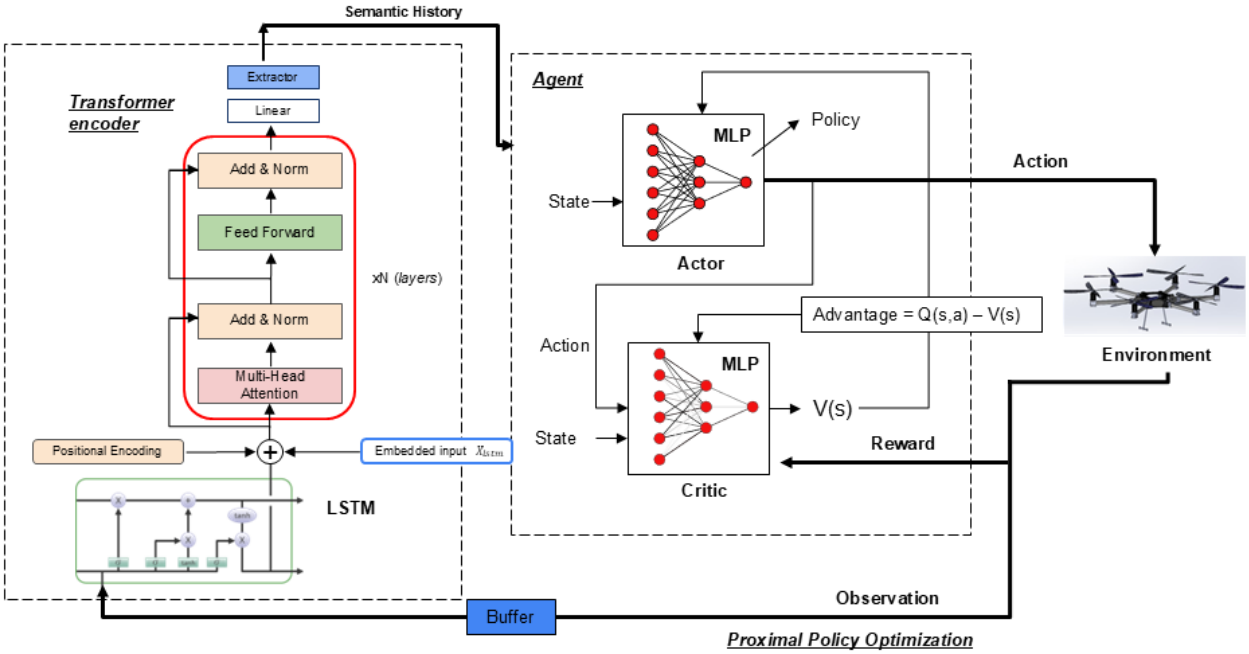


Figure 6.5: transformer-encoder PPO Network

Table 6.1: Reward function

Description	Reward Term
Distance Goal (R_{SE})	$\|\|e_p\|\|^2 = (x - x_d)^2 + (y - y_d)^2 + (z - z_d)^2$
LAP (R_{LAP})	$\mathcal{L} = (T - V)$

maximizing the reward. Among the infinite possible trajectories, the LAP ensures that those which minimize the system action are inherently the same for which the system equation of motion are respected. By implementing a reward component based on LAP, the aim is to directing the search along more "natural" trajectories which should provide smoother and less control intensive paths. The action is computed by the following relation:

$$\text{Action} = \int_{t_0}^{t_f} \mathcal{L}\, dt \tag{6.11}$$

Given the fact that the reward is cumulative, $R_{LAP} = \mathcal{L}$ is set so that, by minimizing the Lagrangian cumulative contribution, the learning is steered towards trajectories with least action. The Lagrangian incorporates both the system's kinetic (T) and potential (V) energy of the hexacopter, and its implementation in the reward function allows for a physic-informed learning of trajectories which takes into account the hexacopter dynamics.

6.3 Conclusions

This chapter provided an overview of the selected DRL network architecture and its integration for autonomous exploration using a Martian hexacopter. An initial taxonomy of DRL algorithms highlighted key categories: model-based, policy-based, and value-based approaches. Given the continuous action space of the environment, two algorithms, the probabilistic PPO and the deterministic DDPG, were selected for implementation. The architectures of the transformer and transformer-encoder networks were introduced and integrated with PPO and DDPG, respectively. Additionally, the reward function design, incorporating the physics informed Least Action principle strategy, is presented to then enable the agent to learn optimized and natural flight trajectories. The performance and application of the proposed network are evaluated in the following chapter.

Chapter 7

Deep Reinforcement Learning: Simulated Experiments

This chapter covers the setup implemented to create a dedicated simulation environment to test the DRL algorithms for autonomous hexacopter navigation. Unlike the previous simulator tailored for Mars exploration, this environment is configured under terrestrial conditions. The choice to begin with Earth-based simulation stems from the need to first validate a novel network architecture that integrates PPO and DDPG algorithms with a transformer-encoder module. Terrestrial environments offer a controlled and well-understood testbed, making them ideal for evaluating the learning stability, convergence behavior, and generalization capabilities of the proposed architecture. This initial phase ensures that the network can learn coherent navigation policies and operate reliably across standard scenarios. Once validated, the trained model will undergo a second-phase adaptation to Mars-like conditions through established transfer learning methodologies. This progression reflects standard space exploration protocols, where terrestrial validation is followed by analog site testing and eventual deployment.

7.1 Simulator Architecture in ROS and Gazebo

The hexacopter's physics and dynamics are modeled in Gazebo and ROS using nested scripts that handle physics plugins, connections, and configuration of the simulation environment. The API manages the interaction between the tested algorithms and the simulation environment in ROS/Gazebo by serving as a bridge for data exchange. It initializes the robot and environment, collects sensor data from ROS topics to form observations, and passes actions from the DRL algorithms back to the robot via ROS commands. The plugin calculates rewards based on task performance, handles episode resets, and ensures synchronization between training logic and the simulated robot. For the presented experiments, the DRL training runs externally in Python, communicating with Gazebo in real time through ROS interfaces.
The 3D simulator provides the environment (7.1) for training and testing the PPO agent

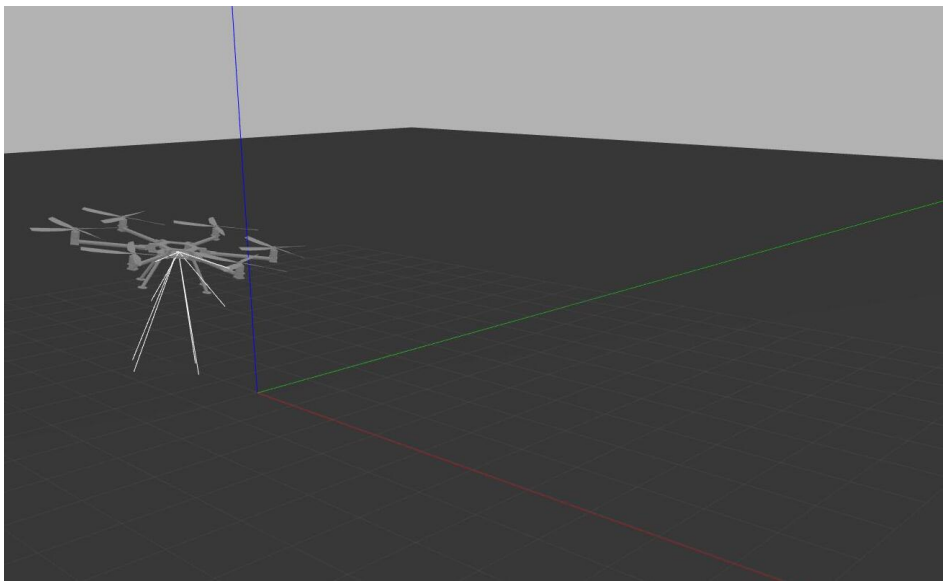


Figure 7.1: ROS Gazebo simulation environment

across episodes, enabling it to learn and effectively execute the waypoint navigation. According to the proposed task, the initial step is to train the agent to track the target waypoint $wp_g = [3.0, -3.0, 10.0]$ in an obstacle-free environment, serving as a foundation for training future models in more complex scenarios with obstacles. Once the environment is defined, the hexacopter takes off to the desired position in the 3D space without prior knowledge of the surroundings.

The episode terminates based on one of the following conditions:

- *Out of bounds*: The UAV moves outside the defined bounding box;
- *Timeout*: The episode reaches the maximum step limit of 1024;
- *Goal reached*: The UAV arrives at the waypoint within a tolerance of ±0.20m.

The learning process is developed according to a goal-oriented navigation method [9]. At the start of each episode, the environment is reset and the initial position is generated randomly. This enables the agent to learn a policy that generalizes across different starting conditions while consistently reaching the same goal ensuring a dynamic initialization of the path. The agent navigates from a variable starting position to a fixed goal, thus simulating real-world conditions where initial positions can vary but the destination remains constant.

7.2 Experimental Results

7.2.1 Training

The training is assessed over 500 episodes for a total of 512k steps, with each episode further divided into 8 mini-batches of 128 samples each, for both PPO and DDPG and

augmented PPO and DDPG with the transformer. The training performances of the agent are evaluated according to the reward total average return (TAR)

$$TAR = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=0}^{T}\gamma^{t} r_t\right) \tag{7.1}$$

where N is the number of episodes, γ the reward gain determining how future rewards are valued, and T the episode length. The hyperparameters of the simulation are given in Table 7.2, while 7.2 shows the cumulative training reward of the two algorithms.

Overall, all the four tested algorithms show stable and gradual learning over 500

Table 7.1: PPO and DDPG Parameters

PPO Network		
Hyperparameter	Coefficient	Value
Learning rate (Actor)	lr_a	0.0001
Learning rate (Critic)	lr_c	0.0003
Actor Network	–	[64, 64]
Critic network	–	[64, 64]
Clip ratio	ϵ	0.1
GAE factor	τ	0.001
Batch number	bt	8
Batch size	bt_s	128
Episodes	Ep	500
DDPG Network		
Hyperparameter	Coefficient	Value
Learning rate (Actor)	lr_a	0.001
Learning rate (Critic)	lr_c	0.002
Actor Network	–	[100, 100]
Critic network	–	[100, 100]
Clip ratio	ϵ	0.1
Buffer size	–	1e5
Batch number	bt	8
Batch size	bt_s	64
Episodes	Ep	500
Transformer-encoder		
Embedding Dimension	d_{model}	13
Attention Heads	h	4
Transformer Layers	L	2
Dropout Rate	P	0.1
Feedforward Dimension	d_{ff}	13
Layer Norm Epsilon	ϵ_{norm}	1e−6

episodes. The evolution of the TAR follows a similar trend for both algorithms, with both converging over the training episodes. The baseline PPO demonstrates moderate final performance, while PPO combined with the transformer-encoder shows significantly improved return, faster convergence, and overall higher stability, highlighting the advantage of temporal attention. It stabilizes around 270 episodes, while the stan-

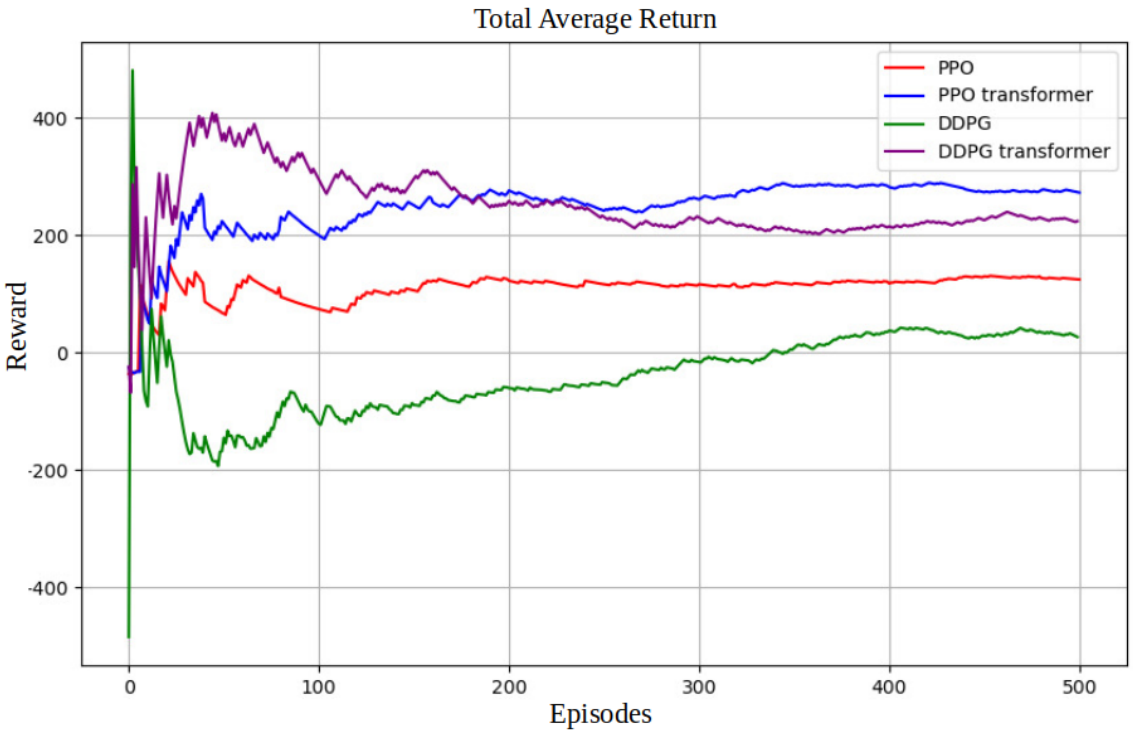


Figure 7.2: PPO and Augmented PPO: TAR Comparison

dard PPO without the transformer settles at approximately 125 episodes. The later convergence, around 350 episodes, suggests that the augmented PPO requires more training to stabilize and ultimately outperform PPO. In contrast, DDPG exhibits high volatility and poor convergence with initial marked undershoot, struggling to learn effective policies. However, when augmented with the transformer, DDPG's performance improves substantially, achieving higher returns and faster learning close to the augmented PPO, though with slightly more variability.
Overall, transformer-based models clearly outperform their standard counterparts, particularly enhancing learning efficiency and policy generalization. Indeed, this improvement can be attributed to the benefits of the integration of LSTM and transformer-encoder in handling the Partially Observable Markov Decision Process (POMDP). As the LSTM contributes to retain past information, allowing the agent to infer hidden states and make better decisions in partially observable environments, it results particularly effective at capturing long-term dependencies and filtering out noise, enabling the agent to maintain a more informed belief state over time.

7.2.2 Trajectory Planner Testing

In the test phase, the starting waypoint was generated randomly $wp_s = [0.75, -0.30, 0.45]$, while the final goal is set to $wp_g = [5.6, 2.3, 7.0]$. The test was conducted over 50 episodes, yielding a success rate of 91.6% for the augmented PPO and 71.6% for the augmented DDPG in reaching the target waypoint. 7.2 summarizes the main evaluation criteria for each algorithm. The proposed methodology is then applied for PPO

Table 7.2: Algorithm Comparison

Parameter	Augmented PPO	Augmented DDPG
Test episodes	50	50
Goal reached	45	36
Success rate (%)	91.6	71.6
Average travel time (sec)	8.17	13.18

with the transformer, as it outperforms DDPG in test, and the implemented trajectory is compared with a polynomial trajectory planner by computing the action value of a batch sample for the obtained trajectories in contrast to their respective 9th-order polynomial equivalents. The 9th-order polynomial is selected as it ensures continuous differentiability up to the jounce (the fourth derivative of position). Overall, the proposed approach results in an action value with a mean and median of 11% and 13% respectively lower than that of the polynomial trajectory planner. 7.3 illustrates one of the generated trajectories along with its corresponding 9th-order polynomial.

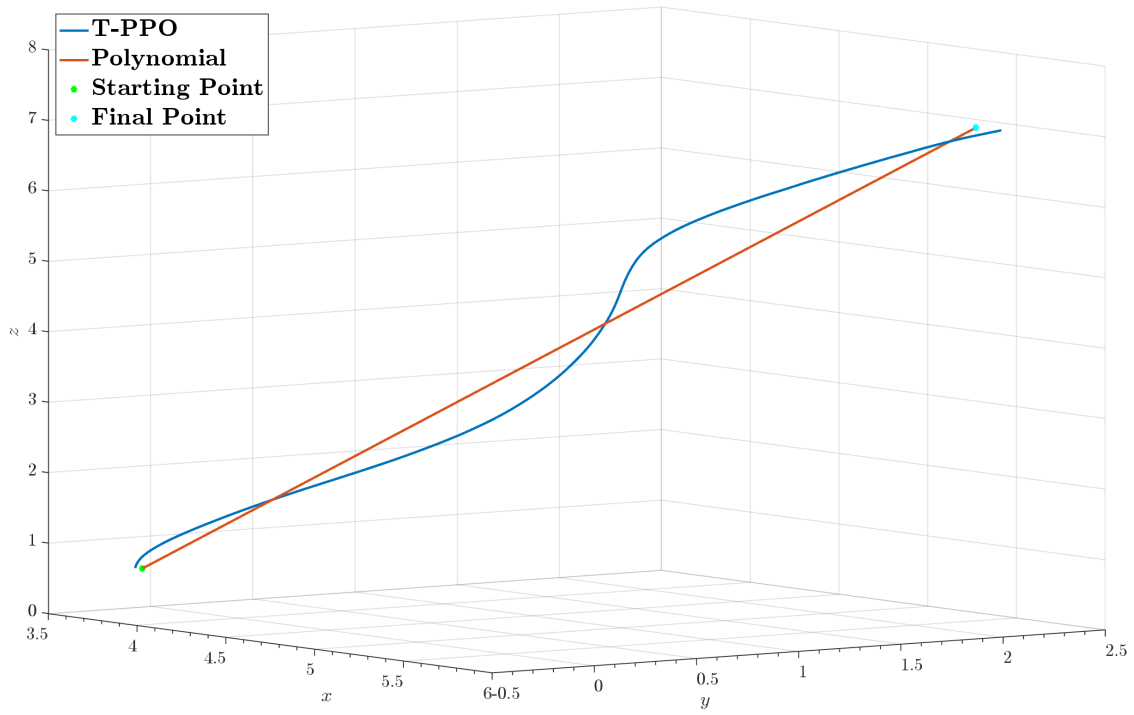


Figure 7.3: Hexacopter Trajectory, PPO with transformer

7.3 Conclusions

This chapter introduced a novel PPO network architecture enhanced with LSTM and a transformer-encoder to tackle complex autonomous navigation challenges for UAVs, specifically for the hexacopter. First, the simulation of the 3D environment and the hexacopter dynamics were modeled using Gazebo and ROS plugins. The framework was evaluated in an obstacle-free environment to assess its suitability and focus on

the agent's ability to reach the final waypoint following a goal-oriented manner. The comparison with the standard PPO algorithm highlighted the benefits of integrating the transformer architecture, enhancing the augmented PPO's effectiveness in complex, partially observable environments that require long-term memory and strategic decision-making. The reward function was specifically designed for continuous action spaces by integrating two key components: a position error term and a physics-informed element that minimized the LAP. Then, the trained agent was tested for evaluation demonstrating the superiority of the transformer-enhanced approach and showing that, by incorporating a reward component based on LAP, the agent was guided the search toward more natural trajectories, resulting in smoother paths that require less intensive control.

Chapter 8

Conclusions and Contributions

This final chapter provides the analysis and interpretation of the results obtained from the simulations conducted in this research, highlighting the future works that will complete the study for the validation of the MHex autonomous navigation.

8.0.1 Conclusion and Contribution

In this thesis, the analysis of the autonomous exploration for a Martian hexacopter is presented. Firstly, the mission requirements setting the autonomous mission context and operations are defined. Then, starting from the investigation of the MSH conceptual design previously presented by NASA in 2020, the main subsystems of the MHex are described, emphatizing the structure of a system able to fly on Mars without the aid of GPS, with limited onboard power, and aerodynamically optimized. Secondly, the entire structure of the GNC subsystem is described and implemented according to the mission scenario. Rather than considering only the well known family of Kalman filters and their application to the MHex system, the research introduces a new approach by analyzing and then implementing the UIO observer, and comparing its performances to the EKF for the same system.
Following the Matlab and Simulink implementation, two different scenarios are investigated. In the first case, only external disturbances acting on the MHex are considered, simulating the wind impacting the trajectory, while the second setting also takes into account the bias coming from the onboard sensors. The results showed better performances of the UIO, and the observer is then implemented in the ROS/Gazebo simulator to simulate the whole mission scenario along with the SLAM navigation. Based on the simulation results, the principal contributions of this part can be summarized as follow:

- The mission analysis follows specific requirements for both the MHex system and the concept of operations to autonomously explore the Belva crater;
- The MHex architecture presents a realistic and tangible system to autonomously explore the Martian environment. The onboard sensors are chosen to ensure

accurate odometry data and detailed point clouds to represent the explored surroundings;

- The implemented simulator offers a realistic testing platform for simulating the Martian exploration, according to the Mars aerodynamic parameters;
- The GNC system integrates the UIO observer to estimate the unknown states for a precise SLAM navigation, facilitating precise SLAM navigation and offering a promising solution for the autonomous exploration in future missions on Mars.

As a parallel contribution, a DRL-based navigation strategy is presented to address the same exploration tasks, but considering a novel system based on the network integration of both PPO and DDPG algorithms, and a transformer-encoder architecture. For this second part of the research, the contributions are:

- The integration of DRL algorithms, LSTM layer, and transformer encoder shows improved performances in learning partially observable scenarios;
- The reward function design combines position error minimization with a physics-informed term derived from the LAP model;
- A full simulation environment, constructed by integrating Python, Gazebo, and ROS, offers a robust platform for UAV navigation tasks. Additionally, a fully integrated Python-based test bench is implemented to support DRL algorithm evaluation, ensuring compatibility with hardware-in-the-loop (HIL) systems and future flight tests.

8.0.2 Future (On-going) Work

In addition to the contributions gained from the presented study, two main phases are planned as a complementary contribution to the implemented SIL analysis:

Python Test-bench

As a final step for SIL validation and future HIL implementation, a secondary simulation architecture was developed by integrating the Python-based mathematical model of the hexacopter with the DRL network described in Section 6.2.4. This setup serves as a bridge toward HIL testing and deployment in real flight scenarios. The proposed framework is given in 8.1

To progress toward a flight-ready system, the DRL agent, originally trained in a fixed setup, is then trained to navigate toward randomly positioned waypoints. This adaptation allows the agent to generalize across diverse scenarios and optimize its trajectory along multiple waypoints, always in accordance to the LAP principle integrated in the reward function for smoothing the final trajectories.

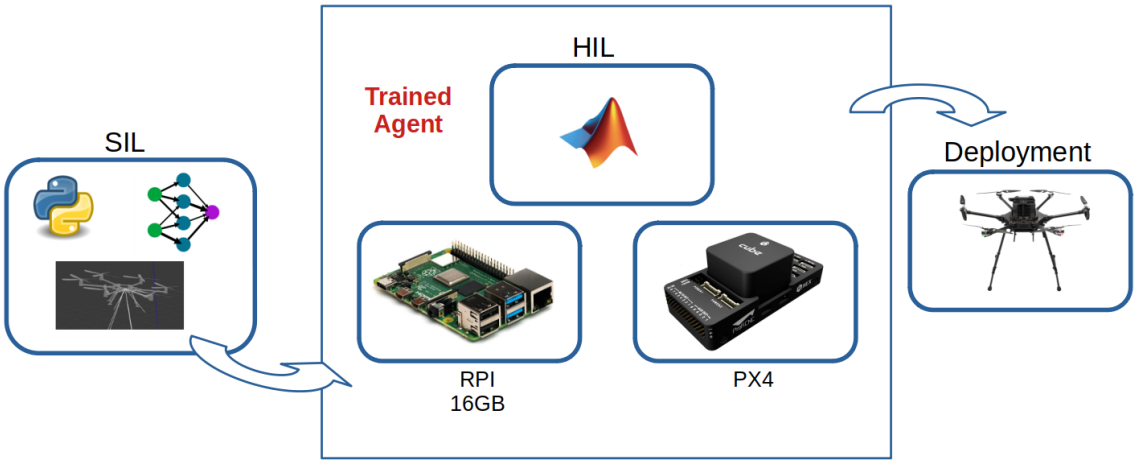


Figure 8.1: SIL, HIL and Deployment integration

HIL Implementation

The examination of the hexacopter gave consistency and substantial capability for autonomous operation on Mars in the context of future exploration, even if the experimental verification of both rotor structural design and aerodynamic performance requires further analysis to validate the entire MHex architecture,
The Ardupilot integration in the simulator pipeline would allow for reducing the effort required to transition from SIL to HIL. Coupled with the Python-based test bench, this setup would provide a flexible foundation for developing and deploying DRL-based HIL experiments and prototyping to validate the GNC system, even if limited to terrestrial flight conditions.
Opportunities with space agencies are also under consideration to validate the real testing.

1. **Sopegno L.**, Casalino L., *Optimization of a Sample Return Mission from mars: Mars Ascent Vehicle Design*, MSc Thesis, Polytechnic of Turin, December 2019.

2. **Sopegno, L.**, Valavanis, K. P., Rutherford, M. J., Casalino, L. "*Mars sample Return Mission: Mars Ascent Vehicle Propulsion Design*". In 2020 IEEE Aerospace Conference (pp. 1-9), March 2020, IEEE.

3. **Sopegno, L.**, Livreri P., Stefanovic M., Valavanis K.P, "*Linear Quadratic Regulator: A Simple Thrust Vector Control System for Rockets*", IEEE Mediterranean Conference on Control and Automation MED 2022, Athens, Greece.

4. **Sopegno, L.**, Livreri P., Valavanis K.P "*Using UAVs for future mission on Mars*", International Aerospace Conference IAC, 2022, Paris.

5. **Sopegno, L.**, Livreri P., Stefanovic M., Valavanis K.P, "*Thrust Vector Controller Comparison for a Finless Rocket*", MDPI Machines, March 2023.

6. **Sopegno, L.**, Martini S., Pedone S., Fagiolini A., Rizzo A., Stefanovic M., Rutherford M.J., Livreri P., Valavanis K.P., "*An Advanced Hexacopter for Mard Exploration: Attitude Control and Autonomous Navigation*", IEEE Aerospace and Electronic Systems Journal, February 2024.

7. **Sopegno L.**, Valavanis K.P., Rutherford J. M., Livreri P., *An Advanced Hexacopter for Autonomous Exploration of Mars: Attitude Control and Navigation Strategies*, PhD Thesis for the Comprehensive Exam, in partial fulfillment of the requirements for the PhD Degree, May 2024.

8. **Sopegno, L.**, Pedone, S., Rutherford, M. J., Livreri, P., Valavanis, K.P., "Evaluation and Comparison of State and Disturbance Observers for a Hexacopter", accepted at the International Conference on Unmanned Aircraft Systems, Chania (Greece), June 4-7, 2024.

9. **Sopegno L.**, Cirrincione G., Martini S., Rutherford M.J., Livreri P., Valavanis K.P., "Transformer-based Physic Informed Proximal Policy Optimization for UAV Autonomous Navigation", accepted at the 2025 International Conference on Unmanned Aircraft Systems (ICUAS), Charlotte, NC, USA, May 14-17, 2025.

Bibliography

[1] Farzin Amzajerdian, Diego Pierrottet, Glenn D Hines, Larry Petway, Bruce Barnes, and John M Carson. Development of navigation doppler lidar for future landing missions. In *AIAA SPACE 2016*, page 5590. 2016.

[2] Sheikh Izzal Azid, Krishneel Kumar, Maurizio Cirrincione, and Adriano Fagiolini. Robust motion control of nonlinear quadrotor model with wind disturbance observer. *IEEE Access*, 9:149164–149175, 2021.

[3] Sheikh Izzal Azid, Krishneel Kumar, Maurizio Cirrincione, and Adriano Fagiolini. Wind gust estimation for precise quasi-hovering control of quadrotor aircraft. *Control Engineering Practice*, 116:104930, 2021.

[4] J Balaram, MiMi Aung, and Matthew P Golombek. The ingenuity helicopter on the perseverance rover. *Space Science Reviews*, 217(4):56, 2021.

[5] J Balaram, IJ Daubar, J Bapst, and T Tzanetos. Helicopters on mars: Compelling science of extreme terrains enabled by an aerial platform. In *Ninth International Conference on Mars*, volume 2089, page 6277, 2019.

[6] Simone Baldi, Danping Sun, Xin Xia, Guopeng Zhou, and Di Liu. Ardupilot-based adaptive autopilot: Architecture and software-in-the-loop experiments. *IEEE Transactions on Aerospace and Electronic Systems*, 58(5):4473–4485, 2022.

[7] Blender Foundation. Blender. `https://www.blender.org/`, 2024. [Accessed May 3, 2024].

[8] Teng Hooi Chan, Henrik Hesse, and Song Guang Ho. Lidar-based 3d slam for indoor mapping. In *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, pages 285–289. IEEE, 2021.

[9] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[10] Cheng Chen, Jiantao Yu, and Songrong Qian. An enhanced deep q network algorithm for localized obstacle avoidance in indoor robot path planning. *Applied Sciences*, 14(23):11195, 2024.

[11] Daniel R Cremons. The future of lidar in planetary science. *Frontiers in Remote Sensing*, 3:1042460, 2022.

[12] Abhijit Das, Frank Lewis, and Kamesh Subbarao. Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, 56(1):127–151, 2009.

[13] Soumyo Dutta, Christopher D Karlgaard, David Kass, Michael Mischna, and Gregorio G Villar III. Postflight analysis of atmospheric properties from mars 2020 entry, descent, and landing. *Journal of Spacecraft and Rockets*, 60(3):1022–1033, 2023.

[14] Andreas Elsaesser, David J Burr, Paul Mabey, Riccardo Giovanni Urso, Daniela Billi, Charles Cockell, Hervé Cottin, Adrienne Kish, Natalie Leys, Jack JWA van Loon, et al. Future space experiment platforms for astrobiology and astrochemistry research. *npj Microgravity*, 9(1):43, 2023.

[15] M Ferguson and K Law. Learning robust and adaptive real-world continuous control using simulation and transfer learning. *ArXiv, abs/1802.04520*, 2018.

[16] Garmin. Altimeter garmin v3 lite, 2023. Accessed: September 2023.

[17] Mohinder S Grewal, Angus P Andrews, and Chris G Bartone. Kalman filtering. 2020.

[18] Bao-Zhu Guo and Zhi-Liang Zhao. *Active disturbance rejection control for nonlinear systems: An introduction.* John Wiley & Sons, 2016.

[19] Surbhi Gupta, Gaurav Singal, and Deepak Garg. Deep reinforcement learning techniques in diversified domains: a survey. *Archives of Computational Methods in Engineering*, 28(7):4715–4754, 2021.

[20] Hovermap. Hovermap 3d lidar, 2023. Accessed: September 2023.

[21] Jianghang Huang, Huifang Li, Yaoyao Lu, and C Senchun. A comparison study of dqn and ppo based reinforcement learning for scheduling workflows in the cloud. In *The 7th International Workshop on Advanced Computational Intelligence and Intelligent Informatic*, 2021.

[22] Ignitarium. 3d lidar slam – graph slam explained, 2025. Accessed: March 2025.

[23] IntelRealSense. t256 tracking camera, 2023. Datasheet camera.

[24] Wayne Johnson, Shannah Withrow-Maser, Larry Young, Carlos Malpica, Witold JF Koning, Winnie Kuang, Mireille Fehler, Allysa Tuano, Athena Chan, Anubhav Datta, et al. Mars science helicopter conceptual design. Technical report, 2020.

[25] Gerhard Kminek, Michael A Meyer, David W Beaty, Brandi L Carrier, Timothy Haltigin, and Lindsay E Hays. Mars sample return (msr): planning for returned sample science. *Astrobiology*, 22(S1):S–1, 2022.

[26] Maxim Lapan. *Deep reinforcement learning hands-on*. Packt Publishing Ltd, 2024.

[27] Pascal Lee, E Kommedal, A Horchler, E Amoroso, K Snyder, and AF Birgisson. Lofthellir lava tube ice cave, iceland: Subsurface micro-glaciers, rockfalls, drone lidar 3d-mapping, and implications for the exploration of potential ice-rich lava tubes on the moon and mars. In *50th Annual Lunar and Planetary Science Conference*, number 2132, page 3118, 2019.

[28] Peng Li, Di Liu, and Simone Baldi. Adaptive integral sliding mode control in the presence of state-dependent uncertainty. *IEEE/ASME Transactions on Mechatronics*, 27(5):3885–3895, 2022.

[29] Peng Li, Di Liu, Xin Xia, and Simone Baldi. Embedding adaptive features in the ardupilot control architecture for unmanned aerial vehicles. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 3773–3780. IEEE, 2022.

[30] Hanchen Lu, Hongming Shen, Bailing Tian, Xuewei Zhang, Zhenzhou Yang, and Qun Zong. Flight in gps-denied environment: Autonomous navigation system for micro-aerial vehicle. *Aerospace Science and Technology*, 124:107521, 2022.

[31] Simone Martini, Serhat Sönmez, Alessandro Rizzo, Margareta Stefanovic, Matt J Rutherford, and Kimon P Valavanis. Euler-lagrange modeling and control of quadrotor uav with aerodynamic compensation. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 369–377. IEEE, 2022.

[32] Simone Martini, Margareta Stefanovic, Alessandro Rizzo, Matthew J Rutherford, Patrizia Livreri, and Kimon P Valavanis. A benchmark framework for testing, evaluation, and comparison of quadrotor linear and nonlinear controllers. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 471–478. IEEE, 2023.

[33] Simone Martini, Kimon P Valavanis, Margareta Stefanovic, Matthew J Rutherford, and Alessandro Rizzo. Correction to the euler lagrange multirotor model with euler angles generalized coordinates. *Journal of Intelligent & Robotic Systems*, 110(1):17, 2024.

[34] João J Martins, Alexandre Amaral, and André Dias. Deep reinforcement learning framework for uav indoor navigation. In *2024 7th Iberian Robotics Conference (ROBOT)*, pages 1–8. IEEE, 2024.

[35] Fernando Mier-Hicks, Havard Fjær Grip, Arash Kalantari, Scott Moreland, Benjamin Pipenberg, Matthew Keennon, Timothy K Canham, Michael Pauken, Emmanuel Decrossas, Theodore Tzanetos, et al. Sample recovery helicopter. In *2023 IEEE Aerospace Conference*, pages 1–11. IEEE, 2023.

[36] Robert Milijas, Lovro Markovic, Antun Ivanovic, Frano Petric, and Stjepan Bogdan. A comparison of lidar-based slam systems for control of unmanned aerial vehicles. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1148–1154. IEEE, 2021.

[37] Jeroen Minnema. Towards component-based, sensor-independent and back-end independent slam. Master's thesis, University of Twente, 2020.

[38] NASA. High Resolution Imaging Science Experiment. [Accessed: 24-April-2024].

[39] NASA.org. Technolgy helicopter status, 2022. Accessed: September 2023.

[40] NASA.org. Viking project, 2022. Accessed: February 2024.

[41] Akash Patel, Avijit Banerjee, Björn Lindqvist, Christoforos Kanellakis, and George Nikolakopoulos. Design and model predictive control of a mars coaxial quadrotor. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–11. IEEE, 2022.

[42] Salvatore Pedone and Adriano Fagiolini. Racecar longitudinal control in unknown and highly-varying driving conditions. *IEEE Transactions on Vehicular Technology*, 69(11):12521–12535, 2020.

[43] Salvatore Pedone and Adriano Fagiolini. Racecar longitudinal control in unknown and highly-varying driving conditions. *IEEE Trans. on Vehicular Technology*, 69(11):12521–12535, 2020.

[44] Salvatore Pedone and Adriano Fagiolini. Robust discrete-time lateral control of racecars by unknown input observers. *IEEE Transactions on Control Systems Technology*, 31(3):1418–1426, 2023.

[45] Salvatore Pedone, Maja Trumić, Kosta Jovanović, and Adriano Fagiolini. Robust and decoupled position and stiffness control for electrically-driven articulated soft robots. *IEEE Robotics and Automation Letters*, 7(4):9059–9066, 2022.

[46] Salvatore Pedone, Maja Trumić, and Adriano Fagiolini. Lateral wind estimation and backstepping compensation for safer self-driving racecars. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2023.

[47] Salvatore Pedone, Maja Trumić, Kosta Jovanović, and Adriano Fagiolini. Robust and decoupled position and stiffness control for electrically-driven articulated soft robots. *IEEE Robotics and Automation Letters*, 7(4):9059–9066, 2022.

[48] Huy X Pham, Hung M La, David Feil-Seifer, and Luan V Nguyen. Autonomous uav navigation using reinforcement learning. *arXiv preprint arXiv:1801.05086*, 2018.

[49] Michael K Pitt, Ralph dos Santos Silva, Paolo Giordani, and Robert Kohn. On some properties of markov chain monte carlo simulation methods based on the particle filter. *Journal of Econometrics*, 171(2):134–151, 2012.

[50] Maria Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46):3736–3741, 2004.

[51] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[52] Shikai Shao, Shuangyin Xu, Yuanjie Zhao, and Xiaojing Wu. Unknown input observer-based fixed-time trajectory tracking control for quav with actuator saturation and faults. *Drones*, 7(6):344, 2023.

[53] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[54] JI Simon, K Hickman-Lewis, BA Cohen, LE Mayhew, DL Shuster, V Debaille, EM Hausrath, BP Weiss, T Bosak, M-P Zorzano, et al. Samples collected from the floor of jezero crater with the mars 2020 perseverance rover. *Journal of Geophysical Research: Planets*, 128(6):e2022JE007474, 2023.

[55] Laura Sopegno, Simone Martini, Salvatore Pedone, Adriano Fagiolini, Matthew J. Rutherford, Margareta Stefanovic, Alessandro Rizzo, Patrizia Livreri, and Kimon P. Valavanis. An advanced hexacopter for mars exploration: Attitude control and autonomous navigation. *IEEE Transactions on Aerospace and Electronic Systems*, 60(3):3569–3581, 2024.

[56] Laura Sopegno, Kimon P Valavanis, Matthew J Rutherford, and Lorenzo Casalino. Mars sample return mission: mars ascent vehicle propulsion design. In *2020 IEEE Aerospace Conference*, pages 1–9. IEEE, 2020.

[57] Vivian Z. Sun and Kathryn M. Stack. Geologic map of jezero crater and the nili planum region, mars. *Scientific Investigations Map*, 2020.

[58] Shreyas Sundaram and Christoforos N. Hadjicostis. Delayed observers for linear systems with unknown inputs. *IEEE Transactions on Automatic Control*, 52(2):334–339, 2007.

[59] Hamid Taheri, Seyed Rasoul Hosseini, and Mohammad Ali Nekoui. Deep reinforcement learning with enhanced ppo for safe mobile robot navigation. *arXiv preprint arXiv:2405.16266*, 2024.

[60] Theodore Tzanetos, MiMi Aung, J Balaram, Havard Fjrer Grip, Jaakko T Karras, Timothy K Canham, Gerik Kubiak, Joshua Anderson, Gene Merewether, Michael Starch, et al. Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout. In *2022 IEEE Aerospace Conference (AERO)*, pages 01–19. IEEE, 2022.

[61] Theodore Tzanetos, Jonathan Bapst, Gerik Kubiak, Luis Phillipe Tosi, Sam Sirlin, Roland Brockers, Jeff Delaune, Håvard Fjær Grip, Larry Matthies, J Balaram, et al. Future of mars rotorcraft-mars science helicopter. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–16. IEEE, 2022.

[62] USGS, U.S. Geological Survey. Scientific Investigations Map 3464. PDF, 2024. PDF file.

[63] Roan Van Hoa, Tran Duc Chuyen, Nguyen Tung Lam, Tran Ngoc Son, Nguyen Duc Dien, and Vu Thi To Linh. Reinforcement learning based method for autonomous navigation of mobile robots in unknown environments. In *2020 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 266–269. IEEE, 2020.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[65] Inc. Velodyne Lidar. Velodyne vlp-16 lidar, 2023. Datasheet.

[66] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[67] Zhuang Wang, Hui Li, Zhaoxin Wu, and Haolin Wu. A pretrained proximal policy optimization algorithm with reward shaping for aircraft guidance to a moving destination in three-dimensional continuous space. *International Journal of Advanced Robotic Systems*, 18(1):1729881421989546, 2021.

[68] Xiaobin Xu, Lei Zhang, Jian Yang, Chenfei Cao, Wen Wang, Yingying Ran, Zhiying Tan, and Minzhou Luo. A review of multi-sensor fusion slam systems based on 3d lidar. *Remote Sensing*, 14(12):2835, 2022.

[69] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[70] Minghui Zheng, Xu Chen, and Masayoshi Tomizuka. Extended state observer with phase compensation to estimate and suppress high-frequency disturbances. In *2016 American Control Conference (ACC)*, pages 3521–3526. IEEE, 2016.

Appendix A

Unknown Input Observer and Extended State Observer Comparison

Another potential alternative method to the proposed analysis between the EKF filter and UIO observer is the Extended State Observer (*ESO*), the main task of which is the estimation of both system states and a large class of uncertainties, relying on the assumption of slow time-varying/low-frequency disturbances. To extend the performance range of the ESO from low frequencies to high frequencies, solutions in literature have been proposed, such as the use of a phase compensator to recover the phase loss of traditional ESOs [70]. This complementary part of the research uses a terrestrial flight hexacopter as the testbed platform to compare and evaluate both UIO and ESO in a Matlab/Simulink environment.

A.1 Modeling of disturbance phenomena

During hexacopter flights, realistic assumptions affect not only attitude stability but also trajectory tracking, as well as accuracy of the gathered sensor data. According to the presented analysis, additional modeling of both external and system disturbances affecting the hexacopter dynamic is added to the model.

Gyrocopic Effect

The gyroscopic effect is caused by rotating components in the hexacopter such as propellers and IMU gyroscope sensors. The gyroscopic effect is modeled as [31]

$$G_{\text{gyro}}^{6\times6} = \sum_{i=1}^{6} S(\nu) \times I_{\text{r}} \omega_i e_3 \tag{A.1}$$

with the skew-symmetric matrix $S(\nu)$ of ν components.

Parasitic Drag

Aerodynamic forces acting on the hexacopter are modeled according to different types of drag. The parasitic drag is an important factor that directly affects hexacopter efficiency. Parasitic drag is defined and modeled as

$$D_{\text{par}} = K_{\text{par}}|\mathbf{V}_B|\mathbf{V}_B \tag{A.2}$$

$$K_{\text{par}} = \frac{1}{2}\rho S C_{D_{\text{par}}} \tag{A.3}$$

where V_B is the linear velocity in body coordinates, while ρ is the air density, C the hexacopter lateral area, and C_{par} the drag coefficient.

Sensor Noise

Random fluctuations in measurements from onboard sensors usually arise due to electronic interference and manufacturing imperfections. The adopted approach considers adding a stochastic term to the sensor measurements, representing the random fluctuations associated with the sensor's biases. Even though the mission scenario differs from the one simulated in ROS/Gazebo for the MHex, having a both different aerodynamic conditions and system parameters, such as gravity, inertia and mass, the observer performances are compared and evaluated, showing the superiority of the UIO over the ESO expecially when it comes to sensor bias.
The next sections provide the analytical description of the ESO, together with the results obtained in the simulations. For the UIO analytical description, the author is pointed to 3.4.

Extended State Observer

The proposed ESO is based on the well-established theory described in [18]. In this setting, based on (3.6), $z = (x^\top, \delta^\top)^T$ defines the augmented state, having included the unknown input as an additional state variable. Assuming, as in UIO context, as system's output vector $y_z = y$, the following Luenberger-like relation is obtained

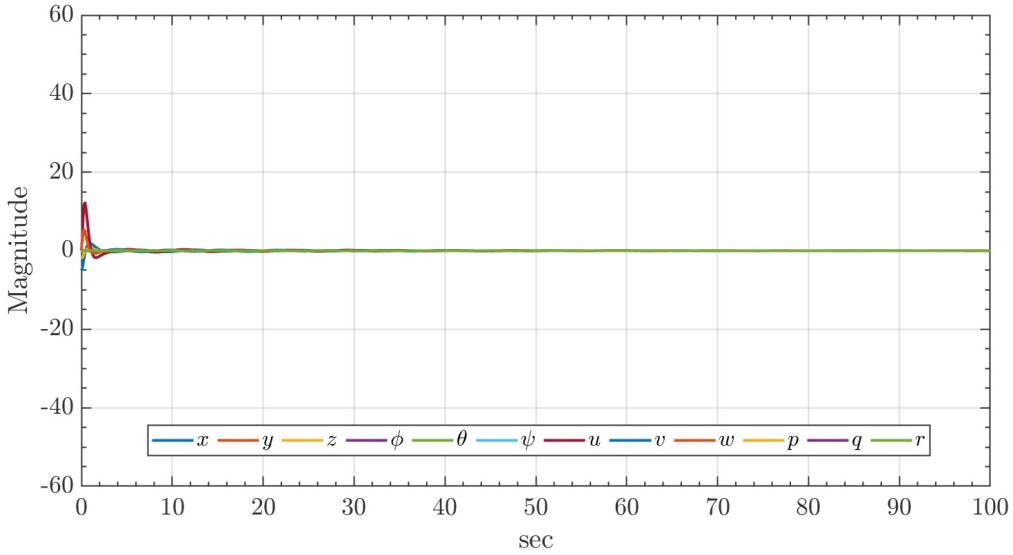
$$\hat{z} = A_z\,\hat{z} + B_z\,u_z + L\,(y_z - \hat{y}_z) \tag{A.4}$$

where $u_z = u$, L is a suitable gain matrix such that the closed-loop dynamic matrix $A_{\tilde{z}} = A_z - LC_z$ is Hurwitz, ensuring the asymptotic boundedness of the augmented state estimation error $\tilde{z} = \hat{z} - z$ if and only if the couple (A_z, C_z) is fully observable and the additional state variable has slow dynamics, i.e., $\dot{\delta} \approx 0$. Finally, the proposed ESO in

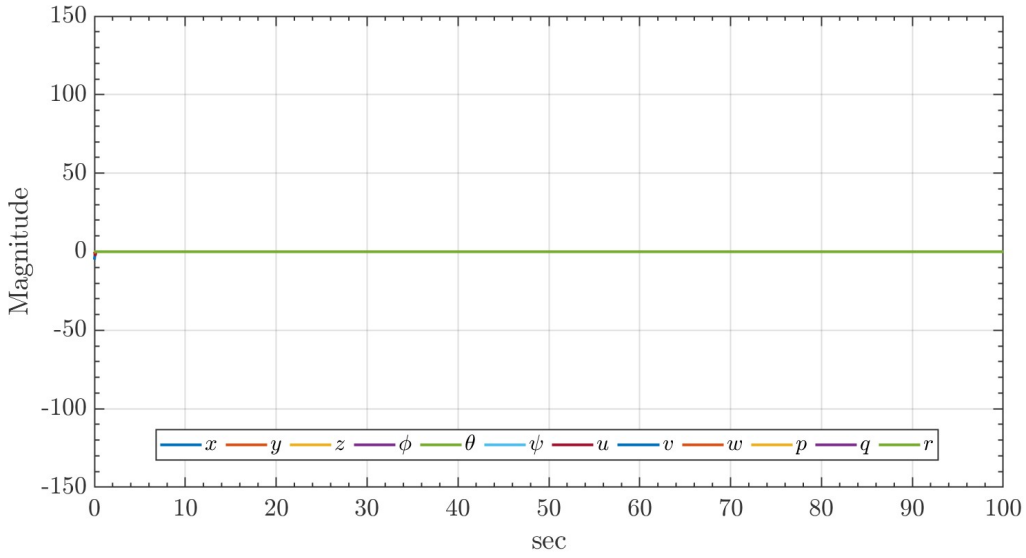
A.4 is defined by $u_z = u$ and

$$A_z = \begin{bmatrix} 0_{6\times6} & I_{6\times6} & 0_{6\times6} \\ 0_{6\times6} & 0_{6\times6} & I_{6\times6} \\ 0_{6\times6} & 0_{6\times6} & 0_{6\times6} \end{bmatrix}, \qquad B_z = \begin{bmatrix} B \\ 0_{6\times3} \end{bmatrix}, \qquad C_z = \begin{bmatrix} I_{6\times6} & 0_{6\times6} & 0_{6\times6} \end{bmatrix}$$

In the comparison of the UIO and ESO, two simulations are presented: the first does not consider the sensor noise affecting the sensor's measurements, while in the second simulation a random bias is introduced impacting the measured states. In the first case, as shown in A.1a and A.1b, both observers exhibit comparable performance, effectively tracking the states of the MHex, with the UIO showing slight overshooting during the initial phase.



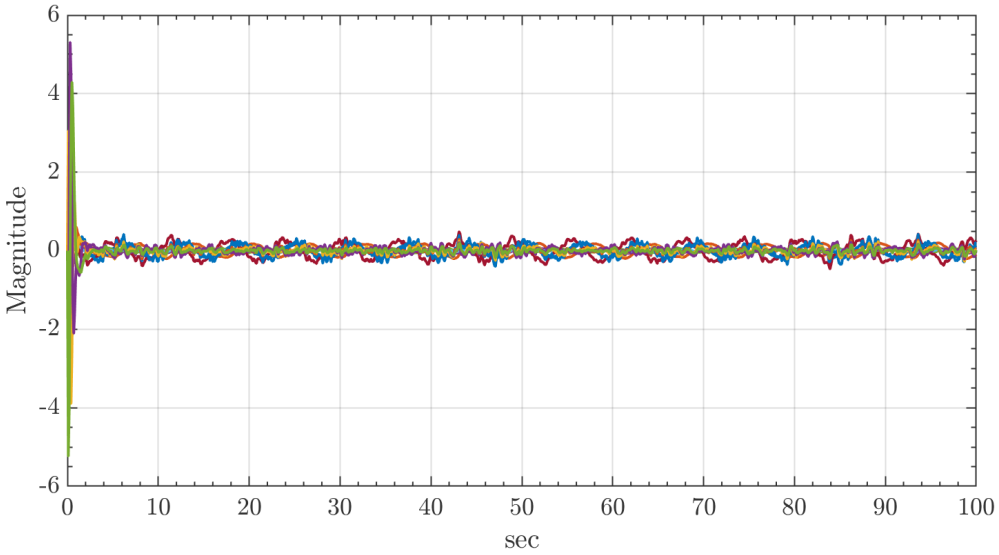
(a) UIO state estimation error



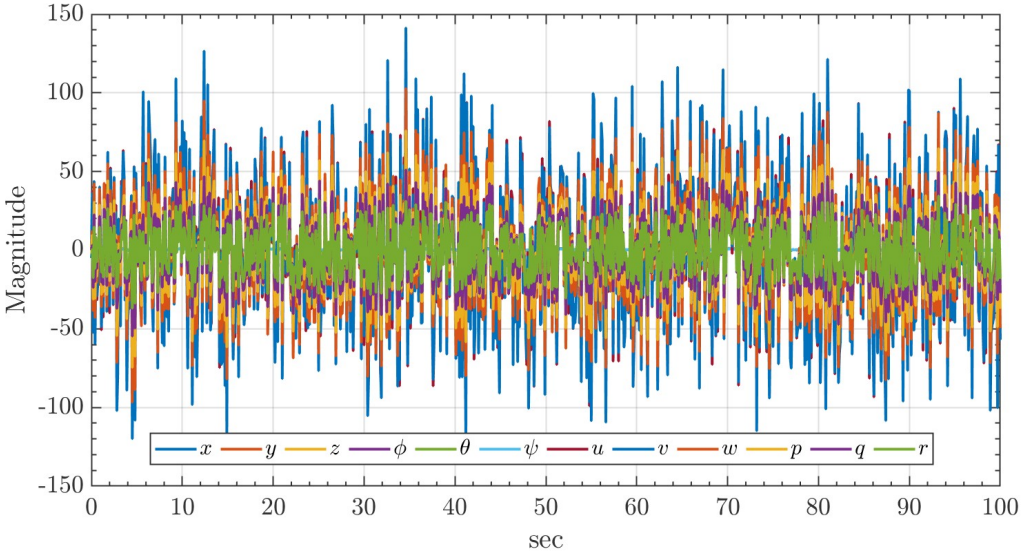
(b) ESO state estimation error

Figure A.1: Observers state estimation

In contrast, when it comes to the sensor bias in the second simulation, the gained results point out the accuracy and superiority of the UIO in mitigating noise disturbances affecting the system even when the bias undergoes sudden changes, as shown in A.2a and A.2b.



(a) UIO state estimation error with bias



(b) ESO state estimation error with bias

Figure A.2: Observers state estimation with bias

A.3-A.4b give the tracking of the MHex states based on both observer measurements, confirming the accuracy of the UIO despite the sensor bias changing randomly. For completeness, the Root Mean Square Error (*RMSE)* for both observers is given in A.1.

Table A.1: Observer estimation wwith sensor bias, *RMSE*

RMSE	*Position*	*Orientation*	*Linear vel*	*Ang vel*
UIO	0.930	0.866	1.038	0.278
ESO	1.168	1.047	79.832	39.053

According to the data, while the ESO error in the pose q closely resembles that of the UIO, the estimated velocities $\dot{q}$ for the ESO are significantly less accurate, differing by several orders of magnitude compared to the UIO estimation.

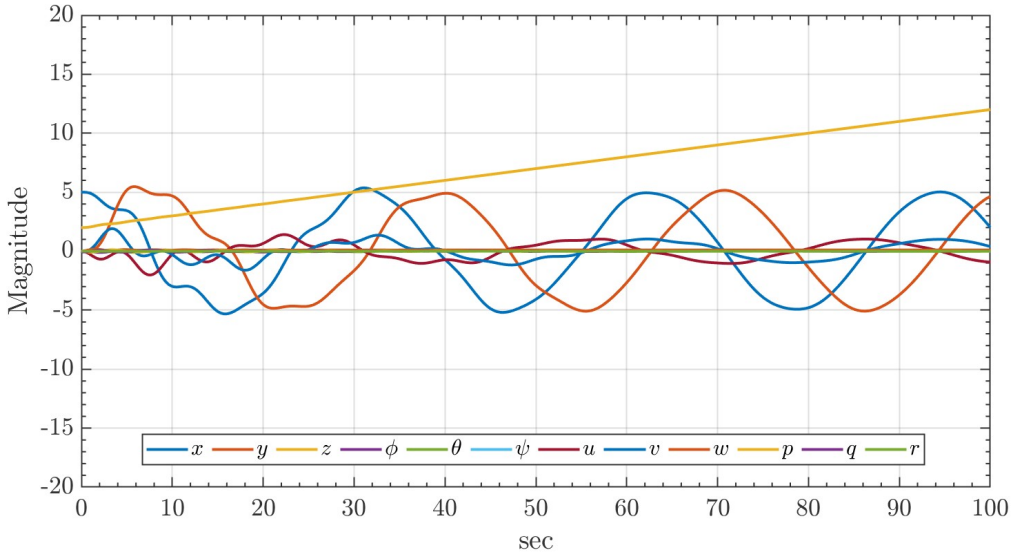
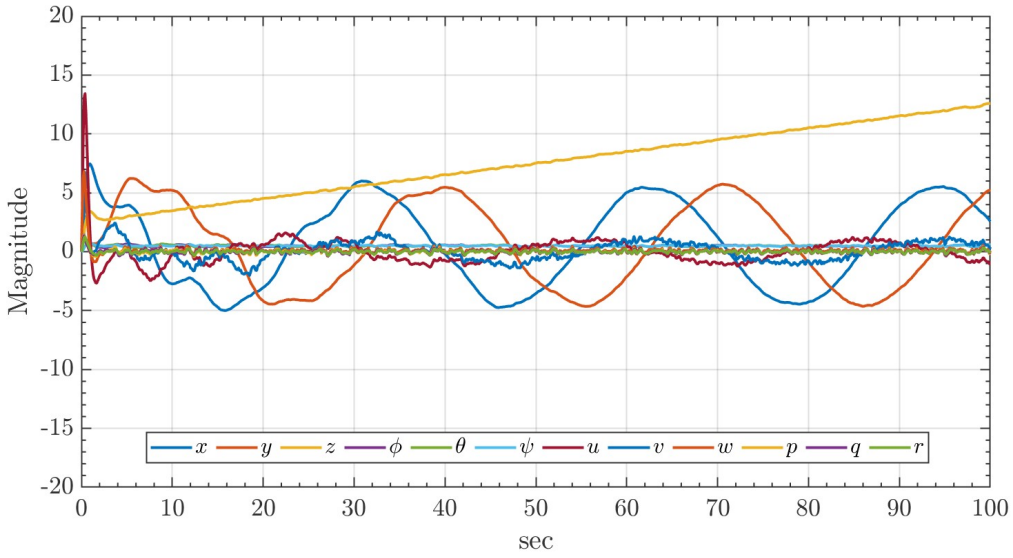
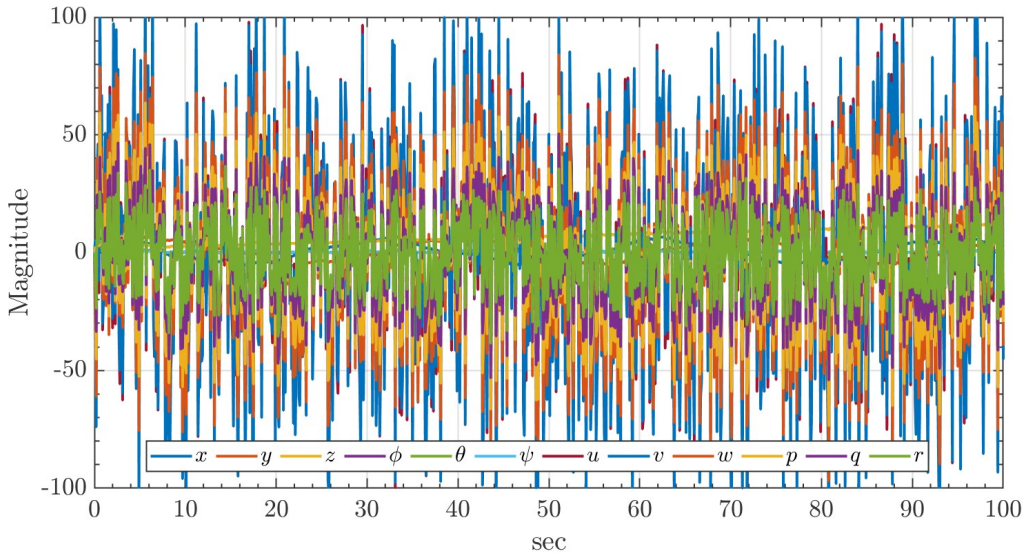


Figure A.3: Hexacopter states



(a) UIO states tracking



(b) ESO states tracking

Figure A.4: UIO and ESO state tracking, considering external wind and sensor bias

In the second simulation, despite the uncertainty affecting the system, the desired trajectory is accurately tracked by the UIO, proving the precise estimation of the disturbances, while the ESO exhibits poor performance especially when the disturbance phenomena change rapidly.

A.1.1 Conclusions

This section showed the comparison between the UIO and ESO observers. A terrestrial hexacopter with the same MHex symmetrical structure is considered for the evaluation of the state estimation in the autonomous navigation scenario. The hexacopter dynamic is first derived, and then the mathematical model is implemented in Matlab/Simulink together with the PID controller. The results showed how the UIO

tracks the trajectory by estimating the system states even in the presence of both aerodynamic disturbances and system uncertainties, as well as sensor noise. Conversely, the performance of the ESO is directly affected by bias random variations, while in the UIO the lumped disturbances are observed and mitigated without prior knowledge of either the boundary of the disturbance or its derivative, proving that the UIO can rapidly and accurately respond to sudden changes in noise.