DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
UNIVERSIDAD DE SEVILLA

Contributions to deconfliction advanced U-space services for multiple unmanned aerial systems including field tests validation

por

Héctor Pérez León

Ingeniero en Electrónica Industrial, Robótica y Automática

PROPUESTA DE TESIS DOCTORAL
PARA LA OBTENCIÓN DEL TÍTULO DE

DOCTOR POR LA UNIVERSIDAD DE SEVILLA
SEVILLA, 2022

Directores
Dr.-Ing. Jesús Iván Maza Alcañíz, Profesor Titular
Dr.-Ing. Aníbal Ollero Baturone, Catedrático

UNIVERSIDAD DE SEVILLA

Memoria para optar al grado de Doctor por la Universidad de Sevilla

Autor:	**Héctor Pérez León**
Título:	**Contributions to deconfliction advanced U-space services for multiple unmanned aerial systems including field tests validation**
Departamento:	**Departamento de Ingeniería de Sistemas y Automática**

V° B° Director:

Jesús Iván Maza Alcañíz

V° B° Director:

Aníbal Ollero Baturone

El autor:

Héctor Pérez León

Life before death.
Strength before weakness.
Journey before destination.

Acknowledgements

First and foremost, I want to express my gratitude to my supervisors, Professor Iván Maza and Professor Aníbal Ollero, for all of their guidance and support all along this Thesis. In particular, I would like to express my deep gratitude and admiration to Professor Iván Maza for the personal teaching and time dedicated to setting my work on the right path to achieve the objectives.

Without the amazing team of people at the Robotics, Vision, and Control Laboratory, research activities like the ones carried out for this Thesis would not have been feasible. I would want to express my gratitude to my coworkers for their generosity, support, and encouragement. Many wonderful moments have resulted from the mix of superb professional individuals in a friendly environment. Alejandro Braza, José Andrés Millán, Alejandro Castillejo, Rafael Salmoral, Pedro Sánchez, Arturo Torres, and Alfonso Alcántara, thank you for the kindness, and the amusing times. José Joaquín Acevedo and Fran Real, in particular, deserve special thanks for spending many hours programming, and developing with me, I learned a lot from you.

A hearty thanks to my life partner, Alejandra, who was always there for me. Unintentionally you have been a part of this journey, encouraging me during the hardest times and celebrating with me every small victory.

Finally, I would want to express my gratitude to my family. Thanks to my sister, for giving me her point of view on tricky situations and for pushing me out of my comfort zone. Thanks to my parents for their constant love and support. They have given me the tools I have needed to face any problem throughout my life.

Héctor Pérez León

Abstract

Unmanned Aerial Systems (UAS) will become commonplace, the number of UAS flying in European airspace is expected to increase from a few thousand to hundreds of thousands by 2050. To prepare for this approaching, national and international organizations involved in aerial traffic management are now developing new laws and restructuring the airspace to incorporate UAS into civil airspace. The Single European Sky ATM Research considers the development of the U-space, a crucial step to enable the safe, secure, and efficient access of a large set of UAS into airspace.

The design, integration, and validation of a set of modules that contribute to our UTM architecture for advanced U-space services are described in this Thesis. With an emphasis on conflict detection and resolution features, the architecture is flexible, modular, and scalable. The UTM is designed to work without the need for human involvement, to achieve U-space required scalability due to the large number of expected operations. However, it recommends actions to the UAS operator since, under current regulations, the operator is accountable for carrying out the recommendations of the UTM. Moreover, our development is based on the Robot Operating System (ROS) and is open source.

The main developments of the proposed Thesis are monitoring and tactical deconfliction services, which are in charge of identifying and resolving possible conflicts that arise in the shared airspace of several UAS. By limiting the conflict search to a local search surrounding each waypoint, the proposed conflict detection method aims to improve conflict detection. By splitting the issue down into smaller subproblems with only two waypoints, the conflict resolution method tries to decrease the deviation distance from the initial flight plan.

The proposed method for resolving potential threats is based on the premise that UAS can follow trajectories in time and space properly. Therefore, another contribution of the presented Thesis is an UAS 4D trajectory follower that can correct space and temporal deviations while following a given trajectory. Currently, commercial autopilots do not offer this functionality that allows to improve the airspace occupancy using time as an additional dimension.

Moreover, the integration of onboard detect and avoid capabilities, as well as the consequences for U-space services are examined in this Thesis. A module capable of detecting large static unexpected obstacles and generating an alternative route to avoid the obstacle online is presented.

Finally, the presented UTM architecture has been tested in both software-in-the-loop and hardware-in-the-loop development enviroments, but also in real scenarios using unmanned aircraft. These scenarios were designed by selecting the most relevant UAS operation applications, such as the inspection of wind turbines, power lines and precision agriculture, as well as event and forest monitoring. ATLAS and El Arenosillo were the locations of the tests carried out thanks to the European projects SAFEDRONE and GAUSS.

Acronyms

UAS	Unmanned Aerial System
UTM	Unmanned Aerial System Traffic Management
SESAR	Single European Sky ATM Research
NASA	National Aeronautics and Space Administration
4D-TBO	4D-Trajectory-Based-Operations
VLL	Very Low Level
ATM	Air Traffic Management
ATC	Air Traffic Control
ConOps	Concept of Operations
GNSS	Global Navigation Satellite System
DAA	Detect And Avoid
VLOS	Visual Line Of Sight
BVLOS	Beyond Visual Line Of Sight
OV	Operational Volume
USSP	U-space Service Provider
NRI	Network Remote ID

DRI Direct Remote ID

ADS-B Automatic Dependent Surveillance-Broadcast

ROS Robot Operating System

USM U-space Service Manager

ICAO International Civil Aviation Organization

GIS Geographic Information System

SITL Software In The Loop

ROCK Robot Construction Kit

ORoCoS Open Robots Control Software

GenoM Generator of Modules

UAL UAS Abstraction Layer

HITL Hardware In The Loop

RPS Remote Pilot Station

GCS Ground Control Station

MQTT Message Queuing Telemetry Transport

M600 Matrice 600 Pro

ATLAS Air Traffic Laboratory for Advanced unmanned System

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This chapter introduces the motivation and main objectives of this Thesis, while describing its outline. It also provides some information on the framework of the research, the U-space context. Finally, it presents the scientific output.

1.1 Motivation

Unmmaned Aerial Systems (UAS) have mostly worked as intelligence collecting devices to assist military activities over the previous two decades. However, in recent years, UAS have generated a lot of interest in civil, industrial, and commercial uses and are becoming more valuable as tools and helpful assistance in a wide range of human activities. These tasks are currently used in a variety of contexts including surveillance Basilico and Carpin (2015); Acevedo et al. (2014); Balampanis et al. (2017), wildfire tracking Merino et al. (2012); Pham et al. (2018), bridge inspection Sanchez-Cuevas et al. (2019); Yoder and Scherer (2016), smart farming Barrientos et al. (2011); Lottes et al. (2017), and transportation Kondak et al. (2015); Bernard et al. (2011). These systems have grown more appealing and practicable as a result of downsizing and cost reductions, particularly for jobs that may represent a risk to humans or are, in essence, logistical problems that are continually being carried out manually. Recent technical advancements have made UAS more accessible to a wider audience, with UAS now having access to off-the-shelf long-range wireless communications, high-resolution light sensors, on-board computational capacity, and power-efficient hardware.

UAS are being used in different spheres of business and are projected to become a commercial delivery method in the near future, while convergent technology has been expanding recently. With an ambitious approach, in December 2016, the first PrimeAir delivery service was launched Amazon (2016). DHL Express and EHang have formed a strategic relationship to jointly create a fully automated and intelligent smart UAS delivery system in China's urban areas DHL-Ehang (2019). Google Project Wing has been awarded permission to transform its public-access UAS delivery experiments in Australia into a full-fledged business. Just over 100.000 drone deliveries were made in Australia by Wing in 2021, and they reached 30.000 in the first two months of 2022 Alphabet (2022).

According to the 2016 European Drones Outlook Study SESAR (2016), the growing UAS business will support potential economic growth. In most economic areas, unmanned aircraft will become a part of daily life. Indeed, by 2050, the number of UAS flying in European airspace is estimated to rise from a few thousand to hundreds of thousands, mostly for government and commercial purposes. By 2035, the yearly economic impact in Europe might surpass EUR 10.000 million, with 100.000 new direct jobs created to support UAS operations. The agriculture industry, for example, is expected to have 150.000 UAS operating by 2035. In the domains of utilities and security, roughly 60.000 unmanned aircraft will be employed to aid in natural disaster management and traffic control, among other activities.

To address this nearing scenario, national and international organizations related to aerial traffic management are currently elaborating new regulations and reorganizing the airspace to integrate the UAS in the civil airspace. U-space is the UAS Traffic Management (UTM) solution for Europe, where the Single European Sky ATM Research (SESAR) considers the development of the U-space, a key step to enable the safe, secure and efficient access of a large set of UAS to the civil airspace SESAR (2017), through a set of services related to flight planning, tracking, conflict management, etc. A similar concept is being developed by the National Aeronautics and Space Administration (NASA) in the United States of America Kopardekar (2015).

According to Unmanned Airspace's December 2021 prediction for the worldwide UTM market in 2021-2025 Butterworth-Hayes and Mahon (2021), the new sector value is USD 1.380 million, which includes awards from commercial, strategic national UTM development programs, and tactical UTM service operational charges.

SESAR also created the 4D-Trajectory Based Operations (4D-TBO) concept, which was originally aimed for manned commercial aircraft but is now being examined in the U-space context. The addition of time as a fourth dimension in the flight plan specification improves airspace occupancy, which is mostly connected to conflict resolution U-space services CORUS (2019). As a result, establishing accurate and precise 4D trajectory control methods becomes a critical problem for the development of the 4D-TBO in the U-space, which will allow the integration of an increasing number of UAS into civil airspace.

1.2 Objectives

The main objectives of this Thesis are:

- Build a modular and scalable implementation of several modules based on advanced U-space services.
- Develop a method that can detect different types of conflicts between UAS trajectories and geofences.
- Develop an algorithm that can solve different types of conflicts between UAS trajectories and geofences proposing alternatives routes to the operator.
- Implement an onboard module that can detect large static obstacles and avoid them by using a path planner to generate alternative routes.
- Implement a trajectory follower that improves the performance of the position controller of an autopilot when following a list of waypoints.

1.3 Contributions

The Thesis meets the objectives previously outlined, producing the following main contributions:

1. An implementation of two advanced U-space services, tactical deconfliction and monitoring, which are part of an architecture framed in the U-space ecosystem. The architecture is flexible and general, and it is focused on advanced services for automated conflict detection and resolution. The system is capable of detecting and resolving different types of conflicts based on 4D UAS trajectories.

2. A conflict detection method for multiple 4D UAS trajectories sharing the same space and a conflict resolution method. The modules that contain these methods are also able to detect and solve conflicts between UAS and geofences.

3. A 4D trajectory follower which improves the performance of the position controller of an autopilot when following a list of waypoints. It can correct an infeasible trajectory given by the user before doing it, which increases the performance of the follower.

4. An autonomous detect and avoid onboard system for UAS. This system uses a LIDAR to get data from the real world, uses octomap to transform the point cloud from the sensor into a 3D map and uses the Lazy Theta* algorithm to avoid unexpected large static obstacles.

Chapter 1 presents the motivation, objectives and framework in which this Thesis has been developed. Moreover, it presents the contributions and scientific output of this Thesis.

Chapter 2 describes the existing work related to this Thesis. More precisely, it presents the U-space framework and its set of new services that rely on a high level of digitalization and automation of functions and specific procedures designed to support safe, efficient, and secure access to airspace for a large number of UAS. As such, U-space is an enabling framework designed to facilitate any kind of routine mission, in all classes of airspace and all types of environments, even the most congested, while addressing an appropriate interface with manned aviation and air traffic control.

Chapter 3 presents the developed services, Contribution 1 of this Thesis. It details the modules that contribute to the architecture, which services are related to each module, and how they are implemented in software. This chapter contains a detailed explanation of how the modules are connected and how are the messages used, as well as which robot frameworks have been chosen for this Thesis and a comparison with other frameworks. Moreover, it details the onboard autonomous detect and avoid module presented in this Thesis, Contribution 4.

Chapter 4 is focused on two modules: Monitoring and Tactical Deconfliction. These modules are in charge of conflict detection and conflict resolution, respectively, Contribution 2 of this Thesis. The monitoring module detecting conflicts between 4D trajectories and geofences should warn of all conflicts in the shortest possible time. The tactical deconfliction module solves threats one by one depending on the threat type. Each threat should be solved with more than one alternative flight plan and each alternative solution must have been made with an independent strategy. The methods used for each module are described in this chapter.

Chapter 5 describes the work related to the Contribution 3. A 4D trajectory follower is needed to perform flight plans in the U-space context. This chapter details the follower based on the carrot chasing algorithm and presents the problems that can appear if, in this context, a 4D trajectory follower is not used.

Chapter 6 presents the experiments carried out on the field, the results of which are presented in this chapter. Moreover, it presents a comparison between a brute force based method, detailed in appendix A, and the discretized based method chosen for this Thesis, which can be scaled without problems. The brute force based method was developed in the first place to be able to make tests quickly in the field, which is not scalable, however, it is fast and does not need to set up.

1.4 Thesis framework

This Thesis has been developed within the context of a number of R&D projects funded by the European Commission in H2020. In all of them, Very Low Level (VLL) airspace scenarios were an important part, as well as U-space operations. The PhD candidate designed and implemented the developments presented in this Thesis within the Robotics, Vision and Control group at the University of Seville.

SAFEDRONE (H2020-783211). The scope of the SAFEDRONE project was to acquire practical experience in VLL operations where general aviation, state aviation, and optionally piloted aircraft and UAS will share the airspace. It is important to highlight that this project had a clear practical focus, which primary activities were innovation, integration, and especially, demonstrating activities with flight tests.

GAUSS (H2020-ICT-776293). The main objective of the GAUSS project was the achievement of an acceptable level in terms of performance, safety, and security for both current UAS and future U-space operations. The research was focused on the consecution of precise and secure positioning to enable U-space operations, supporting the management and coordination of all UAS in the VLL airspace.

1.5 Scientific output

The following peer-reviewed publications have been published based on the work presented in this Thesis:

- Perez-Leon, H., Acevedo, J. J., Maza, I., and Ollero, A. (2021a). Integration of a 4D-trajectory Follower to Improve Multi-UAV Conflict Management Within the U-Space Context. *Journal of Intelligent and Robotic Systems*, 102(3):62

- Perez-Leon, H., Acevedo, J. J., Maza, I., and Ollero, A. (2020a). A 4D trajectory follower based on the 'Carrot chasing' algorithm for UAS within the U-space context. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, page 1860–1867. IEEE

- Perez-Leon, H., Acevedo, J. J., Millan-Romera, J. A., Castillejo-Calle, A., Maza, I., and Ollero, A. (2020b). An Aerial Robot Path Follower Based on the ‘Carrot Chasing’ Algorithm. In Silva, M. F., Luís Lima, J., Reis, L. P., Sanfeliu, A., and Tardioli, D., editors, *Robot 2019: Fourth Iberian Robotics Conference*, volume 1093, page 37–47. Springer International Publishing

- Capitan, C., Perez-Leon, H., Capitan, J., Castaño, A., and Ollero, A. (2021). Unmanned Aerial Traffic Management System Architecture for U-Space In-Flight Services. *Applied Sciences*, 11(9):3995

- Faria, M., Ferreira, A. S., Perez-Leon, H., Maza, I., and Viguria, A. (2019). Autonomous 3D Exploration of Large Structures Using an UAV Equipped with a 2D LIDAR. *Sensors*, 19(22):4849

- Millan-Romera, J. A., Perez-Leon, H., Castillejo-Calle, A., Maza, I., and Ollero, A. (2019b). ROS-MAGNA, a ROS-based framework for the definition and management of multi-UAS cooperative missions. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, page 1477–1486. IEEE

- Millan-Romera, J. A., Acevedo, J. J., Castano, A. R., Perez-Leon, H., Capitan, C., and Ollero, A. (2019a). A UTM simulator based on ROS and Gazebo. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, page 132–141. IEEE

- Castillejo-Calle, A., Millan-Romera, J. A., Perez-Leon, H., Andrade-Pineda, J. L., Maza, I., and Ollero, A. (2019). A multi-UAS system for the inspection of photovoltaic plants based on the ROS-MAGNA framework. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 266–270. IEEE

The following open source software based on the work presented in this Thesis have been released and field tested:

- Perez-Leon, H. (2020). UPAT Follower: UAV Path and Trajectory Follower. `https://github.com/hecperleo/upat_follower` (accesed on 17 February 2022)

- Perez-Leon, H., Braza, A., Jose Joaquin, A., Capitan, C., and Real, F. (2021b). GAUSS UTM system architecture. `https://github.com/grvcTeam/gauss` (accesed on 17 February 2022)

Chapter 2

Related work

One of the most important concepts and technologies currently being developed in the field of drones is the future system of Air Traffic Management (ATM) in the very low level altitude, which will be controlled by an unmanned aerial system traffic management. It will allow effective and organized management of the large volume of civilian unmanned aircraft that will be able to use the airspace in the medium-long term. Despite ATM has traditionally depended on voice communication through an Air Traffic Control (ATC) entity, the restricted workload and communication capacity of this centralized resource makes it a bottleneck for system expansion. As a result, the growth of UAS operations needs a new airspace management paradigm, in which digital communication plays a key role and responsibilities are shared among several stakeholders rather than a single central actor.

2.1 U-space

This section mainly gathers the information contained in the following official documents SESAR (2017); CORUS (2019, 2020). To foster the expansion of the UAS sector and the usage of these aircraft in Europe, the European Union has devised a concept termed U-space: the progressive implementation of procedures and a set of services meant to make it easier for large groups of drones to access the airspace in a safe, efficient, and secure manner. These services and operations, whether they are on board the drone or part of the ground-based environment, rely on a high level of digitisation and automation of tasks. NASA has also developed its own UTM model in the United States of America Kopardekar (2015), but this Thesis is focused on the U-space concept.

U-space provides a simple and effective interface to manned aircraft, ATM service providers, and authorities, as well as an enabling architecture to facilitate normal drone operations. As a result, U-space should not be regarded as a specified volume of airspace reserved only for the operation of drones.

U-space can ensure that drones operate smoothly in all sorts of operating settings and airspace, including but not limited to VLL airspace. It addresses the requirements for supporting a variety of tasks and may affect all drone users and types of drones. U-space, according to CORUS[1], is an environment that allows for commercial activities including drone use while maintaining an acceptable level of safety and public acceptance. CORUS came up with this Concept of Operations (ConOps) by thinking about the most common U-space use cases first.

The following are the key principles that govern U-space delivery: to protect the safety of all airspace users in the U-space framework, as well as ground personnel; to provide a scalable, flexible and adaptable system that can respond to changes in demand, volume, technology, business models and applications, while managing the interface with manned aviation; to allow for high-density operations with several autonomous drones under the guidance of fleet operators; to ensure that all users have equal and fair access to the airspace; to assist business models of drone operators by enabling competitive and cost-effective service offering at all times; to reduce deployment and operational costs as much as possible by using current aviation services and infrastructure, such as Global Navigation Satellite System (GNSS), as well as services from other industries, such as mobile communication services; to accelerate implementation by incorporating technology and standards from other industries when they fulfill U-space requirements; while establishing adequate standards for safety, security, and resilience, to use a risk-based and performance-driven approach, while minimizing environmental effect and preserving privacy of the people.

[1]`https://www.sesarju.eu/projects/corus`

2.1.1 Implementation

The U-space services will be gradually introduced over four phases, from U1 to U4, based on the increasing availability of blocks of services and enabling technologies, the increasing level of drone automation, and advanced forms of environmental interaction, primarily enabled through data exchange and digital information, see Figure 2.1. All services offered in a particular phase may use other services provided in that phase, as well as those provided in earlier stages.

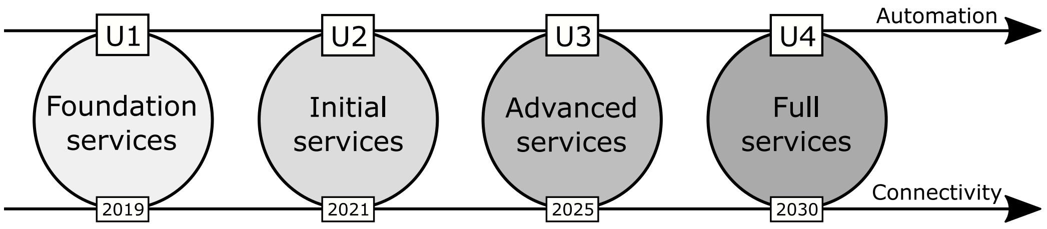


Figure 2.1: U-space levels CORUS (2019). Each level increases the level of drone automation and connectivity. Advanced services are planned to be developed in 2025, and full services after 2030.

The U-space foundation, U1, offers e-registration, e-identification, and geofencing services. Moreover, the initial U-space services corresponding to the U2 stage, assist drone operations management and may include flight planning, flight permission, tracking, airspace dynamic information, and procedural interfaces with ATC. Advanced U-space services, U3, support more complicated activities in congested locations, such as capacity management and conflict detection aid. Indeed, the availability of automatic Detect And Avoid (DAA) functions, as well as more reliable communication methods, will result in a considerable rise in operations in all situations. Finally, U-space full services, U4, especially those with integrated interfaces with human aircraft, provide U-space full operating potential and rely on a high degree of automation, connection, and digitalization for both the drone and the U-space.

2.1.2 Very low level airspace

The VLL airspace is below the usual airspace utilized by aircraft operating under visual flight rules. Although this airspace is normally unoccupied, it is utilized for emergencies, take-offs and landings, certain high-precision airborne operations, and training on occasion. Small drones, such as those used for inspection, photography, or delivery, will make extensive use of it in the near future. VLL airspace is predicted to grow more congested as UAS operations become more popular, demanding more standard structures and procedures to reduce collision risk. These risk reduction methods are expected to be provided by U-space services. All aircraft in that area will be required to participate, similar to how existing flight regulations apply to all manned aircraft flying in that airspace today.

Some operations are subject to a height limit under European drone rules. "During flight, the unmanned aircraft is maintained within 120 meters from the closest point of the earth's surface, except when overflying an obstacle," according to Commission Implementing Regulation (EU) 2019/947[2]. For open operations, "the unmanned aircraft is maintained within 120 meters from the closest point of the earth's surface, except when overflying an obstacle". Vehicles for C0, C1, C2, and C3 are limited to 120 meters above the take-off point, according to Commission Delegated Regulation (EU) 2019/945[3]. However, the European Union Aviation Safety Agency (EASA), in the ED Decision 2022/002/R[4], updated the threshold to 150 meters for the operational volume under certain conditions for the specific category.

The majority of drones now fly in Visual Line Of Sight (VLOS) operations, which means that the remote pilot can see the aircraft. Beyond Visual Line Of Sight (BVLOS) operations, in which the pilot cannot see the drone directly, are uncommon, if not outright prohibited. However, for many future commercial drone operations, such as deliveries, BVLOS flights are projected to be the standard.

U-space divides VLL airspace into three sections based on the services it provides CORUS (2020), see Figure 2.2. These volumes differ in two ways: the services provided and, as a result, the sorts of operations that may be performed, as well as the access and entry criteria. The three forms of airspace volume are referred to as X, Y, and Z. There is no conflict resolution service available in X volumes. On the other

[2]`https://eur-lex.europa.eu/eli/reg_impl/2019/947`
[3]`https://eur-lex.europa.eu/eli/reg_del/2019/945`
[4]`https://www.easa.europa.eu/downloads/135912/en`

hand, preflight conflict management is the only option in Y volumes. Moreover, there are options for preflight conflict management and in-flight separation in Z volumes.

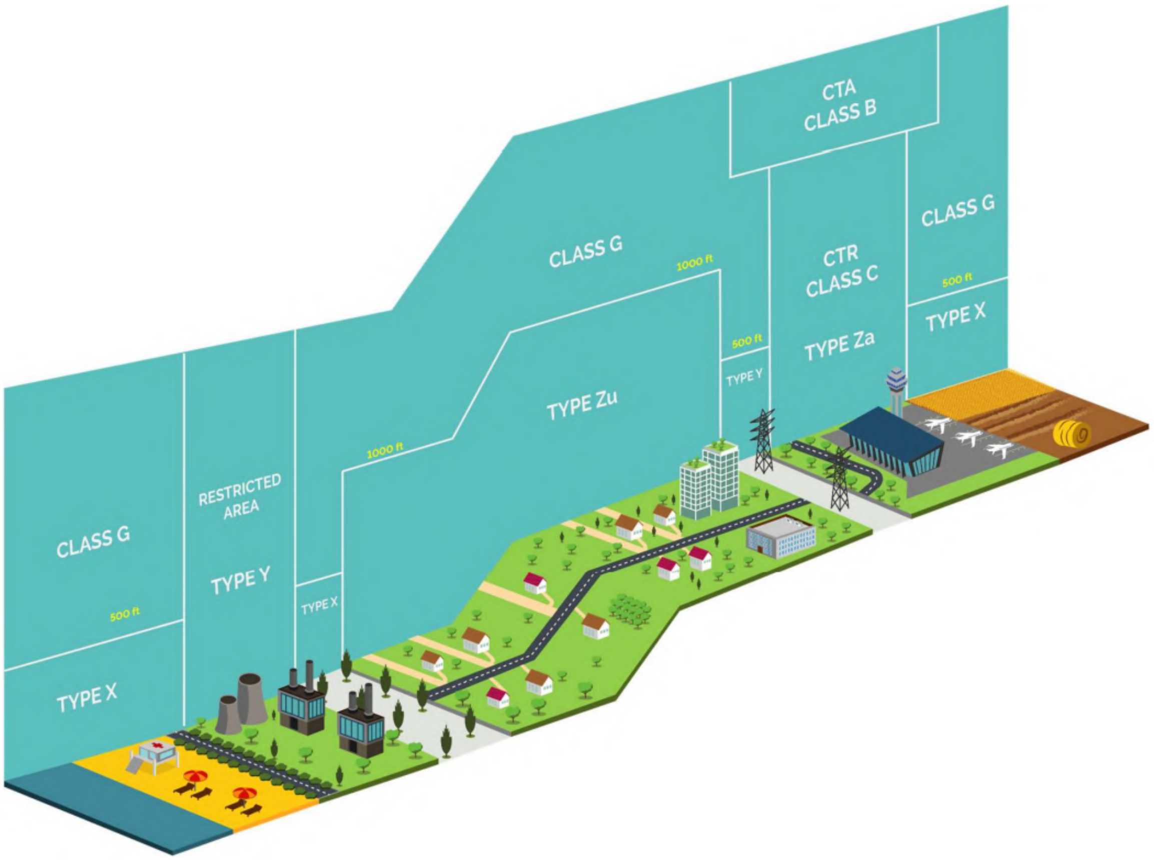


Figure 2.2: Graphical representation of the VLL airspace sections divided by U-space.

From U2, Y airspace will be accessible, allowing for VLOS and BVLOS flying. Because of the risk mitigation afforded by U-space, Y airspace is more conducive to alternative flight modes than X airspace. Type Z airspace can be separated into Zu and Za zones, with UTM and ATM in charge of each. Za is just a standard regulated airspace, and it is therefore instantly accessible. From U0 forward, Zu airspace will be available. U-space is more receptive to different flight modes and permits greater density operations than Y airspace because it provides more risk mitigation for Z volumes. Z supports VLOS, BVLOS, and automated drone flight.

As well as the services offered, these types of airspace differ by their requirements for access, see Table 2.1. Y airspaces may also have specific technical requirements attached to the drone. Moreover, in Z airspaces the drone may be fitted with collaborative detect and avoid system for collision avoidance.

Table 2.1: Necessary requirements to access the different types of airspaces.

Type	**Access requirements**
X	The operator, the pilot, and the drone must meet a few prerequisites. The pilot is still in charge of avoiding collisions. It is straightforward to fly in VLOS and EVLOS. Other types of flying in X necessitate major risk minimization.
Y	An operation plan that has been authorized. A pilot who has been trained for the Y operation. A U-space-connected remote piloting station. A drone with a remote piloting station that can report its location.
Z	An operation plan that has been authorized. A pilot who has been trained for the Z operation. A U-space-connected remote piloting station. A drone with a remote piloting station that can report its location.

Because there are minimal fundamental prerequisites for the operator, pilot, or drone to access X airspace, few services are available. The pilot is in charge of separation at all times in X volume. VLOS flights are simple to achieve, however, other sorts of operations in X necessitate a high level of air risk reduction.

An authorized operating plan is required to gain access to Y. Specific technical criteria for Y volumes may exist, and showing that these are satisfied is part of the operating plan approval process. A remote piloting station connected to U-space and an UAS capable of submitting location reports are frequently included in these technological criteria. VLOS and BVLOS flights are made easier using Y volumes. U-space provides risk mitigation in Y volumes that are not accessible in X.

Z volumes allow for larger densities of operations than Y volumes, and are thus expected in places where traffic demand exceeds Y capacity or where there is a special danger. Access to Z, like Y, needs a preapproved operating plan. In addition, the pilot must be linked to U-space at all times, and a position report for the aircraft with sufficient performance to permit tracking is required.

2.1.3 Services

As technology and operational capabilities allow, U-space services will be offered in four phases as it has been previously mentioned. These services address many areas of need for drone integration with ATC and other airspace users. Although our architecture is generic enough to accommodate all types of services, we focus on advance functionalities in this Thesis to construct an UTM. We integrate services such as tracking, monitoring, emergency management, and tactical deconfliction that are linked to the handling of unforeseen situations while the UAS is flying. These services are part of the U2 and U3 phases of deployment.

The following is a list of services differentiated by deployment U-space phases. First, the U-space foundation services, U1:

- *Registration.* Interaction with the register to enable the registration of the drone, its owner, its operator, and its pilot. Different classes of users may query data or maintain or cancel their own data, according to defined permissions.
- *Registration assistance.* Provides assistance to people undertaking the registration process.
- *e-Identification.* Enables information about the drone and other relevant information to be verified without physical access to the unmanned aircraft.
- *Geo-awarenes.* This provides geofence and other flight restriction information to operators for their consultation up to the moment of take-off. To produce an overall picture of where drones may operate, it includes existing aeronautical information, such as: restricted areas and danger areas, Controlled Traffic Regions; Notice to Airmen; temporary restrictions from the national airspace authority.
- *Drone Aeronautical Information Management.* The drone equivalent of the Aeronautical Information Management service. This service maintains the map of X, Y, and Z airspaces, and permanent and temporary changes to it. (e.g., a weekend festival will change an area from sparsely to densely populated). This

service provides information to the geofencing services as well as the operational planning preparation service.

Moreover, the initial services, U2:

- *Tracking and position reporting.* Receive reports about the location, fuses multiple sources and provides tracking information about drone movements.
- *Surveillance data exchange.* It exchanges data between the tracking service and other sources or consumers of track - radar, other drone trackers, etc.
- *Geofence provision.* An enhancement of geo-awareness that allows geofence changes to be sent to drones immediately. The drone must have the ability to request, receive, and use geofencing data.
- *Operation plan preparation and optimisation.* Provides assistance to the operator in the filing of an operation plan. This service functions as the interface between the drone operator and the operation plan processing service.
- *Operation plan processing.* A safety-critical, access-controlled service that manages live operation plans submitted via the operation plan preparation service and checks them with other services. The service manages authorisation workflows with relevant authorities and dynamically takes airspace changes into account.
- *Risk analysis assistance.* Provides a risk analysis, mainly for specific operations, combining information from other services - drones, environment, traffic information, etc. This can also be used by insurance services.
- *Strategic Conflict Resolution.* Checks for possible conflicts in a specific operation plan and proposes solutions during operational plan processing.
- *Emergency Management.* Provides assistance to a drone pilot experiencing an emergency with their drone, and communicates emerging information to interested parties.

- *Accident / Incident Reporting.* A secure and access-restricted system that allows drone operators and others to report incidents and accidents, maintaining reports for their entire life cycle. A similar citizen access service is possible.

- *Citizen Reporting* A secure and access-restricted system that allows citizens to report what they have observed when they believe incidents or accidents involving drones have occurred.

- *Monitoring.* Provides monitoring alerts about the progress of a flight (e.g., conformance monitoring, weather compliance monitoring, ground risk compliance monitoring, and electromagnetic monitoring).

- *Traffic Information.* Provides the drone pilot or operator with information about other flights that may be of interest to the drone pilot; generally where there could be some risk of collision with the pilot's own aircraft.

- *Navigation Infrastructure Monitoring.* Provides status information about navigation infrastructure during operations. This service should give warnings about the loss of navigation accuracy.

- *Communication Infrastructure Monitoring.* Provides status information about communication infrastructure during operations. The service should give warnings about the degradation of communication infrastructure.

- *Legal Recording.* A restricted-access service to support accident and incident investigations by recording all inputs to U-space and giving the full state of the system at any moment. A source of information for research and training.

- *Digital Logbook.* Produces a report for a user based on their legal recording information.

- *Weather Information.* Collects and presents relevant weather information for the drone operation, including hyperlocal weather information when available required.

- *Geospatial information service.* Collects and provides the relevant terrain map, buildings, and obstacles with different levels of precision for the drone operation.
- *Population density map.* Collects and presents a population density map for the drone operator to assess ground risk. This could be proxy data, e.g., mobile telephone density.
- *Electromagnetic interference information.* Collects and presents relevant electromagnetic interference information for the drone operation.
- *Navigation Coverage information.* Provides information about navigation coverage for missions that will rely on it. This information can be specialised depending on the navigation infrastructure available (e.g., ground or satellite based).
- *Communication Coverage information.* Provides information about communication coverage for missions that will rely on it. This information can be specialised depending on the communication infrastructure available (e.g., ground or satellite based).
- *Procedural Interface with ATC.* A mechanism invoked by the operation plan processing service for coordinating the entry of a flight into controlled airspace before flight. Through this, ATC can either accept or refuse the flight and can describe the requirements and process to be followed by the flight.

Finally, the advanced services, U3:

- *Dynamic Capacity Management.* Responsible for balancing traffic demand and capacity constraints during operational plan processing.
- *Tactical Conflict Resolution.* Checks for possible conflicts in real time and issues instructions to the aircraft to change their speed, level or heading as needed.
- *Collaborative Interface with ATC.* Offers verbal or textual communication between the remote pilot and ATC when a drone is in a controlled area. This

service replaces previous ad hoc solutions and enables flights to receive instructions and clearances in a standard and efficient manner.

2.1.4 Separation and conflict resolution

Separation is a concept that keeps planes at a safe distance from one another to decrease the danger of a collision. ATC is responsible for maintaining a minimum separation between manned aircraft in controlled airspace, based on the flight rules, followed by the aircraft and the airspace class. The remain-well-clear rule ensures self-separation in uncontrolled airspace and for airplanes in controlled airspace for whom separation is not provided by ATC. Minimum separation is maintained in both circumstances thanks to procedural restrictions or a situational observation approach.

The capabilities of the service providing it or the capabilities of the aircraft involved are typically the basis for each separation requirement. With the advent of compact, high-accuracy positional positioning and tracking devices, the minimum separation distance for safe aircraft separation may now be determined by the total navigation and capability of the surveillance system. Separation is a performance-based navigation function that is described in terms of accuracy, integrity, continuity, availability, and functionality. Weather conditions can have an impact on tiny drones in a variety of ways, thus they must be considered when determining the needed gap between two aircraft.

Conflict management is divided into three levels, which also apply to U-space. The strategic deconfliction is focused on the preflight stage of the operation, however, the tactical separation provision is focused on the in-flight stage of the operation.

- **Strategic deconfliction**. Prior to departure, the ability to organize a flight that does not conflict with other users. This includes operators discussing operation plans with appropriate parties and minimizing any potential loss of separation through agreed-upon procedural separation or route planning that avoids other planes.

- **Tactical separation provision**. The capacity to retain situational awareness through visual or instrumental means. Radar is used by ATC to anticipate aircraft trajectories and to give clearance to avoid potential conflicts. Similarly, visual flight rules denote the tactical steps required to deal with the possibility of a loss of separation between two aircraft in uncontrolled airspace.

- **Collision avoidance**. If all preceding separation plans and precautions fail, the capacity to avoid a collision as a last resort. Collision avoidance is accomplished in U-space using detect and avoid methods.

These three conflict management levels are still applicable in U-space. These services can benefit from collision avoidance systems that anticipate a possible loss of separation. When in controlled airspace, the *Collaborative interface to ATC* service or the *Tactical conflict resolution* service can be used to propagate clearances. U-space will also assist in the development of DAA technologies to ensure that missions are carried out safely. DAA on-board drones aim to provide UAS with capabilities similar to those already available to manned aircraft via see and avoid.

Geofences are a new tool for risk mitigation. They are used to putting up geographical barriers to prohibit unwanted drones from entering and exiting a certain area. Geofences consist of a three-dimensional volume and a temporal component representing the time it will be active within the U-space. The following are the characteristics of geofences: it is mandatory to obey them, but exceptions, which will have a standard technical implementation, may be granted. While most geofences exclude aircraft from restricted or controlled areas, a drone may be restricted to staying inside a geofence. They may be temporary, have time of operation, and may be created with immediate effect.

In the U-space, the Operational Volume (OV) of an UAS operation is a critical operational component for airspace control. OVs are like geofences, see Figure 2.3, they are 3D segments with a temporal component that represents the time it will be active within the U-space. The UAS should remain inside the OV at all times. Moreover, geographic overlapping between operations is permitted due to the temporal component of an OV as long as they are separated in time.

The concept of OV is key to the development of this thesis, since it is one of the main parameters taken into account for the detection of conflicts, as well as for their

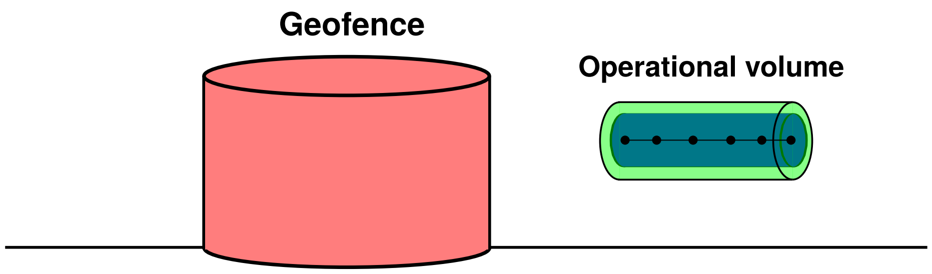


Figure 2.3: Graphical representation of a geofence and an OV of an UAS operation. The OV is made up of two parts: the flight geometry, which describes the extent of airspace in which the UAS will operate, and the contingency volume, which is an outside surrounding volume that accounts for environmental or performance uncertainty.

resolution. These methods of conflict detection and resolution are detailed in Section 4. It must be taken into account that not all UAVs have the same dynamics, it is not the same a multirotor than a fixed wing, therefore, their OVs have to be different, being those of the multirotor much smaller as they have a high maneuverability. In addition, it is not possible to assume a high precision in the localization of the UAVs, which influences the size of these OVs.

2.1.5 European UAS Direct Remote ID

The basis of regulating low-altitude UAS traffic is identification, which successfully permits responsible ownership and allows for the safe integration of UAVs into airspace. The aircraft information, owner or possessor information, and operator information make up the basic content of identification. The ASD-STAN[5] association introduced a technical standard to the European UAS digital remote ID ASD-STAN (2021).

Some drone activities require the drone operator to be readily identifiable by the entity controlling the flight during the flight, U-space operating drones are in this situation. In this instance, the U-space Service Provider (USSP) must guarantee that the remote identification of the drone is processed continuously throughout the flight and that the remote identification of the drone is sent to authorized users. Drones

[5]`https://asd-stan.org/`

operating in the U-space must have a Network Remote ID (NRI) function to ensure that they can connect to the U-space through the Internet.

Such oversight does not apply to operations carried out outside the U-space. To make it possible to read the registration number, the drone should allow easy digital access to it. To that purpose, the Commission Delegated Regulation (EU) 2019/945 requires that drones designed for open category or declaration activities have a Direct Remote ID (DRI) function in their design. During the whole flight, this feature broadcasts the registration number of the operator as well as position of the drone in a form that may be received by existing mobile devices. Drones of class C0, for which the operator is not required to be registered, and drones of class C4, which are aeromodels, are the sole exceptions. The legislation also specifies an add-on that allows drones without DRI to meet extra national criteria if necessary.

Practical implementation of DRI on the UAS may be predicted in two ways. Most drones that have already been sold to customers or are now on the market will require a firmware upgrade from suppliers to meet the DRI standards. UAS that do not receive this level of support from suppliers, or that lack the necessary Wi-Fi or Bluetooth technology onboard, can nevertheless retrofit with add-on devices from third parties. The first option has the benefit of being simple to install, consumers do not have to do anything but update the firmware. The benefit of the add-on, on the other hand, is that it adds new functionality to the drone, maybe including NRI. The downside of the add-on is that it necessitates the purchase and installation of extra equipment.

2.2 UTM related work

Traditional ATM systems rely heavily on the human aspect, which is incompatible with drone operations' inherent automation. Aside from that, ATM systems would be unable to grow to accommodate the enormous volume of UAS operations expected in the future. As a result, it is clear that new control systems are required to allow the efficient concurrent operation of UAS while also ensuring the safety of other aircraft and residents. Even though it has lately been a rising topic, the development of UTMs, such as the NASA' model in the United States Kopardekar (2015) and Europe's U-space SESAR (2017), is still at an early stage. There are a number of services embedded in each framework that will include separation and collision avoidance, route planning services, and airspace design. Other UAS traffic management frameworks that are comparable to UTM models include China's Civil UAS Operation Management System EU-China-APP (2018) and Japan's UTM Consortium (2017). The NASA Technology Capability Level, UTM Pilot Program FAA (2019), and UAS Integration Pilot Program FAA (2017) demonstrations in the United States, and the European Network of U-space Stakeholders in Europe Commission (2021), overseen by Eurocontrol and SESAR, support these UTM frameworks with real-world demonstrations and testing.

There are also a number of commercial UTM applications on the market. They provide the majority of strategic services but only a few tactical services. AirMap (2018), for example, focuses on unmanned aircraft system registration, geographic information systems, flight communication, traffic monitoring, and user interfaces. The Unifly platform connects authorities and pilots to ensure that UAS are securely integrated into the airspace Unifly (2020). This platform has been used in the experiments detailed in Section 6.2. On the one hand, authorities can see and authorize UAS flights in real time, as well as regulate no flight zones. Pilots may manage their UAS, plan and acquire flight authorisations that are compliant with international and local laws. The Thales ECOsystem UTM Thales (2017), which combines UAS and pilot registration, is another framework. Airspace regulation and situational awareness are used as guidelines in ECOsystem's flight planning features. It also comes with tools for working with map overlays and 3D landscape views. Moreover, Indra is working on two solutions for managing unmanned aerial traffic, UTM Hub and UTM Connect Indra (2019). The UTM Hub is a critical tool for law enforcement to track and manage drone operations, on the other hand, the UTM Connect is a multiplatform unmanned service provider solution that enables the connection between a drone and the UTM. Furthermore, Airbus is developing digital ATM technologies to help new aerial vehicles, such as air taxis and delivery drones, enter and use the airspace securely Airbus (2018). The Galicia Institute of Technology is working on Automatic

Intelligent SeRvices for U-space, AIRUS ITG (2018), which will focus on the development of key services such as validation of flight plans according to technical and regulatory requirements, establishment of prohibited areas, air traffic decongestion, strategic and tactical resolution of collision conflicts and real-time flight tracking. The UTM from OneSky is an enterprise-ready system with a variety of deployment choices OneSky (2019). It is intended to help national authorities manage UAS traffic in a unified airspace while maintaining regulatory compliance and flight safety. UAS and pilot registration, flight plan management, airspace analytics, flight monitoring, collision avoidance, and more are all supported by the framework.

Strategic UTM services and some tactical capabilities, such as UAS monitoring, are available through the preceding UTM solutions. They do not handle operations independently throughout the flight phase, despite the fact that they are capable of providing real-time information on the UAS. It is also worth noting that all of those applications are commercial products that may not be available to the public as open source software.

The scientific community has also been working on functional UTM frameworks. Several works Rumba and Nikitenko (2020); Xu et al. (2020) published reviews of related topics. In Taiwan, a prototype UTM for aviation surveillance was suggested Lin et al. (2019). The Automatic Dependent Surveillance Broadcast (ADS-B) technology utilized for surveillance is one of its fundamental qualities, allowing it to monitor cars. There is a preflight protocol for scheduling and approving flights, and the UTM can then provide surveillance warnings throughout the operation, however, the pilot must make all conflict resolution choices. In Sweden, a new UTM has been introduced Lundberg et al. (2018). It includes a comprehensive toolbox for managing traffic, geofences, flight altitude separation, and advanced visualization, as seen in general aviation. By modelling future urban airspace, this research has also uncovered concerns that intense traffic in the VLL airspace would bring to city users.

In general, the preceding systems' functions have only been shown through basic simulations, with a number of works devoted to field flying campaigns for early tests Aweiss et al. (2019); Alarcon et al. (2020). Carramiñana et al. (2021) presented an agent-based simulation platform with a microservice architecture that can replicate UTM information sources including flight plans and telemetry messages. It also takes into account the manual definition of events to replicate unanticipated contingencies such communication interruptions or pilot actions. Shoufan and Alkadi (2021) presented a system that incorporates counter-drone technology into the UTM for information transmission and coordination, based on a set of clarifying protocols, to provide a responsible reaction to the detected drones.

Furthermore, there are several techniques for conflict resolution and emergency management regarding the deployment of certain tactical services. Although automated decision-making has not been fully resolved in UTM systems, some studies Tan et al. (2019); Ho et al. (2019); Sacharny et al. (2020) have focused on flight planning and scheduling at a strategic level in the preflight phase. In general, because of the large search space required to identify the optimal conflict resolution in VLL airspace scenarios, approximation solutions based on heuristic solvers Tan et al. (2019) or lane maneuvers Sacharny et al. (2020) dominate the optimal deconfliction alternatives. In Rubio-Hervas et al. (2018), a probabilistic framework for expressing the risk associated with UAS operations is provided, that approach might be incorporated for automatic, real-time data processing.

2.3 Project results overview

The European Commission mandated the SESAR JU in 2017 to coordinate all U-space and drone integration research and development efforts. This section summarizes the work done, focusing on the conclusions of demonstration projects that covered all elements of drone operations, as well as the supporting technologies and services necessary SESAR (2019).

The outcomes of these initiatives demonstrate the progress made on the building blocks of U-space. Simultaneously, the programmes indicated significant gaps in the performance of specific technologies or areas where more study is required, particularly in the field of urban air mobility operations and the interface with manned aircraft.

2.3.1 Demonstrations

DIODE: D-flight Internet Of Drones Environment

The DIODE project aimed to show that it was possible to successfully operate many drones flying in extremely low-level airspace at the same time while carrying out different operations and missions. The proposal assumed that all aircraft, both manned and unmanned, would report their locations. In other words, the entire flow is cooperative, which reduces the complexity.

The demos took a risk-based strategy to provide initial and advanced U-space services that met drone operators' expectations. D-Flight, a specialised platform that enables e-registration, e-identification, and static geofencing in conformity with European rules set to take effect in 2020, was used to keep track of the drones. Onboard drones, DIODE exhibited developed and mature capabilities that facilitate the implementation of a risk-based and operation-centric U-space concept.

DOMUS: Demonstration Of Multiple U-space Suppliers

Three service providers interacted with one ecosystem manager, while many drone operators used drones from various manufacturers in the DOMUS demos. In this method, the ecosystem manager serves as the primary USSP, ensuring data integrity and serving as a single point of truth for the system: ensuring safety, security, and privacy, as well as facilitating the entry of additional service providers. It also served as a single point of contact for ATC.

DOMUS exhibited e-registration, e-identification, geofencing, tracking, flight planning, dynamic flight management, and interfaces with ATC, which are all described in the U1 and U2 specifications of U-space. Some U3 services were also put to the test, including tactical deconfliction between two drones and dynamic geofencing in coordination with ATM.

EuroDRONE: A European UTM Testbed for U-space

EuroDRONE experimented with several concepts, technologies, and architectures to enhance stakeholder collaboration in a U-space environment. The research tested on U-space features ranges from basic services to more complex services such as automatically identify and avoid utilizing cloud software and hardware.

EuroDRONE conducted highly automated unmanned flights using drones that were completely capable of flying mission planning, employing a cloud-based UTM coupled with a small, intelligent transponder processing board. The experiments featured a unique vehicle-to-infrastructure link that was connected to a self-learning UTM platform and could communicate real-time flight data. The flights showcased end-to-end UTM solutions for both VLOS and BVLOS logistics and emergency services. The insights were utilized to develop a viable, automated cloud-based UTM architecture, which was then validated through simulation and live demonstrations.

GEOSAFE: Geofencing for safe and autonomous flight in Europe

The goal of the GEOSAFE study was to identify state-of-the-art geofencing U-space systems and make suggestions for future geofencing system definitions. The project was built on a one-year flight test campaign that evaluated basic and sophisticated geofencing services in pretactical flight, tactical operations, and dynamic circumstances.

The project found that the majority of drones fit the criteria for pretactical geofencing and that the existing technology is ready for early U-space services. Despite the absence of standardization, solutions exist to provide tactical geofencing, which is required to deliver enhanced U-space services. However, the technology capable of supporting dynamic geofencing is not yet mature enough to meet full U-space service levels, although this is expected to change quickly in the near future, not least because dynamic geofencing is a critical function for unmanned vehicles operating beyond the line of sight.

GOF-USPACE: Safe drone integration in Gulf of Finland

The GOF-USPACE partners created a preoperational authority FIMS by developing an interoperability architecture for combining existing solutions from three USSPs to demonstrate U-space in all phases of drone operations. The GOF U-SPACE architecture, in particular, allowed data to flow between two air navigation service providers, multiple USSPs, eight drone operators, and two manned aircraft operators.

The GOF U-SPACE design incorporated microservices from U-space service providers, allowing for the collective and cooperative administration of all drone traffic in a given geographic region. The research highlighted the necessity for a single source of trustworthy airspace and aeronautical information for all airspace users, as well as uniform standards for system communication.

PODIUM: Proving Operations of Drones with initial UTM

Drone operators must now go through a series of manual procedures before they can take flight. All of this necessitates more time and effort, which may compromise drone operations' financial sustainability. By presenting a web-based UTM, PODIUM aimed to mitigate the hazards associated with the operational and industrial deployment of U-space.

PODIUM found that there is a considerable need from all stakeholders for U-space solutions that may reduce the burden of acquiring flight authorisations for drone flights, and that greater situational awareness improves flight safety and efficiency. It discovered that U-space services in the preflight phase are nearly ready for deployment, but that further work is needed to ensure that U-space services can truly take off in the flight execution phase. PODIUM provided specific suggestions for tracking the human-machine interface for drone pilots and access to reliable data, all of which have consequences for standardization and legislation, as well as future research and development.

SAFEDRONE: Unmanned and manned integration in very low-level airspace

The SAFEDRONE project aimed to define and detail in-flight services such as geofencing, flight tracking, and automatic technologies to detect and avoid obstacles in order to demonstrate how to integrate manned aviation and drones into non-segregated airspace. Pre-flight services included e-registration, e-identification, planning, and flight approval. The goal was to gather information and experience on the services and practices required to operate drones in a safe, efficient and secure way within U-space.

The project also took into account the need for higher levels of autonomy when operating in non-segregated airspace in order to perform dynamic in-flight tasks like on-board re-planning trajectories within the approved U-space flight plan, and autonomous generation of coordinated trajectories within an approved U-space area of operation. This project appears in section 6.2 where autonomous detect and avoid experiments are detailed.

SAFIR: Safe and Flexible Integration of Initial U-space Services in a Real Environment

SAFIR demonstrated the capabilities of drones to protect key places such as an international port or a city. The Port of Antwerp was shown how it might request a drone to investigate a specific area if there was a cause for worry, as well as designate no-drone zones to control safety in the port.

SAFIR proved that the following services are fully operational: e-identification; pre-tactical, tactical, and dynamic geofencing; strategic and tactical deconfliction; and tracking and monitoring. The project successfully evaluated beginning, advanced, and complete U-space services, as well as making research suggestions. It was found, for example, that tracking data from various sources must be fused, and that a connection to ATC, preferably in an automated manner, is essential. It also discovered that satellite mobile connectivity works well, but that 4G weakens at higher altitudes and that a specialised 4G drone overlay network would be beneficial, especially for BVLOS activities.

USIS: Easy and safe access to the airspace

The USIS project evaluated both basic U-space services such as e-registration and e-identification, as well as the more complex flight planning, authorisation, and tracking services required for BVLOS operations and above humans. It also looked at airspace management and scheduling. At sites in France and Hungary, USIS partners conducted live demos using a secure and robust cloud-based platform. Drone operators may submit flying requests to a specialised app, which were then analyzed and authorized or denied by the proper authorities.

The experiment demonstrated that early U-space services can handle numerous drone flights without adding to an operator's workload or compromising airspace safety. It emphasized the importance of flexibility in flight planning and approval management systems to comply with a variety of national and local rules.

VUTURA: Validation of U-space by Tests in Urban and Rural Areas

VUTURA aimed to achieve four major objectives: validating the use of shared airspace by existing manned airspace users and drones; validating more than one USSP, providing U-space services in a specific airspace and the procedures required to support drone flights; ensuring regulatory and standardization alignment between SESAR development and USSPs; and speeding up the rate at which European cities and businesses exploit emerging technologies.

VUTURA's research showed that commercial drone traffic can coexist securely with traditional air traffic in a variety of settings, and that the technology to properly control drone traffic is practical, scalable, and interoperable. It also identified areas in which more study is needed. This contains a set of rules for cross-border flight planning, a common interface for transferring information, and an acceptable transmission latency, as well as a reliable detect and avoid capability.

Chapter 3

Developed services

This chapter presents the services that contribute to the architecture developed to test the U-space services addressed in this Thesis. The architecture has a modular design that is independent of the particular vehicle characteristics and the onboard autopilot. One of the main goals of the design is reusability through hardware abstraction and imposing no requirements on the execution capabilities of the aircraft. Therefore, aerial vehicles from different research groups and manufacturers can be easily fitted within the architecture. Moreover, this chapter presents an onboard autonomous module to detect and avoid unexpected large static obstacles, it is external but linked to the UTM. Section 3.1 presents the modules that compose the UTM architecture, section 3.1.7 exposes the external onboard module that detects and avoids unexpected obstacles, section 3.2 describes why the Robot Operating System (ROS) Quigley et al. (2009) has been chosen as the main framework, and section 3.3 details the communications between the modules of the UTM built upon ROS.

3.1 UAS traffic management architecture

This section describes the proposed UTM architecture, which is built upon ROS, detailed in Section 3.2.1, thus each module of the architecture consists of a software process implemented as a ROS node. The communication between nodes is carried out by ROS topics and services, but the system uses mainly services, as they offer the possibility of confirming the receipt of messages. In these cases, there is asynchronous communication between the modules, whereby one module acts as a server and the others as clients. Topics provide a synchronous communication and are used by modules that need to publish information at a constant rate, for example, the reception of UAS telemetry.

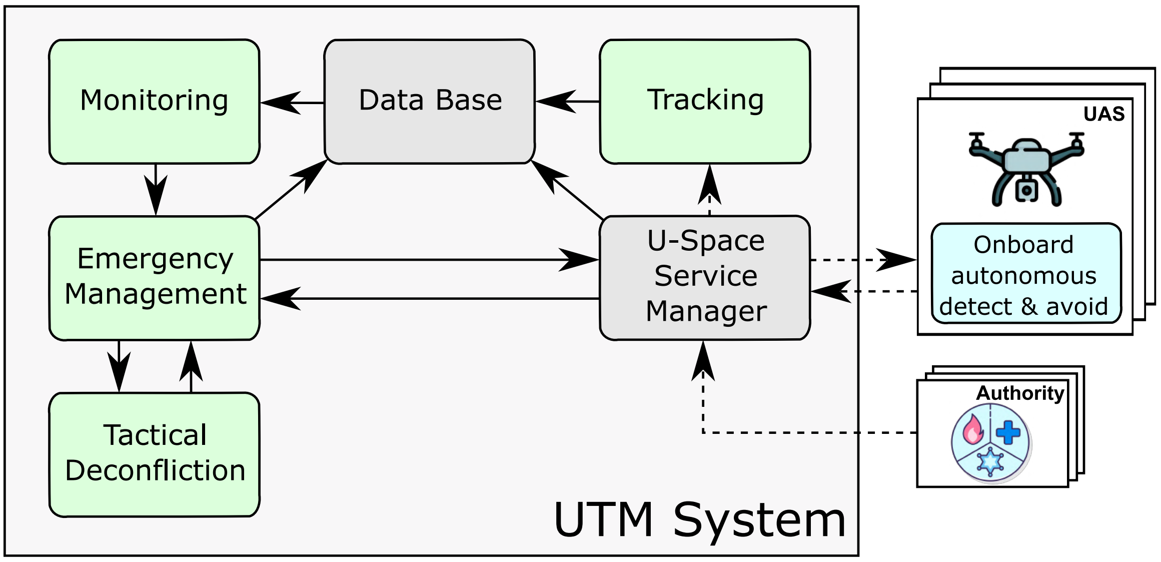


Figure 3.1: Overview of the proposed UTM architecture. Green modules implement U-space services, gray are auxiliary modules, and the blue one is an external module of the UTM used to DAA unexpected structures. Moreover, outside the UTM are all active UAS, and the authorities, which may interact with the UTM warning of threats. Arrows show services between modules, on the other hand, dashed arrows represent topics.

Figure 3.1 shows all software modules involved in the architecture and their interactions. The green modules implement specific U-space services, services that are required during multiple UAS flight operations. In particular, four services with their corresponding modules should be highlighted: Tracking, Monitoring, Emergency Management and Tactical Deconfliction. These services were chosen because they provide the minimum flow of information that can be used to detect, manage and resolve conflicts during the flight of multiple operations. Other modules related to registration have been validated in the SAFEDRONE project by the company Unifly, details can be seen in Section 6.2, therefore they are not highlighted in the presented architecture.

There are additional modules that provide support to the architecture. The Database module handles the information about the state of the airspace, this includes the UAS operations that are currently active and the ones that are accepted. It also includes the geofences that have been activated by auxiliary stakeholders and the ones that have been created by the UTM itself. U-space Service Manager is the interface between the UTM and the rest of the U-space, it receives alerts and state information from the UAS and the external auxiliary stakeholders. This module also communicates recommended actions to deal with threatening events, for example, a loss of separation between UAS or a geofence intrusion. These actions to avoid threats are generated by the interaction between the Monitoring, Emergency Management, and Tactical Deconfliction modules. Finally, an external onboard autonomous DAA module has been developed, and it is detailed in Section 3.1.7.

It is important to remark that the presented architecture can work autonomously without the need of a human intervening in the whole process, aiming to match the final objective of the U-space framework. However, according to the current regulation, the UTM architecture just suggests actions to the UAS operator, which is the one who has the final word and the responsability to go ahead with the proposed action of the UTM.

The following sections provide details of the modules that we have developed for our UTM. For each module, the functionality and interactions with other modules are described, as well as the methodologies that have been used to implement them.

3.1.1 U-space Service Manager

The U-space Service Manager (USM) is the interface between the UTM and the stakeholders of the U-space. It receives the telemetry from each UAS, which is transmitted by their onboard autopilot and ADS-B[1] transceivers if available. The USM

[1]ADS-B is a surveillance technology that allows an aircraft to be monitored by determining its position using satellite navigation and broadcast it.

is in charge of transforming this position data from geographic to local coordinates, thus our UTM works in local coordinates internally.

External stakeholders can use the USM module to transmit warning information to the UTM, for example, a wildfire declared by the fire-fighters. Each UAS can also transmit warning information, like the detection of a jamming[2] or spoofing[3] attack. The USM module is in charge of commanding to the UAS operators the actions recommended by the UTM to avoid imminent or future threats. These recommended actions to variate the flight plan of the UAS must be confirmed or rejected by the involved UAS operator due to actual regulatory restrictions. In case of acceptance, the USM would notify the Database module to update the state of the corresponding operation.

3.1.2 Database

The Database module is in charge of storing and handling the necessary information for the UTM related to the actual situation of the airspace. This situation means the information of the operation of each UAS flying or that have an operation scheduled, and the information related to the geofences that are active. Other modules of the UTM can read the Database or write to it to perform their functions, so the Database works as a server for the whole system. For example, the Monitoring module needs to check the estimated trajectory of every active UAS, so it reads from the Database information at every step to carry out its task. On the other hand, the Tracking module calculates the estimated trajectory of all active UAS and updates this information writing it in the Database.

Table 3.1 and Table 3.2 details the data structures that compose the Database, which are geofences and operations. A geofence is a 4D volume of the airspace, a 3D geometrical space with an activation period of time, which is restricted to fly inside for regular UAS. An operation is composed of the reserved 4D trajectory, the estimated trajectory that the UAS will do, safety-related volumes around the flight

[2]A jamming attack consists of an attempt to endanger the GNSS signal of an UAS.

[3]In a spoofing attack, fraudulent GNSS signals that are stronger than the real unencrypted signals are sent into the UAS to cause it to deviate from its intended course.

Table 3.1: Attributes of a geofence object.

Attribute	**Description**
Identifier	Unique number for geofence identification
Type	Cylindrical or polygonal
Min/max altitude	Altitude range where the geofence is active
Start/end time	Time period in which the geofence is active

Table 3.2: Attributes of an UAS operation object. Some ConOps examples are: power line inspection, long forest surveillance, wind turbine inspection and event monitoring.

Attribute	**Description**
Identifier	Unique identification of the aircraft
Priority	Priority of the operation in the airspace
Flight plan	Reserved 4D trajectory for the operation
Next waypoint	Waypoint index that the UAS is currently targeting
Estimated trajectory	Prediction of the future UAS trajectory
ConOps	Description of the concept of the operation
Flight geometry	Radius of the cylindrical volume where the UAS is intended to remain during its operation
Operational Volume	Radius of the outer cylindrical volume to account for environmental or performance uncertainties

plan, and the information needed to have a unique identification of the operation, like the International Civil Aviation Organization[4] (ICAO) address.

3.1.3 Tracking

This module is in charge of tracking all active UAS in the airspace. Within a certain time horizon, the Tracking module updates in real time the current position and predicts trajectory of each UAS.

The module computes the tracks by fusing information from different sources, UAS telemetry, and ADS-B transceivers if they are available, that it receives through the USM. The estimated trajectory of each active UAS is predicted given its current position and velocity, as well as its flight plan. The Tracking module updates the

[4]The ICAO addresses are 24-bit numbers to identify aircraft uniquely worldwide.

tracks of the active UAS in the Database module, to make this information available for the rest of the modules.

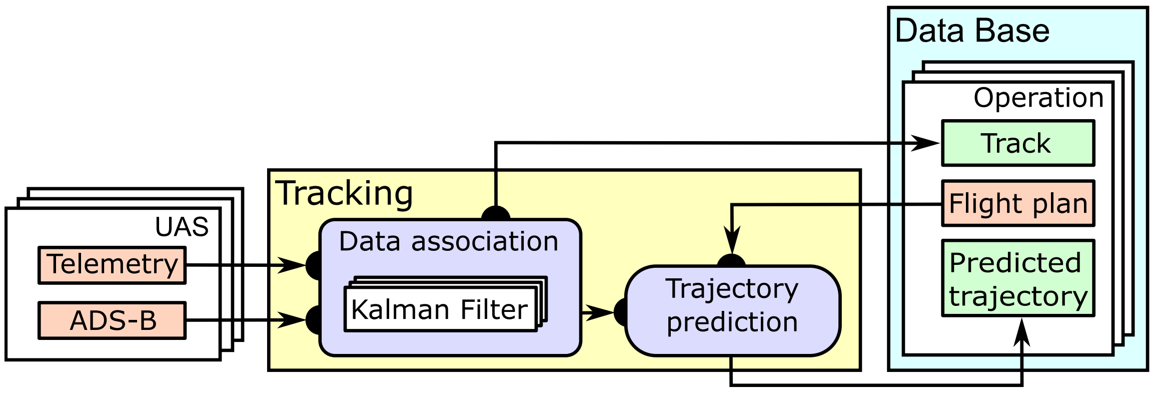


Figure 3.2: Scheme with the internal components (purple) of the Tracking module. The *data association* component matches the measurements from the UAS with their previously position calculated to update the corresponding Kalman filters. The future UAS trajectories are predicted using the output of the data association and the flight plans, which are read from the Database module. Finally, the predicted trajectories are stored in the Database after being calculated, as well as the tracks after being updated.

The Tracking module estimates through a Kalman Filter that integrates the measurements coming from the onboard ADS-B transceiver and the UAS telemetry. Figure 3.2 details how Tracking implements a stochastic filter that maintains a list of objects to estimate the state of all UAS in the airspace. Each state consists of a 3D position and velocity. Irregular sensor rates, noisy and delayed measurements can be handled by the filter and make the data more usable by the UTM.

At a constant rate, the list of active UAS is read from the Database module to identify if there is a new operation. The Tracking module gets constantly just the information needed of the active UAS operations from the Database to perform its tasks, but if there is a new UAS active, it gets the information of the operation. This avoids updating unnecessary information of the operation every step.

The state of all those UAS is predicted and then updated with the received observations that can be easily associated with its corresponding flight plan, since they all come with a unique UAS identifier. If an operation has an unknown UAS identifier, it will be considered as a noncooperative aircraft and the filter will ignore it, an attempt has been made to replicate the behavior of the NRI shown in Section 2.1.5. Furthermore, the current waypoint for each UAS is computed by searching for the waypoint in its flight plan that is closest to its current position. The estimated trajectory is also predicted for each trajectory, but it is limited by a given time horizon to reduce the computational load. If the current position of the UAS is close enough to its current waypoint, the prediction of the estimated trajectory sticks to the flight plan. Otherwise, the Kalman Filter is used to predict a trajectory given the current UAS position and velocity. Finally, the Tracking module updates the information about the trajectories in the Database module after each step.

The difference between our implementation and what is established in the U-space ecosystem definition is that our present implementation only handles cooperative UAS instead of being designed to take both cooperative and non-cooperative UAS into account. This is because our work has been dedicated to enabling automated decision-making for UAS that are actively operating, which is useless for non-cooperative aircraft. These non-cooperative aircraft need to be handled as uncontrollable hazards that are invading the controlled airspace. Moreover, our Tracking module does have the capacity to update and store data in real-time from many sources. If necessary, other services may also access these data via the Database module.

3.1.4 Monitoring

The functionality of the Monitoring module is to detect potential conflicts of the observed airspace that needs to be managed by the UTM. This module deals with conflicts related to UAS trajectories, it detects the case of an UAS getting out of its reserved flight volume, if it conflicts with a geofence or whether two UAS lose a minimum required separation. To perform this detection, the module reads periodically information from the Database module about the UAS estimated trajectories and geofences, and it uses that data to report detected conflicts in the monitored airspace.

When Monitoring notifies a conflict, it should indicate the type of the detected conflict, an estimation of the time instant when the conflict will occur, and a snapshot of the parts of the airspace involved in the conflict. This means the estimated trajectories of the involved UAS and, if necessary, the information related to the conflicting geofence. This snapshot is key so that the modules resolving the conflicts use the same information to evaluate the situation and to avoid time glitches and incoherent solutions.

Monitoring detects possible conflicts in sequence. It first checks if an UAS is out of its OV, which is a 4D volume around the flight plan with a temporal component that means at what time the spatial volume will be active. This is the first check of the Monitoring module because if an UAS is out of its OV, the UTM treats it as a noncooperative UAS and as an out of control UAS. This UAS is doing something that was not expected, so the UTM notifies the operator that the UAS involved should return as soon as possible to its flight plan. Further checks are not necessary if the OV conflict is detected, it is an early return of the module to avoid extra computational time.

If the observed UAS is inside its OV, the Monitoring module starts checking for possible conflict with active geofences. As an early return, Monitoring checks if the UAS is right now inside a geofence, if it is the case, the UAS should leave it as soon as possible. For example, this case can appear if an UAS is monitoring a field for agriculture and the firefighters warn of a wildfire in the same zone that the UAS is having its operation. This conflict is called geofence intrusion and if it is matched, further checks are not necessary to avoid extra computational time. Another possible conflict related to geofences is geofence conflict. To detect this conflict, Monitoring checks if the estimated trajectory of the observed UAS intersects with an active geofence. Every waypoint belonging to the estimated trajectory of each UAS is compared against the active geofences, to determine whether the UAS is already intruding a geofence or it is estimated to enter one in a short future time.

The last check that makes the Monitoring module is called loss of separation, this appears when two UAS loss a safety distance. It is computed last because it needs more computational load to determine if there is a conflict, so the Monitoring module gets here as previous early returns were not matched. This check is done with a geometrical approach whose details can be seen in Acevedo et al. (2019). The Monitoring module discretizes the airspace to model it as a 4D grid, where each cell represents a 4D volume in space and time (dX, dY, dZ, dT) and stores a list of all UAS whose trajectory is estimated to be inside. Hence, each waypoint of an UAS trajectory only needs to be compared with other waypoints of other UAS within the neighboring 4D cells. For each waypoint in the 4D grid, the distances to the waypoints in the list of its neighboring cells are calculated. A loss of separation conflict will be reported, if any of these distances is shorter than a safety distance.

As mentioned above in the Tracking method, our present implementation does not take non-cooperative UAS into account, as established in the U-space ecosystem definition. Our Monitoring method does not provide a particular communication link to give traffic information to the UAS operators, however, this could be done simply through the USM module. On the other hand, our Monitoring implementation achieves all the other expected features of the U-space ecosystem definition; for example, it detects and warns in real time about conflicts involving geofences, flight non-conformances, and inter-UAS separation.

3.1.5 Emergency Management

The Emergency Management module is the component of the UTM that decides which alternative route will be given to the UAS operator. It interacts with the Monitoring, USM, Database and the Tactical Deconfliction modules. The Emergency Management receives reports about external warnings coming from UAS operators or auxiliary stakeholders in the U-space through the USM module. For example, a jamming attack, the declaration of a wildfire, or any other threatening event notified by the authorities. If the Emergency Management module receives a detected threat from the authorities through the USM module, it creates a geofence to isolate that threat. This can result in several conflicts that should be detected by the Monitoring module, like having an UAS inside that geofence or an UAS flight plan intersecting it, see Section 3.1.4.

This module also receives conflicts detected by the Monitoring module and asks the Tactical Deconfliction module for support to have a list of alternative flight plans, see Section 4.2.2. The Emergency Management should choose among all possibilities and send the result to the USM to notify the corresponding UAS operator of the proposed alternative flight plan. The selection of the alternative flight plan is done minimizing the cost and the risk level of the route, more details can be seen in Capitan et al. (2021). The architecture proposes safe alternative flight plans to maintain a secure airspace.

In the U-space ecosystem definition, the Emergency Management service notifies UAS operators of alerts and any other emergency assistance. Moreover, our EM implementation can make automatic decisions to manage hazardous situations in real time by proposing safe and effective deconfliction maneuvers to the UAS operator.

3.1.6 Tactical Deconfliction

The module Tactical Deconfliction is in charge of resolving conflicting situations, calculating a list of different alternative flight plans for UAVs involved. The request to solve a conflict is sent by the Emergency Management module, and it contains all information related to the event to solve, like the data of the involved operations and the active geofences. The response is sent by the Tactical Deconfliction with the alternative flight plan list, every alternative has an associated cost and risk that will be used by the Emergency Management to choose the proposed alternative flight plan. The Tactical Deconfliction module has two strategies to solve conflicts, if the conflict involves one single UAS or if the conflict involves multiple UAS.

The first strategy is used to solve conflicts with a single UAS involved, for example, an UAS that is out of its OV or has a conflict with a geofence. If it is out of its OV, the Tactical Deconfliction calculates two alternative flight plans: the first one from the current UAS position to the closest point of its flight plan, and the second one from the UAS position to its next waypoint in the flight plan, no matter how long the UAS remains out of its OV. There are two possible cases if an UAS has a conflict with a geofence: geofence conflict and geofence intrusion. The first case appears when an

UAS flight plan goes through a geofence, so one of the computed alternative flight plans should go around the geofence to avoid it. On the other hand, if the UAS is already within a geofence, it gets out of the geofence as soon as possible, and then it goes around the geofence to resume its flight plan afterwards.

The second strategy is used to solve conflicts with multiple UAS involved, which mainly result in a loss of separation between two UAS. The number of alternative flight plans returned in this case is directly related to the priorities of the operations. If the UAS involved has the same priority, the module returns the possibilities, but if the priorities are different, the Tactical Deconfliction module returns just the alternatives that involve the UAS with less priority. This is key to reduce the computational time and to avoid alternative routes for aircraft with less maniobrability than the others. In the case of multiple UAS involved, the Tactical Deconfliction module uses a geometric approach based on repulsive forces to modify the current flight plan. The details of the implemented algorithm can be seen in Acevedo et al. (2020), it generates several alternative flight plans applying vertical and horizontal separation to the involved UAS trajectories.

Even if a conflict is solved, the chosen alternative flight plan could still produce additional conflicts with geofences or even with other flight plans. In this case, the Monitoring module would report those new pending conflicts in subsequent iterations, see Figure 3.3, until there is no conflict.

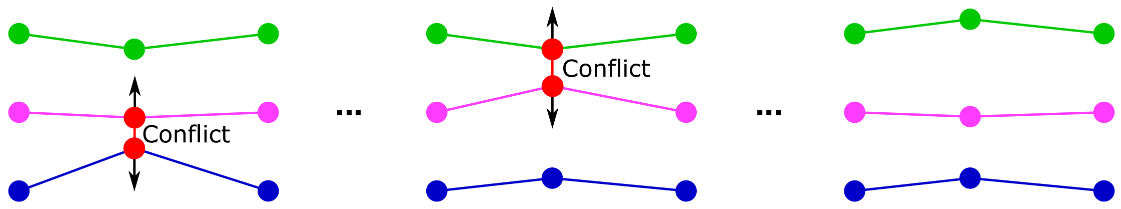


Figure 3.3: Iterative procedure to solve a conflict in the case of a loss of separation from left to right. The flight plans of the two lower UAS are in conflict and need to be separated. Then, the middle UAS enters in conflict with the upper UAS, so these two get separated again to achieve a final solution without loss of separation.

If all alternative flight plans are too risky to continue with the operation, the Emergency Management module could select between the two options given by the

Tactical Deconfliction module, which are an alternative flight plan to return to home and another one to go to its closest landing spot. These landing spots are selected by the operator and stored in the Database module.

In the U-space ecosystem definition the Tactical Deconfliction service is designed to provide UAS operators deconfliction information via the USM. However, in the presented architecture, the EM module is an additional step that fills this duty. The EM requests assistance from the TD module in generating possible alternative plans and incorporates the automated decision-making capacity. Thus, the decision on the appropriate course of action for real time deconfliction is left to the EM.

3.1.7 Onboard autonomous detect and avoid module

Based on the notion of future intelligent drones equipped with onboard sensors with high computing capabilities, we examined the integration of advanced functionality in U-space. The integration of onboard DAA capabilities, as well as the consequences for U-space services, were examined. In the framework of the European SAFEDRONE project[5] we developed an onboard autonomous DAA module, which is external and independent of the UTM architecture, and we tested it in a specific scenario. It was considered a scenario with an UAS flying a previously approved flight plan at a very low altitude and encountering an unanticipated static ground obstacle. The UAS then had to compute an alternative flight plan autonomously to avoid the obstacle. Our objective was to figure out which processes should be followed and how they should change in the near future to help DAA capabilities. A multirotor platform was chosen for this situation because of its higher manoeuvrability compared to fixed-wing systems, as well as its ability to hover while it awaits approval of the new flight plan.

Geographic Information System (GIS) support will be key in the future to enable safe and secure trajectories in VLL and urban environment, it provides information about several layers related to a city: buildings, population, traffic, vegetation, etc. This information will allow future developments to optimize trajectories in those

[5]`https://cordis.europa.eu/project/id/783211`

complex scenarios, however, temporary static large obstacles like cranes, scaffolding, etc., may not be included in GIS maps. Therefore, the development of an onboard DAA module is needed to ensure that UAS can accomplish their trajectories even when encountering unforeseen obstacles.

To detect those unexpected obstacles, the multirotor had a 3D LIDAR mounted, which gave a stream of point cloud measurements that were to be integrated continuously into a virtual representation of the world. The selected environment representation is the octree implementation of the octomap Hornung et al. (2013) library[6], which implements a 3D occupancy grid mapping approach, providing data structure and mapping algorithms. Once the representation was done, the module was able to calculate whether the accepted flight plan crossed an obstacle. In such cases, the path planner computed an alternative route to avoid the unexpected obstacle.

The path planner chosen was the Lazy Theta* Nash et al. (2010), which is based on the well known A* algorithm. A considerable disadvantage of the A* algorithm is that it creates paths using only the edges of the grid. On the other hand, Theta* algorithms do not have the constraint of using only the edges of the grid, as it uses any angle. Each extended vertex does a line-of-sight check for each extended visible neighbor, allowing the vertex's parent to be any other graph vertex, resulting in realistic and short-looking routes. In comparison to Theta*, Lazy Theta* decreases the number of line-of-sight inspections, allowing it to locate routes quicker while maintaining the same length.

The DAA module integrates the presented algorithms of this section using a single onboard computer that allows the UAS to generate a virtual representation of the world, detect unexpected obstacles, generate an alternative route to avoid it, communicate with the operator in charge of the operation, autonomously and online. Figure 3.4 shows a simulation of the module, the rectangle prism is being detected by the simulated 3D LIDAR. Pink dots represent the point cloud, which is the input of the octomap library that transforms this information into 3D voxels. The representation allows the module to differentiate between three types of space: obstacle, free, and unknown. If the beam of the sensor goes from the sensor to the maximum range,

[6]`https://octomap.github.io/`

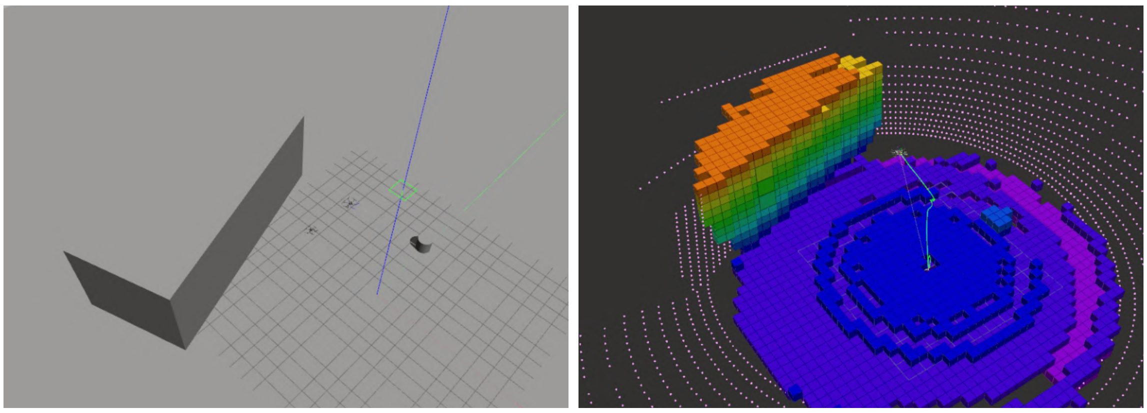


Figure 3.4: Left, Gazebo world with a 3D model of a rectangular prism, and a multirotor with a 3D LIDAR attached. Right, RViz visualization of the output of the LIDAR in form of pointcloud, which is transformed in an octomap. Colors represent the height of the obstacle.

the space travelled by the beam is free. If the beam hits something, the voxel that surrounds it is an obstacle, the space between the sensor and the obstacle is free, and the space that is beyond the maximum range of the sensor is unknown.

3.2 Robotics framework for the implementation of the architecture

This section details a comparison between the foremost robotics frameworks, what offers ROS to the UTM architecture, and how are the communications between the modules using ROS.

We have chosen a robotics framework to implement our UTM architecture, mainly due to its advantages for system integration and realistic Software In The Loop (SITL) simulation. The trend of the last decade within the field of robotic software development aims towards the creation of open source frameworks that make the method of robotic software development much easier by enabling, as much as possible, the reuse of code. All coincide within the approach of separating the code into components or software modules that fulfill a particular functionality of the whole robotic system.

The Global UTM Association is a group of UTM stakeholders from across the world that have come together to debate which critical attributes should be included in future UTM systems GUTMA (2020). These characteristics are a perfect match for several of the ideas used in robotics frameworks, so they can be considered for our UTM development. The following considerations should be addressed throughout the design process of any UTM architecture:

- **Open source** technologies provide worldwide compatibility, and these components can help UTM services be deployed and developed more quickly.
- It is more effective to have an UTM that provides **automated** services to help UAS operators. As a result, to ensure safe operations for both human and unmanned aircraft, the system should provide support through automated features for flight planning, monitoring, and tactical deconfliction.
- **Flexibility and adaptability** are required to include new stakeholders and services as they emerge. To make the process of developing increasingly complicated capabilities easier, the system should be constructed out of reusable modules.
- A **scalable** architecture is required to accommodate additional actors and services. Not only is the preceding modularity desirable, but also a paradigm with decentralized responsibilities, rather than the outdated approach of a centralized ATC to achieve this.
- The system must be **secure and safe**. It should be able to tell who is piloting each unmanned aircraft, where they are flying to, and if they are adhering to the operating requirements.

The foremost open-source frameworks that implement a hybrid architecture concept are ROS, Robot Construction Kit (ROCK), Open Robots Control Software (ORoCoS) and Generator of Modules (GenoM). The pros and cons also are summarized in Table 3.3 and the following points present a brief comparison between the existing robotics frameworks:

- GenoM exports data using public structures, ROS communicates using a topic-based model. ORoCos and ROCK are connection-based middleware. A topic communication requires less management cost and it is easier to use, but it is harder to control the data flow.
- GenoM, ORoCos and Rock modules are defined using an abstract module description, while ROS does not need it.
- Real time applications can easily be implemented in ORoCos and Rock, while ROS and GenoM provide support for near real-time with special interfacing.
- ROS does not have the policy to separate between libraries and middleware, but GenoM and Rock force the user to do it.
- Rock uses standard C++ classes as interface types. GenoM is C structure oriented and supports C++ classes. ROS uses Interface Definition Language making the interface cleaner.
- Rock and ORoCos have native support for flexible module deployments. This can be also possible in ROS with the assistance of nodelets, but not as transparent.

Table 3.3: Summary of the pros and cons of existing robotics frameworks.

Framework Feature	**ROS**	**ROCK**	**OROCOS**	**GENOM**
Scale Down	↑	↑ ↑	↑↑	↑
Scale Up	↑	↑↑	↑↑	↑
Number of libraries	↑↑	↓	↓	↓
Middleware independent	↓	↑↑	↑↑	↑
Hard Real Time	↓	↑↑	↑↑	↑
Framework tooling	↑↑	↑	↑	↑
Supported OS	↑↑	↑	↑	↑
Documentation	↑↑	↓	↓	↓
Community size	↑↑	↓	↓	↓
Model driven	↓	↑↑	↑	↑↑

3.2.1 Robot Operating System

ROS is an open-source framework for robot software development. It consists of a set of tools, libraries, and conventions to ease the deployment and use of complex applications in robot systems, such as low-level device control, hardware abstraction, implementation of frequently used features, package management and passing messages between processes. ROS provides drivers to communicate with a large spectrum of both open-source and commercial autopilots and onboard sensors. The utilization of ROS for multi-UAS systems is extending fast and is accepted among the community, because it facilitates the way for the integration of heterogeneous hardware and software systems. ROS is a framework based on multiple processes called nodes. These nodes are usually grouped into packages and communicate with one another by passing messages, which are typed data structures. ROS implements asynchronous communication through a publish-subscribe paradigm where nodes can stream messages over different topics. On the other hand, synchronous communication is implemented through services for request-response interactions.

ROS offers multiple features that fit our design guidelines, so we decided to use it as middleware for our architecture. It is designed to create modular and reusable components. Therefore, ROS produces flexible and scalable systems that can be adapted easily to include new features. Communication solutions and drivers for the most popular autopilots (e.g., PX4, ArduPilot, DJI, etc.) are already available in ROS. Moreover, it provides remarkable tools for system integration and testing.

We have chosen ROS to implement our UTM architecture, mainly due to its advantages for system integration and realistic SITL simulation. It is important to remark that the proposed UTM architecture is a more general concept that could be adapted to alternative middleware solutions if needed.

3.3 ROS communication of the UTM modules

This section provides a more detailed view of the communication between the modules that conforms the UTM architecture. The architecture follows the concepts[7] of ROS having a peer-to-peer network of processes that are processing data together. The basics of ROS are nodes, messages, topics, and services:

- **Nodes** are the processes that perform computation. ROS is designed to be modular at a fine-grained scale; an UAS traffic management comprises many nodes. For example, one node stores relevant information (Section 3.1.2), one node detects conflicts (Section 3.1.4), one node solves these conflicts (Section 3.1.6), and so on.

- A **message** is a data structure, including typed fields. Standard primitive types like integer, boolean, or floating point are supported. By passing messages, nodes communicate with each other.

- **Topics** are a transport system with publish and subscribe semantics. It can be said that a node is a strongly typed message bus. Each bus has a name, and any node can connect to the bus to send or receive messages as long as they are the right type. The idea is to decouple the production from the consumption of information.

- **Services** represent the request and response interactions, which are often required in a distributed system. A providing node offers a service under a name and other nodes acting as clients use the service by sending the request message and waiting the reply.

Each module described in section 3.1 is a ROS node and communicates with other nodes using mainly services to assure that the information sent is received without any problem. Table 3.4 details the services offered by each module, which one is the server and which ones are the clients.

[7]wiki.ros.org/ROS/Concepts

Table 3.4: Summary of the services used in the UTM architecture.

Name	**Server**	**Clients**
Write Operations	Database	*Operator*
Read Operations	Database	Tracking & Monitoring
Write Geofences	Database	Emergency Management
Read Geofences	Database	Monitoring
Read ICAO	Database	Tracking & Monitoring
Write Tracking	Database	Tracking
Update Flight Plans	Tracking	USM
Change Flight Status	Tracking	USM
Threats	Emergency Management	Monitoring
Deconfliction	Tactical Deconfliction	Emergency Management
Notifications	USM	Emergency Management

The Database module acts as a server offering different services to interact with it. Other modules can read an operation using the *read operations* service or write a geofence using the *read geofences* service. The Emergency Management module uses the service *write geofences* to create geofences if the stakeholders warn the UTM architecture of a threat. As shown in Table 3.4, an operator can interact with the Database, adding operations manually through the *write operations* service, this is a special case because the operations should be written in the preflight stage, the UTM architecture offers this service to the operator in case it becomes necessary. Another module that can write to the Database using the service *write tracking* is Tracking, which is in charge of updating the position, estimate and track the trajectory of the active UAS in the airspace. In this case, the message sent is simplified and it does not contain the data of an operation, it contains just the necessary information to update the position of the UAS: ID, next waypoint, track, estimated trajectory and position. This reduces the amount of data sent to the Database, lighweighting the communications and taking care of the possibility of scaling the system. The service *read ICAO* is used for the same purpose as the last one explained, it is a lightweight manner to read from the Database how many UAS are in the airspace, because it consults just the ICAO instead of the whole operation of the UAS stored. The Monitoring module needs to check how is the airspace at every iteration and it uses,

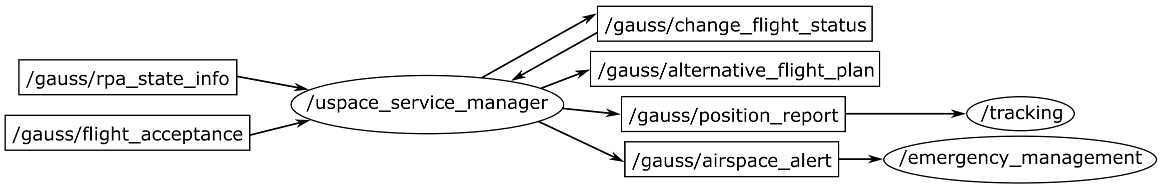


Figure 3.5: The UTM architecture in the ROS graph. The nodes (ellipsoid) are the modules described in Section 3.1 and the rectangles are the topics used in the system.

as a client, the services provided by the Database module. When it detects a conflict, it sends a threat to the Emergency Management module using the service *threats*, which contains a snapshot of the current state of the airspace, this way time glitches can be avoided not being necessary for Tactical Deconfliction to consult the Database. It is in charge of returning alternative flight plans for a received conflict, thus it acts as a server of the service *deconfliction*. The response is received by the Emergency Management module, which selects the alternative route that will be proposed to the operator, this module uses the service *notifications* as a client being the USM module the server. In the case of the services *update flight plans* and *change flight status*, the USM module is a client of the Tracking module, as this one is in charge of writing to the Database, it should store the flight plans accepted by the operator and store the status of the flight plan, whether its flight plan has started or finished.

In ROS, when submitting a request to a service, a synchronous client will block the calling thread until the response has been received, nothing else can happen on that thread during the call. The call might take any duration to finish, and the answer is sent immediately to the client once it is finished. On the other hand, there are cases in the UTM architecture when the communication between nodes does not need to be synchronous, thus these modules communicate using topics for certain tasks, see Figure 3.5. For example, whenever an operator accepts an alternative flight plan or whenever an authority warns to the UTM about a threat. Moreover, the telemetry received from the UAS is also asynchronous, even though it is sent periodically.

The module USM is the one with more communications using topics, this is due to the need of listening to certain sources on standby until a certain event triggers a process. For example, if a stakeholder warns the UTM of a wildfire, it uses this module, and it is in charge of sending to the Emergency Management module the warning as soon as possible. On the other hand, the USM is constantly listening to the topic *rpa_state_info* where is the information related to all active UAS in the observed airspace. There is one topic for all active UAS, so the module just needs to check one, translate the coordinates from geographic to local, and send the information using the topic *position_report*, with the Tracking module being the receiver. Moreover, when the UTM detects a conflict in the observed airspace, solves it, and proposes to the operator an alternative route, it proposes the new flight plan by using the topic *alternative_flight_plan* and the response of the operator is sent back using the topic *flight_acceptance*.

To summarize, it can be said that the internal communications of the UTM architecture are done using mainly services using a request and response interaction and, on the other hand, the communications between the UTM and the external actors are done using mainly topics using a publisher and subscriber interation.

3.4 Conclusions

This chapter presents the architecture developed for our UAS traffic management, which has a modular and reusable design, does not impose requirements on the execution capabilities of the aircraft, runs at a very high level and is built upon ROS. The UTM architecture can work without a human intervening in the process, aiming to match the final objective of the U-space framework. However, it suggests actions to the UAS operator, due to the current regulation, where the operator is the one who has the responsability to go ahead with the proposed actions of the UTM.

The functionalities implemented by the UTM architecture are compared with the ones implemented by the definition of the U-space ecosystem. In the first place, the proposed UTM architecture does not take into account noncooperative UAS. It is mainly focused on enabling automated decision-making for operation UAS, hence all UAS involved in the system must be cooperative. Despite this, the Tracking module can record and update data in real time reported from different sources, like UAS onboard telemetry, and ADS-B. This information is constantly updated and available for other modules if needed in the Database. In terms of Monitoring, this module does not provide traffic information to UAS operators, however, it accomplishes other expected functionalities, for example, it detects and alerts in real time about conflicts related to multiple UAS loss of separation, flight nonconformances and geofences. Moreover, the Tactical Deconfliction module transmits deconfliction information in real time, it does from the USM to the UAS operator, however, in our scheme an additional step is added. It computes different alternatives and sends them to the Emergency Management module instead of to the USM directly, because the Emergency Management decides automatically which alternative will be proposed to the UAS operator.

Finally, an onboard autonomous DAA module, which is external and independent of the UTM architecture, has been presented. It is in charge of detecting and avoiding temporary static large obstacles like cranes, scaffolding, etc., using an onboard 3D LIDAR to detect the obstacle, and the Lazy Theta* algorithm to avoid it. The DAA module is not intended to detect other non-cooperative aircraft, which is a very complex task that is not solved at the time of writing of the presented Thesis. This is due to the scarcity on the market of on-board sensors for UAS that allow long-range detection of other small aircraft.

Chapter 4

Conflict detection and resolution

This chapter presents a method for multi-UAS conflict management at the tactical level, based on the estimation of 4D UAS trajectories. The proposed solution represents the scenario by means of a 4D grid and uses a geometric approach to resolve conflicts in an iterative manner, minimizing the deviation with respect to the initial estimated trajectory. This method is divided in two main modules of the UTM architecture of the presented Thesis, the Monitoring module, which is in charge of detecting conflicts in the observed airspace, and the Tactical Deconfliction module, which calculates alternative flight plans to avoid the detected conflicts. These modules are previously explained in Section 3.1. Moreover, this chapter details the need to have a 4D trajectory follower onboard UAS to maximize the available VLL airspace.

4.1 Introduction and related work

In terms of U-space, the concept of multi-UAS conflict management becomes a key challenge that is considered at two levels: strategic, which relates to preflight approaches, and in-flight approaches that are called tactical. This Thesis is focused on the tactical phase in which the UAS are flying, i.e., tactical conflict management is addressed as a conjunction of conflict detection and tactical conflict resolution.

At a strategic level, most works focus on studying the probability of conflicts between pairs of UAS, as in Liu and Hwang (2011). A quite relevant work related to strategic multitrajectory conflict detection is Kuenz and Peinecke (2009). They propose the sequential division of the airspace to get subspaces containing an unique trajectory. However, its performance may decrease significantly for too dense scenarios. In Mercado Velasco et al. (2015), the authors incorporate intent information from predefined trajectories to complement a *Velocity Obstacle* approach, but focusing on the 2D problem. On the other hand, at the tactical level, conflict detection is also a well-studied problem in the literature. The conditions to define a potential conflict between two UAS, based on their tracked trajectories, are stated in Alonso-Ayuso et al. (2013). Authors in Yang et al. (2015) pose a hierarchical approach, which reduces the computation time to detect potential imminent conflicts by checking the relative distance between pairs of UAS in an asynchronous way. In Tang et al. (2010), they show how the intended flight plan may enhance the conflict detection in a tactical phase. Another reactive approach, rather spread, is using the concept of velocity obstacles Lalish and Morgansen (2012). These methods search a velocity space to assign the vehicle velocities leading to collision-free trajectories in the future. Another geometric approach based on space-time prisms is proposed in Siqi et al. (2018). In Besada et al. (2020) the authors present a flight planning tool for safe urban operations, which combines both the strategic and tactical levels. It predicts trajectories and provides the flight plan designer with visual data so he can build flights that are compliant with the status of the airspace at the time.

Another related area is multivehicle path finding, where a set of vehicles need to find collision-free paths Wolfgang et al. (2019). Different optimization methods have been applied to solve the problem in continuous Mellinger et al. (2012) or discrete space Yu and Lavalle (2016). Centralized constrained optimization solvers have been proposed. For instance, Turpin et al. (2014) proposed to model the problem as a task allocation and then compute the optimal trajectories in terms of traveled distance. In Karamouzas and Guy (2015), linear programming in a velocity space is used. Moreover, sequential convex programming has been proposed to obtain solutions in non-convex scenarios Alonso-mora et al. (2016). The main issue with these methods for multivehicle path finding is that they do not scale well with the size of the team and hence, they do not allow aerial vehicles to replan online in case of contingency or failures. However, Besada-Portas et al. (2010) proposed a method for multiple UAVs based on evolutionary algorithms for realistic scenarios that presented a good scalability. The fitness of the solutions is evaluated using a method that enables for easy addition of additional optimization indices and modification of the objectives and priorities, which gives this planner its flexibility.

Finally, the concept of 4D-TBO is another relevant issue in the U-space context. It consists in the integration of the temporal dimension into the traditional flight plan, which includes only the intended three dimensions spatial trajectory. Therefore, any delay in the time schedule should be assumed as a separation from the intended trajectory, just as a vertical or horizontal deviation. This concept is especially relevant for U-space tactical services, such as monitoring and tactical deconfliction, where 4D-TBO increases the aerial traffic predictability, maximizing the airspace capacity and improving the overall safety in UAS traffic management. Robust, efficient, and precise autonomous following of predefined trajectories is a requirement of the UAS for all these applications. The trajectory tracking problem for UAS is well studied in the literature, and there are different geometric or control methods.

The rest of the chapter is structured as follows. First, Section 4.2 states the conflict detection, tactical deconfliction, and 4D trajectory tracking problems. In Section 4.3, a conflict detection and resolution system and a 4D trajectory follower are presented. Section 4.4, summarizes the set of simulation results used to validate the proposed approach. Finally, Section 4.5 closes the chapter with the conclusions and future work.

4.2 Problem statement

We address conflict detection and resolution for multi-UAS systems from a tactical point of view and based on 4D trajectories. There are several challenges and complementary issues to take into account to approach this problem. From a tactical point of view, the conflicts can not be detected at the current time t, but predicted with enough time to decide and execute the most proper actions to avoid the potential conflict, for example, following an alternative flight plan. Therefore, to handle the conflict detection problem, it is required to have an estimation of the 4D trajectory of the UAS for a given time horizon.

4.2.1 Conflict detection problem

The UTM architecture proposed in this Thesis is able to detect conflicts in an observed airspace using the Monitoring module, presented in Section 3.1.4. In this section, we will focus on three conflicts that should be detected, loss of separation, geofence conflict, and geofence intrusion. The first one is the only case that involves two UAS, the others are related just to the UAS 4D trajectory and a geofence.

- A **geofence intrusion** event is a conflict between an UAS and a geofence, and it appears when an UAS is suddenly inside a geofence, see Figure 4.1. This can happen if an UAS is doing its operation, and an external stakeholder warns the UTM of a threat, for example, firefighters warn of a wildfire in the zone where the UAS is flying. Geofences can be polygons or cylinders, thus the system should be able to detect if an UAS is inside a geofence no matter the shape of the geofence.

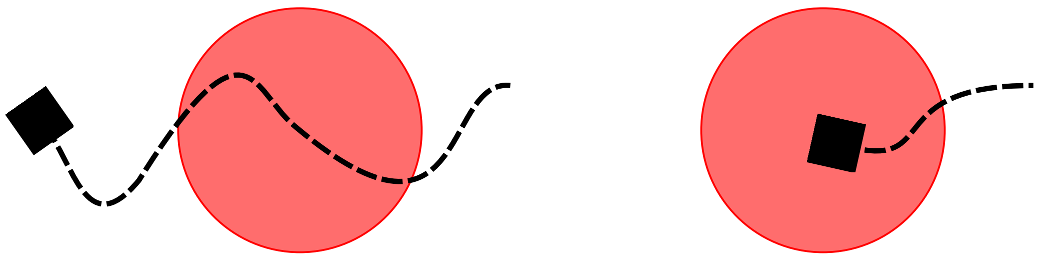


Figure 4.1: Left, geofence conflict. Right, geofence intrusion

- A **geofence conflict** event appears when the estimated 4D trajectory of an UAS is intersecting with a geofence in space and in time, see Figure 4.1. As in the previous conflict, the UTM should be able to detect the conflict no matter the shape of the geofence.

- A **loss of separation** event is a conflict between two UAS, and it is defined as a situation where they approach below their *safety distance*. Depending on the type of vehicle, different separation parameters for the lateral, longitudinal, and vertical dimensions may be considered and different shapes around the vehicle may be considered as their safety zones: an elliptical cylinder, an ellipsoid or a sphere. For the presented methods the cylinder has been chosen in order to be able to use the operational volume concept described in Section 2.1.4. Therefore, the safety distance will have to be at least equal to or greater than the sum of both OVs. A conflict between two UAS will happen at time t if and only if $|p_i(t) - p_j(t)| \leq \delta$, being $p_i(t)$ and $p_j(t)$ the positions at time t of UAS A_i and A_j, respectively, and δ the safety distance parameter. Let assume a 4D trajectory as an ordered list of waypoints, being it defined by an expected three-dimensional position and its estimated arrival time. Let us also consider that every pair of consecutive waypoints are equally time-spaced by the same *inter-waypoint period* τ for any trajectory. Therefore, a potential conflict between a pair of trajectories will happen if there exist a pair of waypoints of both trajectories with a time difference below the inter-waypoint period and distance below the *safety distance* δ, see Figure 4.2.

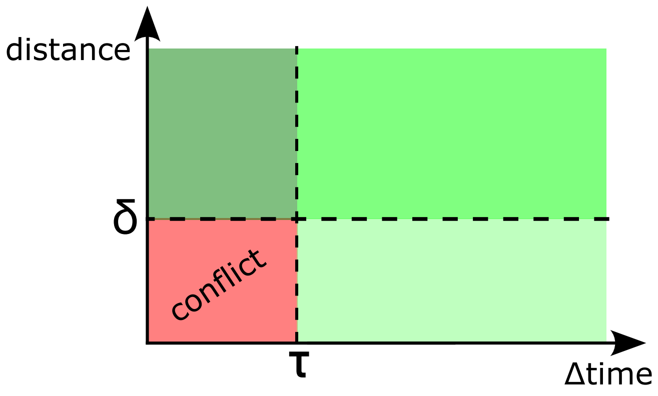


Figure 4.2: Zones where a pair of waypoints of two different trajectories can be free of conflict. The red area is the only one in which a pair of waypoints have a time difference below the inter-waypoint period τ and a distance below the safety distance δ. The dark green area represents a conflicting time between waypoints, but it is a safe area to fly because there is enough distance. On the other hand, the light green area is a safe area to fly due to enough time between waypoints despide the conflicting distance.

4.2.2 Tactical deconfliction problem

Let us consider a set of 4D trajectories for UAS sharing a common airspace defined within a given time horizon, see Figure 4.3. The conflict detection and resolution problem implies not only to detect potential conflicts according to the definitions of the trajectories presented in Section 4.2.1, but also to propose an alternative set of 4D trajectories which ensure no potential conflicts among them.

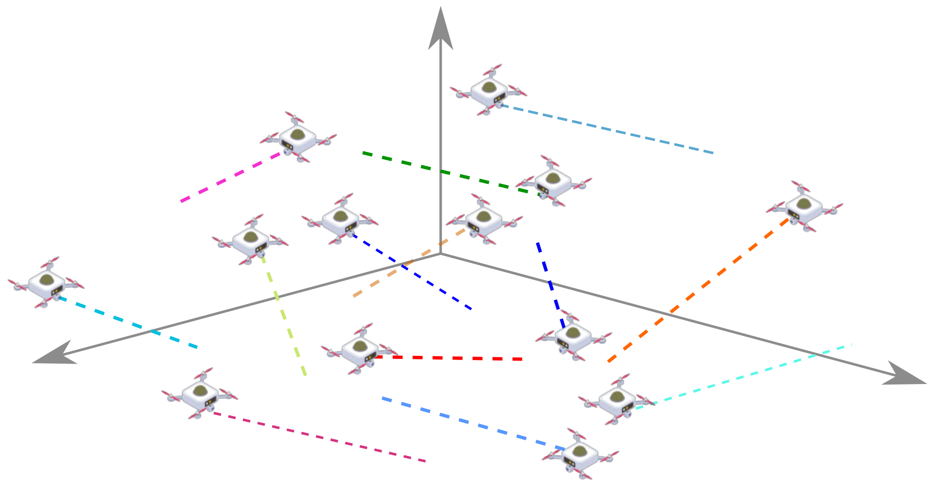


Figure 4.3: Estimated trajectories within a given time horizon of several UAS sharing a common airspace. Each estimated trajectory is marked with a different color.

There are infinite possible solutions to the posed problem. To assess the relevant solutions, it will be assumed that the initial trajectories are the reference ones from an operational point of view, since they come from the flight plans provided by the UAS operators. Therefore, the mean distance deviation from the initial trajectories to the alternative ones will be chosen as the minimization criteria. This distance deviation may be calculated as the three-dimensional distance between a pair of waypoints with the same estimated arrival time from the alternative and initial trajectories.

It is interesting to take into account the priorities associated with the different UAS to solve the problem. A potential conflict between two UAS with different priorities should be solved modifying only the trajectory of the UAS with lower priority. Therefore, the optimization criteria should consider weighting the trajectory deviations according to these priorities.

On the other hand, since this problem should be solved in-flight and the alternative trajectories have to be submitted, accepted, and executed by the UAS, the required processing time to obtain a suitable solution has to be minimized.

4.2.3 4D trajectory tracking problem

The problem described in Section 4.2.2 is based on the accurate tracking of 4D trajectories by the involved UAS: each UAS has to follow its assigned waypoints matching each associated arrival time. Delays or advances with respect to the estimated arrival time may cause unexpected conflicts and require to update the estimation of the trajectories, to detect new conflicts and to generate alternative trajectories continuously. However, commercial autonomous navigation systems for UAS do not usually manage 4D trajectories and set a cruise flight speed to follow a given three-dimensional path. The 4D trajectory could hardly be tracked using these systems, even when the arrival times associated with each waypoint had been properly chosen according to the UAS cruise speed. Moreover, matching the specified arrival times manually is not an easy task that can do every single UAS pilot.

Let us assume that an UAS can be directly controlled through three-dimensional velocity commands. Known the intended 4D trajectory of the UAS, the objective is to implement a system which tracks accurately the positions and the associated arrival times based on velocity commands. The maximum flight speed should be also considered for the UAS.

Formally, the criteria to minimize are two: the mean minimum distance between the actual travelled trajectory and the estimated one; and the mean difference between the actual and the estimated arrival times of every waypoint. The 4D trajectory follower is detailed in Chapter 5.

4.3 Solution adopted

The proposed solution is based on the assumption that the updated versions of the estimated 4D trajectories for all UAS which share the controlled airspace are continuously available within a given time horizon T, and this task is done by the Tracking module detailed in Section 3.1.3. These trajectories are defined as an ordered set of 4D waypoints, equally time-spaced by τ seconds.

Conflict detection and resolution modules are executed on the UTM, which receives information from all UAS. It detects potential conflict events between UAS and provides alternative plans to avoid them. On the other hand, for each UAS, its on-board computer executes a trajectory following algorithm. It receives the alternative plans provided by the UTM and commands the UAS to match the 4D trajectories as closely as possible, not only in space, but also in time. Figure 4.4 shows this software architecture.

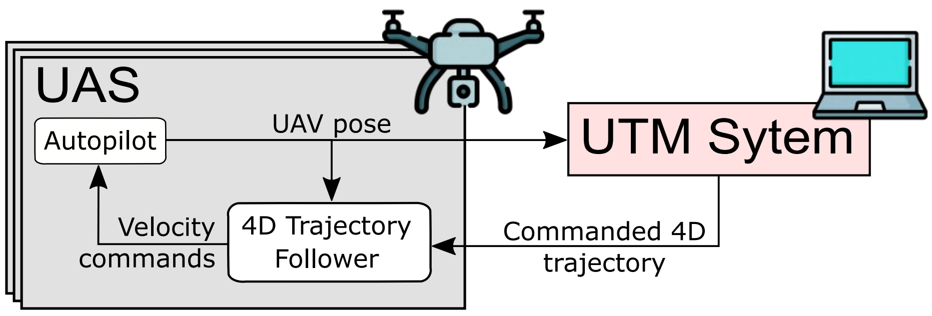


Figure 4.4: Interactions between UAS and UTM modules. The UTM sytem is out of the scope of this chapter, details shown in Figure 3.1.

The solution adopted is based on two interconnected services: monitoring and tactical deconfliction. The alternative flight plans generated by the deconfliction service will be executed by the UAS and registered by the UTM. Therefore, if the new trajectories cause new conflicts, they will be reported again by monitoring to the deconfliction node, generating an iterative process.

4.3.1 Monitoring module

In terms of detecting conflicts, taking a geofence into account, the shape of the geofence is relevant information that should be considered. The 3D volume of the geofence and its activation time are taken into account, as well as the 4D volumes of the operations.

Geofence intrusion and geofence conflict

To detect a geofence intrusion event, Monitoring checks the shape of the geofence. If the geofence is cylindrical, the distance of the given waypoint to the cylinder center is computed and compared with the geofence radius. If it is defined by a polygonal shape, the signed angle method is applied, it computes the sum of the angles between the segments that connect the observed waypoint and each pair of points in the polygon. The waypoint is within the polygon if this sum is 360°, while if the sum is 0° it is outside, see Figure 4.5. For both shapes, the waypoint to check is the current position $p_i(t)$ at the current time t.

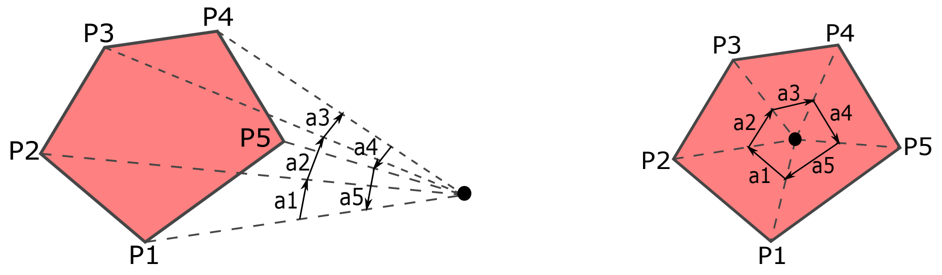


Figure 4.5: The signed angle method is used to evaluate whether a tested waypoint (black dot) is inside or outside a polygonal geofence. Left, an example where the angles of an external waypoint sum up to 0°. Right, an interior waypoint whose angles sum up to 360°.

This case can appear if an UAS is monitoring a field for agriculture and an external stakeholder, like the firefighters, warns of a wildfire in the same zone that the UAS is having its operation. This conflict is called geofence intrusion and if it is matched, further checks are not necessary to avoid extra computational time.

Another possible threat related to geofences is geofence conflict. Monitoring checks if the estimated trajectory of the observed UAS intersects with an active geofence. As said before, an estimated 4D trajectory is defined as an ordered set of 4D waypoints, equally time-spaced by τ seconds and limited by a time horizon to lightweight the computation. In this case, the Monitoring module checks if any waypoint of the estimated 4D trajectory is inside an active geofence, using the same method explained in the geofence intrusion event.

Loss of separation

The event loss of separation involves two UAS at the same time. The method to detect this conflict is based on modeling the considered scenario as a 4D grid. The whole controlled airspace within a time horizon T is divided into 4D cells of size $dX \times dY \times dZ \times \tau$, being τ length associated to the time dimension, see Figure 4.6. Each cell stores a list with the waypoints from the received set of 4D trajectories which are in its associated 4D space.

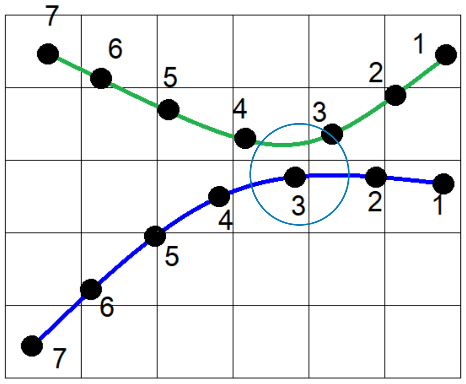


Figure 4.6: Airspace shared by two trajectories divided into a grid. The altitude is not shown for the sake of clarity. Black circles represent the waypoints and the associated number its arrival time.

Periodically, each τ seconds, the monitoring service updates the 4D grid using the estimated trajectories of the UAS. Conflict evaluation is made only between waypoints which are in the same or neighboring cells. Therefore, when an updated waypoint is stored into its associated cell, the neighboring cells are checked to find potential conflicts, see Figure 4.7. This approach reduces dramatically the total number of required checks with respect to other methods based on an exhaustive search approach, especially for large-scale scenarios.

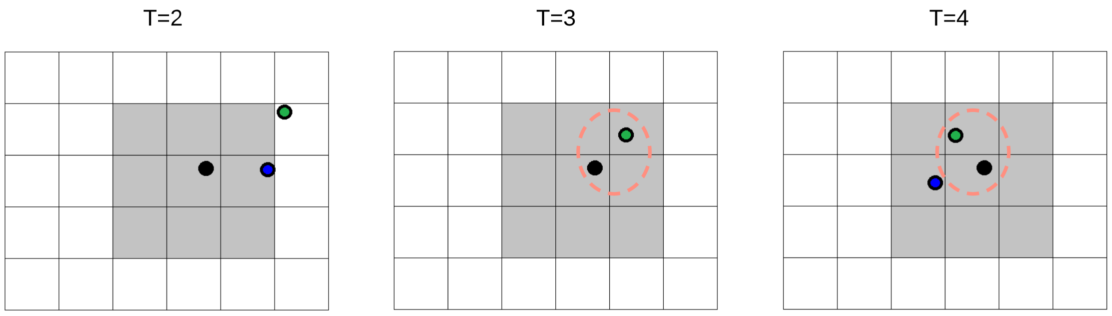


Figure 4.7: Conflict detection process associated to the third waypoint from the blue trajectory in the scenario represented in Figure 4.6. This waypoint is stored into its cell and its neighboring cells (dashed cells in the figure) are checked to look for waypoints from other trajectories (green trajectory, in this case). The analyzed waypoint has to be validated against the third and four waypoints from the green trajectory, as it is shown in the example.

After each iteration, the monitoring service provides a list of potential conflicts between pairs of waypoints of different trajectories, including their information.

Let us consider a controlled airspace of size $dX \times dY \times dZ$ and a time horizon of T seconds. Then, the dimensions of the required grid would be $\lceil X/dX \rceil \times \lceil Y/dY \rceil \times \lceil Z/dZ \rceil \times \lceil T/\tau \rceil$, being $\lceil \rceil$ defined as the ceiling function.

Another issue to consider is the lower limit of the cell size. On the one hand, since conflicts are checked only between neighboring cells, the size of the cells cannot be less than the safety distance δ. In the other case, a potential conflict could exist between two waypoints which are not in the neighboring cells. Therefore, $dX, dY, dZ \geq \delta$.

Moreover, the cell size has to be bounded down by the speed of the involved UAS. The UAS cannot fly fast enough to go through two cells in a single period τ because the potential loss of separation events could be ignored. Therefore, defining $v_{\max}$ as the maximum speed of the fastest UAS involved in the scenario, the size of the cell cannot be less than $\tau v_{\max}$: $dX, dY, dZ \geq \tau v_{\max}$.

The relation between the safety distance and the 4D cell dimensions influences the system efficiency. On the one hand, the lower the value of the safety distance, the larger the 4D grid size, the memory requirements, and time to create the grid, but lower processing time since the number of checks between waypoints in neighboring cells will be minimized. On the other hand, the larger these dimensions are, the conflict detection algorithm will look more like an exhaustive search.

Algorithm 1 summarizes the Monitoring module. It shows how the module clears the 4D matrix of the previous loop and gets all operations and geofences calling the Database services, being just the active operations the ones to be checked. Each waypoint of every active operation is checked for conflicts related to geofences, first geofence intrusion, and then geofence conflict. Both types can be detected for a single trajectory, however, it is up to other modules to decide which conflict is given the highest priority for resolution. The waypoint that is being checked is stored in a 4D matrix, and after that, the module checks this waypoint with the previously stored in the matrix for conflicts related to loss of separation, as well as with its neighbours. When a conflict is detected, the module creates a threat with the following related information: the identification of the UAS involved, the type of the threat, and the waypoints in conflict. The result is a list of potential conflicts that must be solved by other modules. Before notifying the list, it is filtered by the Monitoring module to send only no repeated threats. The method *manageThreatList* is in charge of filtering, checking if a threat has been previously notified and, if a notified threat is not solved after a certain time, notify it again. This is crucial to keep the communications between modules as light as possible. If we had not introduced this filter, Monitoring would notify the conflict uninterruptedly, which could collapse communications in the case of a larger scale scenario. Moreover, this is not a problem, we need to have a margin time for the operator to see the warning, analyze the alternative flight plan proposed, and decide to accept or decline it. When the operator accepts the new route, the Database updates it and Monitoring will not detect the same conflict more times, thus the potential conflict is removed from the threat list.

Algorithm 1: Monitoring pseudocode. Each waypoint of every operation is checked for conflicts related to geofences, it is stored in a 4D matrix, and is checked for loss of separation with the previously stored waypoints. The Monitoring module is detecting the potential conflicts every iteration until the conflict is solved due to the acceptance of an alternative flight plan by the operator involved in the conflict. However, there is a method in charge of filtering the conflicts detected to send them just once, avoiding multiple notifications per conflict.

```
while utm.ok() do
    matrix4d.clear()
    operations = readOperations.call()
    geofences = readGeofences.call()
    for op ∈ operations do
        if op.started then
            for wp ∈ op.estimated_trajectory.waypoints do
                if insideGeofence(wp, geofences) then
                    threat = createThreat(wp, op.uav_id, geofence_intrusion)
                    threat_list.push_back(threat)
                else if intersecGeofence(wp, geofences) then
                    threat = createThreat(wp, op.uav_id, geofence_conflict)
                    threat_list.push_back(threat)
                end
                matrix4d.store(wp)
                for m_wp ∈ matrix4d do
                    if checkDistance(m_wp, wp) > safety_distance then
                        threat = createThreat(wp, op.uav_id, loss_separation)
                        threat_list.push_back(threat)
                    else if checkNeighbours(m_wp, wp) > safety_distance
                     then
                        threat = createThreat(wp, op.uav_id, loss_separation)
                        threat_list.push_back(threat)
                    end
                end
            end
        end
    end
    list_to_notify = manageThreatList(threat_list)
    threats_client.call(list_to_notify)
end
```

4.3.2 Tactical deconfliction module

The Tactical Deconfliction module is in charge of calculating different alternative routes for a given conflict. The generated alternative flight plans are sent with two relevant parameters: the cost, which is determined by the total distance travelled by the UAS, and the riskiness, which is determined by the minimum distance between the alternative route and any geofence, or, in case that the solution goes through any geofence partially, by the length of the route portions that remain within a geofence. These parameters are needed by the Emergency Management module to determine which alternative flight plan should be proposed to the operator of the UAS involved in the conflict.

It should be noticed that the proposed approach requires that UAS can adapt their flight velocities to properly follow the generated trajectories. Although the initial flight plans could have been chosen to fly at a single nominal speed, alternative trajectories do not respect this principle. For example, an UAS has a flight plan of four waypoints P_1, P_2, P_3, P_4 with its corresponding times t_1, t_2, t_3, t_4 and a conflict is detected in the third waypoint (P_3, t_3). The Tactical Deconfliction module modifies the position of P_3 to avoid the conflict, but t_3 remains the same, therefore the distances P_3P_2 and P_4P_3 are larger than initially and the UAS should cover more distance in the same time.

Loss of separation

In terms of multi-UAS conflict, the algorithm proposed is based on a geometric approach to solve sequentially each conflict between pairs of waypoints. The idea is separating each pair of conflicting waypoints independently and taking advantage of the periodic checks performed by the monitoring service to solve iteratively more complex situations, for example, with more than two UAS involved.

For each conflict received, it computes a new pair of waypoints separated enough to match the safety distance, as it is shown in Figure 4.8. These new waypoints are calculated modifying the three-dimensional position, but not the arrival time, generating an estimated variation of the nominal speed of the UAS with respect to the original one. Then, they are fused with the associated trajectories and reported to the corresponding UAS.

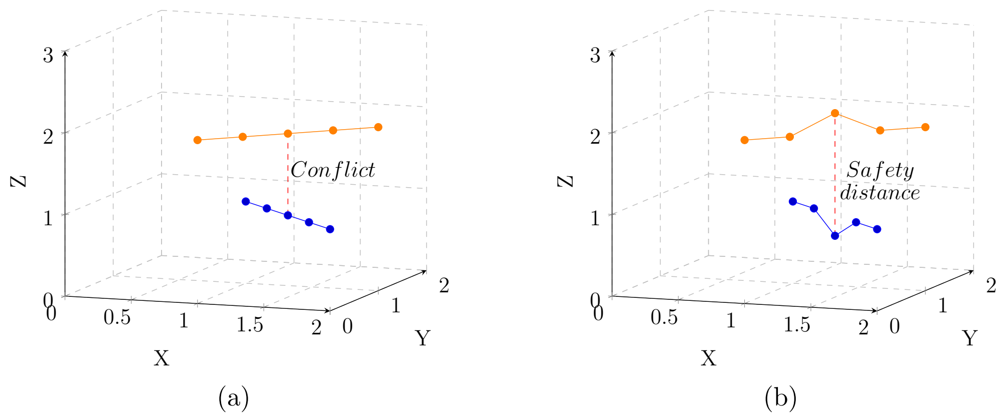


Figure 4.8: On the left two trajectories with a pair of waypoints in conflict. On the right, the alternative trajectories provided by separating the conflicting waypoint to match the safety distance.

To calculate the new waypoints, different issues have to be considered. First, UAS operations may be assigned with different priorities. We keep without variation the trajectories associated with UAS with higher priority, proposing alternative trajectories for the UAS with lower priority.

In addition, waypoint separation may be performed following different directions: the direction which joins both waypoints, vertically or horizontally. The first option seems to be more efficient from an airspace capacity point of view and according to the conditions specified in Section 4.2.2. However, the latter options are safer since they are more predictable for the rest of manned and unmanned aircraft.

Despite each conflict being solved independently, not considering the rest of the trajectories, the monitoring service runs periodically and generates an iterative process which was illustrated in Figure 3.3.

Finally, it has to be checked if the UAS can follow the alternative trajectory, matching the arrival time requirements according to its speed capabilities. In summary, this one may be assured always the required cruise speed of the UAS to perform the initial plan is lower than the half of its maximum speed. In case this condition is not met, different strategies may be adopted: modifying the previous or next waypoint to the conflicting one; or assigning a different separation weight to each conflicting waypoint depending on the difference between the original cruise speed and the maximum speed.

Geofence intrusion and geofence conflict

There are two possible cases if an UAS has a conflict with a geofence. The first case appears when an UAS flight plan goes through a geofence, here the Tactical Deconfliction computes an alternative flight plan around the geofence to avoid it, see left Figure 4.9. On the other hand, if the UAS is already within a geofence, it gets out of the geofence through the closest point and then it goes around the geofence to resume its flight plan afterwards, see right Figure 4.9. When this module needs to compute an alternative flight plan to go around a geofence, it uses a heuristic path planner based on the well-known A* algorithm. This path planner has been developed by our research group, is public[1] and has been used in several applications like in autonomous planning for multiple aerial cinematographers Caraballo et al. (2020).

4.3.3 4D trajectory follower based on the carrot chasing algorithm

The 4D trajectory tracking problem is solved with a method based on the carrot chasing algorithm. It is composed by two algorithms: the generator and the follower. This 4D trajectory follower has been developed in this Thesis, and it is detailed in Chapter 5.

The generator algorithm creates a much more dense list of waypoints based on the ordered list of waypoints received and it can be approximated to a continuous curve. To follow a 4D trajectory, it should interpolate the initial list of times matching the number of waypoints of the more dense list.

[1] `github.com/Angel-M-Montes/path_planner`

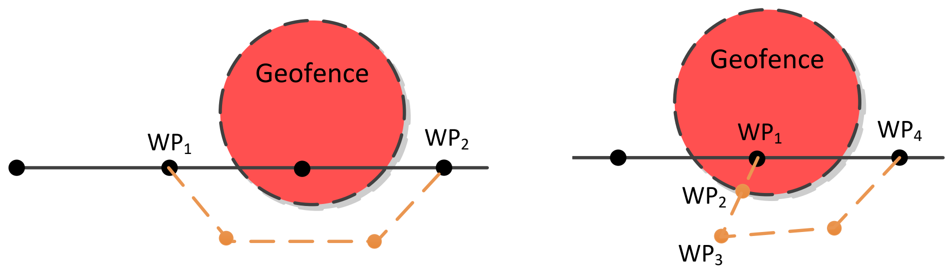


Figure 4.9: Left, an UAS flight plan intersecting a geofence. The last waypoint of its flight plan before entering the geofence (WP_1) and the first waypoint after leaving it (WP_2) are obtained, and this segment of the flight plan is replaced by the alternative route (dashed orange line). Right, an UAS inside a geofence. The escape point (WP_2) is that on the geofence's border closest to the UAS (WP_1). From WP_3 to the first point of the flight plan after leaving the geofence (WP_4), an alternative route avoiding the geofence is inserted to modify the original flight plan and respecting at every moment the safety distance.

4.4 Validation results

This section details the different simulation tests carried out in order to validate the approaches proposed in this chapter. Large scale simulations based on MATLAB are provided to analyze the scalability of the conflict detection and resolution approach, and ROS-Gazebo based tests are performed to validate the entire system integration in operation.

4.4.1 Scalability analysis

Time-based algorithms are a traditional solution for ATM conflicts. Thus, the proposed solution is compared with a conflict resolution approach based on the recursive time shift algorithm Peinecke and Kuenz (2017). It proposes to shift the trajectories in time, starting at conflicting waypoints, to find the successful trajectories. Regarding the conflict detection approach, a detailed comparison with respect to the traditional exhaustive search approach was presented in Acevedo et al. (2019).

MATLAB simulations for large-scale scenarios have been carried out on an Intel i7@2.2GHz, to analyze the scalability of the developed conflict detection and resolution system.

The first element which can influence the optimization criteria described in Section 4.2.2 is the number of involved UAS. Therefore, a test battery increasing the number of UAVs and setting the time horizon to 100 seconds and the 4D cell dimensions to $100 \times 100 \times 100$ meters was carried out. Figure 4.10 shows the processing time and the mean deviation with respect to the original trajectories using the proposed and alternative approaches.

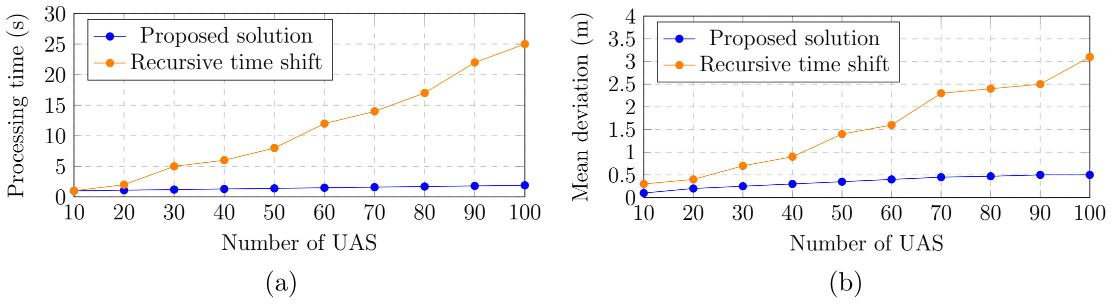
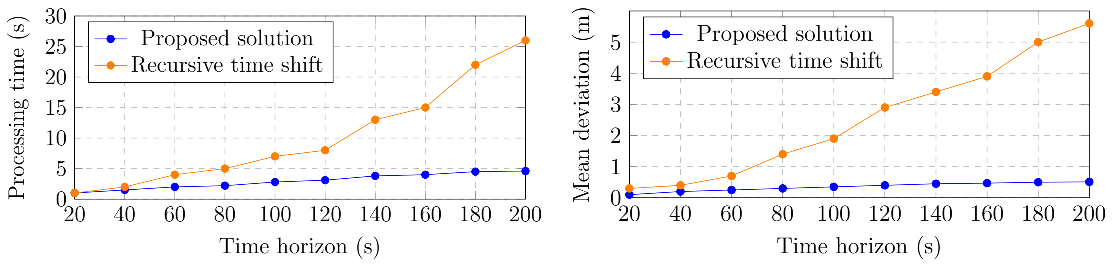


Figure 4.10: Summary of the test battery increasing the number of UAS and comparing the proposed solution against the time shift based approach:
(a) Processing time by test to detect and resolve all the conflicts.
(b) Mean deviation by waypoint between the generated and the original trajectories.

The time horizon is another interesting element whose impact on the system efficiency should be analyzed. A large time horizon may be necessary to match the U-space requirements, allowing the UAS operators to receive, analyze, and execute proper commands to avoid potential conflicts. Therefore, a second test battery increasing the time horizon and setting the number of involved UAS to 30 and the 4D cell dimensions to $100 \times 100 \times 100$ meters has been performed. Processing time and mean deviation with respect to the original trajectories are shown in Figure 4.11.

The results show that the proposed solution requires much less processing time and gets much less deviation with respect to the original trajectories than the more traditional recursive time shift algorithm. These improvements are highly related to



(a) (b)

Figure 4.11: Summary of the test battery increasing the time horizon and comparing the proposed solution against the time shift based approach:
(a) Processing time by test to detect and resolve all the conflicts.
(b) Mean deviation by waypoint between the generated and the original trajectories.

the number of involved trajectories and the considered time horizon. The proposed solution exploits the assumption of following accurately the trajectory both in time and space to modify only the conflicting waypoints. However, shifting a trajectory in time implies to modify every waypoint in the trajectory, generating new conflicts which have to be detected and solved again.

4.4.2 Multi-UAS tests for loss of separation

The proposed solution was tested on ROS Quigley et al. (2009) Kinetic under the Ubuntu 16.04 operating system using GAZEBO Koenig and Howard (2004), the PX4 v1.7.3 SITL Meier et al. (2015) functionality to simulate the autopilot and the UAS Abstraction Layer (UAL) v3.0 Real et al. (2020) to interact with the simulated UAS. The simulation environment allows running the same software used in a real UAS, with the advantage of not needing pilots or permissions. In addition, it supports testing the same real scenario quickly and as many times as the developer wants. The previous section presented a scalability analysis running tests involving 100 UAS, this section is focused on the behaviour of UAS on the simulation framework, thus the tests use 3 UAS to clarify the results.

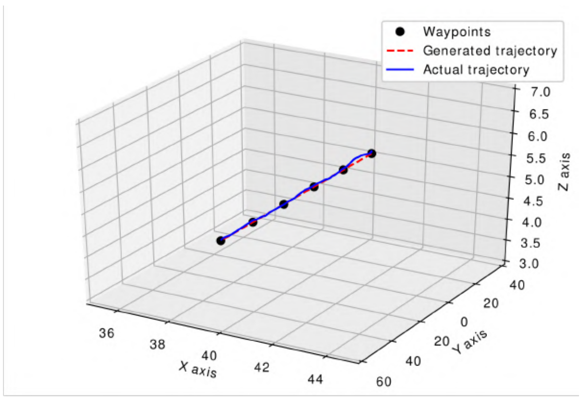
The following tests use multiple UAS with flight plans that have the same characteristics, such as the longitude (100 meters), the arrival time of the last waypoint (37.5 seconds), and cruising speed (2.7 m/s), however they differ in the origin and arrival poses. The conflict detection and resolution system has a safety distance δ of 10 meters and an inter-waypoint distance τ of 5 seconds. For each multi-UAS test, two plots are provided: 3D visualization of the UAS travelling along their flight plans and the distance between the UAS involved in the test.

Figure 4.12 shows an example of these flight plans in detail, with a trajectory that can be followed by the simulated UAS which has a maximum velocity of 4 m/s. Figure 4.12a has different scales on the axes to better visualize the error between trajectories. Moreover, how the trajectory follower caps the commanded velocity to the maximum velocity in the first seconds is shown in Figure 4.12b. Figure 4.12d shows the difference between the UAS actual time and the reference time: if it is negative, the UAS is behind of the schedule, otherwise the UAS is ahead of schedule. The 4D trajectory follower always tries to make the time difference equal to zero. It also tries to minimize the normal distance between the UAS and the trajectory. The space and time errors that the 4D trajectory follower tries to minimize are shown in Table 4.1.

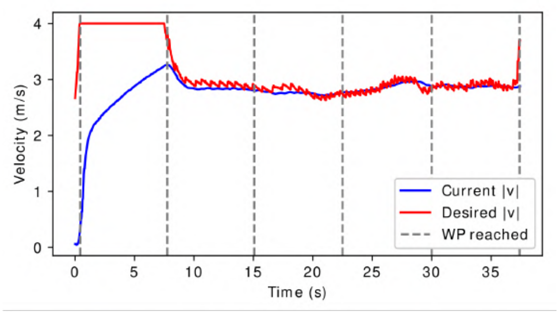
Table 4.1: Simulated test errors

Error	**Min**	**Mean**	**Max**	**Std**	**Var**
Space (m)	0.002	0.036	0.129	0.030	0.001
Time (s)	0.000	0.196	1.047	0.311	0.097

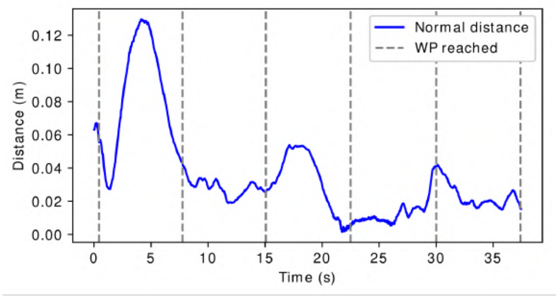
The first test presents a conflict after 15 seconds between two UAS that start their flight plans at the same time. UAS 0 and 1 have priorities 0 and 1, respectively. The conflict detection and resolution module decides to leave one flight plan as it is and modifies the other flight plan due to the UAS priorities to solve the conflict. Despite the increment of the flight plan longitude, the UAS manages to fly through the trajectory in time because it uses the 4D trajectory follower, which minimizes its time and space errors. UAS 0 finishes its flight in 37.46 seconds even modifying its flight plan, and UAS 1 finishes in 37.51 seconds. Figure 4.13b shows the distance



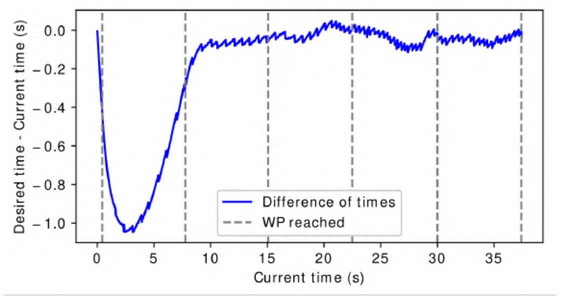
(a)



(b)



(c)

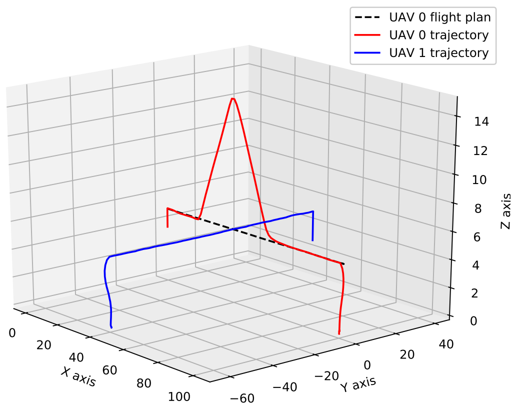


(d)

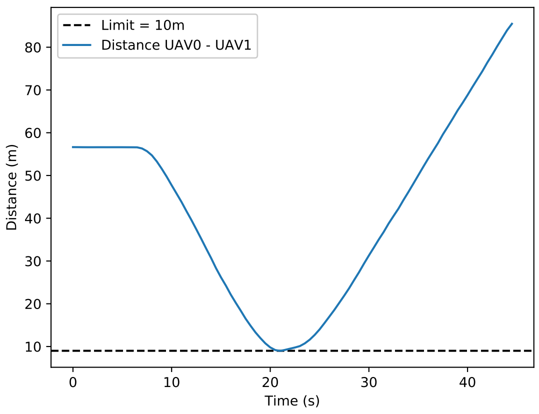
Figure 4.12: Simulation results. (a) Three dimensional view of the initial waypoints, the reference trajectory generated and the actual trajectory flown. (b) reference velocity and current velocity of the UAS. (c) Normal distance between the UAS and the generated trajectory. (d) Difference between the reference time and the current time.

between UAS which never goes below the limit which is 10 meters. A 3D visualization of the trajectories can be seen in Figure 4.13a.

The second test presents three UAS flying with flight plans similar to the ones used in the previous simulations except for UAS 2, which has a flight plan that lasts 7.5 seconds more. The conflict between UAS 1 and UAS 2 is found in the same position and time that happened in the previous test. As the priorities do not change, the conflict detection and resolution module calculates the same solution. The solution of the conflict between UAS 0 and UAS 1 does not create another conflict between UAS 0 and 2, and they fly through the same waypoint at different times, so the



(a)



(b)

Figure 4.13: (a) 3D Visualization of the traveled trajectories. (b) Distances between UAS. UAS 0 has a conflict with UAS 1 but the solution of the conflict detection and resolution module keeps the distance above the limit.

minimum distance between UAS is not violated, UAS 0 goes through the waypoint at 30 seconds, and UAS 2 at 37.5 seconds. Figure 4.14b shows distances between the UAS which never goes below the limit. UAS 0 finishes its flight in 38.14 seconds even modifying its flight plan, UAS 1 finishes in 37.44 seconds, and UAS 2 in 44.94 seconds. A 3D visualization of the trajectories can be seen in Figure 4.14a and a video of the simulation is publicly available[2].

The third test presents the relevance of using a 4D trajectory follower. The conflict detection and resolution module gives its solution to the UAS 0 which follows the new trajectory with a cruising speed of 2.7 m/s to match the mean of the velocity of the last test. The new trajectory is longer than the initial flight plan, so while going along the modified trajectory to avoid UAS 1, UAS 0 is delayed. This causes a conflict between UAS 0 and 2 where their flight plans cross at 37.5 seconds, which did not appear in the previous test and the distance between UAS 0 and 2 goes below the limit, see Figure 4.15b. The new trajectory causes a new conflict, it will be reported again from the monitoring to the deconfliction node, generating an iterative process

[2]`https://youtu.be/0U42krj1MTM`

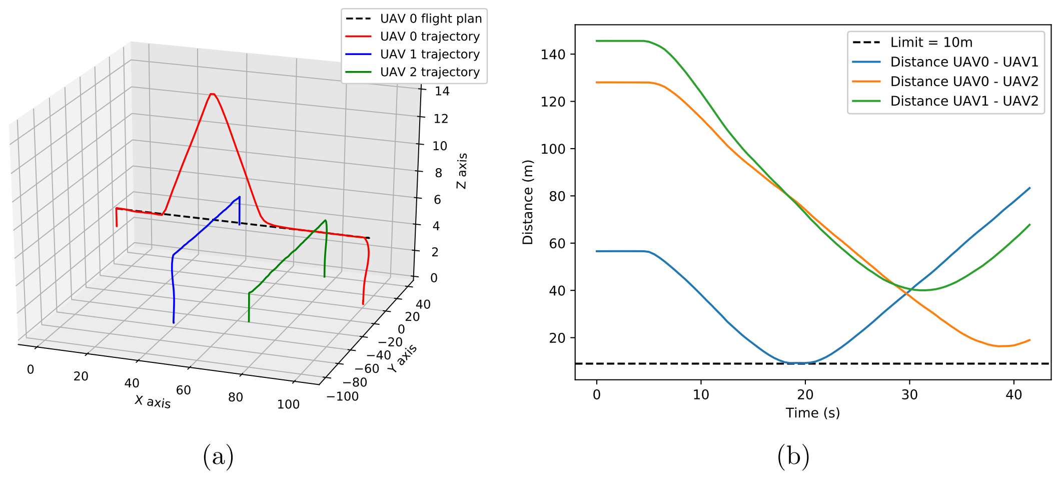


Figure 4.14: (a) 3D Visualization of the traveled trajectories. (b) Distances between UAS. UAS 0 has a conflict with UAS 1 but the solution of the conflict detection and resolution module keeps the distance above the limit. UAS 0 is using the 4D trajectory follower.

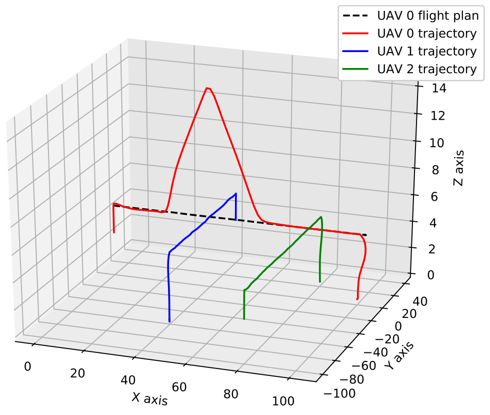
to solve the conflict, but in this test, the conflict will not be solved to emphasize that the distance between UAS goes below the limit. A 3D visualization of the trajectories can be seen in Figure 4.15a and a video of the simulation is publicly available[3].

4.5 Conclusions

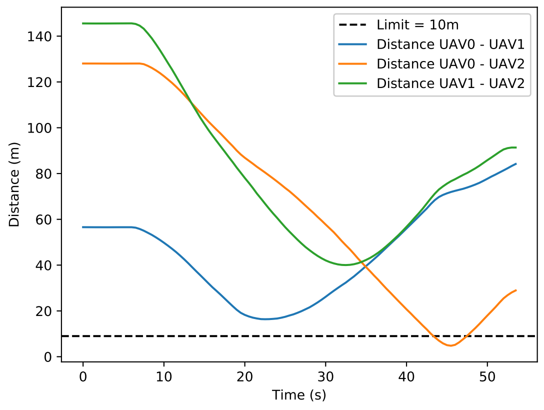
The use of UAS will be increased in the next decades, being the ATM much more complex. In this context, the concept of 4D trajectory becomes a key element to increase the flight safety and to maximize the shared airspace usage. This chapter faces this challenge, presenting two modules for conflict detection and conflict resolution.

The presented conflict detection method tries to optimize the search of conflicts, limiting it to a local search around each waypoint. Regarding the conflict resolution method, it tries to minimize the deviation distance with respect to the initial trajectory, splitting the whole problem into several simpler subproblems with just two

[3]`https://youtu.be/8oKwk7tL-dI`



(a)



(b)

Figure 4.15: (a) 3D Visualization of the traveled trajectories. (b) Distances between UAS. UAS 0 has a conflict with UAS 1 but the solution of the conflict detection and resolution module keeps the distance above the limit. UAS 0 is not using the 4D trajectory follower, and it causes a conflict with UAS 2, lowering their distance below the limit.

waypoints. The validation results demonstrate that the conflict detection method improves the processing time with respect to other optimized methods based on exhaustive search, and this improvement is strongly dependent on the number of trajectories and the time horizon considered. Moreover, the iterative geometric resolution method has proven to achieve much better performance than other resolution methods based on the traditional time shift approach, both in terms of processing time and deviation from the initial trajectories.

The proposed approach to detect and solve the loss of separation events takes advantage of the assumption that UAS may track accurately the trajectories in time and space. Based on this, the resolution approach only modifies the conflicting waypoint positions, not their arrival times, requiring the velocity adaptation of the UAS. The developed UAS 4D trajectory follower can fix the space and time errors while tracking a given trajectory. It minimizes at every time the mean minimum distance between the actual travelled trajectory and the estimated one, and the mean difference between the actual and estimated arrival times to every waypoint.

Chapter 5

4D Trajectory Based Operation Follower

In this chapter, a four-dimensional trajectory follower implementation based on the carrot chasing algorithm is presented. It can be used based on a list of parameters to generate and follow a 4D trajectory. It has been developed as an extension of the path follower presented in Perez-Leon et al. (2020b) and it has been integrated with the UAS Abstraction Layer[1] Real et al. (2018) previously developed by our research group.

5.1 Introduction and related work

As it has been previously presented, an interesting concept developed by SESAR, initially oriented to manned commercial aviation, but also considered in the U-space framework, is the 4D-TBO. Including time as the fourth dimension in the flight plan definition optimizes the airspace occupancy, mainly related to the conflict resolution U-space services CORUS (2019). Consequently, developing accurate and precise 4D trajectory control methods becomes a very relevant challenge to allow the development of the 4D-TBO in the U-space, allowing the integration of an increasing number of UAS in the civil airspace.

[1] `https://github.com/grvcTeam/grvc-ual`

A common requirement for all these applications is the precise, robust, and efficient autonomous tracking of predefined paths by aerial robots. The trajectory following problem for UAS is well studied in the literature, and there are different control-based or geometric methods. Some common geometric algorithms are pure pursuit Coulter (1992), carrot chasing Micaelli and Samson (1993), line-of-sight Fossen et al. (2003) methods, and vector field Nelson et al. (2007).

Sujit et al. (2013) compared path following algorithms that are easy to implement, take less implementation time, and are robust to disturbances in straight lines and loiter paths. Vector field algorithms are more accurate than the other two-dimensional path following algorithms presented in the paper, and carrot chasing algorithms have the worst performance due to wind disturbances. Nuñez et al. (2015) took into account the wind gusts as they play a key role in small prototypes to fix wind disturbances. Xavier et al. (2018) demonstrated that vector field algorithms have the largest errors than carrot chasing and pure line-of-sight Kothari et al. (2010) methods for loitering paths with and without wind disturbances taking into account a three-dimensional space.

Several methods have been proposed in the literature that take into account four-dimensional space, such as Farid et al. (2018) who presented a comprehensive formulation for generating various optimal and constrained trajectories. Sarabakha and Kayacan (2019) presented an approach for a high-level control of UAS that improves online trajectory following performance by using deep learning. The exact model of the system to be controlled is not required, and it is robust against operational uncertainties as well as variations in system dynamics.

The rest of the chapter is structured as follows. First, the concept of 4D-TBO is addressed in Section 5.2. Based on this definition, Section 5.3 states the trajectory following the problem based on 4D trajectories. In Section 5.4, a four-dimensional trajectory follower is proposed. The set of simulation and experimental results are summarized in Section 5.5 to validate the proposed approach. Finally, Section 5.6 closes the chapter with the conclusions and related future work.

5.2 4D Trajectory-Based-Operations

The concept of 4D-TBO is another relevant issue which development was studied by SESAR in Europe and NextGen in the United States Enea and Porretta (2012). Basically, it consists in the integration of the temporal dimension into the traditional flight plans, which include only the intended three dimensions spatial trajectory. The aerial systems must not only follow a predefined path, but also accomplish a time schedule. Therefore, any delay in the time schedule should be assumed as a separation from the intended trajectory, just as a vertical or horizontal deviation.

In practice, the 4D-TBO may be defined as a list of waypoints, including the intended location plus the intended arrival time to this location. It means a curve of four dimensions composed by three spatial dimensions plus time. This new definition is especially relevant for U-space tactical services, such as monitoring and tactical deconfliction. Moreover, it will allow to optimize the use of the airspace, since a flight plan should not to lock the whole 3D-volume during the time operation. Thus, the 4D-TBO will increase the aerial traffic predictability, maximizing the airspace capacity and improving the overall safety in aerial traffic management.

However, this concept implies that unmanned aerial systems should perform a very accurate trajectory following control, considering the time schedule Ramasamy et al. (2014). In this chapter, the flight plan assigned to each UAS is assumed as a 4D trajectory, which should be followed accurately to minimize uncertainties about its future location to facilitate the UAS traffic management.

5.3 Problem statement

This chapter poses the trajectory following problem for velocity-controlled UAS. An UAS Q, which current position is defined by $\mathbf{p}(t) \in \mathbb{R}^3$ at any time t, has to follow a 4D trajectory Γ of length L. The 4D trajectory is defined by two functions:

- A curve in the space $\gamma(\lambda) \in \mathbb{R}^3$, with $\lambda \in [0, L]$ to define the intended UAS location in the space.

- A function of times $\tau(\lambda) \in \mathbb{R}$, with $\lambda \in [0, L]$, to define the intended arrival times to the associated locations in $\gamma(\lambda)$.

Let us assume that Q is holonomic and velocity-controlled, being its velocity defined as $\mathbf{v}(t)$ at any time t. Then, the UAS motion is controlled via velocity commands, such that $\frac{\mathrm{d}\mathbf{p}(t)}{\mathrm{d}t} = \mathbf{v}(t)$. On the other hand, $\mathbf{v}(t)$ is bounded by $v_{\max}$, such that $|\mathbf{v}(t)| \leq v_{\max}$ at any time t.

During the flight, at any time t, the tracked distance λ_p may be defined as

$$\lambda_p(t) = \underset{\lambda \in [0,L]}{\operatorname{argmin}} |\mathbf{p}(t) - \gamma(\lambda)| \tag{5.1}$$

The objective is to implement a control system to generate velocity commands to follow the trajectory, minimizing the minimum normal distance between the actual trajectory travelled by Q and the reference trajectory, which is given by

$$J_1 = \frac{1}{T} \int_0^T |\mathbf{p}(t) - \gamma(\lambda_p(t))| dt, \tag{5.2}$$

and minimizing the difference between the current time t and the reference time given by $\Gamma(\lambda_p(t))$, which is defined as

$$J_2 = \left| \frac{1}{T} \int_0^T (\Gamma(\lambda_p(t)) - t) dt \right|, \tag{5.3}$$

where T is the time taken to complete the task.

5.4 UAS Path And Trajectory Follower

The trajectory generator and the trajectory follower are the two main components that form the proposed module. The default way to use it is interacting with the trajectory follower, but the user can interact also with the generator if it is required as it can be seen in Figure 5.1.

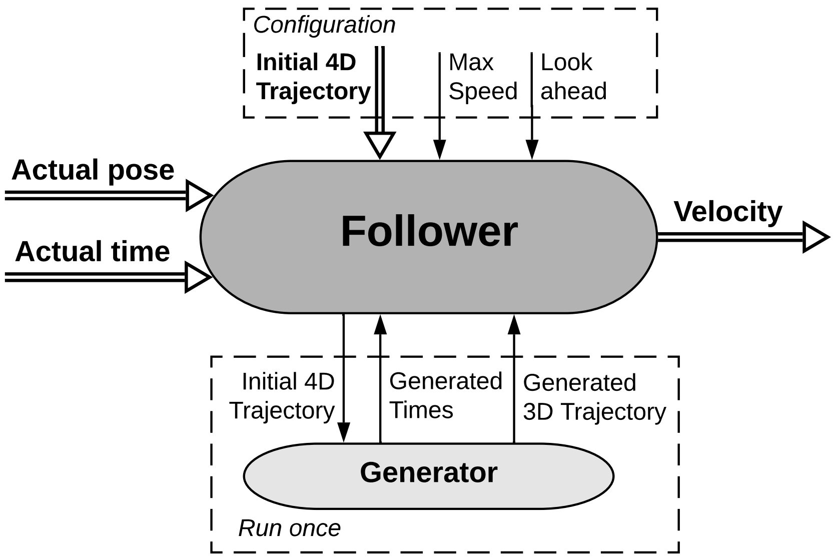


Figure 5.1: The trajectory follower design allows to use it by simply configuring the initial 4D waypoint list. It also provides more configuration options to suit the user needs. The generator is called by the follower and runs once to generate a discrete curve with the reference time on each point of the curve.

5.4.1 Trajectory generator

The trajectory generator is in charge of generating a trajectory Γ based on the ordered list of 4D waypoints received. The generated trajectory is a much more dense list of 3D waypoints, which can be approximated with the continuous curve $\gamma(\lambda)$ described in Section 5.3. It also generates an interpolated list $\tau(\lambda)$ of the initial times that matches the amount of the more dense list of waypoints. Both interpolated lists are necessary for the presented module to follow a 4D trajectory.

5.4.2 Trajectory follower

Initially, the trajectory follower receives the reference trajectory Γ defined as a list of 4D waypoints. It may receive the look-ahead distance d and the maximum speed v_{max}. Setting these parameters properly is key to get a good performance, depending on the reference 4D trajectory. A much more dense list of waypoints is required to apply the trajectory following method efficiently. Hence, it uses the trajectory generator to get a discrete curve $\gamma(\lambda)$ from the ordered list of waypoints W. Then, continuously, it receives the UAS pose $\mathbf{p}(t)$ and the actual time t to generate the velocity commands $\mathbf{v}(t)$, based on the method described below.

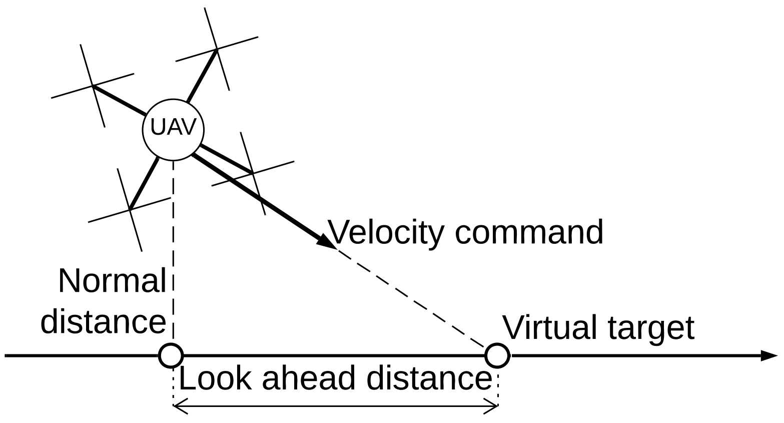


Figure 5.2: Top view of the three-dimensional path follower based on the carrot chasing algorithm without taking into account the orientation error.

The proposed trajectory following method is based on the carrot chasing algorithm, see Figure 5.2. The method runs as follows, first the λ_p argument is obtained using expression 5.1, which minimizes the distance from the UAS position to the trajectory. The fixed look-ahead distance d is added and the virtual target pose in the trajectory is achieved as

$$\mathbf{p}_t(t) = \gamma(\lambda_p(t) + d). \tag{5.4}$$

To fix the time error, the cruising speed is calculated for the time t as

$$v_c = \frac{d}{\tau(\lambda_p(t) + d) - t}. \tag{5.5}$$

The last step is to calculate the velocity command based on the cruising speed, as

$$\mathbf{v}(t) = v_c \frac{\mathbf{p}_t(t) - \mathbf{p}(t)}{|\mathbf{p}_t(t) - \mathbf{p}(t)|} \tag{5.6}$$

to reach the target virtual pose.

5.4.3 Policy adopted to deal with infeasible 4D trajectories

It is possible that the user can set up an infeasible list of 4D waypoints according to the UAS capabilities, where some waypoints w_i are feasible and others are not. In that case, the proposed module should fix the initial list and replace it with feasible times to let the UAS perform a *normal* navigation through the trajectory. The difference of time between going through a segment at maximum velocity and going at the reference velocity $v_{reference}$ should be checked, being $v_{reference}$ the reference velocity to match the times, see Equation 5.7.

$$\Delta T = \frac{w_{i+1} - w_i}{v_{reference}} - \frac{w_{i+1} - w_i}{v_{\max}} \tag{5.7}$$

Each segment s_i is classified in three different ways: infeasible if ΔT is negative, modifiable if it is positive and non-modifiable if ΔT is equal to zero. Non-modifiable segments are the ones where the UAS goes through them at the maximum speed. Negative segments should be extended in time because the UAS has not enough time to go from its beginning to the end of the segment even at the maximum speed. Modifiable segments S are the ones where the UAS can increase its velocity, so they can be reduced in time if needed, see Figure 5.3.

To fix the infeasible time list, the trajectory generator starts extending the segments that are infeasible until they are non-modifiable and stores how long it is extended. At that point, the list is formed by modifiable and non-modifiable segments. The last part is to share equally, if possible, the amount of time stored between the modifiable segments. Every modifiable segment can receive just an amount of time that converts it into a non-modifiable segment, if the time that it should receive is longer than that the other modifiable segments should receive the rest of the time if possible, see Algorithm 2.

5.4.4 Software implementation details

The work described in this chapter has been integrated with the UAL, which tries to abstract the user programmer from the platform's autopilot, defining a common

Algorithm 2: Algorithm to correct the times of the initial waypoint list. It takes the extra time accumulated by infeasible segments and tries to share it equally between the modifiable segments.

```
$t_e$: total time extra,
$d_{t_e}$: time extra division,
$t_s$: subtract time,
foreach $w_i \in W$ do
    calculate $\Delta T$;
    if $\Delta T > 0$ then
        $S$.push_back($s_i$);
    else if $\Delta T < 0$ then
        $t_e = t_e + \Delta T$;
        add $\Delta T$ to the following waypoints in $W$;
    end
end
while $t_e \neq 0$ AND $S.size() > 0$ do
    $d_{t_e} = \frac{t_e}{S.size()}$;
    foreach $s_i \in S$ do
        calculate $\Delta T$;
        if $\Delta T \geq d_{t_e}$ then
            $t_s = d_{t_e}$;
        else
            $t_s = \Delta T$;
        end
        $t_e = t_e - t_s$;
        subtract $t_s$ to the following waypoints in $W$;
    end
    clear $S$;
    foreach $w_i \in W$ do
        calculate $\Delta T$;
        if $\Delta T > 0$ then
            $S$.push_back($s_i$);
        end
    end
end
```

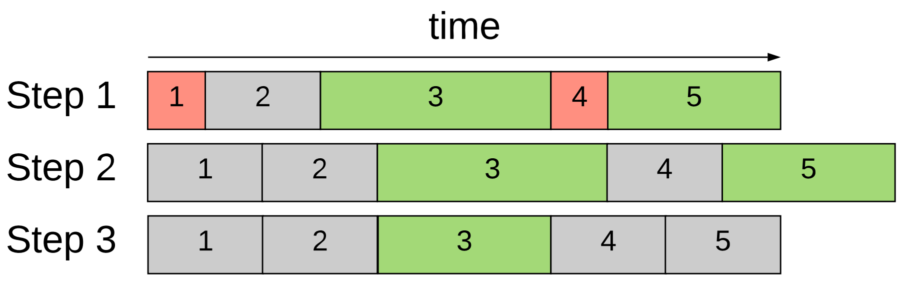


Figure 5.3: Graphical example of Algorithm 2. Red segments are infeasible and should be extended to a grey segment. Green segments are modifiable and can be reduced to match the initial times.

interface with a collection of the most used information and functionalities of an UAS. In particular, the developments presented in this chapter are based on the release 3.0 of UAL and the Kinetic version of ROS. The proposed system receives a list of 4D waypoints, generates a 4D trajectory, and calculates which velocity vector should use UAL as the reference at every instance through the trajectory.

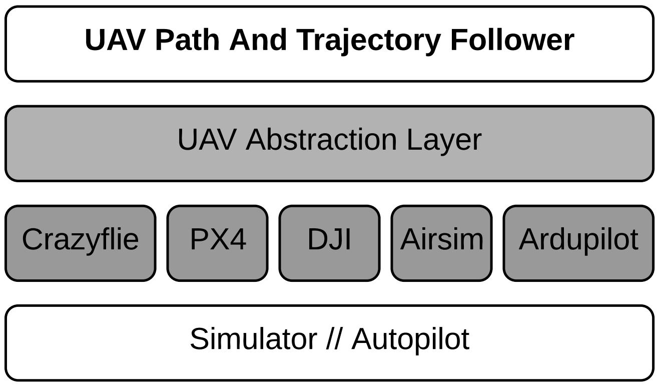


Figure 5.4: The different layers of the software architecture make the system modular. Different autopilots and simulators can be used.

The software architecture is split into four main layers, as depicted in Figure 5.4. In the upper half is the developed UAS path and trajectory follower, which has been packaged as a ROS node to facilitate experimentation and integration, and is built on top of UAL. The lower half of the software architecture is composed by the autopilot, simulators, and communication drivers. The UAL provides a back-end that works with MAVROS[2] which is in charge of providing a communication driver to ROS for various autopilots that uses MAVLink MAVLink (2013) as communication protocol. MAVROS is the ROS adaptation of MAVLink protocol. The simulator used in these developments is based on the PX4 SITL development which is the official SITL environment for the Pixhawk autopilot Meier et al. (2011). UAL has implemented others back-end which provides communications with Crazyflie, Airsim, Ardupilot or DJI protocols. The developed trajectory follower includes two modes: following the trajectory without changing the yaw or aiming at the virtual point.

The proposed module is under continuously development and publicly available in a stable version along with examples and a guide of how to use it. It can be found in the GitHub repository called UPAT Follower[3] under the MIT License.

5.5 Validation results

This section presents different results using the proposed system and a hexacopter as an aerial platform. We ran in simulation large missions and we tested short real experiments inside our indoor testbed to see the scalability of the proposed module. The difference between a large and a short mission is the distance between waypoints.

5.5.1 Simulation results

Our module was tested using GAZEBO, the PX4 SITL functionality to simulate the autopilot, and UAL to interact with simulated UAS. This simulation environment allows running and trying different tests without flying a real UAS using the same software.

We tested a large mission with different velocities between segments, to achieve that the mission should have different times to travel every segment. Table 5.1 shows the initial waypoint list and the reference time t_d of each waypoint. It also shows at which time t_r the simulated UAS reached every waypoint. The mission is feasible for

[2]`https://wiki.ros.org/mavros`
[3]`https://github.com/hecperleo/upat_follower`

the simulated UAS which has in this case a maximum velocity of 4 m/s. Figure 5.5a has different scales of axes to appreciate the error between trajectories. Moreover, how the trajectory follower caps the commanded velocity to the maximum velocity in the first seconds is shown in Figure 5.5b. On the other hand, Figure 5.5d details how the difference between the UAS actual time and the reference time is, if it is negative, the UAS is behind of the schedule, otherwise the UAS is ahead of schedule. The proposed system always tries to make the time difference equal to zero. It also tries to minimize the normal distance between the UAS and the trajectory. Figure 5.5c presents how the UAS after reaching a waypoint moves away by inertia, but recovers while following the next segment of the trajectory. Those errors that the system tries to minimize are shown in Table 5.2, and it is worth to mention that the farthest behind the schedule was 0.830 seconds and the farthest ahead schedule was 0.625 seconds.

Table 5.1: Simulated large mission

	$\mathbf{w_1}$	$\mathbf{w_2}$	$\mathbf{w_3}$	$\mathbf{w_4}$	$\mathbf{w_5}$	$\mathbf{w_6}$	$\mathbf{w_7}$	$\mathbf{w_8}$
x (m)	40.0	-40.0	-40.0	40.0	40.0	-40.0	-40.0	40.0
y (m)	30.0	30.0	-30.0	-30.0	30.0	30.0	-30.0	-30.0
z (m)	2.0	2.0	2.0	2.0	10.0	10.0	10.0	10.0
$\mathbf{t_d}$ (s)	0.0	28.3	66.4	95.0	133.1	161.7	204.6	247.4
$\mathbf{t_r}$ (s)	0.0	28.4	66.2	95.1	132.9	161.9	204.4	247.3

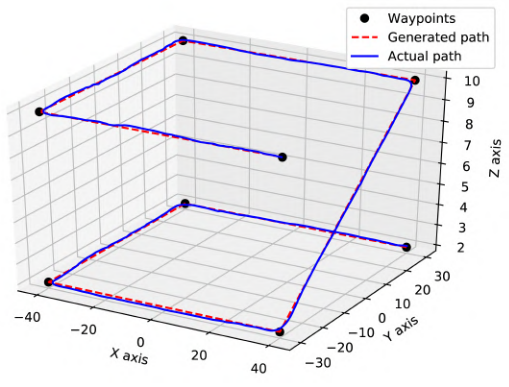
Table 5.2: Simulated large mission errors

Error	**Min**	**Mean**	**Max**	**Std**	**Var**[*]
Space $\mathbf{J_1}(m)$	0.007	0.188	1.02	0.195	0.038
Time $\mathbf{J_2}(s)$	0.000	0.086	0.830	0.100	0.010

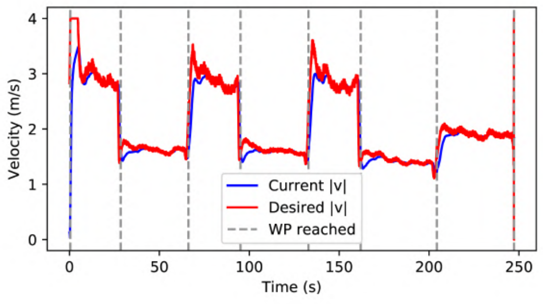
[*]Variance units are squared.

5.5.2 Real experiments

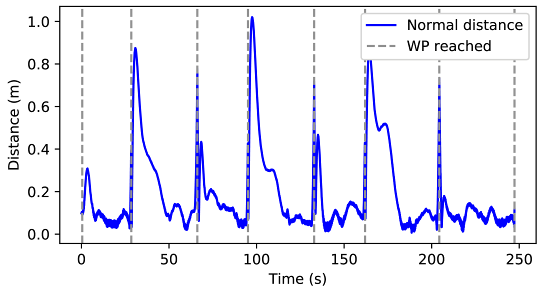
The real experiment is scaled with respect to the simulated due to the limited volume available inside our OptiTrack testbed, which is located the Aerial Robotics Laboratory of the University of Seville, see Figure 5.6. It is equipped with an overhead



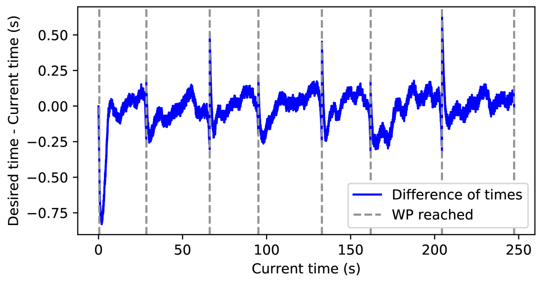
(a)



(b)



(c)



(d)

Figure 5.5: Simulated experiment. (a) Three dimensional view of the initial waypoints, the trajectory generated and the trajectory described. (b) reference velocity and current velocity of the UAS. (c) Normal distance between the UAS and the generated trajectory. (d) Difference between the reference time and the current time.

motion capture system which tracks the 6 degrees of freedom pose of one or more objects. Each object tracked needs to be defined as a rigid body, which is a cluster of reflective markers arranged in a unique configuration for each object. A computer processes the motion capture data, matches it to one predefined rigid body of the UAS, and transmits the pose to the aerial platform at a reference frequency, in this case at 30 Hz to replicate the frequency at which a global position system used outdoors would send information.

The aerial platform used for the real experiment is a DJI F550 hexacopter, see Figure 5.7. It has an on-board computer Intel NUC5i7RYH with 16 GB RAM and

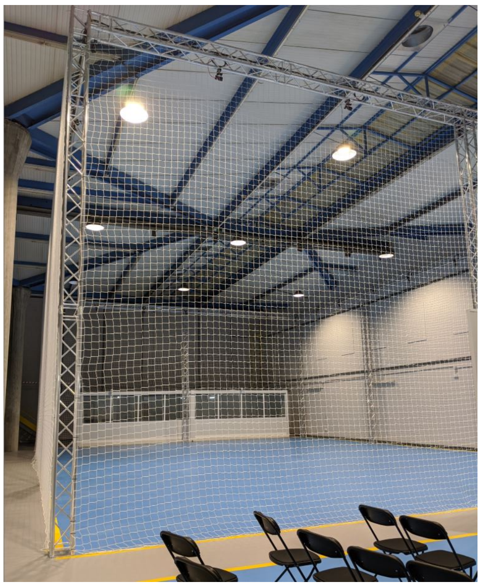


Figure 5.6: OptiTrack Testbed

a 256 GB Samsung 950 PRO M.2 SSD hard disk. The on-board computer can be connected to the Pixhawk autopilot through a serial or USB port. It uses two independent 4S 5500 mAh batteries, one for the motors and autopilot, and the other for the on-board computer. This configuration allows the developer to keep the computer turned on between experiments, to modify the code, or to check or download data, even if the battery for the motors and autopilot is low and needs to be replaced.

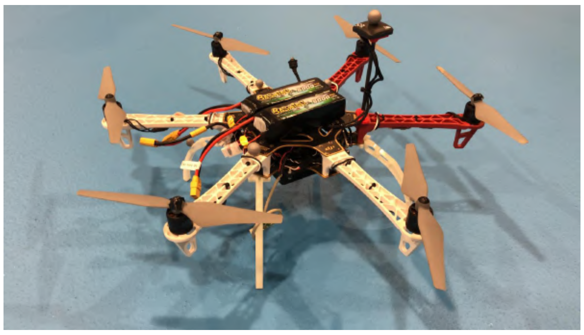
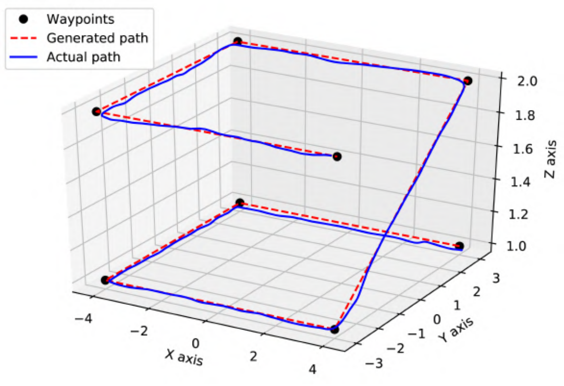
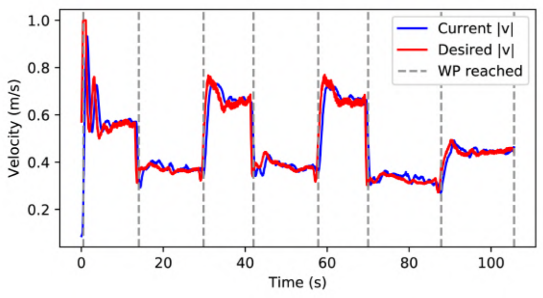


Figure 5.7: DJI F500 Hexacopter used in the flight tests

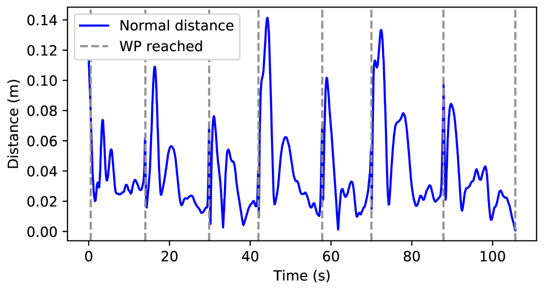
The real experiment is scaled with respect to the simulated one and the UAS should go at different speeds through the segments. Table 5.3 shows the initial waypoint list and the reference times and the reached times. Figure 5.8 shows the results of the real experiment, where the real UAS has a similar behaviour comparing it with the simulated one. A video of the experiment is publicly available[4]. The proposed module tries to minimize online the normal distance error and the time error through the trajectory, see Table 5.4. It is worth to mention that the farthest behind schedule was 0.487 seconds and the farthest ahead schedule was 0.346 seconds.



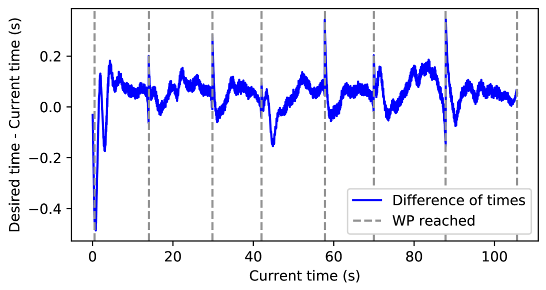
(a)



(b)



(c)



(d)

Figure 5.8: Real experiment inside the OptiTrack Testbed. (a) Three dimensional view of the initial waypoints, the trajectory generated and the trajectory described. (b) reference velocity and current velocity of the UAS. (c) Normal distance between the UAS and the generated trajectory. (d) Difference between the reference time and the current time.

[4]`https://youtu.be/Yv-gHy4_PVs`

Table 5.3: Real scaled mission results

	$\mathbf{w_1}$	$\mathbf{w_2}$	$\mathbf{w_3}$	$\mathbf{w_4}$	$\mathbf{w_5}$	$\mathbf{w_6}$	$\mathbf{w_7}$	$\mathbf{w_8}$
x (m)	4.0	-4.0	-4.0	4.0	4.0	-4.0	-4.0	4.0
y (m)	3.0	3.0	-3.0	-3.0	3.0	3.0	-3.0	-3.0
z (m)	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0
$\mathbf{t_d}$ (s)	0.0	14.0	30.0	42.0	58.0	70.0	88.0	106.0
$\mathbf{t_r}$ (s)	0.0	14.0	29.8	42.0	57.8	69.9	87.8	105.6

Table 5.4: Real scaled mission errors

Error	**Min**	**Mean**	**Max**	**Std**	**Var***
Space $\mathbf{J_1}(m)$	0.001	0.039	0.141	0.026	0.001
Time $\mathbf{J_2}(s)$	0.000	0.071	0.487	0.054	0.003

*Variance units are squared.

5.5.3 Comparison between methods

This subsection analyzes a version of the proposed system without using Algorithm 2, method 1, and another version that uses this algorithm, method 2. We tested in simulation a feasible mission using 4 m/s as v_{max}, and we simulated it three more times lowering its maximum velocity but without modifying the initial 4D waypoint list W, see Table 5.1. By doing that, there is a point where a feasible segment becomes infeasible because of the maximum velocity reduction. Figure 5.9 shows the result of equations 5.2 and 5.3 on the simulations. It also shows the values using the absolute value of the errors, but in these cases, where the reference velocity matches the maximum velocity, the UAS is mostly behind schedule.

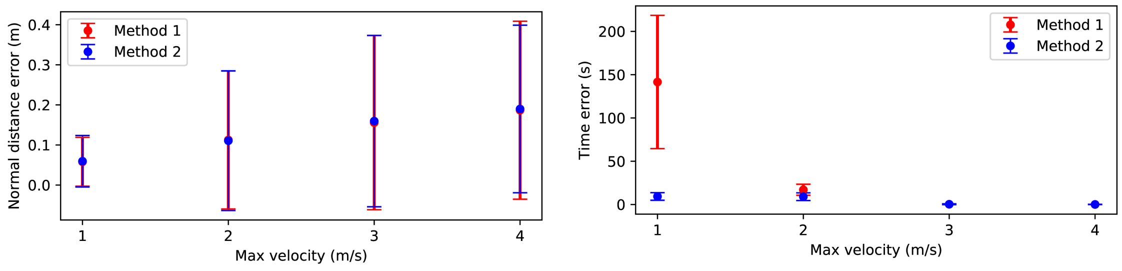


Figure 5.9: Mean and standard deviation of error J_1 and error J_2 with the same initial 4D waypoint list on different simulations modifying the maximum velocity on each experiment.

The difference between methods has been clarified and method 2 presents considerably smaller time errors J_2 than method 1. The simulation with a maximum velocity of 3 m/s is a good example of how the proposed system in this chapter tries to fix at every time the error that has accumulated. This example has segments 1, 3, and 5 at maximum speed, see Figure 5.10. This figure shows at segment 1 that the UAS is behind schedule because it starts the segment at 0 m/s and it fixes while catching the reference velocity, in this case the maximum velocity. Segments 3 and 5 have a similar behaviour, but the UAS is not so far behind the schedule because it did not start these segments at 0 m/s.

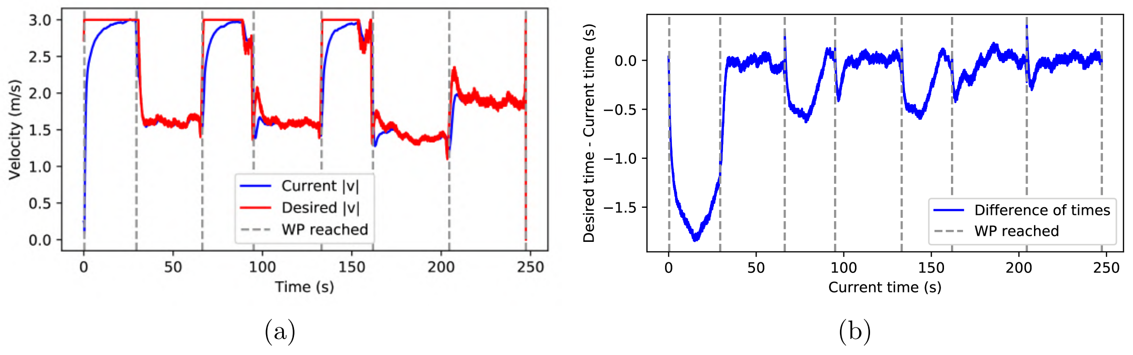


Figure 5.10: Random segments using 3 m/s as v_{max}. (a) reference velocity and current velocity of the UAS. (b) Difference between the reference time and the current time.

Lower maximum velocity experiments have a similar behaviour than the segments showed where the reference velocity matches the maximum velocity. The UAS has no time to fix its current error and it increases time by time. This is why the experiment with v_{max} equal to 4 m/s has a mean of 0.082 meters and a standard deviation of 0.113 m on error J_2 and the experiment with v_{max} equal to 1 m/s has a mean of 9.298 m and a standard deviation of 4.374 m on error J_2.

There are cases where method 1 is more interesting for the user than method 2. For instance, if the user has a mission where every segment is modifiable except the last ones, method 2 would fix the whole mission to return a feasible trajectory, see Table 5.5. This means that the modifiable segments change their defined time to absorb the extra time that the infeasible segments have. The user can be more interested in reaching the waypoint at the time that he defined and have a large error in the last infeasible segment, instead of not reaching the waypoints at the defined time to have a small error over all the trajectory.

Table 5.5: Trajectory with the last segment unfeasible

	$\mathbf{w_1}$	$\mathbf{w_2}$	$\mathbf{w_3}$	$\mathbf{w_4}$	$\mathbf{w_5}$	$\mathbf{w_6}$	$\mathbf{w_7}$	$\mathbf{w_8}$
Initial $\mathbf{t_d}$(s)	0.0	32.0	56.0	88.0	112.2	144.2	168.2	184.2
M1 $\mathbf{t_d}$(s)	0.0	32.0	56.0	88.0	112.2	144.2	168.2	184.2
M1 $\mathbf{t_r}$(s)	0.0	32.1	55.9	87.9	112.2	144.1	168.1	189.8
M2 $\mathbf{t_d}$(s)	0.0	31.3	54.7	86.0	109.5	140.9	164.2	184.2
M2 $\mathbf{t_r}$(s)	0.0	31.4	54.7	86.0	109.6	140.9	164.1	185.9

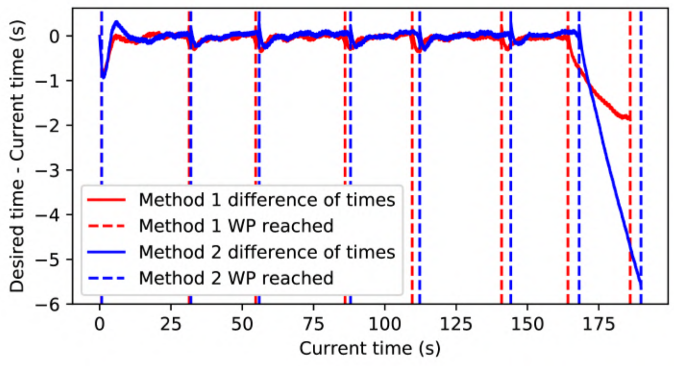


Figure 5.11: Difference of times t_d and reached times t_r of methods 1 and 2 following the same 4D trajectory

Figure 5.11 shows how both methods stay on schedule following each respective 4D trajectory. Method 1 has not fixed the reference time respecting the initial times, as the last segment is not feasible for the UAS, even at the maximum velocity, the difference of time increases until -5.7 seconds, so it can not finish the trajectory at the reference time, see Table 5.5. Method 2 has fixed the reference time respecting the initial time it reaches the waypoints before method 1. As the last segment is feasible, however, the reference velocity matches the maximum velocity, the UAS takes longer to reach that velocity, and while doing that, the difference of time increases until -1.8 seconds. This is why method 2 did not finish the trajectory at the reference time.

5.6 Conclusions

In U-space context, the concept of 4D-TBO becomes a key element to increase the flight safety and to maximize the shared airspace. This chapter faces this challenge, presenting a 4D trajectory follower with low deviation in distance and time with respect to the intended 4D trajectory.

This chapter presents a 4D trajectory follower that can be used based on a list of parameters: look ahead distance and maximum velocity. Trying different values in the simulation several times is recommended before going to fly in the real world because these values directly depend on the input reference trajectory and the behaviour of the trajectory follower would be affected.

The developed UAS path and trajectory follower can fix its space and time errors while following a reference 4D trajectory, it minimizes at every time the equations J_1 and J_2 presented in Section 5.3. The obtained results show that at higher speeds the UAS may oscillate about the trajectory, and it could decrease its performance if a short look-ahead distance is settled. On the other hand, if the user sets a large look-ahead distance, the UAS will cut corners because the UAS tries to turn towards each new virtual point. A good start point for tuning the look-ahead distance is to make it equal to the maximum velocity of the trajectory.

The presented trajectory follower can fix the preflight infeasible trajectories given by the user to increase its performance while following the trajectory. It also warns the user that the given trajectory is not feasible and lets to the user the decision of choosing the initial or the fixed trajectory. The results show that there are cases where for the user is better not to fix an infeasible trajectory.

Chapter 6

Experiments

This chapter presents all experiments carried out with our UTM related to the European GAUSS[1] project, detailed in Section 6.1. There were several partners in the project who were part of the experiments, EVERIS[2] was in charge of flying multiple aircraft, and SATWAYS[3] developed the RPS client application and the server for the communication of the components via Internet. This chapter also shows the experiments carried out with our onboard autonomous detect and avoid module related to the European SAFEDRONE[4] project, detailed in Section 6.2. In this project, Unifly[5] provided a USSP to create, validate and manage the drone operations during the experiments.

Due to the characteristics and needs of the GAUSS project, the real experiments carried out have been planned to have a maximum of three UAS flying sharing the same airspace at the same time. On the other hand, the SAFEDRONE project was carried out with one UAS.

[1]`https://cordis.europa.eu/project/id/776293`
[2]`https://www.everis.com/global/en`
[3]`https://www.satways.net`
[4]`https://cordis.europa.eu/project/id/783211`
[5]`https://www.unifly.aero/`

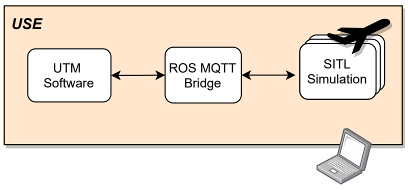
6.1 UTM experiments

6.1.1 Setup used

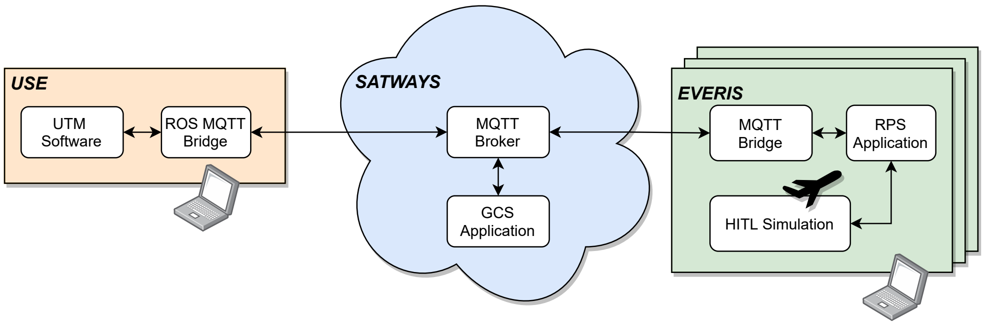
Our UTM architecture was built on ROS Kinetic, and the software is publicly available online[6]. As the UTM architecture is designed at a very high level, the setup does not change drastically between real experiments, preliminary tests, and simulations, see Figure 6.1. This is a key factor to be able to simulate a scenario before doing it on a real aircraft, thus a lot of time can be saved if an error in the definition of the operation appears simulating the scenario. In the simulations, the partners are simulated, and the whole operation can be tested with a single computer using a SITL airspace simulated in Gazebo, see Figure 6.1a. For system integration and preliminary tests involving other partners, we used Hardware In The Loop (HITL) simulating just the flight plan of the aircraft. This allowed us to see if the communications between partners, computers, autopilots, and protocols were working properly, see Figure 6.1b. Before doing HITL tests in the field with the partners in the same place, we carried out several integration tests remotely. Finally, Figure 6.1c shows the configuration used in the real experiments with the aircraft flying and without any simulation. The final experiments were done in the field with SATWAYS working remotely due to the pandemic issues, so one of the advantages of the design of the whole system appeared letting us run the experiments with this partner working from Greece.

EVERIS had a Remote Pilot Station (RPS) for every UAS involved in the field. Furthermore, they were able to make HITL simulations generating telemetry data and send it to us in real time. Thanks to this, the whole project was tested before going to the field, and we were able to see errors in the communications protocols to fix them before the real experiments. For every test, EVERIS had one safety pilot in charge of flying the UAS and keep an eye always in safety, so he had the responsability to go on an alternative route. EVERIS had an operator, who has the responsability of communicate with the UTM all time. The operator is the one who accepts or denies the alternative flight plan after discussing it with the safety pilot. Another

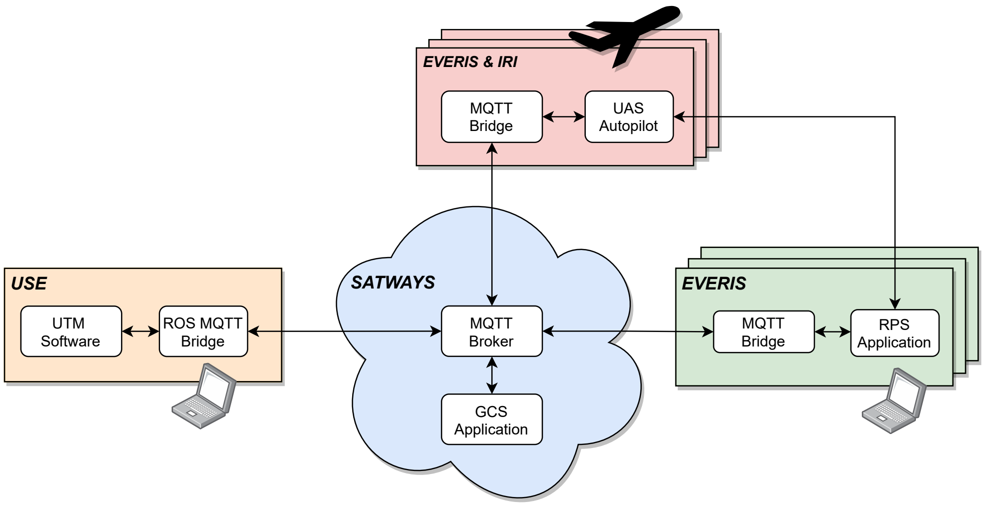
[6]`https://github.com/grvcTeam/gauss`



(a)



(b)



(c)

Figure 6.1: Three setups used along the SITL tests, HITL tests, and real experiments.
(a) SITL setup. We simulated an airspace and its active aircraft using Gazebo.
(b) HITL setup. In this case aircraft were not flying so the onboard modules and autopilots were simulated by EVERIS.
(c) Experiments setup. USE had a laptop with the UTM architecture, EVERIS one with the Remote Pilot Station (RPS) and with access to the Ground Control Station (GCS), finally EVERIS and IRI had their software onboard.

partner involved was IRI[7], and they were in charge of checking the health of the GNSS signal, converting the telemetry to the Message Queuing Telemetry Transport (MQTT) protocol and sending it through 4G communication. Finally, SATWAYS developed the RPS client application and the server for the communication of the components via Internet, see Figure 6.2.

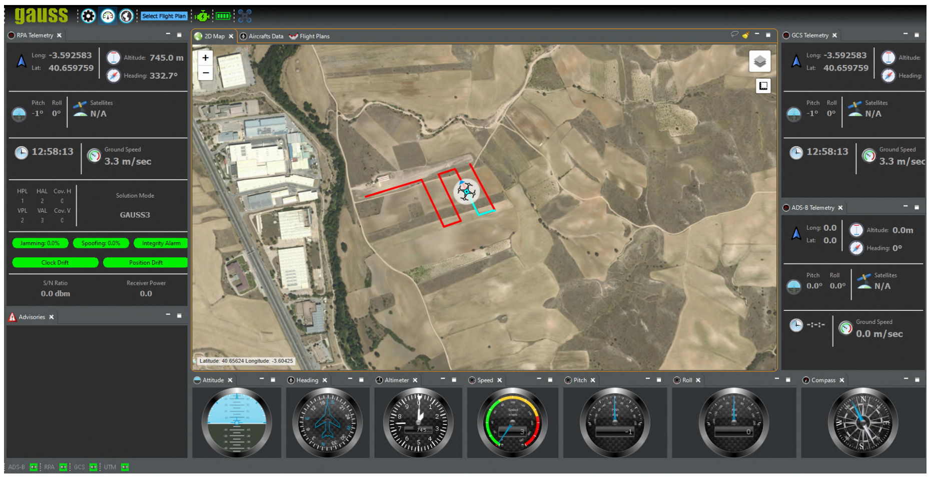


Figure 6.2: Graphical user interface developed by SATWAYS running on the RPS Client Application.

Our UTM was connected via Internet with EVERIS to carry out the experiments, they sent us in real time the telemetry of each active UAS processed by IRI, and, if any conflict or threat appeared, we communicated back with EVERIS proposing to the operators an alternative route to follow and to avoid the potential conflict. This communication was done exchanging JavaScript Object Notation messages between our UTM and each UAS RPS, and it was done using the MQTT protocol. We tried to replicate a real U-space environment using Internet as our communication medium, not being necessary to have the UTM computer in the same network of the RPS computers.

[7]https://www.iri.upc.edu/

Aircraft used

During the experiments, we used three UAS: Matrice 600 Pro, Atlantic I and Scrab II. These aircraft are quite different from each other, first we had one quadrotor and two fixed wings, therefore they have different maneuveravility and different autopilots. Moreover, there is a large gap between the maximum velocity between each UAS and the cruising velocity. These characteristics have to be taken into account to define useful scenarios to test U-space services. For example, if the Scrab II is flying at 200 km/h, the UTM detects a conflict with this aircraft and the UTM creates a geofence in the position of the Scrab II, every second delayed will result in a 28-meter error. This is an unfavourable case, but it shows how accurate the whole system must be to avoid errors.

The characteristics of the UAS involved in the experiments done in the project GAUSS are summarized in Table 6.1.

Table 6.1: Summary of the characteristics of each aircraft used.

Characteristic	**M600**	**Atlantic I**	**Scrab II**
MTOM (kg)	15	50	90
Payload (kg)	5	5	10
Autonomy (min)	15	60	300
Range (km)	3	100	100
Maximum speed (m/s)	18	48	120
Cruising speed (m/s)	-	30	60
Wingspan (m)	-	3.8	2.5
Length (m)	1.2	2.8	2.9

The DJI Matrice 600 Pro (M600), is intended for professional aerial photography and industrial applications, see Figure 6.3. It inherits everything from the previous version of the M600, but with increased flight performance and loading capacity. The system's modular architecture makes it simple to mount additional modules, while preinstalled arms and antennae save setup time. The A3 Pro flying controller, Lightbridge 2 HD communication system, Intelligent Batteries, and Battery Management system are among the newest DJI technologies used in the airframe. The M600 is excellent for professional aerial photography and industrial applications since it is natively compatible with several Zenmuse cameras and gimbals and has complete integration with third-party software and hardware.

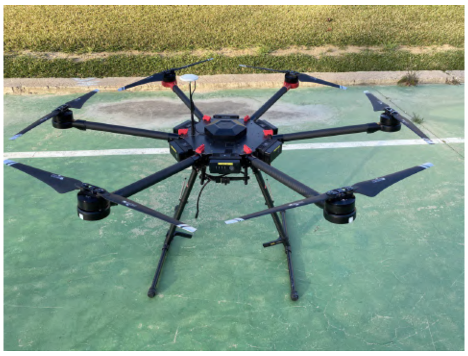


Figure 6.3: DJI Matrice 600 Pro

The Atlantic is an UAS composed completely of composite materials that is intended for quick operational response, see Figure 6.4. Only two people are required to operate it on a mission, whether it is taking off from a runway or launching from a catapult, due to its simplicity and reliability. The Atlantic follows the flight plan autonomously and automatically, and features a parachute recovery mechanism in case of an emergency.

Platform, ground station, and weather resistant laptop with command and control software make up the entire system. The Atlantic is a particularly adaptable unit since it can carry a wide range of payloads, such as EO/IR sensors, radio and radar equipment.

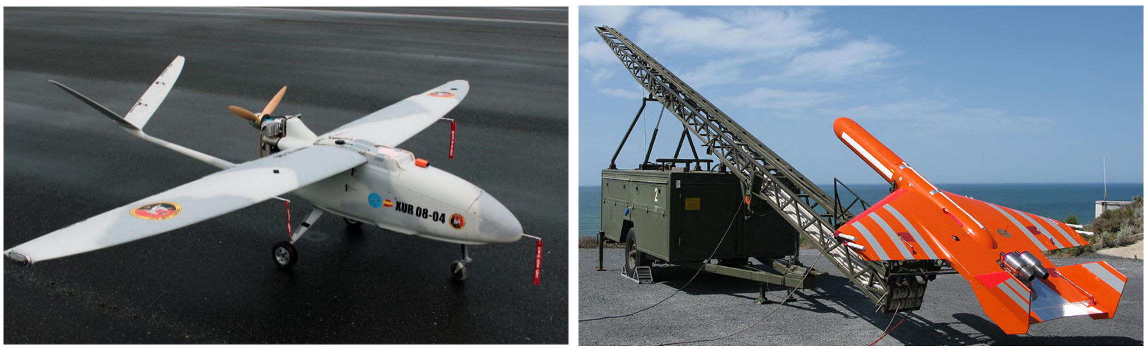


Figure 6.4: Left, the Atlantic I. Right, the Scrab II.

Figure 6.4 shows the Scrab II, a target UAS propelled by twin turbines. It has been developed and engineered to deliver great performance while keeping the operating and maintenance operations as simple as possible. Its propulsion consists of two turbines that are controlled by the onboard electronics of the target aircraft. The great flying capabilities of the Scrab II are owed to the power of the configuration, which allows it to attain speeds of up to 120 m/s.

Its easy functioning, on the other hand, makes it a multifunctional target that may be utilized in a variety of settings. Moreover, the internal components of the aircraft are shielded, this includes for offshore exercises. Finally, the communications of the target UAS, which can reach radio connection lengths of up to 100 kilometers, are extremely reliable.

6.1.2 Scenario definition

This section details the order of the operations carried out in the field experiments as well as which conflicts have been forced to be resolved later. We conducted experiments in two different test flight centers, the first one was in ATLAS, and the aim was to test the land scenarios of the project. The second one was in El Arenosillo and the main purpose was testing the whole system in a marine scenario with fast aircraft.

ATLAS

The first one was in the Test Flight Center ATLAS[8] (Air Traffic Laboratory for Advanced unmanned System) located in Jaen, Spain. It offers an aerodrome equipped with scientific facilities and airspace ideally suited to the development of experimental flights with UAS, see Figure 6.5. In addition to the facilities, ATLAS has a segregated airspace[9] that has an extension of 1000 km^2, which is an incredible advantage to do experimental flight saving time in terms of bureaucracy that should be done when not having a segregated airspace. In these experiments, the aircraft used are the M600

[8] `http://atlascenter.aero/en/atlas_aa23.html`

[9] `http://atlascenter.aero/en/espacio-aereo_aa7.html`

and the Atlantic I, detailed in Section 6.1.1. Due to safety reasons, the Atlantic was allowed to fly at a minimum of 400 meters above the ground. This limited us to create two operations that get in conflict by losing its safety distance, thus we were forced to increase the parameters involved in the detection of the loss of separation conflict. The rest of the conflicts were not customized to get a proper demonstration of the UTM developed.

Figure 6.5: Test Flight Center Air Traffic Laboratory for Advanced unmanned System located in Jaen, Spain.

The three scenarios done in ATLAS are depicted in Figure 6.6, and the following paragraphs detail which conflicts appeared and how the UTM solves them. Table 6.2 summarizes the characteristics of the operations done in ATLAS.

The first scenario carried out in ATLAS was composed by operations 1 and 2, see Figure 6.6a. Operation 1 was an agriculture inspection done by the M600, while operation 2 was a forest inspection done by the Atlantic. Due to the minimum flight altitude of the Atlantic, we could have flown the M600 at a higher altitude, however we decided to keep it at a comfortable altitude for the safety pilot. It also allowed us to see the M600 without binoculars following the alternative flight plan given by the UTM to avoid the loss of separation conflict. In this case, the difference of altitude between UAS were 350 meters, and operation 1 started 16 seconds after operation 2 to be able to make the two UAS coincide in the same latitude and longitude. The alternative flight plan given to the M600 was equal to the one that was previously accepted, but decreasing its height at the last segment of the flight plan, this way the loss of separation conflict were avoided modifying the altitude of the M600. It was the M600 one that modified its flight plan due to the priority of the UAS. Quadrotors are often given lower priority than fixed wings because they have more maneuverability. An exception can be an emergency operation carried out by a quadrotor, for example, delivering rescue and medical supplies to an injured person in the forest.

Table 6.2: Summary of all operations done in ATLAS.

UAS	**Name**	**Description**	**Potential threat**	**Speed** (m/s)	**Altitude** (m)
M600	Op 1	Agriculture inspection	Loss of separation	3.3	70
	Op 3	Windturbine inspection	Geofence intrusion	1.0	30 - 90
	Op 5	Event monitoring	Jamming	3.3	70
Atlantic	Op 2	Forest inspection	Loss of separation	15	600
	Op 4	Powerline inspection	Geofence conflict	15	420

Figure 6.6b shows the second scenario tested in ATLAS, and it was formed by operation 3, which was a windturbine inspection done by the M600. In this case, the difference of altitude imposed by the minimum height of the Atlantic and the difference of the start time of the operations did not matter as much as in the first scenario. This scenario was created to test how the firefighters can interact with the UTM, how it creates geofences in the commanded place, and how the system gives the operator an alternative flight plan to leave as soon as possible the conflictive geofence. To carry out these tests, both UAS started their operations and, suddenly, we simulated the interaction of the firefighters telling the UTM that a wildfire has appeared, and thus the utm created a geofence in the commanded place with the given radius. After the geofence generation, the M600 was inside that geofence, in such a way that the UTM detected a geofence intrusion and sent a notification to the operator in charge of operation 3 telling him to leave the geofence as soon as possible.

The third scenario tested in ATLAS can be seen in Figure 6.6c, and it was formed by operations 4 and 5. In this case, operation 4 was a powerline inspection done by the Atlantic and operation 5 was an event monitoring done by the M600. This scenario was created to test two different threats: the first one was to test the jamming attack on an UAS and see how the UTM acted, the second one was to test a geofence conflict. In this scenario, both UAS started its respective operations and followed its flight plans, suddenly, we simulated a jamming attack on the M600, the UTM sent a notification to the operator in charge of the M600 telling him to land it as soon as possible because of the jamming attack. Therefore, a geofence was created by the UTM to prevent other UAS flying near the jamming zone, and this geofence intersected with the flight plan of operation 4, thus the UTM detects a geofence conflict. To solve this potential threat, the UTM sent a notification to the operator in charge of operation 5, telling him that the flight plan of the Atlantic is intersecting with a geofence and proposing an alternative flight plan to avoid it.

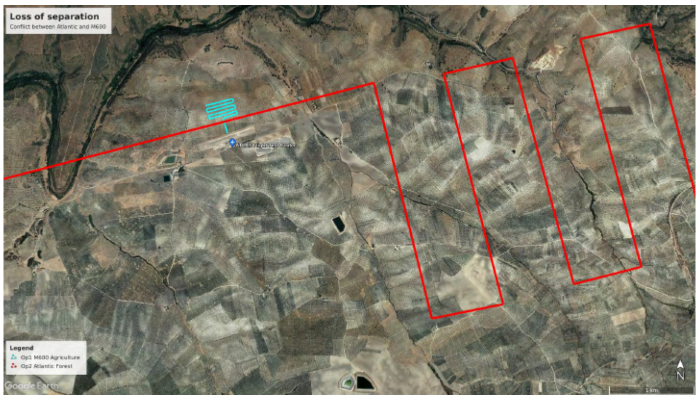
El Arenosillo

The second field experiments were carried out in the testing center of El Arenosillo[10] (CEDEA) located in Huelva, Spain. It was a sounding rocket base, but nowadays, it is active in other National Institute of Aerospace Technology[11] (INTA) and Ministry of Defense programmes, mainly atmospheric studies and tests of unmanned aircraft. This test center has been used by the military to make shooting tests with UAS like the Scrab described in Section 6.1.1.

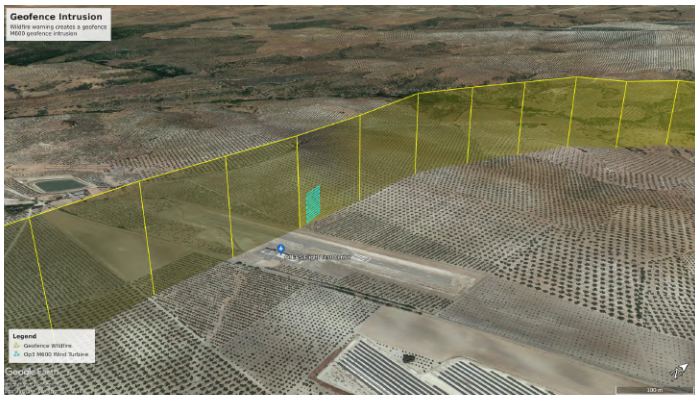
Figure 6.8 shows three marine scenarios tested in El Arenosillo, and the following paragraphs detail which conflicts appeared and how the UTM solves them. Table 6.3 summarizes the characteristics of the operations done in El Arenosillo.

The first marine scenario carried out in El Arenosillo was composed by operations 1 and 2, see Figure 6.8a. Both operations were marine inspections, operation 1 was done by the Atlantic, while operation 2 was done by the Scrab. In this marine scenario, the minimum altitude limitation was the same presented in ATLAS, thus the Atlantic had to fly above 400 meters. In addition to that, the Scrab had also the requisite to fly above that height, therefore we decided to fly the Scrab at 700 meters and the Atlantic at 500 meters of altitude, letting a safety distance between UAS of

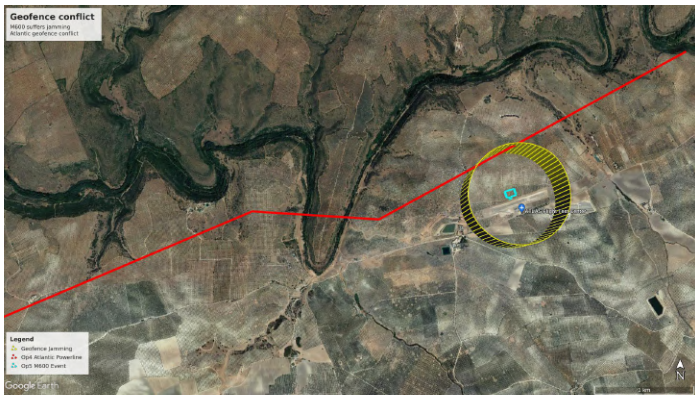
[10]`https://www.inta.es/INTA/en/quienes-somos/historia/el-arenosillo/`
[11]`https://www.inta.es/INTA/en/quienes-somos/`



(a)



(b)



(c)

Figure 6.6: Three scenarios defined in order to solve potential conflicts.
(a) Conflict between M600 and Atlantic due to loss of safety distance.
(b) Firefighters warn about a wildfire, M600 appears inside the created geofence.
(c) M600 suffers jamming, the created geofence intersects with the Atlantic.

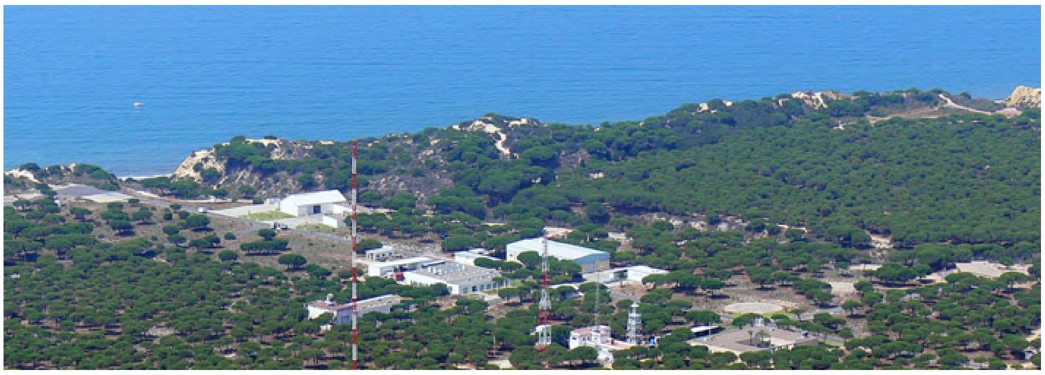


Figure 6.7: El Arenosillo Testing Center (CEDEA) located in Huelva, Spain.

Table 6.3: Summary of all operations done in El Arenosillo.

UAS	**Name**	**Description**	**Potential threat**	**Speed** (m/s)	**Altitude** (m)
Atlantic	Op 1	Marine inspection	Loss of separation	22	500
	Op 3	Transition to next operation	Geogence conflict	22	500
	Op 4	Beach monitoring	Geofence intrusion	22	500
Scrab	Op 2	Marine inspection	Loss of separation and Jamming	55	700

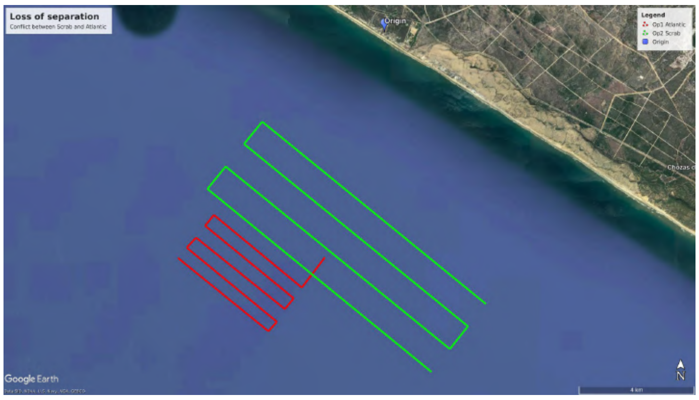
200 meters. To match both aircraft in the same longitude and latitude, operation 2 started 750 seconds after operation 1, since this allowed us to force a loss of separation conflict. The alternative flight plan given to the Atlantic was not equal to the one that was previously accepted, and decreasing its altitude in El Arenosillo, the UAS could not cross in the airspace, and thus the alternative flight plan given to the Atlantic was a safe spot in the airspace. It was the M600 one that modified its flight plan due to the priority of the UAS. In this case, the Scrab had more maneuverability, however we had only 30 minutes of effective fly due to its autonomy, thus the alternative flight plans were given to the Atlantic. We had to manage these 30 minutes of effective fly per day of the Scarb, therefore we decided to schedule the operations of the Atlantic to generate potential conflicts with the one and only operation of the Scrab. This is one of the reasons of having a safe spot as an alternative flight plan to avoid the loss of separation conflict, and it allowed us to confirm the detection of the conflict and continue right away with the next operation wasting as little time as possible.

Figure 6.8b shows the second scenario tested in El Arenosillo, and it was formed by operations 2 and 3, which was a marine inspection done by the Scrab and a transition to another operation done by the Atlantic respectively. In this case, the difference of height imposed on the aircraft for safety reasons did not matter as much as in the first scenario. In terms of timing, the Scrab started its operation in the previous scenario and did not finish, but the Atlantic started operation 3 when the Scrab was in its antepenultimate segment of the flight plan. This scenario was created to test two things: the first one was to test the jamming attack on the Scrab and see how the UTM acted, the second one was to test a geofence conflict. In this scenario, both UAS followed their respective flight plans and, suddenly, we simulated a jamming attack on the Scrab, the UTM sent a notification to the operator in charge of the Scrab telling him to land it as soon as possible because of the jamming attack. Therefore, a geofence was created by the UTM to prevent other UAS flying near the jamming zone, and this geofence intersected with the flight plan of operation 3, thus the UTM detects a geofence conflict. To solve this potential threat, the UTM sent a notification to the operator in charge of operation 3, telling him that the flight plan of the Atlantic is intersecting with a geofence and proposing an alternative flight plan to avoid it. All unexpected routes given to the Atlantic had to be westward facing, on the other hand, the unexpected routes given to the Scrab had to be eastward facing. An unexpected route were the routes generated if we cancelled a scenario, for example, if we saw problems with telemetry or communications, and we needed to restart the operation, the Atlantic went to its starting waypoint in the west part of the reserved airspace in El Arenosillo. This requisite was given to us also to force the generation of the alternative flight plan of the discussed scenario to go around the west part of the geofence instead of the east one.

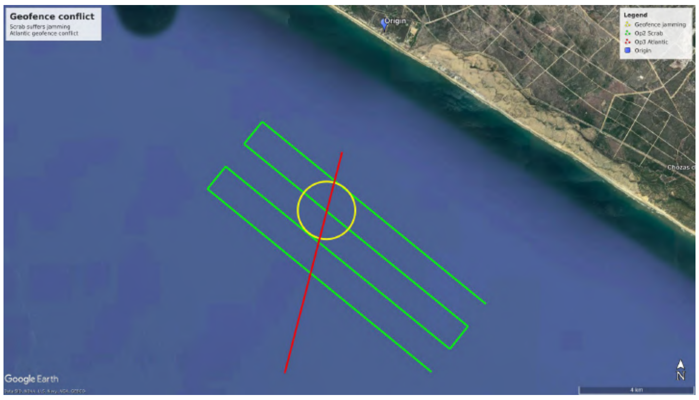
The last scenario tested in El Arenosillo can be seen in Figure 6.8c, and it was formed by operation 4, which was a beach monitoring done by the Atlantic. This scenario was created to test how the firefighters can interact with the UTM, how it creates geofences in the commanded place, and how the system gives the operator an alternative flight plan to leave as soon as possible the conflictive geofence. To carry out these tests, the Atlantic started its operation and, suddenly, we simulated the interaction of the firefighters telling the UTM that a wildfire has appeared, and thus the utm created a geofence in the commanded place with the given radius. After the geofence generation, the Atlantic was inside that geofence, in such a way that the UTM detected a geofence intrusion and sent a notification to the operator in charge of operation 4 telling him to leave the geofence as soon as possible.

6.1.3 Real experiments

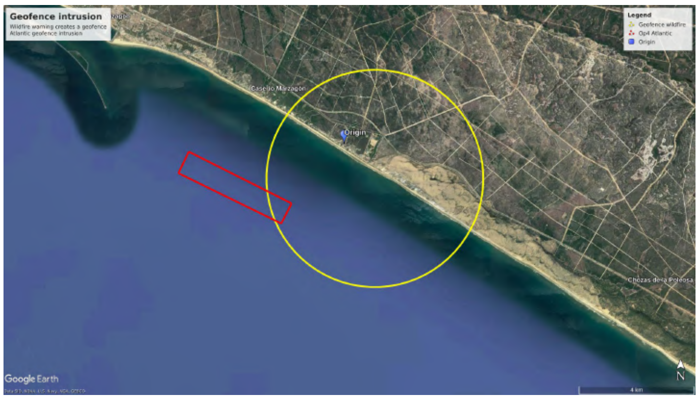
The scenarios presented in Section 6.1.2 were used to test the UTM architecture, and the results of the experiments are detailed in this section, as well as the details of how the UTM worked internally, how the modules communicate with each other to detect and solve a potential conflict, and how long did each module spent computing its tasks. The Monitoring module works with the estimated trajectory of each UAS, and these trajectories are limited by a time horizon predefined to 5 minutes. These minutes cover the UTM necessary computation time to suggest an alternative flight plan, the time of the operator deciding if that alternative route will be accepted or not, and the time of the operator loading to the UAS final solution. It also includes



(a)



(b)



(c)

Figure 6.8: Three scenarios defined in order to solve potential conflicts.
(a) Conflict between Scrab and Atlantic due to loss of safety distance.
(b) Scrab suffers jamming, the created geofence intersects with the Atlantic.
(c) Firefighters warn about a wildfire, Atlantic appears inside the created geofence.

extra time taking into account if the operator denies the proposed alternative flight plan and the time needed by the UTM to suggest another one.

ATLAS

The scenarios detailed in this section are the same as those previously presented, Figure 6.6 has a visual representation of the scenarios, and Table 6.2 summarizes the operations of the experiments.

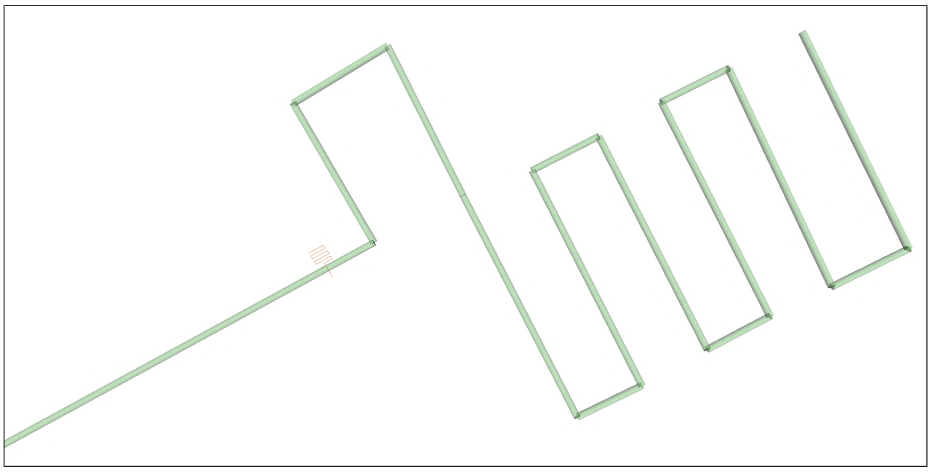


Figure 6.9: First real experiment carried out in ATLAS visualized using RViz. Top view of the loss of separation conflict between the operation 1 done by the M600 (orange) and the operation 2 done by the Atlantic (green).

The first potential conflict forced was the loss of separation, which is a loss of safety distance between two UAS. The operation 1 carried out by the M600 at 3.3 m/s crossed with operation 2 in the airspace, which was done by the Atlantic at 15 m/s. It should be remarked here that both operations can coexist in the airspace crossing each other, as long as they pass through the same longitude and latitude at different times. If this requisite is not matched when both operators upload their flight plans to the UTM, it should be detected the conflict in a preflight stage. However, this Thesis is focused on the tactical stage, thus we skipped the verification of the operations before flying. This allowed us to force a conflict like the one presented in this scenario, loss of separation, which can not appear if the UTM is checking operations before the aircraft involved are flying their flight plan. Hence, to force the conflict we needed to match both UAS in the same latitude and longitude, and with a difference of altitude less than the safety distance, which in this case was 350 meters. Operation 1 started 16 seconds after operation 2 to match both UAS in the airspace, see Figure 6.9. There was a difference of height of 530 meters between the operations of the Atlantic and the M600, which produced a loss of separation conflict. To be able to detect this conflict, the Tracking module was updating the estimated trajectories of all UAS, the Monitoring module used these estimated trajectories to detect the loss of separation conflict, which took 0.17 milliseconds to be detected. Monitoring sent a message to the Emergency Management module warning of a potential conflict detected, which asked for support to the Tactical Deconfliction module. It took 0.5 milliseconds to compute the alternative flight plans that solved the potential conflict and Emergency Management spent 0.19 milliseconds deciding which one was suggested to the operator. This list of alternative flight plans was composed by flight plants with different maneuvers like avoid the conflictive flight plan in the opposite direction, return to home, land in a landing spot, reduce the speed, and increase the speed. Each one of them has two parameters, risk and cost, to help the Emergency Management decide which one should be suggested to the operator. Since this event, the suggestion went from the Emergency Management through the U-space Service Manager module to the operator in 3.4 milliseconds. The operator spent 6 seconds deciding to accept or decline the alternative flight plan suggested, and when accepted, the new flight plan was stored in the Database in 0.3 seconds. Figure 6.10 shows the alternative flight plan accepted by the operator and the one that the M600 followed to finish its operation. Finally, the times of the events described in this paragraph are summarized in Table 6.4.

The geofence intrusion was the second potential conflict tested in ATLAS, which appears if an UAS is inside an active geofence. The operation 3 carried out by the M600 at one meter per second was a vertical zigzag, thus the operation simulated a windturbne inspection. In this scenario, we simulated the interaction between the UTM and the firefighters, which were the ones that warned the UTM of a wildfire in a certain area. It took 39.9 milliseconds between the firefighters warned the UTM

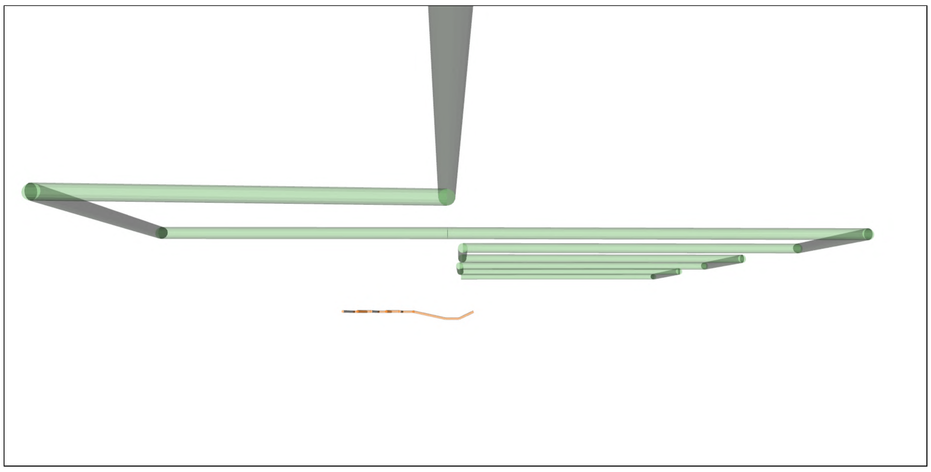


Figure 6.10: First real experiment carried out in ATLAS visualized using RViz. Side view of the alternative flight plan accepted by the M600 operator to avoid the loss separation conflict between the operation 1 done by the M600 (orange) and the operation 2 done by the Atlantic (green).

Table 6.4: Timeline of loss of separation in ATLAS, first experiment.

0.00s	Atlantic starts operation 2.
21.59s	M600 starts operation 1.
23.09s	Monitoring detects the loss of separation.
24.07s	Tactical Deconfliction calculates the alternative flight plans.
24.08s	Emergency Management chooses an alternative flight plan.
24.08s	U-space Service Manager suggests the alternative flight plan.
29.76s	M600 operator accepts and loads the alternative flight plan.
349s	M600 finishes its operation.
1186s	Atlantic finishes its operation.

and a geofence was created in the Database following the characteristics of the warn message, like center, radius and height of the geofence. Figure 6.11 shows the operation inside the geofence, producing a geofence intrusion, which took 0.15 milliseconds to be detected by Monitoring. The procedure is the same as the last scenario presented, Monitoring warned the Emergency Management module, and this one asked for support to the Tactical Deconfliction module, which calculated a list of alternative flight plans. Tactical Deconfliction spent 0.39 milliseconds computing the list and Emergency Management spent 0.32 milliseconds deciding which one was proposed to the operator. In this case, the alternative flight plan proposed was a straight line to a landing spot. When the UTM proposes a landing spot as an alternative flight plan, the spot is not on the ground, it is given at the same altitude that the aircraft has when the conflict was detected. This approach allows the UAS going to a safe spot as soon as possible, which is key while leaving a geofence, and then the aircarft can start the procedures of landing. Therefore, the suggestion went from the Emergency Management through the USM module to the operator in 3.7 milliseconds. The operator spent 14 seconds deciding to accept or decline the alternative flight plan suggested, and when accepted, the new flight plan was stored in the Database in 0.6 seconds. Figure 6.12 shows the alternative flight plan accepted by the operator and the one that the M600 followed to finish its operation. Finally, the times of the events described in this paragraph are summarized in Table 6.5.

Table 6.5: Timeline of geofence intrusion in ATLAS, second experiment.

0.00s	M600 starts operation 1.
44.01s	Firefighters warn about a wildfire in the area.
45.03s	Emergency Management creates a geofence.
45.04s	Monitoring detects the geofence intrusion.
45.12s	Tactical Deconfliction calculates the alternative flight plans.
45.13s	Emergency Management chooses an alternative flight plan.
45.13s	U-space Service Manager suggests the alternative flight plan.
59.28s	M600 operator accepts and loads the alternative flight plan.
174.9s	M600 finishes its operation.

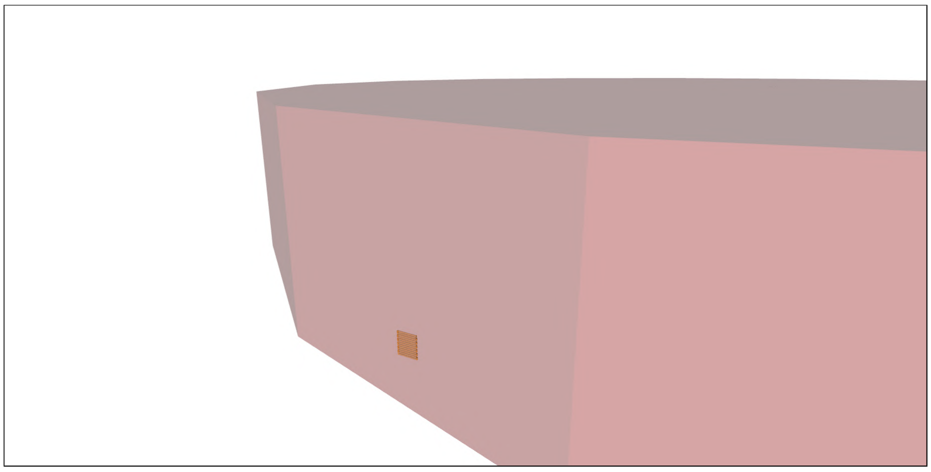


Figure 6.11: Second real experiment carried out in ATLAS visualized using RViz. View of the geofence intrusion conflict between the operation 3 done by the M600 (orange) and the geofence (red) created due to the warning of a wildfire by the firefighters.

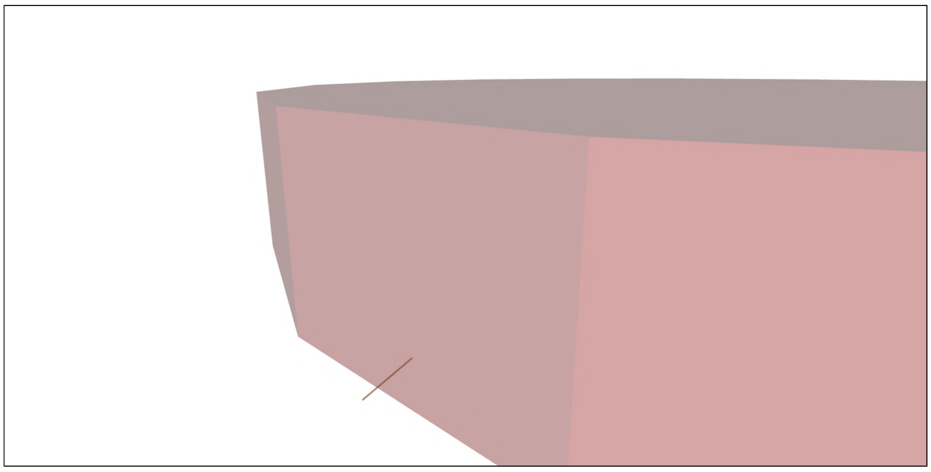


Figure 6.12: Second real experiment carried out in ATLAS visualized using RViz. View of the alternative flight plan accepted by the M600 operator to avoid the geofence intrusion conflict between the operation 3 done by the M600 (orange) and the geofence (red) created due to the warning of a wildfire by the firefighters.

The last scenario tested in ATLAS involved operation 4 done by the M600 at 3.3 m/s and operation 5 done by the Atlantic at 15 m/s. Here we tested a jamming attack on an UAS, and a geofence conflict produced by the geofence created to protect other UAS from the jamming attack. In this case, while both aircraft were doing their operations, we simulated a jamming attack on the M600, therefore a geofence was created in the Database in 3.5 milliseconds by the UTM. This geofence intersected with the flight plan of the Atlantic, see Figure 6.13, and it took Monitoring 0.18 seconds to detect the geofence conflict. The procedure after this is the same presented in the previous scenarios, thus Table 6.6 summarizes the timing of the events of this scenario. The UTM was able to detect, compute multiple solutions, and propose to the operator an alternative flight plan in 3.9 seconds. The solution accepted by the operator can be seen in Figure 6.14. There is a difference of the geofence color between the two figures presented in this scenario, because the system has the ability to distinguish between active (red) and nonactive (yellow) geofences. The geofences are 3D shapes plus the temporal dimension, thus the airspace can be optimized.

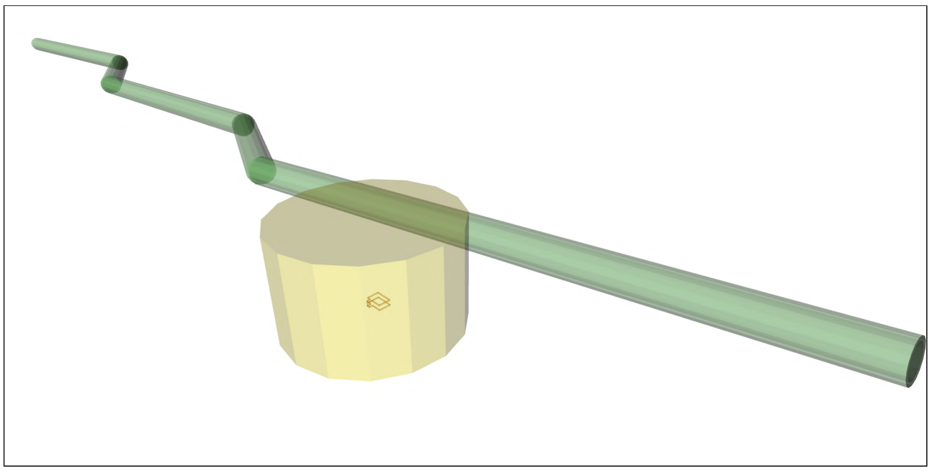


Figure 6.13: Third real experiment carried out in ATLAS visualized using RViz. View of the geofence conflict between the operation 4 done by the Atlantic (green) and the geofence (yellow) created due to the warning of a jamming attack, which was received by the operation 5 carried out by the M600 (orange).

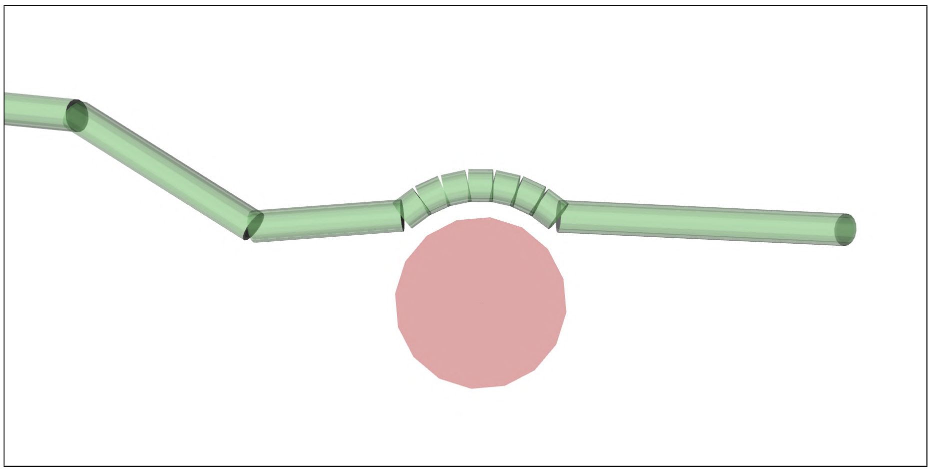


Figure 6.14: Third real experiment carried out in ATLAS visualized using RViz. Top view of the alternative flight plan accepted by the Atlantic operator to avoid the geofence conflict between the operation 4 carried out by the Atlantic (green) and the geofence (red) created due to the warning of a jamming attack.

Table 6.6: Timeline of geofence conflict in ATLAS, third experiment.

0.0s	Atlantic starts operation 4.
31.34s	M600 starts operation 5.
95.15s	M600 receives a jamming attack.
95.16s	Emergency Management creates a geofence.
96.85s	Monitoring detects the geofence conflict.
96.99s	Tactical Deconfliction calculates the alternative flight plans.
97.82s	Emergency Management chooses an alternative flight plan.
98.85s	U-space Service Manager suggests landing the M600.
98.85s	U-space Service Manager suggests the alternative flight plan.
110.1s	M600 operator accepts and decides to land the M600.
123.8s	Atlantic operator accepts and loads the new flight plan.
482.2s	Atlantic finishes its operation.

El Arenosillo

Due to the limitations presented in the Flight Test Center El Arenosillo, where the autonomy of the Scrab was the largest one, we decided to schedule the operations in such a way that the Scrab did one operation while the Atlantic did multiple ones to force different potential conflicts. For this reason, Table 6.3 has just one operation done by the Scrab and three done by the Atlantic. Figure 6.8 has a visual representation of the scenarios and the potential conflicts forced. In the marine scenarios, the Atlantic kept the minimum altitude limitation presented in the experiments carried out in ATLAS, thus the operations done by the Atlantic had an altitude of 500 meters, while the Scrab flew at 700 meters, letting a safety distance between them of 200 meters at the closest point. The flight plans were influenced by another requirement presented in El Arenosillo, which was that UAS could not cross each other in the airspace, no matter at what altitude were the aircraft flying. This influenced the way alternative flight plans were proposed, the procedure of aborting operations if needed, and it limited the airspace given to carry out the experiments.

The first marine scenario carried out in El Arenosillo was composed by operation 1 done by the Atlantic at 22 m/s and operation 2 done by the Scrab at 55 m/s. To match both aircraft in the same longitude and latitude, operation 2 started 750 seconds after operation 1, since this allowed us to force a loss of separation conflict, see Figure 6.15. The requirement of not crossing UAS in the airspace was not violated because they never did. The Monitoring module, using the estimated trajectory, detected the conflict before the aircraft crossed with each other, and it took 0.17 milliseconds to detect the conflict. Due to the requirement of not crossing, the alternative flight plan given to the Atlantic was completely different to the accepted flight plan, see Figure 6.16. The solution proposed was a safe spot in the airspace to avoid having the aircraft crossing each other, and taking into account the rest of the experiments since the Scrab has only 30 minutes of autonomy assigned to run operations.

The second scenario tested in El Arenosillo involved operation 2 done by the Scrab at 55 m/s and operation 3 done by the Atlantic at 22 m/s. Here we tested a jamming attack on an UAS, and a geofence conflict produced by the geofence created to protect other UAS from the jamming attack. In this case, while both aircraft were doing their

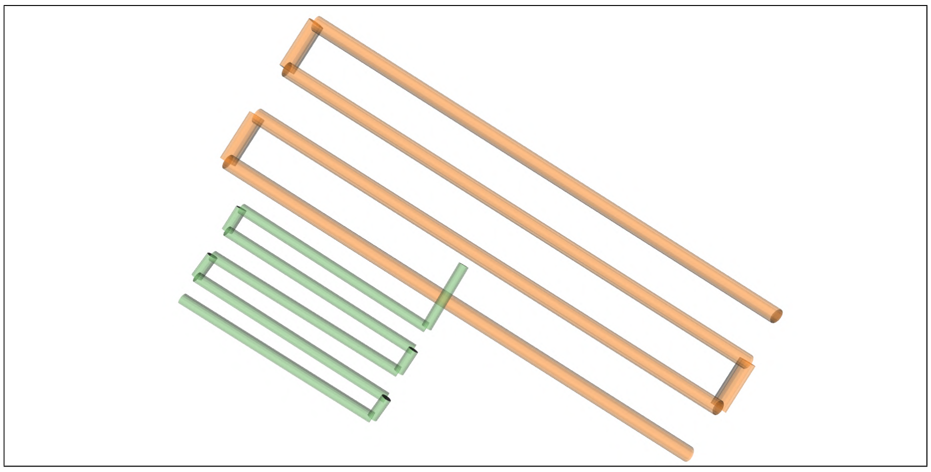


Figure 6.15: First real experiment carried out in El Arenosillo visualized using RViz. Top view of the loss of separation conflict between the operation 1 done by the Atlantic (green) and the operation 2 done by the Scrab (orange).

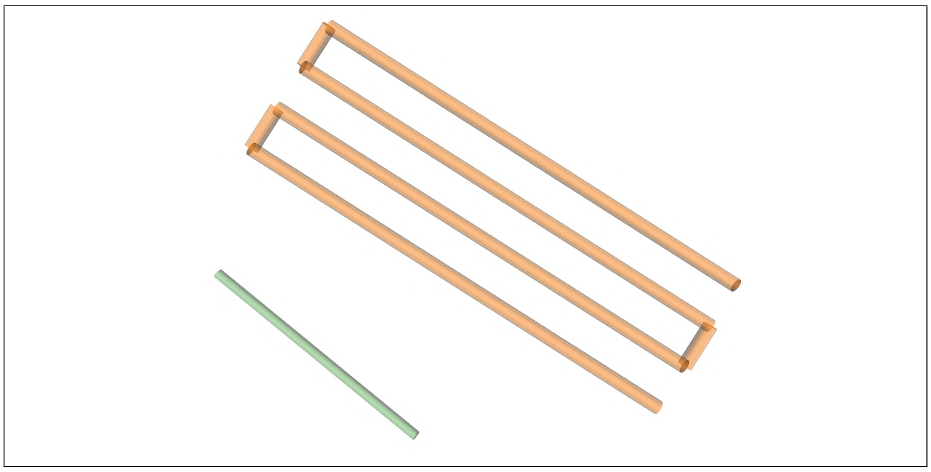


Figure 6.16: First real experiment carried out in El Arenosillo visualized using RViz. Side view of the alternative flight plan accepted by the Atlantic operator to avoid the loss of separation conflict between the operation 1 done by the Atlantic (green) and the operation 2 done by the Scrab (orange).

operations, we simulated a jamming attack on the Atlantic, therefore a geofence was created in the UTM. This geofence intersected with the flight plan of the Scrab, see Figure 6.17, and the Monitoring module spent 0.18 milliseconds to detect the geofence conflict. The procedure after this is the same as the one presented in the previous scenarios. The UTM was able to detect, compute multiple solutions, and propose to the operator an alternative flight plan in 3.8 seconds. The solution accepted by the operator can be seen in Figure 6.18. After avoiding the potential conflict, the Atlantic finished its operation 3 and went to start the last operation of the experiments. On the other hand, the Scrab finished operation 2 and went to its landing spot to start the landing maneuver using a parachute.

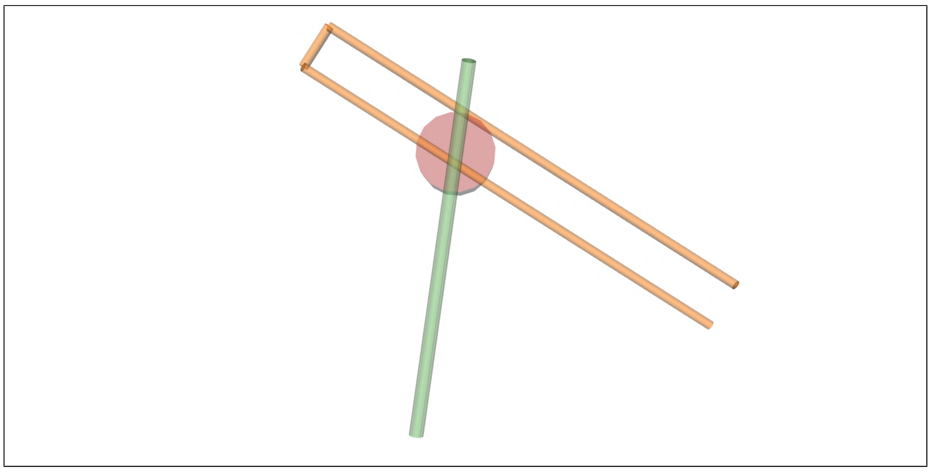


Figure 6.17: Second real experiment carried out in El Arenosillo visualized using RViz. Top view of the geofence conflict between the operation 3 done by the Atlantic (green) and the geofence (red) created due to the warning of a jamming attack, which was received by the operation 2 carried out by the Scrab (orange).

The geofence intrusion was the last conflict tested in El Arenosillo, which appears if an UAS is inside an active geofence. The operation 4 carried out by the Atlantic at 22 m/s was a rectangle, thus the operation simulated a beach monitoring. In this scenario, we simulated the interaction between the UTM and the firefighters, which were the ones that warned the UTM of a wildfire in a certain area. It took 39.1 milliseconds between the firefighters warned the UTM and a geofence was created in the Database. Figure 6.19 shows part of the operation inside the geofence: the

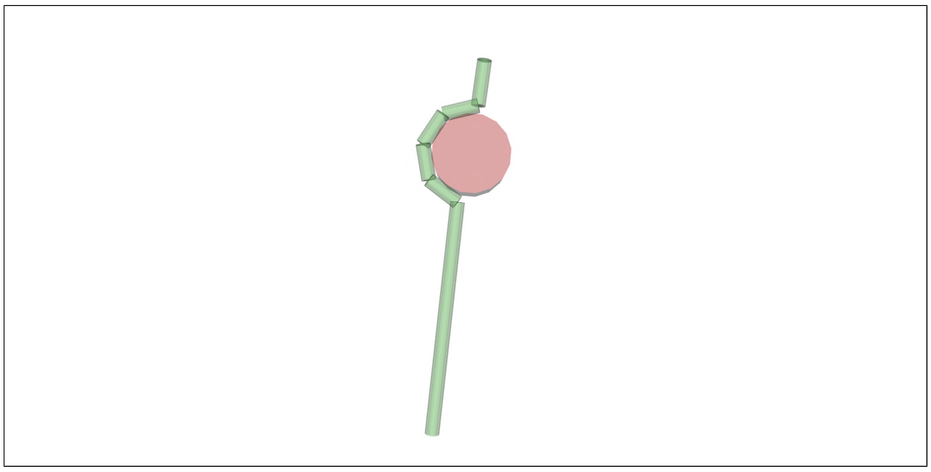


Figure 6.18: Second real experiment carried out in El Arenosillo visualized using RViz. Top view of the alternative flight plan accepted by the Atlantic operator to avoid the geofence conflict between the operation 3 carried out by the Atlantic (green) and the geofence (red) created due to the warning of a jamming attack.

Atlantic was at this moment at this part, thus the geofence intrusion was detected by the Monitoring module in 0.19 milliseconds. The procedure is the same as the last scenario presented: Monitoring warned the Emergency Management module, and this one asked for support to the Tactical Deconfliction module, which calculated a list of alternative flight plans. Tactical Deconfliction spent 0.4 milliseconds computing the list and Emergency Management spent 0.31 seconds deciding which one was proposed to the operator. In this case, the alternative flight plan proposed was a straight line to a landing spot. Therefore, the suggestion went from the Emergency Management through the USM module to the operator in 3.9 milliseconds. The operator spent 13 seconds deciding to accept or decline the alternative flight plan suggested, and when accepted, the new flight plan was stored in the Database in 0.5 seconds. Figure 6.20 shows the alternative flight plan accepted by the operator and the one that the Atlantic followed to finish its operation.

Finally, a timeline is presented in Table 6.7 detailing the events of the experiments carried out in El Arenosillo. Unlike in the case of the experiments carried out in ATLAS, the marine scenarios were designed to be done using one take off of both aircraft, due to the autonomy limitation of the Scrab. This resulted in experiments

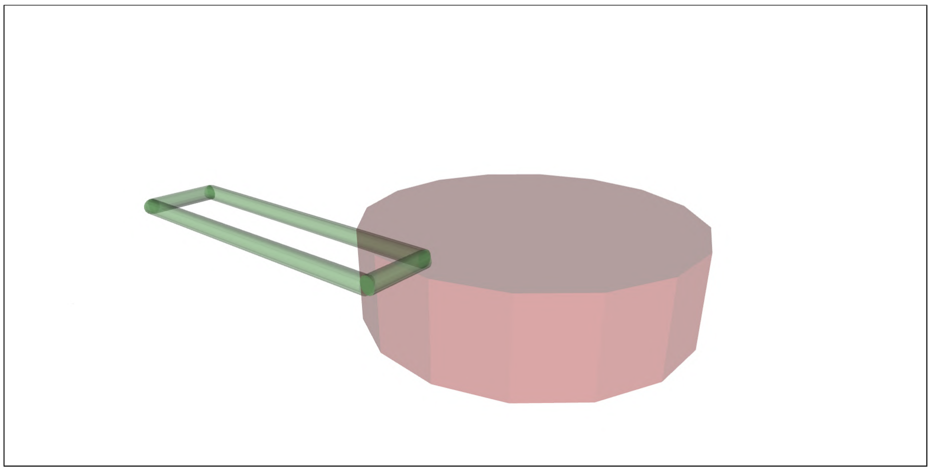


Figure 6.19: Third real experiment carried out in El Arenosillo visualized using RViz. View of the geofence intrusion conflict between the operation 4 done by the Atlantic (green) and the geofence (red) created due to the warning of a wildfire by the firefighters.

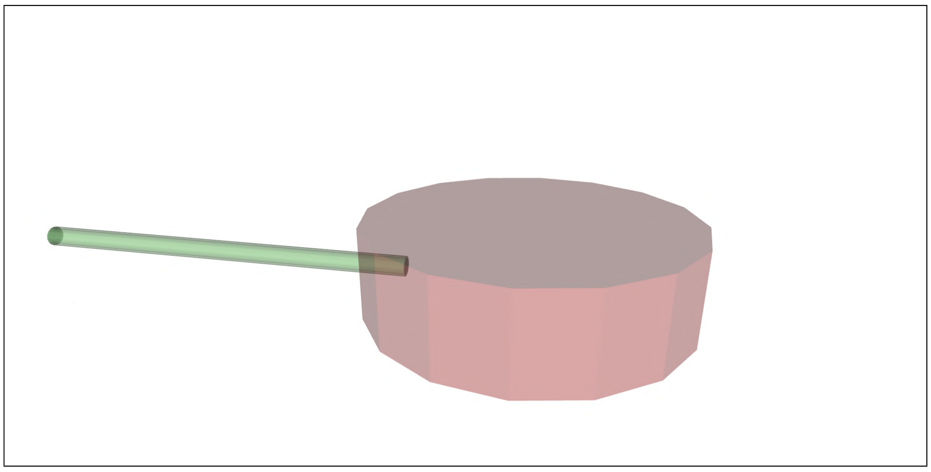


Figure 6.20: Third real experiment carried out in El Arenosillo visualized using RViz. View of the alternative flight plan accepted by the Atlantic operator to avoid the geofence intrusion conflict between the operation 4 done by the Atlantic (green) and the geofence (red) created due to the warning of a wildfire by the firefighters.

with chained operations, i.e. the last point of the first operation of the Atlantic is the start point of its second operation, and the last point of the second operation is the first point of the third operation. Chaining operations allowed us to lose the minimum time possible between operations and we were able to repeat a segment if an operation was aborted due to any unexpected problem. The timeline summarizes the order of the events presented during all experiments carried out in El Arenosillo, where three conflicts were forced to be later detected and resolved, which are differentiated in the timeline by a time jump. The time of each event is referred to the start of operation 2 carried out by the Scrab.

6.1.4 Comparison between methods

This section presents a comparison between two approaches that solve the same problem: how to detect and solve conflicts between 4D UAS trajectories. The method chosen for this Thesis is detailed in Chapter 4 and is the one that can be scaled without problems, and therefore it is the final solution. Before that, a brute force method was developed in the first place to be able to make tests quickly in the field, a method that does not need to set up parameters to work properly, see Appendix A for more details. A brute force method was not a problem knowing that no more than five UAS were going to be tested simultaneously in the field experiments.

A significant number of simulations were run to assess the effectiveness of the algorithm implemented in the Monitoring module in relation to the number of UAS and the size of the planned trajectories, with the latter being proportional to the number of waypoints for each trajectory. Each simulation creates a scenario consisting of a certain number of straight trajectories between two random waypoints in a rectangular prism with predetermined spatial dimensions XYZ and a predetermined time horizon T. Then, using the discretized based method detailed in Chapter 4 and the appropriate time and iterations, each scenario is analyzed. Additionally, each situation is assessed using the brute force approach presented in Appendix A to compare the time and iterations required and check whether they provide the same results.

Table 6.7: Timeline of all experiments carried out in El Arenosillo

0.0s	Atlantic starts operation 1.
693s	Scrab starts operation 2.
693s	Monitoring detects the loss of separation.
693s	Tactical Deconfliction calculates the alternative flight plans.
694s	Emergency Management chooses an alternative flight plan.
694s	U-space Service Manager suggests the alternative flight plan.
711s	Atlantic operator accepts and loads the new flight plan.
712s	Atlantic starts its new flight plan.
1076	Atlantic finishes its operation.
⋮	
1256s	Atlantic starts operation 3.
1368s	Scrab receives a jamming attack, so UTM creates a geofence.
1368s	Monitoring detects the geofence intrusion.
1368s	Tactical Deconfliction calculates the alternative flight plans.
1369s	Emergency Management chooses an alternative flight plan.
1370s	USM suggests landing the Scrab and new plan to Atlantic.
1391s	Scrab operator accepts and decides to land the Scrab.
1394s	Atlantic operator accepts and loads the new flight plan.
1395s	Atlantic starts its new flight plan.
1616s	Atlantic finishes its operation.
⋮	
1796s	Atlantic starts operation 4.
1976s	Firefighters warn about a wildfire in the area.
1977s	Emergency Management creates a geofence.
1977s	Monitoring detects the geofence intrusion.
1978s	Tactical Deconfliction calculates the alternative flight plans.
1978s	Emergency Management chooses an alternative flight plan.
1978s	U-space Service Manager suggests the alternative flight plan.
1994s	Atlantic operator accepts and loads the new flight plan.
1996s	Atlantic starts its new flight plan.
2141s	Atlantic finishes its operation.

Figure 6.21a illustrates the time necessary to find all conflicts based on the number of trajectories for both the method of filling the 4D grid and the brute force approach, depending on the number of trajectories. For all simulations, both approaches discovered the same conflicts. While the needed time for the brute force technique grows quadratically with the number of trajectories, the discretized method grows linearly with the number of trajectories.

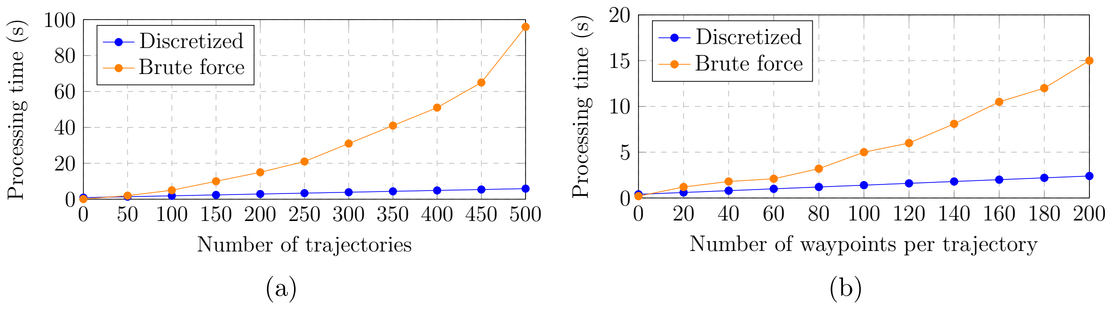


Figure 6.21: Comparison between brute force based method and discretized method. Required time to detect all potential conflicts. The simulations have been performed assuming a common volume of 50x50x10 meters.
(a) Results of a battery of tests increasing the number of trajectories.
(b) Results of a battery of tests increasing the number of waypoints per trajectory.

Furthermore, Figure 6.21b compares both the discretized based method and the typical brute force algorithm in terms of the time necessary to discover all possible conflicts, depending on the number of waypoints per route. For all simulations, both approaches discovered the same conflicts. The required time to detect all conflicts depends quadratically on the brute force based method, however, it depends linearly on the number of waypoints using the discretized based method.

The key disadvantages of the suggested approach, on the other hand, are the needed storage capacity and the time required to initialize the grid. Both are affected by the volume, time horizon, safety distance, and time steps that are taken into account. For example, in the scenario depicted in Figure 6.21b, the grid grows by 50x50x10 = 25000 for every second of the time horizon. Figure 6.22 shows how the time required to generate the grid grows as the time horizon grows.

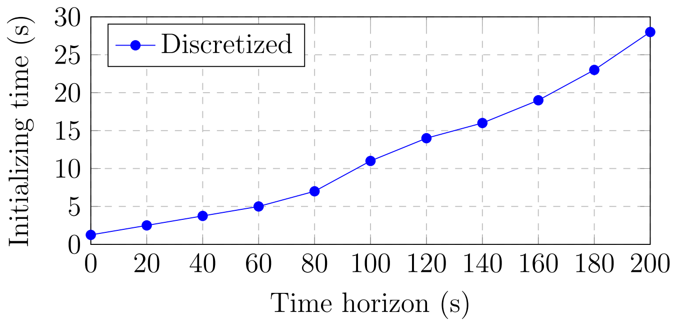


Figure 6.22: Required time to initialize the 4D grid, depending on the time horizon. The simulations have been performed assuming a common volume of 50x50x10 meters.

As a consequence, the results indicate that the discretized based method detects all possible conflicts in a more efficient manner than the brute force method. In addition, the performance of the algorithm improves as the number of UAS and waypoints per trajectory increases. However, depending on the size of the scenario and the time horizon, the findings reveal a growing storage capacity required and initialization time.

6.2 Autonomous detect and avoid experiments

We carried out several experiments for the EU SAFEDRONE project[12] for testing UAS autonomous functionalities in the U-space context. We examined the integration of onboard DAA capabilities, as well as the consequences for U-space services. This section details the experiment done in the SAFEDRONE Open Day Demonstration at ATLAS, where an UAS with the appropriate sensors flew a previously approved flight plan at a very low altitude and encountered an unanticipated static ground obstacle. Then, the UAS had to compute an alternative flight plan on its own to avoid the obstacle. A multirotor platform was chosen for this situation because of its higher manoeuvrability compared to fixed-wing systems, as well as its ability to hover while it awaits approval of the new flight plan.

[12]https://cordis.europa.eu/project/id/783211

The UAS was fitted with an on-board computer (Intel NUC i7) that ran Ubuntu 16.04 as the operating system and ROS Kinetic as the autonomous behavior framework. The NUC was attached to the autopilot and could interact with the ground section through a Ubiquity Rocket M5-based radius link. As a result, the GCS could command and monitor it. The UAS employed a Pixhawk autopilot with APM firmware, which could not only execute preprogrammed flight plans but also track online flying orders from a remote computer using the MAVLINK protocol. The UAS was equipped with a 3D-LIDAR sensor (Ouster OS1-16) to acquire a point cloud of the surrounding area. A path planner based on the Lazy Theta* algorithm Faria et al. (2019) was used to generate an avoidance 3D path based on the octomap generated using the information provided by the 3D LIDAR sensor, until then the path planner was using a 2D LIDAR as input of the generation of the virtual world.

Two laptops were deployed as GCS. The first, which ran on Ubuntu 16.04 and ROS Kinetic, was in charge of giving the UAS particular commands (start mission, take off, land, etc.) as well as monitoring its autonomous behavior. The second, which ran on Windows 10, monitored the telemetry data of the UAS and forwarded it to the USSP, as well as communicated the reserved flight plan to the UAS and requested new flight plans based on the requests of the UAS, using the Mission Planner commercial control station software. The Unifly USSP was connected to the UAS telemetry via Mission Planner and was available over the internet. The flight plans were managed by the USSP, which reserved an OV of 50 meters of radius around the proposed flight plan. At ATLAS, the experiments were also conducted on a structure having a height of roughly 10 meters, which was the one used to be detected and avoided. The following procedure was put to test:

First, a flight plan was created in the Unifly USSP. Then, the Unifly USSP accepted the flight plan since no conflict was found. The flight plan was selected through the GCS and uploaded to the UAS. The operator sent the start command to the UAS, which took off and started following the flight plan. In the meantime, the UAS collected data from the environment using its onboard LIDAR and transformed it into an octomap. The UAS detected an unexpected obstacle that intersected the accepted flight plan, see Figure 6.23. Therefore, the autonomous path planner, Lazy Theta*, generated a safe alternative flight plan, see Figure 6.24. It was not necessary to request a new flight plan to the U-space system since the path deviation was not large enough to exceed the OV reserved in the system. Finally, the generated alternative flight plan was executed by the UAS avoiding the unexpected building.

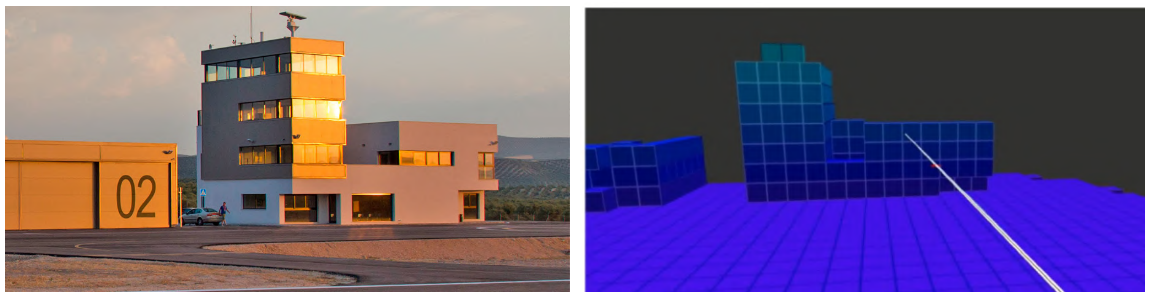


Figure 6.23: Left, ATLAS control tower. Right, RViz visualization of the octomap representation based on the LIDAR point cloud, and the conflictive flight plan under the ROS framework.

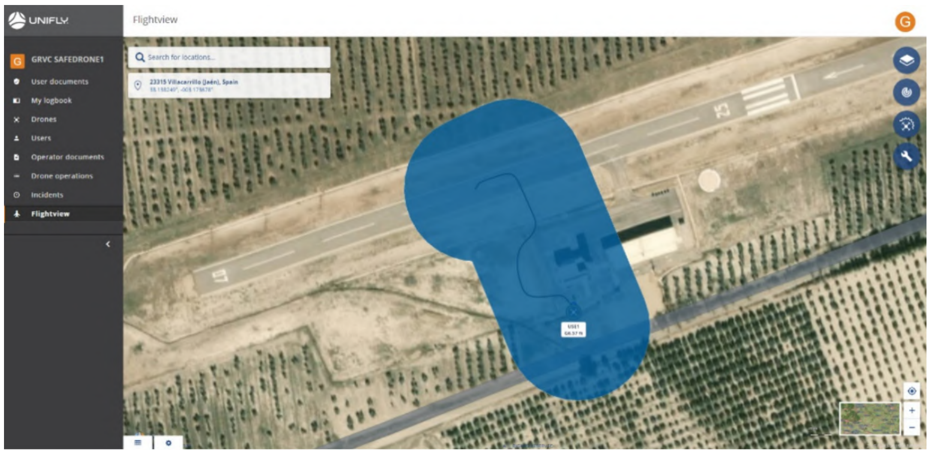


Figure 6.24: The Unifly U-space Service Provicer interface. Representation of the alternative flight plan carried out by the UAS and the OV reserved in the U-space system.

It should be noted that throughout the experiments, the flight plans were planned at a relatively low height of 8 meters to allow for the possibility of colliding with ground facilities. Furthermore, because the generated alternative flight plan was within the OV previously allowed by the USSP, there was no need to request a new flight plan reservation due to the obstacle avoidance maneouver being within the restrictions of the OV. The UAS trajectory shown in the Figure 6.24 was always in the reserved area throughout the demonstration flight.

Till date, no standards body has agreed on the minimum distance that an UAS should maintain when approaching static structures. This figure ranges from a few meters to hundreds of meters, depending on several characteristics such as the weight and type of the UAS. The minimum distance to the building throughout our flights was roughly 20 meters. Given that the data obtained by the 3D-LIDAR outside can be collected consistently at 30 $\sim$ 40 meters, keeping a distance of 20 meters from an unanticipated static obstruction appears to be appropriate.

Regarding the sensor range limitation that we had outdoors, we developed a flyby maneouver to safely follow large flight plans with the UAS. The main idea consisted in dividing the flight plan into segments, shorter than the maximum sensor range, and flying safely between segments. When the UAS reached the final point of a segment, it ensured that the next segment was collision free. The sensor could gain information about an unexpected obstacle while going through the segment, at this moment the UAS started hovering, and the path planner was called to generate an alternative route that started at the UAS position and ended at the final point of the segment, see Figure 6.25.

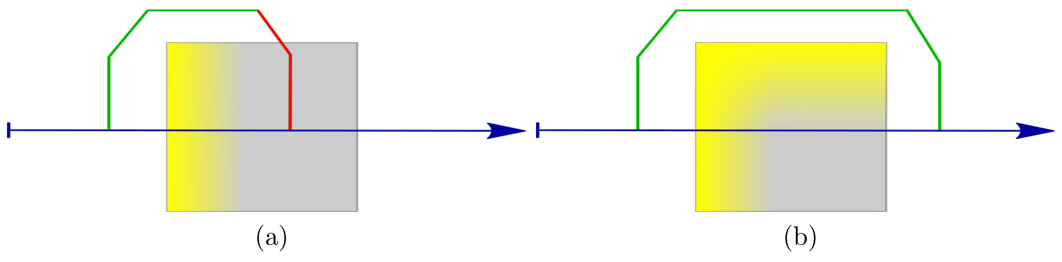


Figure 6.25: Top view of how lack of information may result in collisions. The flight plan is represented with a blue arrow, the green line is the new route generated by the path planner, and the red one is the segment that will collide into a building.
(a) First stage. With insuficient information the path planner generates an alternative route that will collide into the building.
(b) Second stage. The UAS followed a portion of the new route, gained information, detects a potential collision, and the path planner recalculated a new safe route.

There were cases in which the shape of the obstacle concealed part of it, causing the path planner to first give a solution that had to be rectified by gaining more information seconds later. To be able to gain as much information as possible, we decided to mount the 3D LIDAR in such a way that the majority of the sensor was facing forward and down trying to minimize blind spots. Figure 6.26 shows the problems caused by the blind spots of the LIDAR in three different stages.

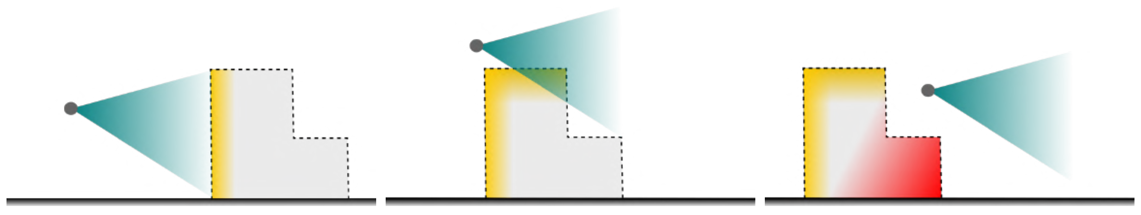


Figure 6.26: Side view of the problem caused by the blind spots of the LIDAR.
(a) First stage. The LIDAR detects a building wall, area colored in yellow.
(b) Second stage. The LIDAR detects the building roof, which concealed the rest of the building. The path planner may give a new route that may cause a further collision.
(c) Third stage. The remaining red parts of the building have been concealed by previous detections, therefore presenting a hazardous situation caused by no information gain due to the blind spots of the sensor.

Finally, after all flight demonstrations, the potential need for reporting unexpected static ground obstacles has been identified, and a new service may be defined to report their detection to the U-space system, allowing it to alert other UAS operators about potential new obstacles, minimizing risks even if these UAS do not have onboard sensors capable of detecting the obstacles on their own.

In terms of noncooperative detect and avoid capabilities, the demonstration flights proved the technological feasibility of incorporating these advanced capabilities into UAS to identify and avoid large unexpected obstacles autonomously. As a result, technical integration of U3 services such as detect and avoid is possible. This feature, however, might limit the nominal speed of the UAS due to the existing range limitation of lightweight 3D sensors. Furthermore, USSPs are not contemplating incorporating advanced capabilities into their systems at this time. It could be beneficial to produce special warnings, notifications, or instructions relating to these capabilities in the U-space system.

6.3 Conclusions and lessons learned

Thanks to the experiments carried out in ATLAS and El Arenosillo, we have learned valuable lessons related to the development of high-level U-space services. One of the more important is the need of good quality signal for communications, this is critical, and we did not face it until we arrived in El Arenosillo since we had no problems in ATLAS and in the preliminary tests regarding of communications. The use of the 4G technology was a requirement of the European GAUSS project, thus the connectivity was checked in the field before going to carry out several experiments, and in conclusion the connectivity was good in ATLAS, however in El Arenosillo the connectivity got worse as it went into the sea. It was concluded that the experiments in El Arenosillo may be affected by the poor health of the 4G signal just if the aircraft flew further than 10 km from the coast. On the first day of experiments in El Arenosillo, we tested the signal quality by flying the Atlantic and checking from where the communication was failing. We found that the signal deterioration was not progressive and, even worse, we had zones below 10 km away from the coast where the signal was completely gone. We carried out the experiments anyway suffering signal blinks, which caused that the UTM stayed blind during these communication losses without being able to track the active aircraft. It should be reminded here that the UTM received the telemetry via 4G, the operator was not losing the signal because it used another communication system, therefore in the perspective of the operator everything was going smooth. Due to the scalability that U-space wants to have, the communication with the aircraft must be a top priority, we can not afford less than excellent signal health, otherwise the U-space services will not be able to know when the UAS is having problems with the connectivity, where it is, what it is doing, and the UTM will not be able to communicate with the operators in case of a potential conflict. We had a radar at our disposal in El Arenosillo to double-check the aircraft position, but it is not reasonable to think about covering a country with radars to solve the communication problems in case of poor health of the signal. Another approach is to develop several algorithms to filter the telemetry reception through the internet and to estimate the trajectory of the UAS during a short period of time. If this time window is exceeded, the UTM should treat the aircraft as noncooperative, and warn the rest of the UAS that fly around it about that noncooperative UAS. This is a very dangerous situation due to the unknown position, course, and velocity of the noncoperative aircraft, we can make estimations, but it can change completely these mentioned parameters, making the estimations useless. Taking into account the experience gained through the experiments, we propose to stablish no-fly zones where the health of the signal is not good enough to track properly active aircraft, which can be done in a preflight stage where the operator is not allowed to fly in certain areas. The levels of danger involved in letting an UAS fly in a bad signal area need to be studied in depth.

We had the opportunity to conduct experiments with different operators during the project, which allowed us to learn how the operator interacts with the messages sent by the UTM and how well could them specify the parameters needed by the system. Taking into account all scenarios tested in ATLAS and in El Arenosillo, operators spent between 5 and 17 seconds, with a mean of 12 seconds, to accept a flight plan since they received it. It is pretty quick, and it is because they knew they were going to get an alternative flight plan, they even knew the main characteristics of the new route, thus the checks that they wanted to make were always considerably fast. Therefore, the times spent by the operators during the experiments are not a good reference to know how long an operator needs to check and accept or decline an alternative flight plan. Despite this, in the preliminary tests, we sent the operators several alternative flight plans when they were paying attention to other tasks, and the total time they needed to realize that the UTM was alerting them, to check all waypoints of the new route, to decide whether to accept or reject it, and to load it into the autopilot, was between 2 and 5 minutes. Apart from that, we found that operators had a hard time setting up the key parameters of their operations. Knowing that the OV is a cylinder that encloses the flight plan and that limits the error that the UAS can have while following its flight plan, the operators did not know which value needs the aircraft. Usually, the error is small while following a straight line, but when the flight plan has 90-degrees course changes, the aircraft can shorten the path or can reach the waypoint, and then head to the next one. This is a behavior that the operator should be aware of because it depends on the autopilot. In both cases, the operator must know how long the maximum distance the UAS can be from the flight plan. We found that operators overstimated this parameter as much as they could, which is a bad habit that can not be replicated when U-space services will be active. All operators will have to be aware that the airspace must be shared, therefore parameters such as OV must be rigorously established to optimize the airspace.

On the other hand, there was no autopilot capable of following a 4D flight plan, they could meet the space requirements but not the time requirements. Moreover, the operators knew that the UAS could follow the flight plan at a commanded cruising speed, however, they could not estimate the times of reaching a waypoint, which are needed by the UTM. Therefore, the Tracking module had difficulty updating the estimated trajectories of the UAS, because the times actually achieved by the aerial vehicles did not coincide with those stored in the Database in the preflight phase. It should be remarked here that the modules of the UTM work with the estimated trajectories to detect and solve potential conflicts that appear during the operations. Commercial autopilots will have to offer the possibility of using a 4D trajectory follower to the operators, and this must be standardized to ensure an efficient and secure access to numerous UAS to the airspace. U-space will need autopilots with this technology to facilitate the operator's registration of a flight plan, and to ensure that all aircraft involved within the U-space reached the waypoints at the times stored in the Database.

The requirements to identify possible conflicts in a multi-UAS situation, based on the 4D planned trajectories, might be handled simply by comparing each pair of waypoints from various UAS carefully. However, in large-scale applications, this strategy becomes ineffective. As a result, a strategy based on filling a 4D grid of cell objects is presented in Chapter 4 as a more efficient way to address the problem, even for large-scale applications. It is assured that only waypoints in nearby cells will be in dispute if the cell dimensions are properly chosen. As a result, the task is reduced to searching in nearby cells of the waypoint. During the experiments carried out in ATLAS and in El Arenosillo, the discretized based method was mainly used, however, we tested a few scenarios with the brute force based method presenting almost no performance difference between the methods due to the low number of UAS involved in the experiments. Therefore, we validated the discretized method in the same conditions as the brute force method giving almost the same performance. Thus the grid-based method may be a better solution in further experiments with an increased number of involved UAS.

Chapter 7

Conclusions and future work

The objectives of this Thesis are contrasted with its implementation and results in this chapter. Moreover, these results are discussed in the context of their limits and potential applications, pointing to possible future work directions.

7.1 Conclusions

This Thesis describes the design, integration, and validation of a set of modules that contribute to our UAS traffic management architecture for advanced U-space services. The architecture is adaptable, modular and scalable, with a focus on conflict detection and resolution features. It has no requirements for the aircraft's execution capability, runs at a high level, and is based on ROS. The UTM design can function without the need for human intervention, aiming to match the final goal of U-space. However, it advises actions to the UAS operator since, under existing regulations, the operator is the one who is responsible for carrying out the actions advised by the UTM. The advanced U-space services that compose the presented architecture are monitoring, tracking, tactical deconfliction and emergency management, in addition to two auxiliary services called Database and U-space service manager.

Monitoring and tactical deconfliction services are the main developments of the presented Thesis, in charge of detecting and solving potential conflicts that appear in the shared airspace of multiple UAS. The proposed conflict detection approach attempts to optimize the conflict search by confining it to a local search around each waypoint. In terms of the conflict resolution approach, it aims to reduce the deviation distance from the starting trajectory by breaking the problem down into smaller subproblems with only two waypoints. The validation results show that the conflict detection approach reduces the processing time when compared to other optimal methods based on exhaustive search, with the amount of improvement being highly dependent on the number of trajectories and the time horizon evaluated.

The suggested method for resolving potential conflicts is based on the premise that UAS can follow trajectories in time and space correctly. As a result, the resolution method only changes the conflicting waypoint placements, not their arrival timings, requiring the UAS's velocity adaption. While following a given trajectory, the designed UAS 4D trajectory follower fixes space and temporal inaccuracies. The mean minimum distance between the actual travelled trajectory and the estimated one, as well as the mean difference between the actual and estimated arrival timings to each waypoint, are minimized at all times. Moreover, the presented trajectory follower can fix preflight infeasible trajectories to increase its performance while following the trajectory.

ATLAS and El Arenosillo were the locations of the tests carried out thanks to the European projects SAFEDRONE and GAUSS, from which several conclusions have been drawn. First, commercial autopilots have to offer the possibility of using a 4D trajectory follower to the operators in order to ensure that all aircraft involved within the U-space reach the waypoints in time. It must be standardized to ensure an efficient and secure access to numerous UAVs to the airspace. Moreover, signal quality is a key aspect to take into account before carrying out an operation. It should be improved to ensure a continuous communication between the UTM and the UAS. Finally, during the tests we encountered misunderstandings between the operators, the pilots, and the UTM. These issues were resolved during the flight campaign itself thanks to the explanations given by all participants. Therefore, the training necessary for pilots and operators to adapt to communicating with the UTM cannot be overlooked.

Moreover, the discretized based technique was mostly employed in the field tests, but we also evaluated a few situations with the brute force based method, with essentially no performance difference between the methods due to the small number of UAVs participating in the experiments. As a result, we verified the discretized approach under the same conditions as the brute force approach, getting almost identical results, suggesting that the grid-based method may be a better answer in future studies involving a larger number of UAVs.

Finally, we examined the integration of onboard detect and avoid capabilities, as well as the consequences for U-space services. A multirotor with an onboard 3D LIDAR flew a previously approved flight plan at a very low altitude and encountered an unanticipated static ground obstacle. The need for reporting unexpected ground obstacles has been identified, and a new service may be defined to report their detection to the U-space system, allowing it to alert other UAS operators about potential new obstacles, reducing risks even if these UAS lack onboard sensors capable of detecting the obstacles on their own.

7.2 Future work

The research described in this Thesis presents an UAS traffic management architecture. Nevertheless, these solutions should be considered as a first step in a long-term research effort. Although the system performs well, there is still room for improvement in its performance and the development process, which raises new questions and opens up new lines of research. The following are the current and future work lines drawn from this Thesis.

First, the real tests should be carried out increasing the number of UAS, since some methods proposed in this Thesis claim to be scalable. Simulation results have demonstrated this scalability, however, we were unable to test the scalability of the proposed methods in real tests due to current regulations and project limitations. Therefore, it would be interesting to see how they work in scenarios with a larger number of UAS. Scalability can not be ignored even in the early stages of the development of the U-space services to have a successful deployment of these functionalities in the future.

Due to the scalability that U-space seeks, communication with the aircraft must be a top priority. Otherwise, the U-space services will be unable to detect when the UAS is experiencing connectivity issues, where it is, and what it is doing. Moreover, the UTM will be unable to communicate with the operators in the event of a potential threat. Future developments should obtain better quality using 5G networks, or even 6G networks in urban areas, which will enhance the speed, reliability and coverage of current networks.

Another interesting research line related to scalability is parallelization. The scalable discretized method chosen for this Thesis has been compared with a brute force method, which theorically presents a lack of scalability. An interesting research would be to parallelize the checks between segments carried out by the brute force method to find conflicts. Instead of checking a segment in the processor one after the other, a performance improvement could be found by parallelizing the checks using graphics cards. U-space does not aim to work in a determined volume of airspace, making a parallelized brute force method worthy of study since it does not discretize the airspace.

On the other hand, the sensors in charge of detecting obstacles should be improved. In this Thesis, tests were carried out using an onboard 3D LIDAR to detect large unexpected static ground obstacles, which gave us reliable measurements only below 30 meters in daytime conditions. These types of sensors will continue to improve to the point where they will be able to detect other drones as well as the ones used on the ground, like active and passive radars, acoustic sensors, and cameras. Future work can study the integration between counter UAS systems and sensors with the UTM architecture, this integration can detect noncooperative UAS flying in the monitored airspace, in addition to confirming that the cooperative UAS are following correctly its operations.

Finally, there are certain practical limits to our UAS traffic management architecture. It is based on a centralized server, which necessitates continuous contact with the other participants. Another research line would be the study of splitting the UTM into a group of dispersed and networked servers to reduce the congestion related to the centralization of communications.

Appendix A

Brute force based method

This appendix presents another approach to detect and solve conflicts between 4D UAS trajectories. The method chosen for this Thesis is detailed in Chapter 4 and is the one that can be scaled without problems, therefore it is the final solution. Before that, a brute force method was developed in the first place to be able to make tests quickly in the field, a method that does not need to set up parameters to work properly. A brute force method was not a problem knowing that no more than five UAS were going to be tested simultaneously in the field experiments, and the method was not going to present a calculation time that would be prohibitively long due to the scalability. We called this module 'Continuous Monitoring' because it does not need to discretize the space neither time. The module that solves conflicts detected by the Continuous Monitoring should not discretize the space to keep the policy, and it is called 'Continuous Tactical Deconfliction'.

A.1 Continuous Monitoring module

This module works as the module detailed in Chapter 4. It detects conflicts between 4D trajectories and geofences. The following subsections detail how the method works in terms of brute force and how it calculates the distance between two 4D segments.

A.1.1 Detect conflicts between two 4D trajectories

Let the input of the problem be a set of N trajectories, each of them composed of n waypoints in four dimensions, three for space and one for time. First, the loss of separation between any pair of trajectories must be checked, therefore the brute force way checks two at a time, making a total of $N(N-1)/2$ trajectory checks. Now let us consider that the trajectory between points is linear, thus for n waypoints, there are $n-1$ segments in four dimensions. Therefore, for each pair of trajectory checks, every segment of $trajectory_i$ must be checked with each segment of $trajectory_j$, see Algorithm 3.

Algorithm 3: Brute force check of all segments of all trajectories with each other.

```
for i = 0 to trajectories.size()-1 do
    for j = i+1 to trajectories.size() do
        for k = 0 to trajectories[i].size()-1 do
            for l = 0 to trajectories[j].size()-1 do
                check(trajectories[i].segment[k], trajectories[j].segment[l])
            end
        end
    end
end
```

That makes $(n_i-1)(n_j-1)$ segment checks for each pair of trajectory checks. In a scenario with 100 trajectories, with 100 points within each trajectory, it makes an approximated total of 50 million checks. However, the checks in the experiments were not even close to the mentioned number of checks, in the worst case scenario we had 3 UAS flying at the same time, with 50 waypoints within each trajectory or even less, making a total of approximatly 1 hundred checks, which is completely affordable by the CPUs nowadays.

To detail how the check for each pair of segments may be computed, first, suppose we can describe the trajectory segment between points
$A = (x_A, y_A, z_A, t_A)$ and $B = (x_B, y_B, z_B, t_B)$ with the following set of parametric equations:

$$x = x_A + m(x_B - x_A) \tag{A.1}$$
$$y = y_A + m(y_B - y_A) \tag{A.2}$$
$$z = z_A + m(z_B - z_A) \tag{A.3}$$
$$t = t_A + m(t_B - t_A), \tag{A.4}$$

where $m \in [0, 1]$ and this 4D segment is denoted as $\mathbf{r}(m) = (x(m), y(m), z(m), t(m))$, where $\mathbf{r}(0) = A$ and $\mathbf{r}(1) = B$. Now suppose we want to calculate the distance between the 4D segments $\mathbf{r}_1(m)$ and $\mathbf{r}_2(m)$ from two different trajectories, as described by these sets of parametric equations:

$$x_1 = x_{A1} + m(x_{B1} - x_{A1}) \qquad x_2 = x_{A2} + m(x_{B2} - x_{A2}) \tag{A.5}$$
$$y_1 = y_{A1} + m(y_{B1} - y_{A1}) \qquad y_2 = y_{A2} + m(y_{B2} - y_{A2}) \tag{A.6}$$
$$z_1 = z_{A1} + m(z_{B1} - z_{A1}) \qquad z_2 = z_{A2} + m(z_{B2} - z_{A2}) \tag{A.7}$$
$$t_1 = t_{A1} + m(t_{B1} - t_{A1}) \qquad t_2 = t_{A2} + m(t_{B2} - t_{A2}). \tag{A.8}$$

Calculate the 3D distance between the segments may be difficult, because they may exist in different times t. However, if the t_α is taken from the segment that starts from A (or come to existence) last and the t_β from the segment that arrives to B (or ceases to exist) first, we may compare the 3D positions at any given time. Defining $t_\alpha = max(t_{A1}, t_{A2})$ and $t_\beta = min(t_{B1}, t_{B2})$ we can immediately say that, if $t_\alpha > t_\beta$, the two segments do not coexist in time and there is no possibility for them to generate any loss of separation nor collision. This is an early return condition that will be used in the implementation to save further calculations. On the other hand, if the two segments coexist for some time, it seems easier to calculate the distance between them considering the two subsegments that do coexist for all considered intervals $t \in [t_\alpha, t_\beta]$. Therefore, if we calculate

$$m_{\alpha 1} = (t_\alpha - t_{A1})/(t_{B1} - t_{A1}) \qquad m_{\alpha 2} = (t_\alpha - t_{A2})/(t_{B2} - t_{A2}) \tag{A.9}$$

$$m_{\beta 1} = (t_\beta - t_{A1})/(t_{B1} - t_{A1}) \qquad m_{\beta 2} = (t_\beta - t_{A2})/(t_{B2} - t_{A2}), \tag{A.10}$$

and insert the values into the correspondent segment equations, we now can describe two new segments contained in the original ones, see figure A.1, but within an unified time interval $[t_\alpha, t_\beta]$. In the new equations $\mathbf{r}_1(\mu)$ and $\mathbf{r}_2(\mu)$, where $\mu \in [0,1]$, the points can be expresed as $\mathbf{r}_1(0) = \alpha_1$, $\mathbf{r}_1(1) = \beta_1$, $\mathbf{r}_2(0) = \alpha_2$, and $\mathbf{r}_2(1) = \beta_2$.

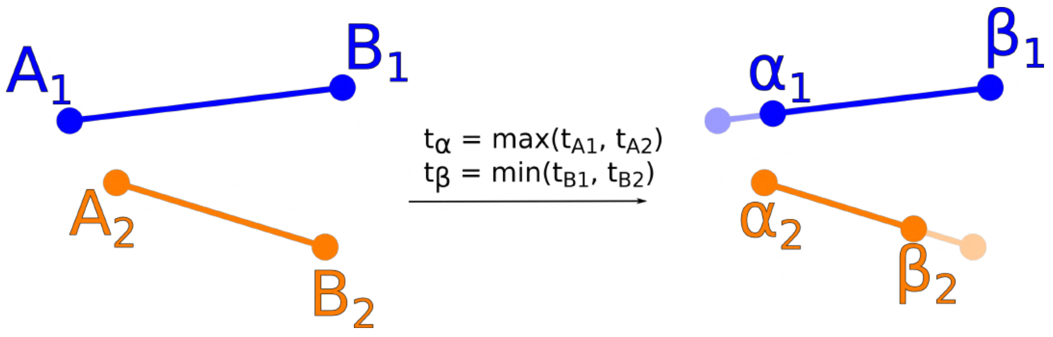

A_1
B_1
A_2
B_2
t_α = max(t_{A1}, t_{A2})
t_β = min(t_{B1}, t_{B2})
α_1
β_1
α_2
β_2

Figure A.1: Perspective view of a 3D example, where $t_\alpha = t_{A2}$ and $t_\beta = t_{B1}$. Segments are now defined in the same time interval.

We can see how the t equation is the same for both segments:

$$x_1 = x_{\alpha 1} + \mu(x_{\beta 1} - x_{\alpha 1}) \qquad x_2 = x_{\alpha 2} + \mu(x_{\beta 2} - x_{\alpha 2}) \tag{A.11}$$

$$y_1 = y_{\alpha 1} + \mu(y_{\beta 1} - y_{\alpha 1}) \qquad y_2 = y_{\alpha 2} + \mu(y_{\beta 2} - y_{\alpha 2}) \tag{A.12}$$

$$z_1 = z_{\alpha 1} + \mu(z_{\beta 1} - z_{\alpha 1}) \qquad z_2 = z_{\alpha 2} + \mu(z_{\beta 2} - z_{\alpha 2}) \tag{A.13}$$

$$t_1 = t_\alpha + \mu(t_\beta - t_\alpha) \qquad t_2 = t_\alpha + \mu(t_\beta - t_\alpha), \tag{A.14}$$

thus if we define Δx, Δy, and Δz as:

$$\Delta x = x_{\alpha 2} + \mu(x_{\beta 2} - x_{\alpha 2}) - [x_{\alpha 1} + \mu(x_{\beta 1} - x_{\alpha 1})] \tag{A.15}$$
$$\Delta y = y_{\alpha 2} + \mu(y_{\beta 2} - y_{\alpha 2}) - [y_{\alpha 1} + \mu(y_{\beta 1} - y_{\alpha 1})] \tag{A.16}$$
$$\Delta z = z_{\alpha 2} + \mu(z_{\beta 2} - z_{\alpha 2}) - [z_{\alpha 1} + \mu(z_{\beta 1} - z_{\alpha 1})], \tag{A.17}$$

we can describe the squared distance between the segments as:

$$s(\mu) = \Delta x^2 + \Delta y^2 + \Delta z^2. \tag{A.18}$$

As the equations are really similar for Δx, Δy, and Δz, let us make the math for one and then we can generalize for the rest. Therefore, if we develop Δx

$$\Delta x = \mu(x_{\beta 2} - x_{\beta 1} - x_{\alpha 2} + x_{\alpha 1}) + x_{\alpha 2} - x_{\alpha 1}, \tag{A.19}$$

renaming

$$\Delta x_\alpha = x_{\alpha 2} - x_{\alpha 1} \tag{A.20}$$
$$\Delta x_\beta = x_{\beta 2} - x_{\beta 1}, \tag{A.21}$$

we get

$$\Delta x = \mu(\Delta x_\beta - \Delta x_\alpha) + \Delta x_\alpha, \tag{A.22}$$

and finally we can square it

$$\begin{aligned}\Delta x^2 &= \mu^2(\Delta x_\beta - \Delta x_\alpha)^2 + 2\mu(\Delta x_\beta - \Delta x_\alpha)\Delta x_\alpha + \Delta x_\alpha^2 \\ &= a_x\mu^2 + b_x\mu + c_x,\end{aligned} \tag{A.23}$$

where

$$a_x = (\Delta x_\beta - \Delta x_\alpha)^2 \tag{A.24}$$

$$b_x = 2(\Delta x_\alpha \Delta x_\beta - \Delta x_\alpha^2) \tag{A.25}$$

$$c_x = \Delta x_\alpha^2. \tag{A.26}$$

If we generalize for y and z dimensions, we have that the equation for the distance squared is a parabola:

$$\begin{aligned} s(\mu) &= (a_x + a_y + a_z)\mu^2 + (b_x + b_y + b_z)\mu + (c_x + c_y + c_z) \\ &= a\mu^2 + b\mu + c. \end{aligned} \tag{A.27}$$

Now let us find the first and second derivatives:

$$\frac{ds}{d\mu} = 2a\mu + b \tag{A.28}$$

$$\frac{d^2 s}{d\mu^2} = 2a, \tag{A.29}$$

and use calculus to find extreme values:

$$\frac{ds}{d\mu} = 0 \rightarrow \mu^* = -\frac{b}{2a}. \tag{A.30}$$

And since a is a sum of squares (see equation A.24), the second derivative in equation A.29 is always positive, so μ^* is always a minimum, indicating the minimum distance between the two segments.

Now suppose that a threshold distance squared $s_{threshold}$ is defined so that, if at any time $s(\mu)$ is less than $s_{threshold}$, it means that a loss of separation is happening. We know that μ^* represents the minimum distance, but it is not assured to be inside the unified segments. However, we saturate its value to keep it in the interval $[0, 1]$ and calculate $\mu_{min} = max(0, min(\mu^*, 1))$ to get the μ corresponding to the minimum distance between the segments.

Again, an early return for the implementation is possible, as if $s(\mu_{min}) > s_{threshold}$, we can assure that there is no loss of separation between the two segments. Otherwise, we can calculate exactly values $\bar{\mu_0}, \bar{\mu_1}$ where the loss of separation starts and ends, solving the roots of:

$$a\mu^2 + b\mu + c - s_{threshold} = 0, \tag{A.31}$$

and as the roots $(\bar{\mu_0}, \bar{\mu_1})$ may be also outside the interval $[0, 1]$, we finally calculate:

$$\mu_{entry} = max(0, min(\bar{\mu_0}, 1)) \tag{A.32}$$

$$\mu_{exit} = max(0, min(\bar{\mu_1}, 1)). \tag{A.33}$$

That allow us to calculate the exact points in which each of the segments enters and exits the loss of separation region, which can be denoted as:

$$\gamma_1 = \mathbf{r}_1(\mu_{entry}) \qquad \gamma_2 = \mathbf{r}_2(\mu_{entry}) \tag{A.34}$$

$$\delta_1 = \mathbf{r}_1(\mu_{exit}) \qquad \delta_2 = \mathbf{r}_2(\mu_{exit}). \tag{A.35}$$

Figure A.2 shows where are the segments within the loss of separation conflict with the segments defined in the same time interval.

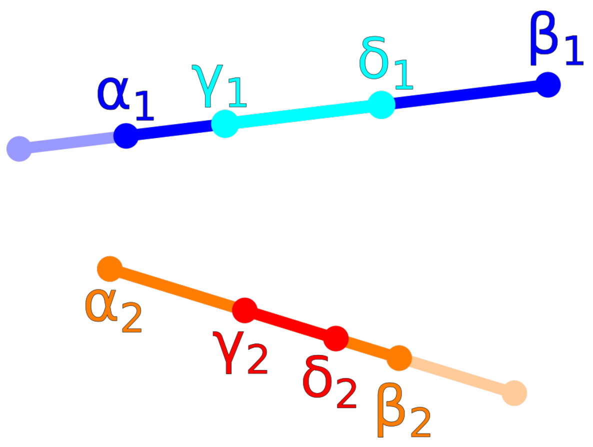


Figure A.2: Perspective view of a 3D example, where $\overline{\delta_1\gamma_1}$ and $\overline{\delta_2\gamma_2}$ are the segments within a loss of separation conflict

A.1.2 Detect conflicts between a 4D trajectory and a geofence

Let the input of the problem be a set of geofences and a set of trajectories. The geofences are composed of a center, a radius, and the minimum and maximum height. The trajectories are composed of a set of waypoints in four dimensions, three for space and one for time. First, each geofence is checked on every trajectory. Considering that the trajectory between points is linear: for n waypoints, we will have $n-1$ segments in four dimensions. Then, for each check, every segment of $trajectory_j$ must be checked with $geofence_i$.

Algorithm 4: Check every geofence for all trajectories

```
for $i = 0$ to geofences.size() do
    for $j = 0$ to trajectories.size() do
        for $k = 0$ to trajectories[j].size() do
            check(geofence[i], trajectory[j].segment[k])
        end
    end
end
```

To detail how the check of a point inside a cylinder may be computed, first, the height of the segment must be checked against the height of the geofence, if the points of the segments are completely above or under the geofence, there is no conflict. This early return is needed to avoid unnecessary calculations in the following steps. This type of conditions are used to save computational time in the implementation, which is key in the proposed system.

Once the component Z has been checked, the next step is to check if the segment, or a portion of it, is inside the geofence in the remaining dimensions X and Y. The following steps are similar as the ones described in Section A.1.1, check it for more details. Suppose we can describe the 2D trajectory segment between points

$A = (x_A, y_A)$ and $B = (x_B, y_B)$ with the following set of parametric equations:

$$x = x_A + m(x_B - x_A) \tag{A.36}$$

$$y = y_A + m(y_B - y_A), \tag{A.37}$$

where $m \in [0, 1]$. The equation of a circle with center (h, k) and radius r is $(x-h)^2 + (y-k)^2 = r^2$, but we will translate the segment to the center of the circle $(0, 0)$ so:

$$x^2 + y^2 = r^2. \tag{A.38}$$

Now we can develop the following equation to see the intersections between the segment and the circle:

$$(x_A + m(x_B - x_A))^2 + (y_A + m(y_B - y_A))^2 = r^2. \tag{A.39}$$

If we develop it we get:

$$\begin{aligned} & m^2(x_B - x_A)^2 + 2m(x_B - x_A)x_A + x_A^2 \\ & + m^2(y_B - y_A)^2 + 2m(y_B - y_A)y_A + y_A^2 \\ & = r^2, \end{aligned} \tag{A.40}$$

where for the component X

$$a_x = (x_B - x_A)^2 \tag{A.41}$$

$$b_x = 2(x_B x_A - x_A^2) \tag{A.42}$$

$$c_x = x_A^2, \tag{A.43}$$

and if we generalize it for Y

$$a = a_x + a_y \tag{A.44}$$

$$b = b_x + b_y \tag{A.45}$$

$$c = c_x + c_y, \tag{A.46}$$

we have that the equation for the radius is a parabola:

$$am^2 + bm + c = r^2. \tag{A.47}$$

We are now in a similar point as detailed in Section A.1.1 in Equation A.31, but here the threshold is the radius of the circle. We can calculate exactly values $\bar{m_0}, \bar{m_1}$ where the conflict with the geofence starts and ends, solving the roots of Equation A.47. And as the roots $(\bar{m_0}, \bar{m_1})$ may be also outside the interval $[0, 1]$, we finally calculate:

$$m_{entry} = max(0, min(\bar{m_0}, 1)) \tag{A.48}$$

$$m_{exit} = max(0, min(\bar{m_1}, 1)). \tag{A.49}$$

That allow us to calculate the exact points in which each of the segments conflicts on the circle, taking the components XYZ into account, and following the notation used in the previous section, the points can be denoted as $\gamma = \mathbf{r}(m_{entry})$ and $\delta = \mathbf{r}(m_{exit})$. We need to check if the time of the potential conflict segment is overlapping with the time of the geofence. For that, we can assume that the times of the segment are $Segment_{t_{start}}$ and $Segment_{t_{end}}$, and the times of the start and end of the geofence are $Geofence_{t_{start}}$ and $Geofence_{t_{end}}$, see Algorithm 5.

Algorithm 5: Check if a segment is overlapping in time with a geofence.

```
if $Segment_{t_{start}} \leq Geofence_{t_{end}}$ and $Segment_{t_{end}} \geq Geofence_{t_{start}}$ then
    return true
end
```

Finally, the Continuous Monitoring should determine the type of the threat, whether it is a Geofence Conflict or a Geofence Intrusion. The algorithm remains the same, but when it checks the first segment of the trajectory, if the first waypoint of that segment is inside the geofence, it is a Geofence Intrusion, otherwise it is a Geofence Conflict, see Figure A.3.

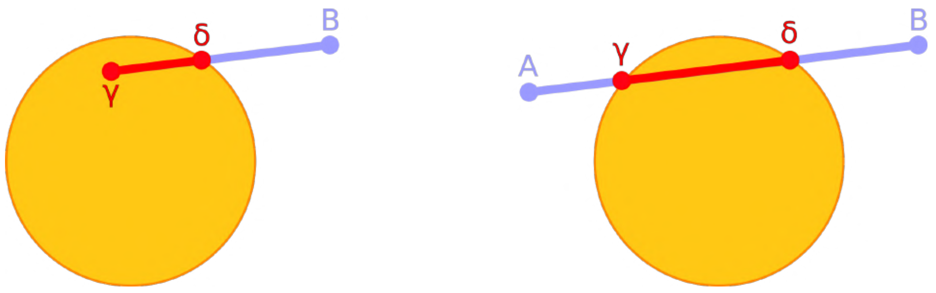


Figure A.3: Graphical representation of the possible conflicts that can appear between a 4D trajectory and a geofence. Left, geofence intrusion due to $\gamma = \mathbf{r}(m_{entry})$ inside the geofence. Right, geofence conflict due to $\delta = \mathbf{r}(m_{exit})$ matching the geofence border.

A.2 Continuous Tactical Deconfliction module

This module works as the module detailed in Chapter 4. The continuous version does not discretize the space either time, therefore it can not use a path planner like A* to respect the policy used in the continuous version. In this section, we will assume that the conflicting geofences are always cylinder shape.

A.2.1 Loss of separation

To solve a threat of type loss of separation, Continuous Tactical Deconfliction takes the trajectories involved in the conflict and the segments that break the separation minima. Each segment has a point where the distance between UAS is minimum, let us call them λ_1 and λ_2 to respect the annotation of subsection A.1.1. The module computes the unit vector $\hat{\overline{\lambda_2\lambda_1}}$ and place it in λ_1, this creates a vector pointing in the opposite direction of the conflict. The vector is multiplied by a safety margin and Continuous Tactical Deconfliction uses it to modify the position of the conflicting segment $\overline{\delta_1\gamma_1}$, it places the segment at the end of the vector, generating an alternative route avoiding the conflict that loss the separation minima, see Figure A.4.

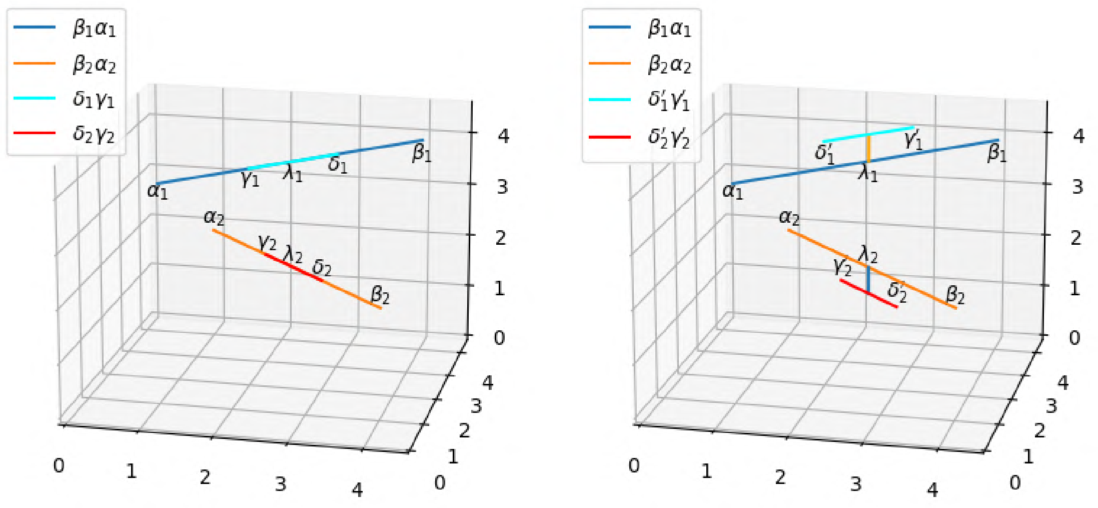


Figure A.4: Left figure shows a loss of separation conflict between the segments $\overline{\beta_1\alpha_1}$ and $\overline{\beta_2\alpha_2}$, their conflicting portions $\overline{\delta_1\gamma_1}$ and $\overline{\delta_2\gamma_2}$. Right figure shows how the module generates alternative routes replacing the conflicting portions $\overline{\delta_1'\gamma_1'}$ and $\overline{\delta_2'\gamma_2'}$.

Continuous Tactical Deconfliction does the same for the other conflictive segment, computing the unit vector $\hat{\overline{\lambda_1\lambda_2}}$, multiplying it by a safety margin and placing it in λ_2, see Algorithm 6. These are two alternatives routes that the module returns, but it also generates two alternatives that modify the time of the conflicting flight plan instead of the space. Continuous Tactical Deconfliction delays one flight plan until the other UAS has exit the conflicting segment. For example, if the module is trying to give an alternative route for the segment $\overline{\beta_1\alpha_1}$ it needs to delay the entry of the UAS in the conflicting portion $\overline{\delta_1\gamma_1}$. This portion starts at t_{δ_1} and ends at t_{γ_1}, so the Continuous Tactical Deconfliction adds a waypoint in δ_1 at t_{γ_1}, calculates the difference of time $\Delta t = t_{\beta_1} - t_{\alpha_1}$ and adds it to the rest of the waypoints delaying the remaining flight plan.

Algorithm 6: Modify the position of the conflictive segments to avoid the loss of separation conflict.

```
**Input:** $\gamma_1$, $\delta_1$, $\gamma_2$, $\delta_2$, $\lambda_1$, $\lambda_2$, $margin_{safety}$, $distance_{safety}$
**Output:** $\gamma_1'$, $\delta_1'$, $\gamma_2'$, $\delta_2'$
$\widehat{\overline{\lambda_1\lambda_2}} = \overline{\lambda_1\lambda_2}/|\overline{\lambda_1\lambda_2}|$
$\widehat{\overline{\lambda_2\lambda_1}} = \overline{\lambda_2\lambda_1}/|\overline{\lambda_2\lambda_1}|$
$distance_{\lambda_1\lambda_2} = \| \overline{\lambda_1\lambda_2} \|$
$distance_{avoid} = (distance_{safety} - distance_{\lambda_1\lambda_2}) \cdot margin_{safety}$
$vector_{\lambda_1} = -\widehat{\overline{\lambda_2\lambda_1}} \cdot distance_{avoid}$
$vector_{\lambda_2} = -\widehat{\overline{\lambda_1\lambda_2}} \cdot distance_{avoid}$
$\gamma_1' = \gamma_1 + vector_{\lambda_1}$
$\delta_1' = \delta_1 + vector_{\lambda_1}$
$\gamma_2' = \gamma_2 + vector_{\lambda_2}$
$\delta_2' = \delta_2 + vector_{\lambda_2}$
**return** $\gamma_1'$, $\delta_1'$, $\gamma_2'$, $\delta_2'$
```

A.2.2 Geofence Conflict

To solve a threat of type geofence conflict, Continuous Tactical Deconfliction takes the points of the start and end conflicting segment, joints them using a radial path between the points, taking into account the radius of the cylinder plus a safety margin. The Algorithm 7 runs as follows, first it needs to calculate the difference of the angles ($\Delta\alpha$) between the initial and final points. The algorithm also calculates the total length of the arc (L_{arc}) between the points taking into account the radius of the circumference and gets how many segments it should be divided into ($segment_{count}$). It should have at least two segments to go from the start point to the end point using an intermediate point. The algorithm is calculating the angles in two dimensions, X and Y, so the step in angle (α_{step}) is the difference in angles divided by the segment count. However, the algorithm should return a fourth dimensional path, so it also computes z_{step} and t_{step}. Finally, a loop must be done to fill the path with all intermediate points. They are calculated taking a safety margin into account to generate enough space between the output path and the conflicting geofence.

Algorithm 7: Create a path avoiding a geofence

```
Input: $point_{initial}$, $point_{final}$, $circle$, $margin_{safety}$, $segment_{min}$
Output: $path$
$\alpha_{initial} = \arctan(point_{initial_Y}, point_{initial_X})$
$\alpha_{final} = \arctan(point_{final_Y}, point_{final_X})$
$\Delta\alpha = \arctan(\sin(\alpha_{final} - \alpha_{initial}), \cos(\alpha_{final} - \alpha_{initial}))$
$L_{arc} = r \cdot \Delta\alpha$
$segment_{count} = max(2, \left\lfloor \frac{L_{arc}}{segment_{min}} \right\rfloor)$
$\alpha_{step} = \frac{\Delta\alpha}{segment_{count}}$
$z_{step} = \frac{point_{final_Z} - point_{initial_Z}}{segment_{count}}$
$t_{step} = \frac{point_{final_T} - point_{initial_T}}{segment_{count}}$
for $i = 0; i < segment_{count} - 1; i = i + 1$ do
    $\alpha_i = \alpha_{initial} + i \cdot \alpha_{step}$
    $waypoint_X = circle_X + (circle_r + margin_{safety}) \cdot \cos(\alpha_i)$
    $waypoint_Y = circle_Y + (circle_r + margin_{safety}) \cdot \sin(\alpha_i)$
    $waypoint_Z = V_{initial_Z} + i \cdot z_{step}$
    $waypoint_T = V_{initial_T} + i \cdot t_{step}$
    $path[i] \leftarrow waypoint$
end
return $path$
```

A.2.3 Geofence Intrusion

To solve a threat of type geofence intrusion, Continuous Tactical Deconfliction can proceed the same way as detailed in the previous and in Algorithm 7, however the conflicting segment that Continuous Monitoring gives has a portion inside the geofence. Therefore, Continuous Tactical Deconfliction should calculate the closest exit point, it adds a safety margin to the closest exit point to avoid further conflicts, and it computes the joint from the closest exit point to the flight plan outside the geofence using the radial method, see Figure A.5.

Algorithm 8 does not take the coordinate Z into account because it assumes that the closest exit point is at the same height as the UAS is at the time of the conflict.

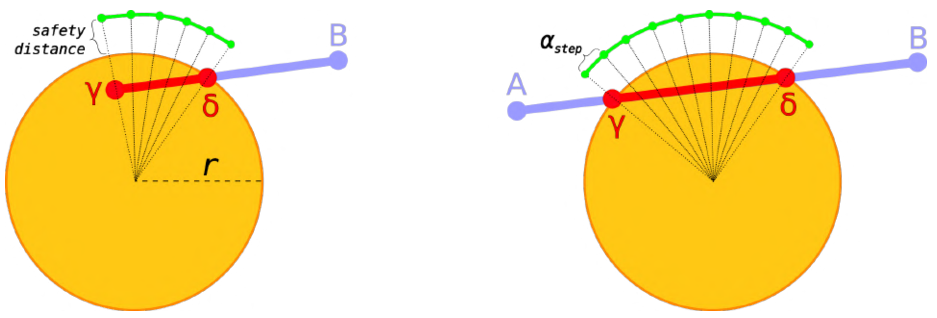


Figure A.5: Left, solution adopted to avoid a geofence intrusion. Right, solution adopted to avoid a geofence conflict. Both solutions are computed using the radial method explained in this section.

Algorithm 8: Calculate the closest exit point of a circumference

```
    Input: $point$, $circle$, $radius$
    Output: $exit$
    $\Delta X = point_X - circle_X$
    $\Delta Y = point_Y - circle_Y$
    $distance = \sqrt{\Delta X^2 + \Delta Y^2}$
    $exit_X = circle_X + \Delta X \cdot radius/distance$
    $exit_Y = circle_Y + \Delta Y \cdot radius/distance$
    return $exit$
```

Bibliography

Acevedo, J. J., Arrue, B. C., Maza, I., and Ollero, A. (2014). A decentralized algorithm for area surveillance missions using a team of aerial robots with different sensing capabilities. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4735–4740.

Acevedo, J. J., Capitán, C., Capitán, J., Castaño, A. R., and Ollero, A. (2020). A Geometrical Approach based on 4D Grids for Conflict Management of Multiple UAVs operating in U-space. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 263–270, Athens, Greece.

Acevedo, J. J., Castaño, A. R., Andrade-Pineda, J. L., and Ollero, A. (2019). A 4D grid based approach for efficient conflict detection in large-scale multi-UAV scenarios. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 18–23.

Airbus (2018). Airbus UTM. `https://www.airbus.com/en/innovation/autonomous-connected/unmanned-traffic-management/airbus-utm` (accessed on 19 November 2021).

AirMap (2018). Five Critical Enablers or Safe, Efficient, and Viable UAS Traffic Management (UTM). Technical report.

Alarcon, V., Garcia, M., Alarcon, F., Viguria, A., Martinez, A., Janisch, D., Acevedo, J. J., Maza, I., and Ollero, A. (2020). Procedures for the Integration of Drones into the Airspace Based on U-Space Services. *Aerospace*, 7(9).

Alonso-Ayuso, A., Escudero, L. F., Olaso, P., and Pizarro, C. (2013). Conflict avoidance: 0-1 linear models for conflict detection & resolution. *TOP*, 21(3):485–504.

Alonso-mora, J., Montijano, E., Schwager, M., and Rus, D. (2016). Distributed Multi-Robot Formation Control among Obstacles : A Geometric and Optimization Approach with Consensus. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5356–5363.

Alphabet (2022). Google's drone delivery business in Australia reaches 100,000 deliveries in 2021. `https://blog.wing.com/2022/03/dispatches-from-australia-supermarket.html` (accessed on 1 Mar 2022).

Amazon (2016). Amazon PrimeAir. `https://www.amazon.com/b?node=8037720011` (accesed on 27 August 2021).

ASD-STAN (2021). Direct Remote ID. Introduction to the European UAS Digital Remote ID technical standard. `https://asd-stan.org/wp-content/uploads/ASD-STAN_DRI_Introduction_to_the_European_digital_RID_UAS_Standard.pdf` (accessed on 5 January 2022).

Aweiss, A., Homola, J., Rios, J., Jung, J., Johnson, M., Mercer, J., Modi, H., Torres, E., and Ishihara, A. (2019). Flight Demonstration of Unmanned Aircraft System (UAS) Traffic Management (UTM) at Technical Capability Level 3. In *IEEE/AIAA Digital Avionics Systems Conference (DASC)*, pages 1–7.

Balampanis, F., Maza, I., and Ollero, A. (2017). Coastal Areas Division and Coverage with Multiple UAVs for Remote Sensing. *Sensors*, 17(4):808–832.

Barrientos, A., Colorado, J., Cerro, J. d., Martinez, A., Rossi, C., Sanz, D., and Valente, J. (2011). Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689.

Basilico, N. and Carpin, S. (2015). Deploying teams of heterogeneous UAVs in cooperative two-level surveillance missions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 610–615. IEEE.

Bernard, M., Kondak, K., Maza, I., and Ollero, A. (2011). Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, 28(6):914–931.

Besada, J., Campaña, I., Bergesio, L., Bernardos, A., and de Miguel, G. (2020). Drone flight planning for safe urban operations. *Personal and Ubiquitous Computing*, pages 1–20.

Besada-Portas, E., de la Torre, L., Jesus, M., and de Andrés-Toro, B. (2010). Evolutionary trajectory planner for multiple uavs in realistic scenarios. *IEEE Transactions on Robotics*, 26(4):619–634.

Butterworth-Hayes, P. and Mahon, T. (2021). The market for UAV traffic management services 2021-2025. `https://www.unmannedairspace.info/wp-content/uploads/2022/01/Unmanned-airspace-forecast-report-Edition-4.2.2021.December-2021.sample.pdf` (accessed on 8 February 2022).

Capitan, C., Perez-Leon, H., Capitan, J., Castaño, A., and Ollero, A. (2021). Unmanned Aerial Traffic Management System Architecture for U-Space In-Flight Services. *Applied Sciences*, 11(9):3995.

Caraballo, L.-E., Montes-Romero, Á., Díaz-Báñez, J.-M., Capitán, J., Torres-González, A., and Ollero, A. (2020). Autonomous Planning for Multiple Aerial Cinematographers. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1509–1515.

Carramiñana, D., Campaña, I., Bergesio, L., Bernardos, A. M., and Besada, J. A. (2021). Sensors and communication simulation for unmanned traffic management. *Sensors (Switzerland)*, 21:1–29.

Castillejo-Calle, A., Millan-Romera, J. A., Perez-Leon, H., Andrade-Pineda, J. L., Maza, I., and Ollero, A. (2019). A multi-UAS system for the inspection of photovoltaic plants based on the ROS-MAGNA framework. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pages 266–270. IEEE.

Commission, E. (2021). European Network of U-space Stakeholders. `https://transport.ec.europa.eu/news/aviation-european-network-u-space-stakeholders-re-launched-2021-11-30_en` (accessed on 7 January 2022).

Consortium, J. (2017). Japan Unmanned System Traffic & Radio Management Consortium. `https://jutm.org/en/` (accessed on 8 October 2021).

CORUS (2019). Concept of Operations for U - Space Enhanced Overview.

CORUS (2020). U-space Concept of Operations. Technical Report October 2019.

Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

DHL-Ehang (2019). DHL Express launches its first regular fully-automated and intelligent urban drone delivery service. `https://www.dhl.com/global-en/home/press/press-archive/2019/dhl-express-launches-its-first-regular-fully-automated-and-intelligent-urban-drone-delivery-service.html` (accessed on 7 May 2021).

Enea, G. and Porretta, M. (2012). A comparison of 4D-trajectory operations envisioned for Nextgen and SESAR, some preliminary findings. In *28th Congress of the International Council of the aeronautical sciences*, pages 23–28.

EU-China-APP (2018). UAS Operation Management System. `https://rpas-regulations.com/wp-content/uploads/2018/06/1.2-Day1_0910-1010_CAAC-SRI_Zhang-Jianping_UOMS-_EN.pdf` (accessed on 8 October 2021).

FAA (2017). UAS Integration Pilot Program. `https://www.faa.gov/uas/programs_partnerships/completed/integration_pilot_program/` (accessed on 9 October 2021).

FAA (2019). UTM Pilot Program. `https://www.faa.gov/uas/research_development/traffic_management/utm_pilot_program/` (accessed on 9 October 2021).

Faria, M., Ferreira, A. S., Perez-Leon, H., Maza, I., and Viguria, A. (2019). Autonomous 3D Exploration of Large Structures Using an UAV Equipped with a 2D LIDAR. *Sensors*, 19(22):4849.

Farid, G., Hamid, H. T., Karim, S., and Tahir, S. (2018). Waypoint-based generation of guided and optimal trajectories for autonomous tracking using a quadrotor uav. *Studies in Informatics and Control*, 27(2):225–236.

Fossen, T. I., Breivik, M., and Skjetne, R. (2003). Line-of-sight path following of underactuated marine craft. *IFAC Proceedings Volumes*, 36(21):211–216.

GUTMA (2020). Designing UTM for global success. Technical report.

Ho, F., Geraldes, R., Gonçalves, A., Rigault, B., Oosedo, A., Cavazza, M., and Prendinger, H. (2019). Pre-Flight Conflict Detection and Resolution for UAV Integration in Shared Airspace: Sendai 2030 Model Case. *IEEE Access*, 7:170226–170237.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, 34(3):189–206.

Indra (2019). Indra Air Drones. `https://www.indracompany.com/en/indra-air-drones` (accessed on 19 November 2021).

ITG (2018). AIRUS. `https://itg.es/en/urban-air-mobility/` (accessed on 20 November 2021).

Karamouzas, I. and Guy, S. J. (2015). Prioritized Group Navigation with Formation Velocity Obstacles. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5983–5989.

Koenig, N. and Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3.

Kondak, K., Ollero, A., Maza, I., Krieger, K., Albu-Schaeffer, A., Schwarzbach, M., and Laiacker, M. (2015). *Unmanned Aerial Systems Physically Interacting with the Environment: Load Transportation, Deployment, and Aerial Manipulation*, pages 2755–2785. Springer Netherlands.

Kopardekar, P. (2015). Unmanned Aerial System (UAS) Traffic Management (UTM): Enabling Civilian Low-Altitude Airspace and Unmanned Aerial System Operations. (April 2014).

Kothari, M., Postlethwaite, I., and Gu, D.-W. (2010). A Suboptimal Path Planning Algorithm Using Rapidly-exploring Random Trees. *International Journal of Aerospace Innovations*, 2.

Kuenz, A. and Peinecke, N. (2009). Tiling the world — Efficient 4D conflict detection for large scale scenarios. In *2009 IEEE/AIAA 28th Digital Avionics Systems Conference*, pages 3.B.5–1–3.B.5–10.

Lalish, E. and Morgansen, K. A. (2012). Distributed reactive collision avoidance. *Autonomous Robots*, 32(3):207–226.

Lin, C. E., Chen, T., Shao, P., Lai, Y., Chen, T., and Yeh, Y. (2019). Prototype Hierarchical UAS Traffic Management System in Taiwan. In *Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–13.

Liu, W. and Hwang, I. (2011). Probabilistic Trajectory Prediction and Conflict Detection for Air Traffic Control. *Journal of Guidance, Control, and Dynamics*, 34(6):1779–1789.

Lottes, P., Khanna, R., Pfeifer, J., Siegwart, R., and Stachniss, C. (2017). UAV-based crop and weed classification for smart farming. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3024–3031. IEEE.

Lundberg, J., Palmerius, K. L., and Josefsson, B. (2018). Urban Air Traffic Management (UTM) Implementation In Cities - Sampled Side-Effects. In *IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–7.

MAVLink (2013). MAVLink: Micro air vehicle communication protocol. `https://mavlink.io/en/`.

Meier, L., Honegger, D., and Pollefeys, M. (2015). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *Proceedings - IEEE International Conference on Robotics and Automation*.

Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M. (2011). Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*, pages 2992–2997. IEEE.

Mellinger, D., Kushleyev, A., and Kumar, V. (2012). Mixed-Integer Quadratic Program Trajectory Generation for Heterogeneous Quadrotor Teams. *2012 IEEE International Conference on Robotics and Automation*, pages 477–483.

Mercado Velasco, G. A., Borst, C., Ellerbroek, J., van Paassen, M. M., and Mulder, M. (2015). The Use of Intent Information in Conflict Detection and Resolution Models Based on Dynamic Velocity Obstacles. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2297–2302.

Merino, L., Caballero, F., de Dios, J. M., Maza, I., and Ollero, A. (2012). An Unmanned Aircraft System for Automatic Forest Fire Monitoring and Measurement. *Journal of Intelligent and Robotic Systems*, 65(1):533–548.

Micaelli, A. and Samson, C. (1993). *Trajectory tracking for unicycle-type and two-steering-wheels mobile robots*. PhD thesis, INRIA.

Millan-Romera, J. A., Acevedo, J. J., Castano, A. R., Perez-Leon, H., Capitan, C., and Ollero, A. (2019a). A UTM simulator based on ROS and Gazebo. In *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, page 132–141. IEEE.

Millan-Romera, J. A., Perez-Leon, H., Castillejo-Calle, A., Maza, I., and Ollero, A. (2019b). ROS-MAGNA, a ROS-based framework for the definition and management of multi-UAS cooperative missions. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, page 1477–1486. IEEE.

Nash, A., Koenig, S., and Tovey, C. (2010). Lazy Theta*: Any-angle path planning and path length analysis in 3D. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24.

Nelson, D. R., Barber, D. B., McLain, T. W., and Beard, R. W. (2007). Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529.

Nuñez, H. E., Flores, G., and Lozano, R. (2015). Robust path following using a small fixed-wing airplane for aerial research. In *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pages 1270–1278. Institute of Electrical and Electronics Engineers Inc.

OneSky (2019). OneSky UTM. `https://www.onesky.xyz/utm-platform` (accessed on 18 November 2021).

Peinecke, N. and Kuenz, A. (2017). Deconflicting the urban drone airspace. *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, 2017-Septe.

Perez-Leon, H. (2020). UPAT Follower: UAV Path and Trajectory Follower. `https://github.com/hecperleo/upat_follower` (accesed on 17 February 2022).

Perez-Leon, H., Acevedo, J. J., Maza, I., and Ollero, A. (2020a). A 4D trajectory follower based on the 'Carrot chasing' algorithm for UAS within the U-space context. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, page 1860–1867. IEEE.

Perez-Leon, H., Acevedo, J. J., Maza, I., and Ollero, A. (2021a). Integration of a 4D-trajectory Follower to Improve Multi-UAV Conflict Management Within the U-Space Context. *Journal of Intelligent and Robotic Systems*, 102(3):62.

Perez-Leon, H., Acevedo, J. J., Millan-Romera, J. A., Castillejo-Calle, A., Maza, I., and Ollero, A. (2020b). An Aerial Robot Path Follower Based on the 'Carrot Chasing' Algorithm. In Silva, M. F., Luís Lima, J., Reis, L. P., Sanfeliu, A., and

Tardioli, D., editors, *Robot 2019: Fourth Iberian Robotics Conference*, volume 1093, page 37–47. Springer International Publishing.

Perez-Leon, H., Braza, A., Jose Joaquin, A., Capitan, C., and Real, F. (2021b). GAUSS UTM system architecture. `https://github.com/grvcTeam/gauss` (accesed on 17 February 2022).

Pham, H. X., La, H. M., Feil-Seifer, D., and Deans, M. C. (2018). A Distributed Control Framework of Multiple Unmanned Aerial Vehicles for Dynamic Wildfire Tracking. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–12.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. *ICRA workshop on open source system.*

Ramasamy, S., Sabatini, R., Gardi, A., and Kistan, T. (2014). Next generation flight management system for real-time trajectory based operations. In *Applied Mechanics and Materials*, volume 629, pages 344–349. Trans Tech Publ.

Real, F., Torres-Gonzalez, A., Ramon-Soria, P., Capitan, J., and Ollero, A. (2018). UAL: an abstraction layer for unmanned vehicles. In *2nd International Symposium on Aerial Robotics (ISAR).*

Real, F., Torres-González, A., Ramón-Soria, P., Capitán, J., and Ollero, A. (2020). Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, 17(4):172988142092501.

Rubio-Hervas, J., Gupta, A., and Ong, Y.-S. (2018). Data-driven risk assessment and multi-criteria optimization of UAV operations. *Aerospace Science and Technology*, 77:510–523.

Rumba, R. and Nikitenko, A. (2020). The wild west of drones: a review on autonomous- UAV traffic-management. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1317–1322.

Sacharny, D., Henderson, T. C., and Cline, M. (2020). Large-Scale UAS Traffic Management (UTM) Structure. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 7–12.

Sanchez-Cuevas, P. J., Ramon-Soria, P., Arrue, B., Ollero, A., and Heredia, G. (2019). Robotic System for Inspection by Contact of Bridge Beams Using UAVs. *Sensors*, 19(2).

Sarabakha, A. and Kayacan, E. (2019). Online Deep Learning for Improved Trajectory Tracking of Unmanned Aerial Vehicles Using Expert Knowledge. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7727–7733. IEEE.

SESAR (2016). European Drones Outlook Study. Unlocking the Value for Europe. `http://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf` (accessed on 8 January 2022).

SESAR (2017). U-space Blueprint. pages 2015–2019.

SESAR (2019). Supporting safe and secure drone operations in Europe. `https://www.sesarju.eu/node/3530` (accessed on 10 December 2021).

Shoufan, A. and Alkadi, R. (2021). Integrating Counter-UAS Systems into the UTM System for Reliable Decision Making.

Siqi, H., Cheng, S., and Zhang, Y. (2018). A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation. *Chinese Journal of Aeronautics*, 31(7):1579–1593.

Sujit, P. B., Saripalli, S., and Sousa, J. B. (2013). An evaluation of UAV path following algorithms. In *2013 European Control Conference (ECC)*, pages 3332–3337.

Tan, Q., Wang, Z., Ong, Y., and Low, K. H. (2019). Evolutionary Optimization-based Mission Planning for UAS Traffic Management (UTM). In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 952–958.

Tang, H., Robinson, J., and Denery, D. (2010). *Tactical Conflict Detection in Terminal Airspace*. American Institute of Aeronautics and Astronautics.

Thales (2017). ECOsystem: Decision support for improved aviation operations. `https://www.thalesgroup.com/en/ecosystem` (accessed on 18 November 2021).

Turpin, M., Mohta, K., Michael, N., and Kumar, V. (2014). Goal assignment and trajectory planning for large teams of interchangeable robots. *Autonomous Robots*, 37(4):401–415.

Unifly (2020). Unifly UTM platform. `https://www.unifly.aero/solutions/unmanned-traffic-management` (accesed on 3 September 2021).

Wolfgang, H., Kiesel, S., Tinka, A., Durham, J. W., and Ayanian, N. (2019). Persistent and Robust Execution of MAPF Schedules in Warehouses. *IEEE Robotics and Automation Letters*, 4(2):1125–1131.

Xavier, D. M., Natassya Silva, B. F., and Branco, K. (2018). Comparison of path-following algorithms for loiter paths of Unmanned Aerial Vehicles. In *Proceedings - IEEE Symposium on Computers and Communications*, volume 2018-June, pages 1243–1248. Institute of Electrical and Electronics Engineers Inc.

Xu, C., Liao, X., Tan, J., Ye, H., and Lu, H. (2020). Recent research progress of unmanned aerial vehicle regulation policies and technologies in urban low altitude. *IEEE Access*, 8:74175–74194.

Yang, J., Yin, D., Niu, Y., and Zhu, L. (2015). Unmanned aerial vehicles conflict detection and resolution in city airspace. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2436–2441.

Yoder, L. and Scherer, S. (2016). Autonomous Exploration for Infrastructure Modeling with a Micro Aerial Vehicle. In Wettergreen, D. S. and Barfoot, T. D., editors, *Springer Tracts in Advanced Robotics*, volume 113 of *Springer Tracts in Advanced Robotics*, pages 427–440. Springer International Publishing, Cham.

Yu, J. and Lavalle, S. M. (2016). Optimal Multirobot Path Planning on Graphs : Complete Algorithms and Effective Heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177.