Alma Mater Studiorum
Università di Bologna

SCHOOL OF ENGINEERING
Forlì Campus

SECOND CYCLE MASTER'S DEGREE in
AEROSPACE ENGINEERING
Class LM-20

GRADUATION THESIS
in Aerospace Technologies and Materials

ADS-B driven implementation of an Airport Control Tower Augmented Reality platform

Student:
Tommaso Fadda

Supervisor:
Prof. Sara Bagassi
Co-supervisor:
Prof. Matteo Zanzi
Eng. Marzia Corsi
Examiner:
Prof. Fabio Olivetti

Academic year 2021/2022

Abstract

Over the years, many instruments have improved airport operations safety, yet making the controllers spend more and more time looking at those instead of looking at the actual air traffic outside the tower. This tendency reduces the operator's situational awareness and increases her/his workload due to continuous gaze shifting between terminals on the controller's working position and traffic. In addition, the increase in safety levels goes in the opposite direction with respect to the efficiency of the air traffic flow, with more instruments and longer procedures involved in the control action. Moreover, in Europe, this situation has been worsened by the chronic suffering of European airspace in terms of route congestion and flight delays. Within the SESAR research partnership, at the University of Bologna, the SESAR DTT Solution 97.1-EXE-002 project has validated in a simulated airport environment the use of a head-mounted display for augmented reality which visualizes the information the operators usually employ in a head-up rather than on terminals and improves operations in low-visibility. Since the project was carried out in a simulated environment, the transfer of the technology to the real world is needed. This work proposes to design and validate an Augmented Reality based platform for the real-time visualization of aircraft in the airport traffic zone. As additional features, the developed platform considers future integration with airport surveillance systems and task-related adaptivity. Thus, the integration of a present weather interface and the possibility to tailor the application to different conditions have been considered. The design followed the user-centred design methodology and joined the use of tracking labels and a weather interface – which resulted from previous projects - with the specific requirements of a real-world implementation of augmented reality and airport surveillance, including the capacity of tracking the aircraft in real-time and designing suitable labels to point out them and provide surveillance information. A conceptual design has been proposed, in which an ADS-B data stream is used to feed the tracking position information to a HoloLens Head Mounted Display, which runs an application to read the ADS-B data and transform them in coordinates for the holograms, visualizing the Aircraft Tracking Labels in correspondence to the real aircraft positions in the physical world. This work describes the passages leading to the final design, namely the decoding of the ADS-B data and their transfer to the HoloLens device, the registration process of the HoloLens with the real world and the implementation of the real-time tracking system of the detected aircraft. The implementation of a real weather interface and the tracing of the runway overlay completed the application and demonstrated its validity in implementing other surveillance systems in a head-up position. A first technical check on the design goodness was done with the help of an ATC operator, and a technical validation followed ensuring the repeatability and reproducibility of the application configuration for a specific environment while fulfilling all the requirements. Future work could complete the fundamental steps for implementing the application in a shadow mode in the tower, by implementing it with tower control airport systems and then testing in a real environment the solutions studied in the previous research campaigns. As a final remark, being SESAR a comprehensive and organic research partnership, the solution developed for this work could be shared in other solutions, such as the remote control towers.

Contents

List of Figures

List of Tables

List of Acronyms

ACAS	Airborne Collision Avoidance System
ACC	Area Control Center
ADS-B	Automatic Dependent Surveillance - Broadcast
ADVS	Air Traffic Advisory Service
ALRS	Alerting Services
ANS	Air Navigation Services
APP	Approach and Terminal Control
AR	Augmented Reality
ASM	Airspace Management
ATC	Air Traffic Control
ATCO	Air Traffic Control Operator
ATFM	Air Traffic Flow Management
ATM	Air Traffic Management
ATS	Air Traffic Service
ATSU	Air Traffic Service Unit
ATZ	Aerodrome Traffic Zone
AWY	Airway
CTA	Control Area
CTR	Control Zone
DF	Downlink Format
DTT	Digital Technologies for Tower
EOBT	Estimated Off-Block Time
ERA	European Research Area
ES	Extended Squitter
FIR	Flight Information Region
FIS	Flight Information Service
FP	Framework Programme
GND	Ground
GNSS	Global Navigation Satellite System
H2020	Horizon 2020
HITL	Human-in-the-loop
HMI	Human Machine Interface
IAS	Indicated Airspeed

ICAO	International Civil Aviation Organization
IFR	Instrumental Flight Rules
IMU	Inertial Measurement Unit
METAR	MEteorological Terminal Air Report
MLAT	Multilateration
MR	Mixed Reality
MRTK	Mixed Reality Toolkit
OOT	Out of the Tower View Generator
OTW	Out-of-the-Window
PSR	Primary Surveillance Radar
RETINA	Resilient Synthetic Vision for Advanced Control Tower
RV	Reality-Virtuality (Continuum)
RWY	Runway
SAR	Search and Rescue
SES	Single European Sky
SESAR	Single European Sky ATM Research
SESAR JU	SESAR Joint Undertaking
SSR	Secondary Surveillance Radar
TCAS	Traffic Collision Avoidance System
TCP	Transmission Control Protocol
TMA	Terminal Control Area
TWR	Aerodrome Control Tower
UCD	User Centered Design
UDP	User Datagram Protocol
UF	Uplink Format
UIR	Upper Information Region
UNIBO	University of Bologna
UWP	Universal Windows Platform
VFR	Visual Flight Rules
VR	Virtual Reality
WLT	World Locking Tool
XR	Extended Reality

Chapter 1

Introduction

Airport traffic control operators (ATCOs) are in charge of the safety and the orderly flow of air traffic at and in the proximity of airports. The airport controllers typically work on the top level of the airport control tower, a tall building from which the aircraft - and other vehicles - on the airport surface and those arriving at the airport are clearly visible. In fact, differently from other air traffic controllers, tower operators highly relate to the out-of-the-window view of the control tower to accomplish surveillance and provide separation and clearances.
Since the dawn of air traffic control services, many technological advancements have been designed to improve airport operations safety, usually resulting in instrumentation the ATCOs could consult to manage the traffic flow with higher situational awareness. As a drawback of these tools being provided as monitor interfaces the ATCOs have to focus on, the controllers' gaze has progressively moved from the out-of-the-window view to workstation terminals for ever-increasing percentages of the working shift. This tendency has actually reduced the operator's situational awareness and increased her/his workload with a continuous shift between the outside view and the work position terminals [1].
Beyond that, the increase in safety levels goes in the opposite direction with respect to the efficiency of the air traffic flow, as more instruments often result in longer procedures and more fragmented operations. This situation has been worsened by the global increase in air traffic volumes over the years [2], which are estimated to be growing even in the near and mid-term future[1]. The European airspace in particular suffered from route congestion and flight delays since the end of the previous century [5], and initiatives were taken to mitigate and counteract these trends.
As part of a general renovation of the whole European Air Traffic Management and airspace led by the Single European Sky ATM Research (SESAR) Joint Undertaking partnership, efforts have been directed also toward the development of new technologies for the airport control tower, aimed to improve the airport capacity and reduce the ATCOs workload while increasing their situational awareness and the overall safety of the system.
Following the successful use of Virtual/Augmented Reality and Synthetic Vision in piloting

[1]Worldwide air traffic volumes have been constantly growing in the last half-century, with the same trend predicted for the next decades [3][4]. Whether the Covid-19 pandemic has brought a substantial downfall in the number of both commercial and private flights for years 2020 and 2021, in 2022 the pre-pandemic traffic volumes have almost recovered worldwide, with a renewed growing trend expected in the next few years [2].

and pilots training activities for all-conditions visibility provision, as tested by SESAR [7], the possibility to transfer these technologies in the control tower has been considered after it was first proposed by Lloyd Hitchcock in 2006 [8]. Two different projects run within the SESAR research framework, RETINA and DTT Solution 97.1-EXE002, have explored the possibilities offered by Augmented Reality in eliminating the trade-off between airport safety and efficiency of operations and in improving the controller's workload and situational awareness. The investigation was led by the "Digital technologies and design methods for aerospace applications" research group of the University of Bologna, which tested different techniques in a simulated environment.
As a result of the validations done on RETINA and DTT, a final concept has been developed which uses a head-mounted display visualizing all the information the operators usually employ in a head-up position - then directly gazing out of the tower - rather than on terminals, and also improves operations capability in low-visibility conditions by means of an aircraft tracking labels system. Moreover, the developed system allowed the testing of additional concepts developed within the SESAR research.

1.1 Statement of the problem

Whether their successful outcomes, these projects are related to a virtual airport scenario simulation, and a real-environment application still misses. Furthermore, a general lack of experience exists in the application of Augmented Reality techniques in a real airport environment, with only a few research currently existing in its primaeval stages.
Many passages are required to bring an augmented reality airport solution from a simulation to a real environment, in terms of tracking technology (how to acquire the aircraft position information and track it in the virtual environment), surveillance and traffic management information visualization and maturity needed for deployment in operational environments. As a first stage, a framework should be developed for the general use of augmented reality techniques with real air traffic. Only at a later time could additional virtual overlays be added increasing the technology capabilities.
Given these objectives, the scope of the thesis follows. This work proposes to design, develop, implement and validate a platform for the real-time visualization of aircraft in the airport traffic zone, paving the way for future integration with all the control tower workstation systems. In doing this, some typical tasks which come out when dealing with augmented reality applications will be solved: the registration between the real and the virtual world, the real-time tracking of the aircraft of interest, and the correct blending of the elements of the real and virtual world.
As additional features, the developed platform should be suitably designed for future integration with airport surveillance systems and for the adaptation to multiple tasks. Thus, the integration of already available information - such as the weather one - and the possibility to tailor the application to different operative conditions (in terms of visibility or controller role, for instance) should be considered even in the early stages of development.
Finally, in the context of a user-centred design methodology, as required for the development

of user-sensitive solutions involving high-risk assignments, results coming from the validations of previous work will be integrated with the delivered solution.

1.2 Thesis structure

For the purpose of a better comprehension of this thesis, the general structure of the work is briefly described. Chapter 2 contains a literature review of the industrial framework inside which this work is collocated with a special focus on the research preceding and motivating this thesis project. Chapter 3 will then explain the methodology used in the application development and the general structure of the implemented solution. In chapter 4, the tools and devices used for the development and implementation of the system are presented. After that, chapter 5 will then describe the implementation of the designed solution and chapter 6 will explain the validation process which assessed the fulfilment of the requirements stated in chapter 3. Comments on the work done and on the possible future development will be assessed in chapter 7.

Chapter 2

Theoretical Background

In this chapter, a literature review of the main theoretical aspects foregoing and motivating the work done in this thesis project is presented.

2.1 Extended Reality

In computing, "virtual", as opposed to "real", is a term referring to anything that is not physically existing, but is likely something physical, while being generated by computer software. Referring to a generic user perception, a virtual element or image is generated by a computer and can be displayed making the user aware of it. In modern days, different display techniques are available which bring to an upper level the way the user interacts with and perceives these virtual elements. As the number of different technologies of this kind is large, the umbrella term "Extended Reality" (XR) has become common, referring to the whole of the technologies which alter reality by adding virtual elements to the physical world to any extent.
Being a generic concept, XR comprises several techniques and technologies that differ substantially from each other. Three main technologies are characterized by the use of virtual elements. The first two, "Virtual Reality" and "Augmented Reality", are the oldest and are only mildly linked. "Mixed Reality" instead, whether still being a young and malleable concept, somehow corresponds to a "missing link" between Virtual Reality and Mixed Reality (Fig. 2.1).

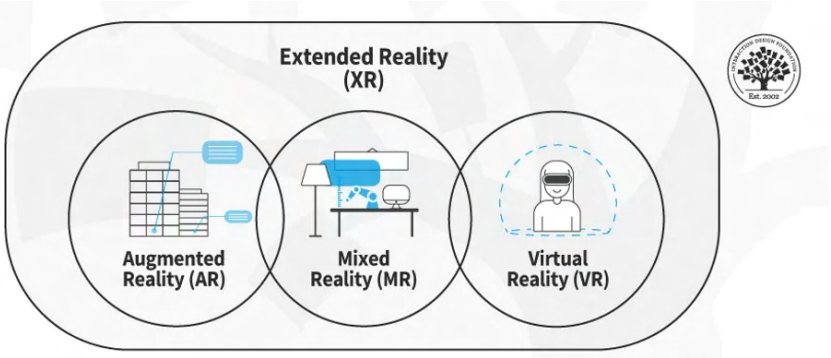


Figure 2.1: Extended Reality Technologies Representation. Copyright: Lara Tremosa and the Interaction Design Foundation.

To better understand the wide world of XR technologies, the main concepts characterizing them will be described in the next paragraphs, better defining the terms "Virtual Reality", "Augmented Reality", "Mixed Reality", and "Reality - Virtuality Continuum".

2.1.1 Virtual Reality

Virtual reality (VR) is a term that relates to different techniques to reproduce a three-dimensional computer-generated simulation of a real image or scenario (usually a fully-digital reproduction) in a way a user can interact with it by means of specific electronic equipment. In modern days, VR has often been related to the term "immersive", meaning that the user has to perceive the virtual scenario as if he is participating in it. However, Robertson et al. [9] state that the VR experience can also be non-immersive, as in the case of a computer-generated simulation which could be visualized through a display terminal[1]. At the same time, immersive VR has become predominant in the years as it unlocks the full potential of the technology. Immersive VR can be achieved by means of a wearable headset (such as Oculus Meta Quest 2, in Figure 2.2) with which the user experiences a full surrounding digital world that he can directly explore by means of real-world displacement or control interfaces. An immersive experience can be also realized by means of multiple projectors and screens surrounding the user and reproducing a virtual scenario as happens in a CAVE environment, like the one built at the University of Bologna (UNIBO) Virtual Reality and Simulation Laboratory (in Figure 2.3).

As reality is considered what we perceive with our senses, in an immersive virtual reality ex-

Figure 2.2: The Oculus Meta Quest 2 virtual reality headset.

perience the user is strongly isolated from the physical world perception while the synthetic[2]

[1]Could be the case of a 3D CAD file visualized on a PC monitor and interacted through a keyboard and a mouse device.

[2]In this case "synthetic" is used as a synonym of "virtual". In aeronautics, however, "synthetic vision" usually refers to a system used on aerial vehicles to enhance the pilot's situational awareness, so that it is similar to what will be later described as augmented reality technology.

environment he perceives is accurate and convincingly enough so that he is fully engaged with it [10][11]. In VR applications, this immersion is typically spatial, meaning the user perceives to be present in the synthetic scenario rather than externally observing it, being her/his involvement enough to lose perception of the real world and/or awareness of time.

Figure 2.3: Reconfigurable CAVE Environment at UNIBO Virtual Reality and Simulation Lab.

2.1.2 Augmented Reality

Whether it is immersive or not, a VR scenario is fully synthetic, in such a way as to be alternative to the physical world rather than complementary to it. At the same time, the way the user can assess this virtual world is fully controlled by the application implementation. The opposite happens instead while using augmented reality (AR) techniques. Augmented reality is in fact a technology consisting of the superimposition of an overlay of digital elements to a real-world scenario. Here the user physically interacts with the real world - moving inside an environment or handling physical objects for example -, while additional information is provided to enhance her/his experience of the scenario. This is the case for example of smartphone applications that provide the user with additional information about the object tracked by the phone camera (Figure 2.4).

Whether AR is used to enhance the user experience of the physical world, nevertheless the same opportunity is usually exploited and integrated into virtual scenarios, with the user experiencing a scenario that levels up the interacting and perceiving capacities he would have in the real world. Hence, the same enhanced features of an AR application are suitable for adding in a virtual reproduction of the real environment and vice versa. However, it is important to notice that this interchangeability is only available in a case of a real-world digital reproduction in VR, as a VR scenario can also overflow the boundaries of the real world, as the laws of physics, cutting off the links with the real world which obviously cannot be changed in AR applications.

Following this concept, it appears that VR and AR are indeed clearly distinguishable and

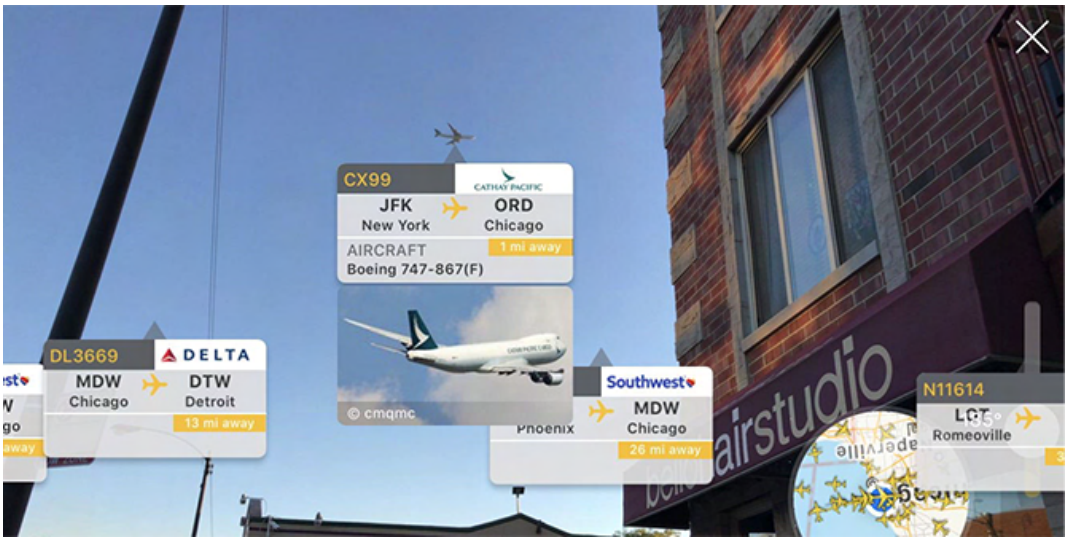


Figure 2.4: An example of an AR application provided by FlightRadar24 to visualize real aircraft in the space surrounding the user.

separated techniques, and this way they have been considered for years during their development [12]. VR separates the user from the real world; then, it does not have to be in any way related to existing reality, except for the fact the user interacts with his natural senses. In opposition, in AR the real world plays a dominant role. All that is represented in AR has to deal with the world behind which is not obliterated; so not only do the contents have to be related to the scene behind, but also the overlay has to be strictly positioned and organized dealing with the reality whose perception has to be "augmented".

2.1.3 The Reality-Virtuality Continuum

Whether VR and AR are not replaceable with each other, AR can still be presented as a spectrum of different shades of "augmentation"; different technologies may rely differently on real or virtual elements to recreate the final reality the user perceives. The term Extended Reality is well suited to comprise all these possibilities due to the previous definition in Section 2.1. Considering what VR and AR aim for, Extended Reality can be thought of as a container for all the technologies which use the virtual elements to boost the user's experience within or with respect to the real world (thus comprising also a full synthetic scenario, i.e. VR).
This concept was first introduced by Milgram et al. [13]. In 1994 they tried to define AR and its relation with VR through the introduction of the new concept of "Reality-Virtuality (RV) Continuum". While AR was being considered more and more unrelated to VR, in this new concept the two techniques are related in a way they occupy opposite positions of a continuum, precisely the RV continuum. In figure 2.5 the proposed concept is illustrated, in which two extremes represent the real environment and the virtual environment. Then on left, we have a reality solely composed of real elements. On the right, "virtuality" is a reality solely composed of virtual elements. Different existing and hypothetical AR technologies continuously occupy all the positions in the middle, which takes the generic name of Mixed Reality (MR) environment. While the authors of the study did not use the term "extended

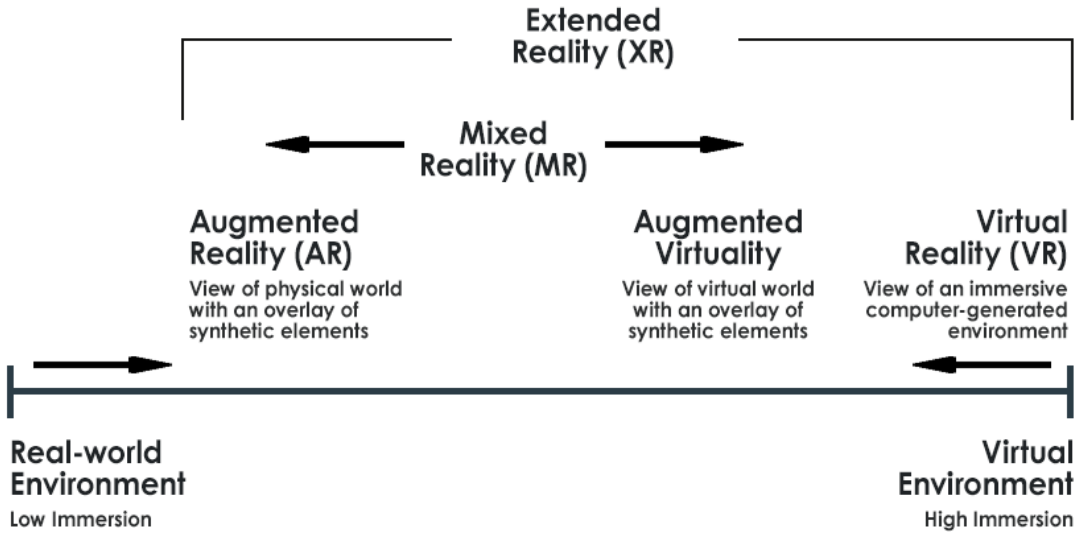


Figure 2.5: Virtuality-Reality (VR) Continuum concept. "Extended Reality" is a modern umbrella term which identifies all the techniques relying on virtual elements to "extend" the perception of reality.

reality", which has aroused only in the last decade, they introduced the term mixed reality (MR), meaning every technique that "blends" the physical and the virtual world introducing both virtual-world and real-world objects at the same time within the same display. Inside this framework, AR is well defined as a technique in which the user feedback of the world is enhanced with virtual clues. At the same time, also the introduced concept of "augmented virtuality" is self-explanatory looking at figure 2.5.
A great evolution of VR, AR and MR has been since 1994, and also the meaning of some terms has greatly changed over the years. In particular, many tech companies have provided their own definition of the terms based on their proprietary technologies and maybe on how they see the future of the industry. While AR was first intended as part of the MR techniques, today the terms are usually considered separate. Milgram's 1994 concepts have recently been reviewed [14] collecting various modern definitions of MR, which can be considered as a synonym of AR, as an extension of AR, as a combination of AR and VR or as Milgram's defined. For Intel® MR is strictly a blend of the real and virtual world in which the user can perceive and interact with both, in opposition to AR in which interaction is not allowed. Microsoft® considers AR the case of a virtual overlay of graphics onto video, as in the case of phone AR applications; MR instead needs a combination of the real and the virtual. This can happen with the Microsoft HoloLens® headset, where holograms can be seen to be placed over the real objects in the environment.
These different descriptions could be summarized by defining mixed reality (MR) as a technique in which "real world and virtual world objects and stimuli are presented together within a single percept" [14]. Still, this definition emphasizes the concept of MR as the blending operation of physical and virtual elements inside the RV continuum. While different techniques rely more on real rather than virtual environments, it is difficult to quantify each role, which

can be related to external conditions. Then, according to the RV continuum, an adaptable technology is possible, which can vary the amount and predominance of virtual elements, such as labels or overlays, which are presented to the user depending on the physical scenario evolution. An application of this idea is presented in section 2.4.4, where the application can be tailored to different operational situations that may occur in the airport control tower, in particular showing more virtual information in low-visibility conditions.

2.1.4 Different industrial applications of VR and AR/MR

AR and VR differences usually result in different usages. In industrial applications, immersive VR could be used to reproduce something that does not exist yet in reality as a machine. This way designers could be aware of their prototype which could be also tested by final users to enlighten ergonomics or safety issues for example. At the same time, virtual reality could be used to make it possible for a buyer to see in a realistic way a new machine before having it shipped, having clear characteristics such as the volume occupied or allowing to train in advance who is going to use it. The same could be extended to an entire working scenario, for example, training personnel on a full-scale scenario of a working environment under development.
AR instead is well exploited in already existing scenarios whose comprehension or usability can be improved by means of virtual information, for instance using an overlay to increase the operator's situational awareness over an environment he is monitoring. The information displayed could be used to make users aware of incoming risks, or to increase the amount of information about an object when looking at it, and in many other manners, also tailoring this information for each specific situation.

Since the aim of this thesis work is not to analyze the coexistence and interrelations of different extended reality technologies - whose definitions are eventually not univocal throughout literature -, and the technologies of interest for this work are somehow contained in the boundaries of augmented or mixed reality, in the subsequent pages the terms AR and MR should be considered as synonyms, generally referring to AR applications in which the user can somehow interact with both the virtual and the real objects. When the intended meaning for the previous terms will be different, this will be explicitly stated.

2.1.5 Considerations in AR implementation

As previously stated, VR and AR differ in that VR completely isolates the user from the real world, while in AR both the real and virtual worlds are present and blended together. Having in mind that it is much more difficult to control the real world than a completely synthetic one, the implementation of AR brings additional requirements with respect to VR for the experience to be adequate [15].

- **Registration**. Being the superposition of the virtual images dependent on the real world, spatial matching between the two worlds is needed which depends on the user's

perspective inside the scenario. This concept is known as registration, meaning the AR system has to be provided with devices able to map the environment and identify what is foreground, what is background, objects extension and dimension, etc. In some environments, such as the airport control tower, the virtual objects should then consider the position, depth of view, shape, dimension, orientation etc. of the real object.

- **Tracking**. For the registration process to be carried out, a complex tracking system is needed to obtain the relative position of the user and of the real world as well as the actual user's view of his surroundings. This system has to both acquire the real scenario configuration (e.g. by means of cameras), and the user orientation/position, which can be done by means of some inertial sensors and/or by reconstructing the tracked surrounding and analyzing the sensors' relative view inside it.

- **Display Technology**. Visualizing both the virtual world and the real world is a crucial aspect of AR and MR. Unlike VR, these techniques need a way to superimpose the virtual layer over the real world. This is done by either using video-based or optical-based display techniques. In the first case, the virtual content is merged with images of the real world captured from cameras, as in the case of smartphone-based AR applications. In optical-based techniques, instead, the user's direct view of the world is augmented with the projection of virtual content in the user's field of view. This is usually achieved thanks to see-through displays, which are transparent letting the user see the physical world behind the projected images. Head-mounted displays (HMDs) are the dominant technology, including the possibility for the user to move while keeping the tracking and displaying apparatus integral to him. If mobility is not a requirement, spatial displays, like a transparent window on which overlays are projected, can be easily used.

- **Real-Time capability**. To allow user-dependent spatial matching, all the evaluations of the changes in the scenario have to be real-time, with the virtual scenario which evolves according to the user's perception of the physical one. Then, while the user view is determined in the fastest possible way, also fast and efficient rendering methods are needed to reconstruct the virtual world.

2.2 Air Traffic Management (ATM)

This thesis relates to the operative environment of an airport control tower. Hence, this section describes Air Traffic Management and its different areas of interest comprising Air Traffic Control and more specifically the Aerodrome Control Services.
Air Traffic Management is an aeronautical term describing all the systems that support an aircraft in all phases of its operational journey - ground movement, the departure from an airport, the air transit (climb, cruise, descent) and the arrival at the destination airport - ensuring safe operations and transit throughout different airspace. Following the ICAO definition [17], ATM is defined as the "dynamic, integrated management of air traffic and airspace [...] - safely, economically and efficiently - through the provision of facilities and seamless services in collaboration with all parties and involving airborne and ground-based

functions". Air traffic management comprises three main services: Air Traffic Services (ATS), including Air Traffic Control (ATC), Air Traffic Flow Management (ATFM), and Airspace Management (ASM).

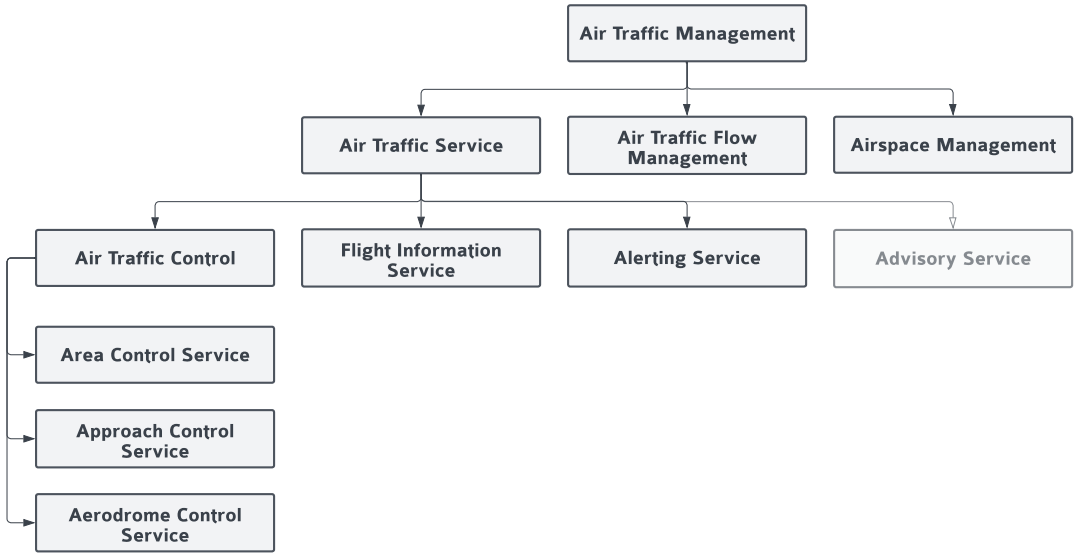


Figure 2.6: Air Traffic Management (ATM) structure.

Figure 2.6 summarises the division of ATM functions. Since this work deals with aerodrome control services, in the next paragraphs the focus will be on ATS, ATC service and ATC sub-activities.

2.2.1 Air Traffic Services

ATS has the purpose of ensuring safe and orderly traffic flow (mainly with the ATC service), providing the flight crews with all needed information (Flight Information Service - FIS), and alerting the appropriate bodies (e.g. Search and Rescue - SAR) in case of a reported or discovered emergency (Alerting Services - ALRS). A fourth service exists called Air Traffic Advisory Service (ADVS), used in uncontrolled airspaces to advise pilots of other aircraft or hazards preventing collisions. In fact, the main role of ATS is to prevent collisions, for instance by controlling flight levels and applying adequate separation standards, and to provide the flight crews with clearances and instructions for an orderly traffic flow. ATS is provided by controllers through tactical interventions and direct communication with the flight crew for the entire duration of the flight.
The ATS system can be split into airborne and ground-based segments. The airborne segment comprises all the elements connected to the aircraft and flight crew, with the ground segment referring to all the aeronautical infrastructures in their more general meaning. Among the objectives identified by the ICAO organization for the ATS, the main ones are to guarantee the subsistence of the safety levels required in the specific serviced area and the implementation of safety-related enhancements whenever necessary.
For the provision of ATS, specific air traffic service units (ATSUs) are designated. An ATSU

can provide more than one service; for instance, ATC units can also provide FIS. The needed services are designated depending on different factors that characterize the served area: traffic types (commercial or general aviation[3]) and density, available surveillance and communication equipment, geographical aspects, meteorological conditions or relevant others.
An ATS unit usually operates in a well-defined area called airspace. After analysis of the factors above, the airspace can be classified (Section 2.2.4), and adequate ATSU can be designated for a sector. Depending on the airspace class and the ATSU designated, different levels of service are provided. While an ATC unit can also provide FIS (usually with lower priority) and ALRS, the opposite usually does not apply. At the same time, an ATC unit can also be responsible for a sector of uncontrolled airspace (Section 2.2.4, Class G). In this situation, the ATC unit will provide only FIS and ALRS in that portion of airspace, and ATC services only in the controlled airspace sectors.
A key factor in defining ATS units is the system capacity, which is associated with each ATS system in terms of the traffic volumes it can manage in normal and peak conditions, which has to be always sufficient for all traffic conditions over an area. The capacity of the system depends on many factors including the ATS route structure, the aircraft navigation system's precision, weather-related aspects, and controller workload (which has not to be excessive). Following ICAO definitions "the capacity corresponds to the maximum number of aircraft which can be accepted over a given period of time within the airspace or the aerodrome concerned" [17]. Since the traffic volumes could greatly variate over time, even exceeding the airspace capacity, procedures have to be implemented to manage the flow and avoid airspace congestion; thus ATFM is needed.

2.2.2 Air Traffic Flow Management

ATFM has the primary objective of regulating the aircraft flow over airspace to avoid traffic congestion, particularly when the system capacity is exceeded in a given control sector. It also contributes to the "establishment of a safe, orderly and expeditious flow of air traffic by ensuring that ATC capacity is utilised to the maximum extent possible and that the traffic volume is compatible with the capacities declared by the appropriate air traffic service providers" [18]. ATFM ensures that the ATS supply meets the demand at each time staggering the traffic over time and space and managing the airspace capacity, for instance planning the number of controllers needed at each time. Also, certain traffic flows could be restricted (e.g. requiring flights to meet certain requirements to travel a given route). ATFM does not focus on actual traffic conditions but is pre-tactical, as it organizes the system's operability in the near future.

[3]General aviation (GA), also called sometimes general aviation /aerial work (GA/AW) comprises all civil aviation operations other than scheduled air services and non-scheduled air transport operations for remuneration or hire. Commercial aviation, or commercial air transport operations are those which involve the transport of passengers, cargo or mail for remuneration or hire.

2.2.3 Airspace Management

ASM has the purpose of managing scarce airspace resources in the most efficient possible way. ASM prevents mutual interference from all users of the airspace. Then, the need is to satisfy all airspace users, both civil and military; in order to do this, two aspects have to be considered: the way the airspace is allocated among different users, in terms of routes, flight levels, zones, and the way it is structured among different control zones to manage the provision of the different traffic services.

2.2.4 Airspace structure

ATS services are provided over specific airspace, where the term refers to the portion of atmosphere above the territory of a country, or more generally to any three-dimensional portions of the atmosphere with defined spatial and temporal dimensions. For ATS, airspace can be identified by means of the type of airspace, the air traffic services provided, the service provider inside the airspace, and the classification of the airspace. The different types of airspace are defined by ICAO [17][19][20]:

- **Aerodrome Traffic Zone (ATZ)**, an airspace of defined dimensions established around an aerodrome for the management of aerodrome traffic.
- **Control Zone (CTR)**, a controlled airspace extending upwards from the surface of the earth to a specified upper limit.
- **Terminal Control Area (TMA)**, a control area normally established at the confluence of ATS routes in the vicinity of one or more major aerodromes.
- **Control Area (CTA)**, a controlled airspace extending upwards from a specified limit above the earth.
- **Flight Information Region (FIR/UIR)**, an airspace of defined dimensions within which flight information service and alerting service are provided. UIR (upper information region) is a FIR in upper airspace.
- **Airway (AWY)**, a control area or portion thereof established in the form of a corridor.

ICAO further categorises airspace based on the adopted flight rules and on the interactions between aircraft and ATC. Some key concepts to better define the classes are:

- **Separation**, refers to a minimum distance (both horizontal and vertical) to be maintained between aircraft to mitigate collision risks or secondary issues, such as wake turbulence interference. Standards exist for both airborne and ground status, and the concept also applies to terrains, obstacles and airspace.
- **ATC clearance**, is an authorization for an aircraft to proceed under the indication of an ATC unit. It is usually abbreviated simply as "clearance". It is provided by ATC and is solely used for expediting and separating air traffic depending on known traffic conditions. It is used for both airborne and ground traffic conditions with regard to in-air

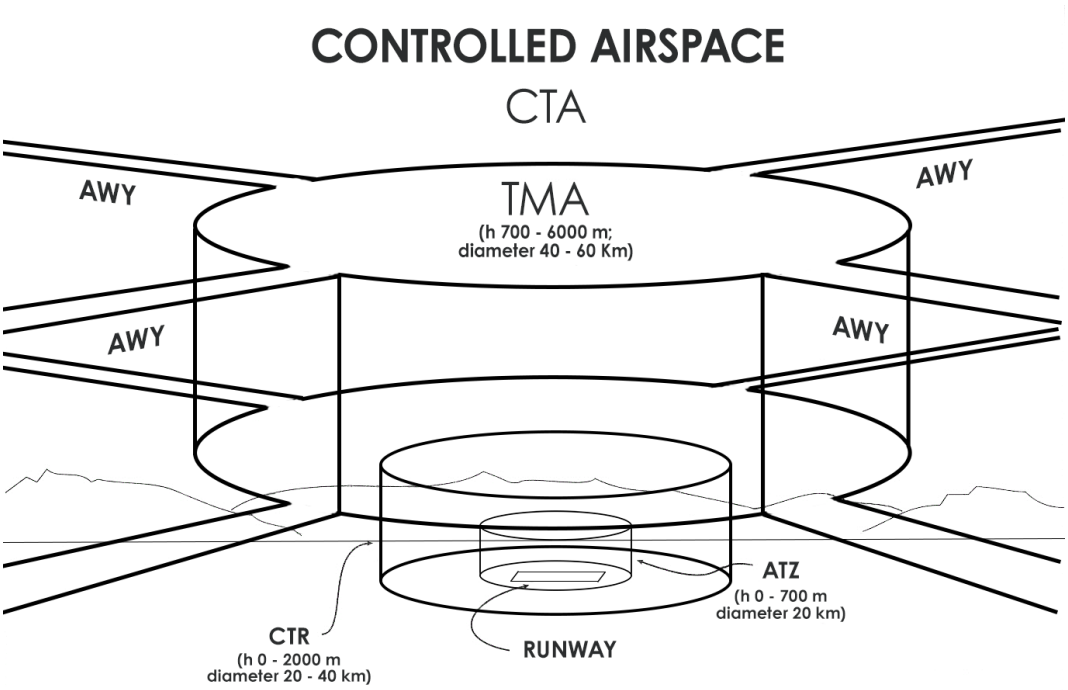


Figure 2.7: An example of airspace structure division.

aircraft or aircraft, ground vehicles or not permanent obstructions in the manoeuvring areas.

- **Traffic Information**, is issued by an ATS unit to alert an aircraft to other known or observed air traffic in proximity to the actual or near-future flight position. It is provided to help the pilot avoid a collision.

- **Flight Rules**, are the rules the pilots follow while flying. They can be visual flight rules (VFR) and instrument flight rules (IFR), which differ in whether they rely on the outside visual rather than on the instrumentation to fly the aircraft. VFR controls the operation of the aircraft under Visual Meteorological Conditions, meaning the pilots mainly rely on visual references. They required limited equipment and instrumentation, so VFR flights can be limited in certain airspace classes. IFR allows the aircraft to be operated under Instrument Meteorological Conditions, mainly relying on instrumentation and radio communication for aircraft piloting.

Based on these notions, several airspace classes can be defined based on ICAO Annex 11: Air Traffic Service, Chapter 2, Section 2.6 [19]:

- **Class A**. IFR flights only are permitted, all flights are provided with air traffic control service and are separated from each other.

- **Class B**. IFR and VFR flights are permitted, all flights are provided with air traffic control service and are separated from each other.

- **Class C**. IFR and VFR flights are permitted, all flights are provided with air traffic control service and IFR flights are separated from other IFR flights and from VFR flights. VFR flights are separated from IFR flights and receive traffic information in respect of other VFR flights.
- **Class D**. IFR and VFR flights are permitted and all flights are provided with air traffic control service. IFR flights are separated from other IFR flights and receive traffic information in respect of VFR flights, VFR flights receive traffic information in respect of all other flights.
- **Class E**. IFR and VFR flights are permitted, IFR flights are provided with air traffic control service and are separated from other IFR flights. All flights receive traffic information as far as is practical. Class E shall not be used for control zones.
- **Class F**. IFR and VFR flights are permitted, all participating IFR flights receive an air traffic advisory service and all flights receive flight information service if requested.
- **Class G**. IFR and VFR flights are permitted and receive flight information service if requested.

If ATC services are offered in a given region, the correspondent airspace of defined dimensions is called "controlled airspace" [19]. Controlled Airspace generally covers ATS airspace classes A, B, C, D and E, and comprises Control Areas, Terminal Control Areas, Airways, and Control Zones. Table 2.1 summarises the airspace class diversification.
An example of a possible aircraft path inside the Italian airspace is reported in Figure 2.8, showing possible classes associated to the type of airspace and the corresponding control service.

2.2.5 Air Traffic Control

Air traffic control is the most important among all the services provided by ATS Units. ATC is a service provided on-ground by ATC operators (ATCOs), which direct the aircraft on the ground and through the airspace section of competence.
A key aspect of ATC service provision is controller/pilot communication. Traditionally, voice communication is used for controller instruction, pilot requests and relative responses. Thus, both ground and onboard radio-communication apparatuses are needed to allow data transmission among control centres and aircraft. With a continuous and widespread increase in worldwide air traffic [2] over the decades, voice communication frequencies are nowadays saturated in many regions. An alternative system that is slowly spreading across the industry is the CPDLC (Controller-Pilot DataLink Communication), which is now mandatory in Europe for UIR flights. The CPDLC is a device which can be used to transmit written instructions and requests in alternative to voice commands and for not time-critical communications. The

[4]In Class D airspace, both IFR and VFR traffic are required to follow ATC clearances; however, ATC are only responsible for IFR against IFR separation.

[5]In Class E airspace, ATC does not provide separation between IFR and VFR traffic; IFR traffic shares responsibility for separation from uncontrolled VFR traffic with that traffic.

Table 2.1: Airspace Classes Summary

Class	**Type of Flight**	**Separation Provided**	**Service Provided**	**Speed limitations***	**Radio - Comm Requirements**	**Subject to an ATC Clearance**
A	IFR only	All aircraft	Air traffic control service	Not applicable	Continuous two-way	Yes
B	IFR	All aircraft	Air traffic control service	Not applicable	Continuous two-way	Yes
	VFR	All aircraft	Air traffic control service	Not applicable	Continuous two-way	Yes
C	IFR	IFR from IFR IFR from VFR	Air traffic control service	Not applicable	Continuous two-way	Yes
	VFR	VFR from IFR	1) Air traffic control service for separation from IFR 2) VFR/VFR traffic information service (and traffic avoidance advice on request)	250 kts IAS below 10000 ft amsl	Continuous two-way	Yes
D[4]	IFR	IFR from IFR	Air traffic control service, traffic information about VFR flights (and traffic avoidance advice on request)	250 kts IAS below 10000 ft amsl	Continuous two-way	Yes
	VFR	Nil	IFR/VFR and VFR/VFR traffic information (and traffic avoidance advice on request)	250 kts IAS below 10000 ft amsl	Continuous two-way	Yes
E[5]	IFR	IFR from IFR	Air traffic control service and, as far as practical traffic information about VFR flights	250 kts IAS below 10000 ft amsl	Continuous two-way	Yes
	VFR	Nil	Traffic information as far as practical	250 kts IAS below 10000 ft amsl	No	No
F	IFR	IFR from IFR as far as practical	Air traffic advisory service; flight information service	250 kts IAS below 10000 ft amsl	Continuous two-way	No
	VFR	Nil	Flight information service	250 kts IAS below 10000 ft amsl	No	No
G	IFR	Nil	Flight information service	250 kts IAS below 10000 ft amsl	Continuous two-way	No
	VFR	Nil	Flight information service	250 kts IAS below 10000 ft amsl	No	No

* When the height of the transition altitude is lower than 10,000 ft amsl, FL100 should be used in lieu of 10000 ft

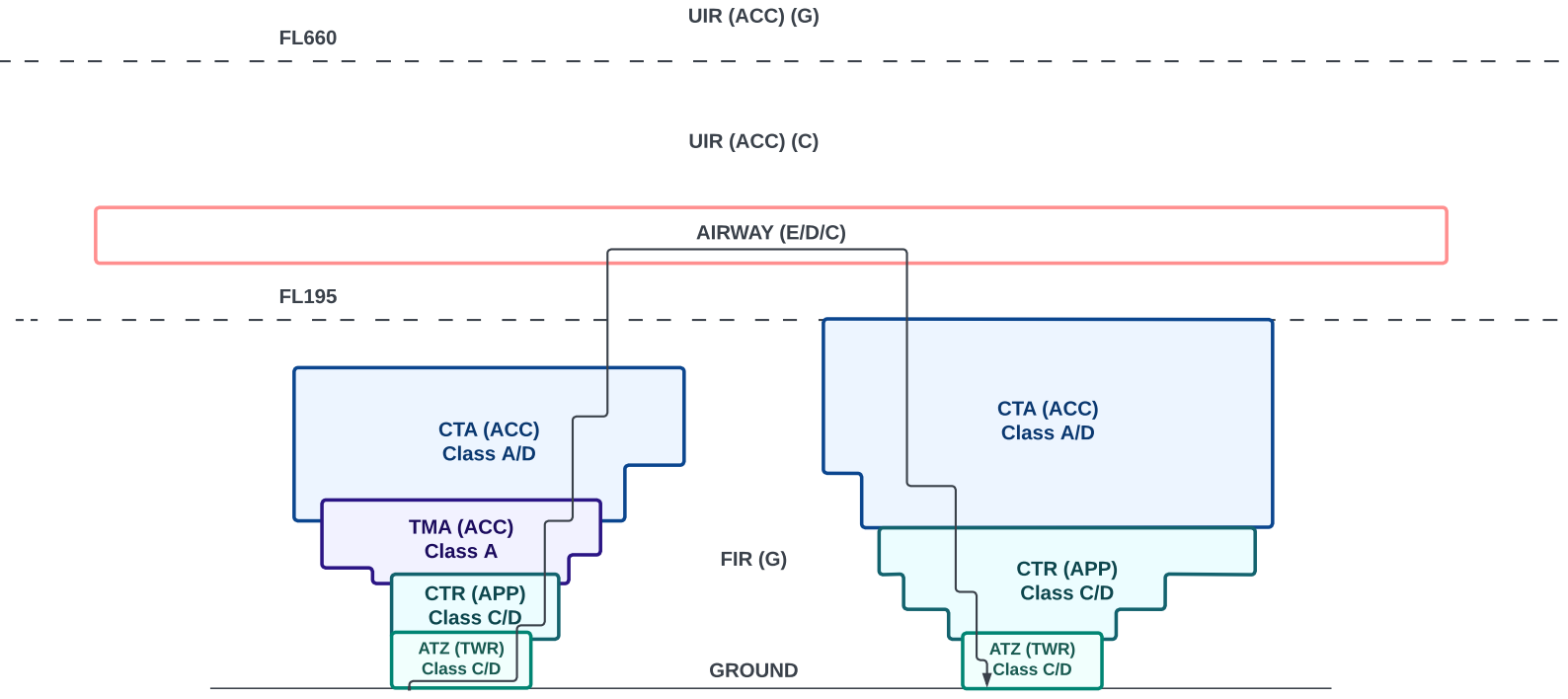


Figure 2.8: A possible route among two airports through different airspace types and classes.

main advantage is the reduced signal content needed to transmit encoded digital information with respect to carrying an acoustic analogical signal. At the same time, digital data transmission is more robust to interference and simpler to be checked for transmission errors.
Within the tasks of the ATS, ATC shall comply with three different objectives indicated by ICAO: prevent collision between aircraft, prevent collision between aircraft in the manoeuvring area and obstructions in that area, and expedite and maintain an orderly flow of air traffic. FIS is instead responsible for providing advice and information useful for the safe and efficient conduct of flights, while the alerting service is responsible for notifying appropriate organizations regarding aircraft in need of search and rescue aid, and assisting such organizations as required.
As for ATC, the control activity is divided into three monitoring groups:

a) **Area Control Service**, the provision of air traffic control service for controlled flights, except for those parts of such flights described in b) and c).

b) **Approach Control Service**, the provision of air traffic control service for those parts of controlled flights associated with arrival or departure.

c) **Aerodrome Control Service**, the provision of air traffic control service for aerodrome traffic, except for those parts of flights described in b).

While Area Control Service and Approach Control Service aim to avoid in-air collisions and manage traffic flow, Aerodrome Control Service is also meant to avoid on-ground collisions.
The three services are provided by distinct entities, which usually are organized in control centres where multiple operators are responsible for the safety of a given airspace:

- **Area Control Center (ACC)** provides Area Control Service inside the Control Region (which starts above a given flight level[6] - FL). It is usually responsible for the traffic of an FIR region at high altitudes between departure and arrival sites. Whether it can provide some approach services, its main focus is on en-route aircraft. Controllers usually accept traffic from - and pass traffic to - a terminal control area, or another control centre. They provide different services based on different control techniques: procedural for a low number of routes and low-traffic and ATS surveillance techniques for terminal areas and high-traffic flows.

- **Approach and Terminal Control (APP)** provides Approach Control Service in Control Zones and Terminal Control Areas (so from the ground up to a given FL) both for IFR and VFR flights. They are intermediaries between the tower controller and the area controller and provide control services during climb, descent and approach. APP controllers may serve multiple (nearby) aerodromes, while tower controllers are assigned to a single aerodrome. APP units are established when traffic levels demand them; in many airports, flights are passed directly from ACC to tower control and vice versa,

[6]In aerospace and meteorology it is an altitude, usually provided as hundred of feet, above the reference altitude level given by a standard pressure of 1013.25hPa. For instance, FL250 means an altitude of 25000 above the altitude level at which pressure is 1013.25hPa. With this standard, altitude is not fixed with respect to the ground level as variates with the pressure envelope depending on meteorological conditions.

without needing an APP control action. Flights control can be transferred directly from tower control to ACC even in case an APP control unit is provided over that aerodrome area. APP controllers apply separation using radar instrumentation, position reporting and ATS surveillance techniques.

- **Airport Control Tower (TWR)** provides Aerodrome Control Service. Tower Controllers (also called Aerodrome Controllers) work from the airport control tower, which is situated inside the aerodrome facility and has a clear view over the apron as well as on taxiways and airways (which will be defined in the next section). Tower Controllers' area of responsibility includes the runway and the taxiway as well as the airspace in the vicinity of the aerodrome. Differently from ACC and APP, the tower controllers greatly rely on visual observation of the controlled area to do their job; other technologies are used if available (such as airport radar, surface movement radar, optical systems, etc.) In recent days, new configurations are under testing called remote towers, where the service is provided from a remote location as in ACC and APP, which rely instead on displays reporting aircraft positions inside the controlled airspace (see Section 2.4.3).

In the next sections, a deeper insight will be provided into the Aerodrome control service.

2.2.6 Aerodrome Control Tower service provision

At first, a brief recap of aerodrome infrastructures is provided, which includes the main operative elements, mainly the apron, the taxiways, the runways, the control tower, and all the apparatus for surveillance and instrumental guidance. The apron is the area where aircraft are parked during the fuelling, loading, unloading and refurbishment phases. In the biggest airports, a dedicated controller supervises aircraft movement from the apron to the taxiways, which are paths the aircraft have to follow to move from aprons or terminals and other facilities to the runways; the latter is the rectangular strip, usually made of asphalt and concrete, dedicated to aircraft take-off and landing runs. The control tower is the facility in which the controllers are located. It is usually a tall building on the top of which the control centre is established, which is provided with a large window extension to provide optimal visibility on all the areas in the aerodrome where aircraft can be present.
From the control tower, the operators focus on the three main sources of traffic which are controlled by TWR: departures, arrivals and flights overflying the aerodrome traffic zone, with these last ones only representing a small part of the total number of controlled aircraft. Depending on the traffic type, different activities are carried out by controllers, which spend more time on departures, contrary to approach controllers who are mainly focused on arrivals. Controllers have to deliver clearances and approve pilots' requests for proceeding in the different departure or arrival phases.

Departures

- **Start-Up**. Usually, a start-up request is the first contact between the pilot and the controller and is generally approved unless there is a good reason for doing otherwise.

The start-Up request is cleared by the controller to allow the pilot to start the aircraft engines.

- **Pushback**. Depending on the aircraft parking position, it could be not possible to move forward to face the taxiway. In this case, a tug is needed which pushes back the aircraft and a pushback request is needed by the pilot. Sometimes, also a direction has to be provided which the aircraft has to face when the manoeuvre is completed.

- **Clearances Provision**. This clearance can contain the indications for the post-departure phase, such as which path to follow and flight level to reach after take-off or the transponder code to be set. In an optimal situation, it is provided during the pushback phase when the pilot workload is still low. However, it can be provided later if it is not still possible to provide the path and the flight level. As the communication is long and very important, the pilot has to copy it back to the controller, which has to pay attention to any error in the pilot reporting and correct it.

- **Taxi Out**. The pilots request to proceed with taxiing and the controller has to define the best route to reach the right runway depending on the airport layout and incoming traffic. In this phase, the controller has to ensure that the aircraft route is conflict-free (meaning there are no risks for collision with other aircraft or vehicles in the manoeuvring area) or, if this is not possible, to provide the pilot with appropriate instructions on intermediate holding points where to wait for further clearances and similar instructions.

- **Line Up**. The taxiing ends when the aircraft reaches the holding position[7]. After that, the controller has to give further clearance to allow the aircraft to enter the runway and align for take-off. If there is incoming traffic on the runway, the hold instruction can be reiterated.

- **Take-off Clearance**. This is the final clearance for the aircraft before departure. The controller has to ensure there are no obstacles on the runway and no incoming traffic arriving. He also has to coordinate with the approach controller to know the arrival traffic situation and to visually check the runway to ensure no irregularities are present. In providing take-off clearances, the controller has also to take into account the separation minima, in particular, due to the wake turbulence interference. After the take-off, the tower operator transfers control to the approach controller.

Tower controllers are also responsible for arrivals during the final landing phase and the subsequent taxi in phase.

Arrivals

- **Landing Clearance**. Pilots have to be cleared for landing, as the controller has first to ensure the runway is free from any hazard for a safe landing. Usually, the pilots do not receive any further communication after the clearance until they have landed and

[7]The runway holding position (called "holding point" in ATC communications) is the last point where the aircraft has to stop and explicitly wait for an ATC clearance before entering the runway.

slowed down the aircraft - due to the critical workload they suffer during landing - unless serious safety hazards are happening. Then, the taxiway or path to be followed after landing has to be indicated before the landing clearance.

- **Taxi In**. As for the taxi out, the controller has to choose an appropriate path for the aircraft to reach the apron and communicate the selected parking slot to be reached.

Transit Overflies

Small VFR flights could be overflying the airport at low altitudes, potentially interfering with the aerodrome traffic. In this case, the tower controller has to provide clearance to allow entering the airport traffic zone as well as provide ATC indications for the route to be kept inside the ATZ. When the overfly has ended, the control has to be transferred to an appropriate unit depending on the airspace structure.

Transfer of Control

When tower controller tasks are completed, which depends on the airspace structure and meteorological conditions, the control is transferred to the APP controller or directly to ACC if the approach service is not established in the area.
In smaller airports, a single figure (tower or "local" controller - TWR) can be in charge of the whole aerodrome area control. With the increase of airport dimension and traffic volume, a separate figure can be established called ground controller (TWR GND), which is in charge of the start-up and taxi phases.
The TWR GND can have control over the apron (if an apron manager is not provided), the taxiways and the inactive runways. The local controller in this case is responsible only for the active runways (she/he is then called TWR RWY) and deals with the line-up and take-off clearances and with landing operations.
Both TWR RWY and TWR GND controllers highly relate to the out-of-the-window visual from the control tower to ensure that the correct separations are maintained and to manage the aircraft movement. However, many apparatus are available in the biggest airports to increase the controller's situational awareness[8] of the managed traffic. These tools comprise surveillance technologies and ATM Systems and support tools. In addition, controllers are helped by regulated procedures to provide safer traffic management. A brief deepening of these aspects is presented in the next section.

2.2.7 Airport Tower Control Procedures, Techniques and Tools

Since the first ATC centres were instituted between 1920 and 1930, the newborn ATS has continuously increased in complexity under the guidelines provided by international organizations such as the ICAO and national authorities. Improvements have been delivered both for the technologies used to manage the traffic and for the procedures the air traffic control operators (ATCOs) have to deal with in solving operative tasks. Standard procedures are

[8]The situational awareness for a controller corresponds to having a clear mental picture of the overall current traffic situation in the aerodrome, including the possible and unexpected progression and changes that are going to happen in the scenario and therefore the conflicts that could be generating in the near future.

established for all civil aviation operators (and also military aviation has to deal with ad hoc procedures or submit those provided by civil authorities if flying in a shared airspace), including controllers and flight crews, and are continuously corrected and improved to ensure enhanced safety levels. With reference to the aerodrome control service, the most important procedures the ATCOs have to follow are the issue of ATC clearances and instructions and the application of separation minima between aircraft (these procedures are already described in section 2.2.4). In addition, other primal procedures are those to detect and solve potential and existing conflicts, manage the traffic flow through basic controller techniques, and coordinate the transfer of control responsibility to other ATC units.

All the procedures require the TWR RWY and TWR GND controllers to be aware of the traffic distribution. While this is usually obtained through a visual inspection from the tower view, some conditions (e.g. the case of blind spots in the tower view or low visibility conditions) require the use of radio and radar instrumentation (Section 2.3) for determining the relative aircraft positions or the position with respect to significant obstacles. In addition, controllers also need reference information about the traffic - like estimated off-block time (EOBT), parking position, destination, wake category -, the meteorological conditions in the aerodrome nearby, and others. All this information is provided on display terminals inside the control tower, which the controller has to continuously look at while working. A brief description of some tower instrumentation is presented below.

The meteorological conditions are usually obtained by a meteorological airport station and are organized on displays giving present and future weather information to be reported to pilots if significant or when requested.

The traffic to be managed is usually organized using flight progress strips, which are traditionally made of paper and are physically moved through different blocks whilst the corresponding aircraft completes the different departure or arrival phases. Nowadays, more and more control centres are using electronic strips, which are visible on displays and can be easily moved from different blocks and transferred to other controllers with a gesture on the screen.

In addition, position reporting technologies are today a common facility in airports. These

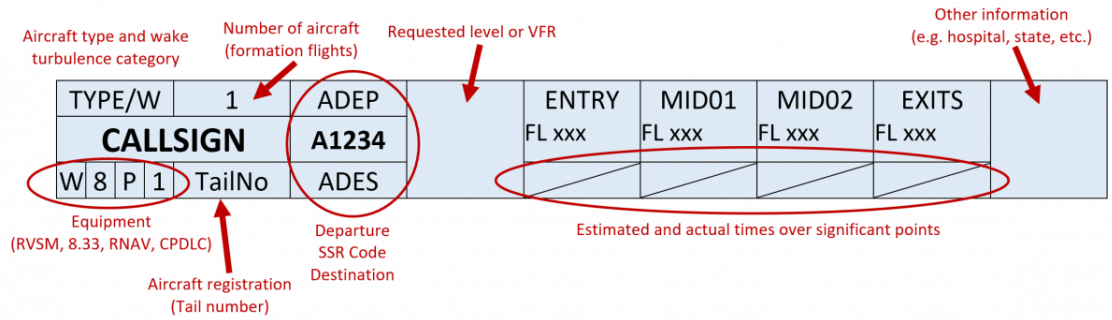


Figure 2.9: Example of a flight progress strip structure. Source: Skybrary Portal [16].

are meant to enhance controller awareness giving a clear picture of the traffic on visual displays and are based on the so-called surveillance techniques.

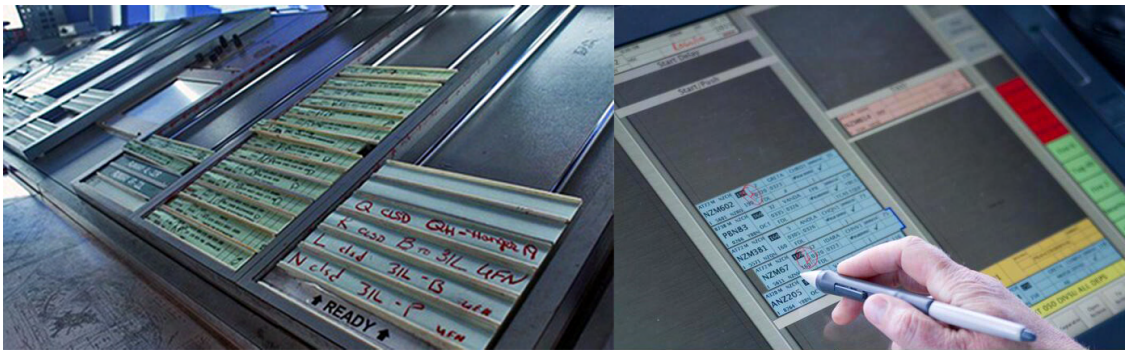


Figure 2.10: Comparison of paper and electronic strips.

2.3 Surveillance techniques

Surveillance is a method to detect and track an aircraft to provide the controller with additional information about the traffic condition. In the aeronautical field, surveillance is based on sensors capable of detecting the aircraft, linked to data processing systems which calculate the position and other information such as aircraft identification and elevation. The computed data are presented to the controller on a situation visual display, which presents in a single airport-referenced view all the detected aircraft (or other relevant objects) in the controlled area.
Traditionally in aviation surveillance, the sensor is a surveillance radar, which physically detects the aircraft. Whether radar facilities are typical of approach and area control, they can be used also for aerodrome tower control, and become an important instrument for the controller in low-visibility conditions. The different sensor technologies are the following:

- **Primary Surveillance Radar (PSR)**. Composed of a rotating antenna which transmits a radio signal or a pulse (usually at 1-5 GHz), and calculates the delay in the response after the signal is reflected on an object. The PSR can calculate the heading, bearing and distance of the detected object. With respect to SSR, PSR is less precise and does not provide any information on the detected object. However, it is the most reliable instrument as it is robust to any aircraft transponder[9] failure [21]. The PSR is usually available in approach and area control centres, while similar technology is used by tower control for airport traffic.

- **Secondary Surveillance Radar (SSR)**. Is an evolution of the PSR, in which a rotating radar antenna periodically sends a radar signal containing a request (interrogation) and collects the corresponding response from any interrogated aircraft. Different transmission modes (A, C, S) exist and also different types of interrogations. With respect to PSR, the SSR requires less power to reach the desired range, and can directly obtain information (such as altitude, callsign[10], wake category) about the detected flight, which

[9]The transponder (short of transmitter-responder) is an instrument which sends a reply signal when properly interrogated. When installed on aircraft, receives interrogations from ground apparatus or other aircraft and replies with the requested piece of information about the flight

[10]In aeronautics, it is a unique code assigned to a flight to keep track of it in a control area and when transferring control to another unit.

is provided by onboard transponders. In this type of radar the position is still determined by the data processing system based on time delays in transmission/reception [22].

- **Automatic Dependent Surveillance - Broadcast (ADS-B)**. Is a technology by which aircraft and airport vehicles can automatically send all the information already transmitted by Mode-S transponders plus their position information, in a broadcast mode. A big advantage of this technology is the simplicity of implementation and the low cost. In fact, being the position information contained in the message, there is no need for the complex data processing of a traditional radar system and a simple quarter-wave antenna is sufficient to receive the messages as opposed to an expensive and cumbersome radar facility. On the market, low-cost solutions are available also for amateur users, and professional stations still cost much less than other types of surveillance facilities.

- **Multilateration (MLAT)**. Is a rather new technique which uses multiple receivers to determine the position of an aircraft. Could be implemented by sending an interrogation to an aircraft and calculating the response difference in arrival time for multiple sensors at different locations. The time differences allow to obtain the aircraft position without the need for a rotating antenna. The implementation is still difficult as very high precision is needed in calculating the time differences to have adequate precision and accuracy in the computed position [23].

In addition to these technologies, a variant of the PSR is used to detect all the principal features on the surface of an airport environment: the Surface Movement Radar (SMR). This radar works by a similar principle but at a higher frequency (10-20 GHz) with respect to the PSR; as a result, the antenna is much smaller and the radar can rotate faster while having a smaller range capability, which is compatible with ATZ needs. This type of radar is specific for controller awareness of the manoeuvring area. In addition, it is the basis for an Advanced Surface Movement Guidance and Control System (A-SMGCS), a modular system comprising multiple functionalities to allow the safe, orderly and expeditious movement of aircraft and vehicles in airports under all traffic and weather conditions. According to the Eurocontrol definition [24], the four A-SMGCS services are the surveillance service, the airport support service (which comprises Runway Monitoring and Conflict Alerting, Conflicting ATC Clearances alerting, and Conformance Monitoring Alerts for Controllers), routing service and guidance service, with the last two services participating in automating air traffic management. In addition to the SMR, also the ADS-B systems are a relatively new and promising technology for airport surveillance. Since Mode-S and ADS-B techniques are key aspects of this thesis project, they are further described below.

2.3.1 ADS-B technology

ADS-B is used on modern aircraft to improve overall safety, ATCO and pilot's situational awareness, and ATM efficiency in the air traffic control field. The surveillance system is based

on the aircraft transmitting its position through a modified Mode S transponder at 1090 MHz, the primary frequency band used worldwide as a standard for interoperability. In addition, another system is used at the regional level, the Universal Access Transceiver (UAT), which works at 978 MHz and is used for example in the USA. Other transmitting systems have been tested in the past but have then been discarded.
The ADS-B system is automatic, as it transmits aircraft data without the need for transponder interrogation. It is dependent, as it depends on the data provided by the aircraft navigation system. In the broadcast operation, the transponder periodically sends aircraft data, which can be received and decoded by every compatible device listening on that frequency. For the ADS-B to work properly, ICAO regulations require a certified high-accuracy GNSS positioning system to be installed on the aircraft.
The technology comprises two services, the "ADS-B In" and the "ADS-B Out", and could replace radar in its worldwide primary surveillance role for aircraft control. The "ADS-B Out" is used by aircraft to transmit data through the transponder periodically. It is mainly used by ATCOs to get real-time high-precision positions and other information about an aircraft. ATCO will then be able to administrate air traffic with improved separation management and timing. The "ADS-B In" allows aircraft to know the position of other air vehicles and is mainly used for collision avoidance.
Practically, ADS-B has to be used in combination with other radar sources and in particular with the primary radar and the Mode S Elementary Surveillance provided by SSR, as those provide independent surveillance - not relying on aircraft data to get the position - while being less accurate. In addition, ADS-B represents an opportunity for the smallest airports, where the low-traffic volumes can not justify the investment in expensive radar facilities, with ADS-B stations being an affordable solution to improve overall safety and situational awareness for both airport operators and pilots.

2.3.2 1090 MHz transponder operations

The 1090 MHz transponder is the standard for EU airlines. This ADS-B transponder functioning was derived from an extension of the available Mode S operation modes. In Mode S, the transponder can be selectively[11] interrogated, differently from Mode A and Mode C interrogations, which are sent to every listening aircraft. The selective call is made by the SSR using the ICAO address of the aircraft[12], which is a 24-bit code transmitted by the transponder which uniquely identifies the aircraft. This way, a ground transmitter can interrogate a precise air vehicle, as only the transponder with a correspondent ICAO code will answer the call. Mode S can also be used for all call replies and is designed to be compatible with Mode A and Mode C calls. Mode S transponder sends replies in 1090 MHz, and ADS-B uses the same frequency for interoperability, with an Extended Squitter (ES) transmission protocol, that is a variation of the Mode S Elementary Surveillance protocol.
A protocol was developed to distinguish the different uses of the 1090 MHz Mode S transponder. In particular, an uplink format (UF) given by 5 message bits identifies the type of call by

[11] The "S" of Mode S means "Selective".
[12] SSR transmits request on a 1030 MHz band.

the SSR. Similarly, a downlink format (DF) is transmitted by transponder with its replying or broadcast message. To highlight the multiple options of the 1090 MHz Mode S protocol, a scheme from *The 1090 MegaHertz Riddle* [25] is reported in figure 2.2.

Table 2.2: Mode S uplink and downlink format

UF/DF	**Bits**	**Uplink type**	**Downlink type**
0	56	Short air-air surveillance (ACAS)	Short air-air surveillance (ACAS)
4	56	Surveillance, altitude request	Surveillance, altitude reply
5	56	Surveillance, identity request	Surveillance, identity reply
11	56	Mode S All-Call	All-Call reply
16	112	Long air-air surveillance (ACAS)	Long air-air surveillance (ACAS)
17	112	-	Extended squitter
18	112	-	Extended squitter/non transponder
19	112	-	Military extended squitter
20	112	Comm-A, altitude request	Comm-B, altitude reply
21	112	Comm-A, identity request	Comm-B, identity reply
24	112	Comm-C (ELM)	Comm-D (ELM)

2.3.3 ADS-B functioning

The ADS-B works with Mode S DF 17, 18 and 19. As stated in table 2.2, there is not any uplink message for those DF, as the ADS-B works in automatic broadcasting. Commercial aircraft use ADS-B DF 17. Whether DF 19 is also used for ES, it is reserved for military aircraft so its decoding is not for the purpose of commercial aviation applications which this work focuses on. Other DF are very rare but they can still be detected by ground systems, like ACAS messages sent by an aeroplane to another one.
Extended squitter downlink messages are composed of a preamble followed by a 112-bit (56 for elementary surveillance) message. The preamble lasts $8\mu s$ and contains 16 bits, composed by a $0.5\mu s$ pulse for the 1 bit and $0.5\mu s$ flat signal for the 0 bit. The messages follow the Pulse Position Modulation (PPM) technique, a type of amplitude modulation. With PPM, bit 1 is represented by a $0.5\mu s$ pulse followed by a $0.5\mu s$ flat signal, so each bit lasts $1\mu s$. Bit 0 is reversed with respect to 1. In figure 2.11 a message pulse structure is shown.

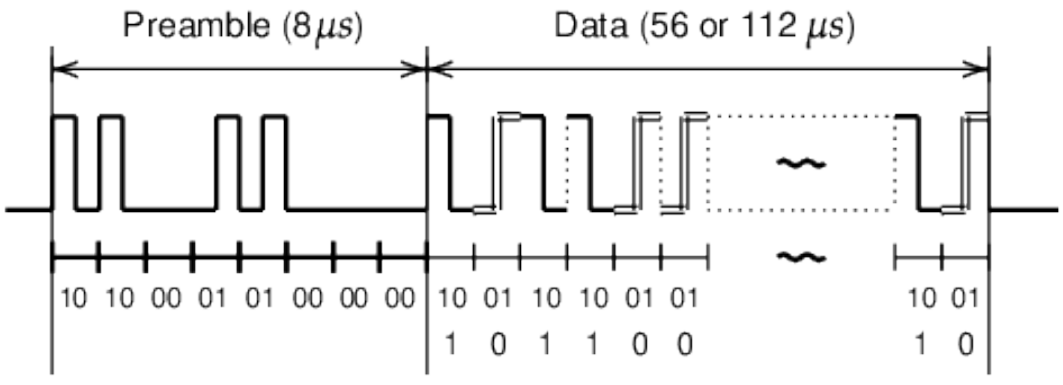


Figure 2.11: An example of ES reply message. Source: [25].

2.3.4 ADS-B data encryption

A good description of ADS-B frame structure and decoding algorithms is contained in [25]. Whenever this guide has been found incomplete, the RTCA/DO-260A document [26] has been used.
The 112 bits data structure is composed of a first byte in which the DF is encoded (5 bits) followed by a value for the transponder capability (3 bits). Then three bytes contain the identification code, a 24-bit unique code assigned by ICAO to every Mode S transponder, thus uniquely identifying each existing civil aircraft. The next 56 bits of the ADS-B frame contain the extended squitter information message. Different information can be transmitted, such as position or airborne velocity. The type of information transmitted is identified by the first 5 bits (type code - TC) of the 56 bits message. The last 24 bits are used for error checking using the CRC error control technique. The basic ADS-B message structure is reported in table 2.3. The different message types sent by ADS-B are listed in table 2.4.

Table 2.3: Structure of ADS-B messages

Bits	**No. bits**	**Information**	
1-5	5	Downlink Format	DF
6-8	3	Transponder capability	CA
9-32	24	Aircraft Address (ICAO)	ICAO
33-88	56	Message	ME
(33-37)	(5)	(Type code)	(TC)
89-112	24	CRC control string	PI

Table 2.4: Type Code and content of ADS-B messages

Type Code	**Message Frame Content**
1–4	Aircraft identification
5–8	Surface position
9–18	Airborne position (using Baro Altitude)
19	Airborne velocities
20–22	Airborne position (using GNSS Height)
23–27	Reserved
28	Aircraft status
29	Target state and status information
31	Aircraft operation status

2.4 The European ATM system: SES and Sesar

2.4.1 The Single European Sky

In order to cope with the crowding of the European countries' national airspace, the European Commission ("the Commission") launched the Single European Sky (SES) initiative in 2000, following the systemic and heavy delays the commercial flights had issued the previous years. The programme aims to restructure the European airspace and the European ATM

system through a series of actions at different levels (institutional, operative, control and supervision and technological) to improve air transport in Europe by reducing the airspace fragmentation[27] at national borders.
The search for improvements aims at enhancing of safety and efficiency of air transport, the reduction of delays by proper and shared management of the scarce airspace and airport resources, the improvement of the offered services and reduction of flight costs per passenger, in particular by reducing inefficiencies due to ATS provision fragmentation, and the integration of military systems into the European ATm system.
The Commission operates through the delivery of Legislative Packages which are the results of recommendations contained in reports produced by a "High Level Group" [28], which was instituted in 2001 by the Commission itself. The High Level Group is composed of the director generals of all the member nations' Civil Aviation Authorities, the Director General of Eurocontrol and senior partners from the Aviation Industry. Eurocontrol (European Organisation for the Safety of Air Navigation) is the intergovernmental organization through which European countries cooperate in improving European airspace.
The first Legislative Package (SES I) has been delivered by the Commission in 2004 and designed a framework for a shared and interoperable airspace (from here the name "Single European Sky") with additional regulations concerning the air navigation services (ANS) provision, the airspace organization and the interoperability of a European ATM network. These four regulations were integrated with more precise implementation criteria developed together with other specially created groups, namely the Single Sky Committee (responsible for the implementation of the regulations) and the Industry Consultation Body (the industry counterpart for advice and feedback on the regulations).
Following the feedback on the SES I implementation, in 2009 the Commission prepared a second legislative package (SES II). This new set of regulations aimed to counteract some difficulties in the implementation of the first package, due to a still too-high fragmentation at national borders and a continuous increase of air traffic levels in European skies. The second package aimed to create a performance framework to set a common target on regulations implementation, create a single safety framework to harmonize the development of safety regulations and ensure their implementation, improve the management of airport capacity, and organize and finance the research for new technologies enabling new operative models, improving the overall system capacity and safety.

2.4.2 SESAR

The technological development of new technologies for the future of the ATM sector in Europe has been a main objective of the Commission, which created the Single European Sky ATM Research (SESAR) project for this purpose in 2004.
As an initiative of the Commission, the SESAR Joint Undertaking (SESAR JU) has been established in 2007 as a public-private partnership to run the SESAR project. SESAR JU is fully in charge of the SESAR research, driving the funding of the programme, both public and private, the technical organization of research and implementation phases, the update of the

work programme and objectives when needed, the involvement of relevant ATM stakeholders, the general supervision and reporting on the advancement of each activity in the work programme. In November 2021, the new SESAR 3 JU has been established with an additional ten years mandate (2021-2031).
SESAR research was originally settled in three phases:

- a **definition phase** (2004-2008) to develop the air traffic modernization plan (SESAR ATM Master Plan) defining the objective, priorities and timetables of the different technological stages of research,
- a **development phase** (2008-2013) to make possible the development of the basic technologies which should pave the way for the next generation ATM systems, and
- a **deployment phase** (2014-2020 and beyond) to implement at large-scales the new ATM infrastructure, enabling a fully harmonic and interoperable system.

The temporal boundaries of the phases have acquired flexibility over the years with multiple lifecycles planned, but the general structure has remained the same. After a definition phase, in which the ATM master plan is updated, a public call is launched to receive and evaluate projects which will enter the development phase. During the latter, the research has been organized following the SESAR Innovation Pipeline structure, which contemplates three stages of research, called for the SESAR 3 JU:

- **Exploratory Research**: Explores and develops new concepts identified in the European ATM Master Plan and emerging technologies.
- **Industrial Research and Validation**: Validates and evaluates new concepts in simulated or real operative scenarios. This stage turns concepts into SESAR solutions.
- **Digital Sky Demonstrator** (called "Very Large Demonstrator" in SESAR JU): Tests the SESAR solutions on a much larger scale, and in real scenarios, for example testing a solution on multiple airports and many different operative scenarios.

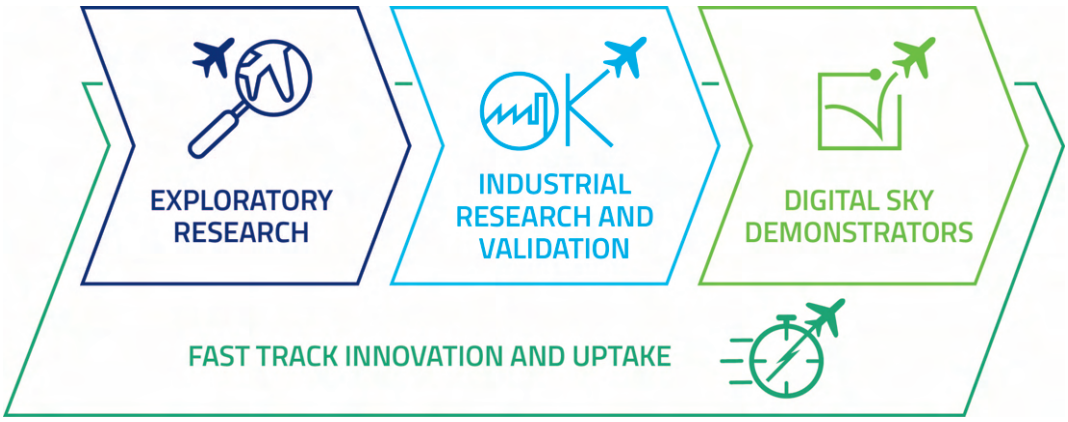


Figure 2.12: The SESAR 3 JU Innovation Pipeline. Source: Sesar JU [6].

The Innovation pipeline has a precise direction, with concepts and solutions being assessed and developed moving from left to right along the pipeline in Figure 2.12 while the research

progresses. New projects can enter the pipeline at the first or second stage (Exploratory Research or Industrial Research and Validation). After a project is validated and concluded, if the research is valid, it can be promoted to a more advanced step inside the same stage, or move to the next stage. This way, solutions are progressively evolved and developed from initial ideas - even theoretical - to developed and validated solutions ready for market and industry deployment at the exit of the pipeline.
With SESAR 3 JU, an additional transversal channel has been added, the "fast track innovation and update", which aims to fasten the development of high-risk / high-profit technologies to shorten the time to deliver to the market highly innovative technologies.
Since SESAR JU is a public-private partnership, the funding mechanism is top-flight, with public funds being progressively substituted by private capital as the maturity level of the developed technologies approaches the one required for private commercialization. Following this trend, exploratory research is usually fully covered by public funds. When entering the industrial research stage, researchers may ask SESAR for funds partially covering the budget requested for the project, while they are responsible for the raising of private partnerships and capital covering the remaining part of the budget. Private capitals become the predominant or even exclusive funding source in the Digital Sky Demonstrator phase. With this mechanism, it is ensured that private companies, which are the final profit-maker above the developed technologies, are stimulated to start researching with public incentives when solutions are still non-profitable. As commercialization gets closer, companies have then more interest in carrying out research with their own investments.
As SESAR is an initiative of the European Commission, the public funds provided for the project come from the European general research programmes established over the years to fund public research in the whole European Research Area (ERA). These programmes are called "Framework Programmes for Research and Technological Development" (In short "Frameworks Programmes" or FP1 to FP9). Starting from 2014 they have been called Horizon programmes (FP8 was called Horizon 2020 and FP9 is called Horizon Europe). The FPs have a duration of seven years. SESAR 1 (the first phase of SESAR) was initially funded with the FP7 for the period 2007-2013, while SESAR 2020 (the phase of the programme looking at objectives for 2020 plus four additional years) took public funds from the Horizon 2020 programme (FP8) for the period 2014-2020. The new SESAR 3 partnership distributes funds coming from the Horizon Europe framework programme (FP9) for the period 2021-2027.

2.4.3 SESAR developed solutions

Many solutions have been developed as part of the SESAR project over the years. Some solutions have completed the innovation pipeline and are now ready for deployment. These are called "delivered solutions", or simply "solutions", and some have been references for further projects, re-entering the innovation pipeline as enabling concepts for the development of new technologies. Other concepts and technologies have already been selected for industrial research after the first exploratory research stage; these are defined as "candidate solutions". The research involves four key areas: high-performing airport operations, advanced air traffic services, optimised ATM network services, and enabling aviation infrastructure. Additionally,

a specific research area is dedicated to Urban mobility research. Some of the most important and disruptive solutions developed and under development by SESAR are presented[13]:

Delivered Solutions

- **Airport safety nets for controllers: conformance monitoring alerts and detection of conflicting ATC clearances**. This solution contributes to the high-performing airport operations research campaign, increasing controllers' situational awareness and safety around and on the runway. Various new generation automatic systems have been tested and validated to alert controllers on existing conflicting clearances, such as an aircraft departing while another is landing or runway incursion during take-off. The safety net solution is now being considered as enabling technology for many other projects and solutions.

- **ATC and AFIS service in a single low-density aerodrome from a remote CWP**. Dealing with high-performing airport operations, this project brought the creation of the world's first remote-controlled airport tower at the isolated airport of Örnsköldsvik, in Sweden. Through the use of sensors, mainly video cameras, around the airfield, a remote control centre has been set up to control the airport operations from a more accessible facility, and with additional deliverable services, such as meteorological data, thus allowing to offer around the clock services, where the dimension of the airport did not allow it to happen on the site. This project has paved the way for many further solutions involving medium-traffic airports and even multiple remote towers located at the same facility. The solution contributes to increasing the cost-efficiency of airport operations and supports regional economies.

- **ACAS Xa European acceptability framework**. As part of advanced air traffic services research, a solution has been delivered that implemented and validated the use of ADS-B for the next-generation ACAS technology[14], ACAS Xa. Using the more precise information coming from the ADS-B system with respect to Mode C/S interrogations, the collision avoidance system can be improved while maintaining the same interface as the current ACAS system for pilots. The use of ADS-B is consistent with the requirement to have ADS-B system installed on large aircraft[15] operating in the European Union, starting from December 2020. The SESAR-developed framework is ready for industrialization and is currently under an international standardization process due to the worldwide interoperability the ACAS system must hold. This technology contributes to increasing safety and reducing nuisance alerts.

[13]The whole catalogue of the SESAR solutions [29] is available at https://www.sesarju.eu/catalogue

[14]The ACAS (Airborne Collision Avoidance System), is an ICAO concept regulating the implementation of an onboard system which activates when two aircraft are under collision risk, alerting both crews and giving consistent instructions to modify the flight routes avoiding collisions. Standards and recommended practices have been updated and evolved over the years, resulting in an evolution of the physically implemented system, the TCAS (Traffic Collision Avoidance System). The current generation uses the Mode S/C of transponders to calculate the distance between aircraft and detect possible trajectory conflicts, giving instructions accordingly.

[15]MTOW greater than 5700 kg or maximum cruise speed greater than 250 knots.

- **Automated support for dynamic sectorisation**. It is a solution dealing with the automatic evaluation of future traffic conditions in a sector of airspace, aimed at automatically calculating the best allocation of sectors. The tool allows the supervisor to plan the sectorization of the airspace for the day and human resources accordingly. As a solution delivered for the optimization of ATM network services, the main advantages result in improved airspace capacity, due to better use of resources, increased cost efficiency and reduced saturation periods and flight delays.

- **ADS-B surveillance of aircraft in flight and on the surface**. A solution enabling the aviation infrastructure of the future. It comprises an ADS-B ground station and a data processing system and uses new techniques to mitigate spoofing external actions on the received signal; it can also be used to cope with the malfunction of avionics equipment. The solution has been validated on a large scale and is already deployed in multiple European countries. During its development, SESAR has contributed to the relevant standard for the implementation of ADS-B ground data processing systems.

Candidate Solutions (currently in the pipeline)

- **Extended airport safety nets for controllers at A-SMGCS airports**. Aimed to improve airport human performance and situational awareness, this candidate solution continues the previous work on safety nets, validating new sensor technologies and new safety procedures while integrating previous solutions (both procedural and technological) developed in SESAR.

- **Dynamic airspace configurations (DAC)**. As stated in the project description "[...] the aim is to harmonise airspace management, flow management, and air traffic control during planning phases to deliver a seamless and dynamic process enabled by collaborative decision making (CDM) between civil and military stakeholders. With DAC in place, it will be possible to manage dynamically all capacity elements and constraints in one single, seamless process and ensure the best possible balance of the different performance targets and operational requirements [...]". SESAR is currently developing algorithms to analyse and forecast traffic flows and an automatic sector management tool proposal for airspace sector allocation.

- **Future ADS-B communications link**. The candidate solution addresses the future of ADS-B communication links, studying a phase overlay methodology for Mode S and ADS-B datalink. Using phase overlay techniques, it has been tested the possibility to extend data transmission from 112 to 448 bits. The additional data capacity can be used to provide additional information, such as meteorological data, add a cybersecurity layer or reduce the transmission rate with a positive impact on the 1090 MHz band congestion. For these reasons, the candidate solution is expected to increase cost efficiency, security[16], and spectrum efficiency.

[16]Following ICAO definitions [30][31] security consists in “safeguarding civil aviation against acts of unlawful interference”, while safety is “the state in which risks associated with aviation activities, related to, or in direct support of the operation of aircraft, are reduced and controlled to an acceptable level”.

- **Multiple Remote Towers**. Following the solution of remote-control towers, a new project has been established which aims to enlarge the previous idea with multiple control towers situated in the same facility. The concept was tested by providing air traffic control services to 15 simulated airports from a single control centre. The project also developed managing tools to efficiently and safely allocate the shared workstations and human resources based on the needing of each airport over time. The solution is part of the Digital Technologies for Tower (DTT) project - referenced by SESAR as PJ05-W2 -, which comprises also other solutions reported in the next sections.

2.4.4 DTT project

Within the SESAR 2020 context, industrial research has been organized in two waves; the first was concluded in 2019, with many solutions ready for pre-industrialization. In 2020, a second wave of projects was prepared to take forward the results of the first wave, selecting the most beneficial solutions with respect to the SESAR objectives of improving ATM safety, capacity, cost efficiency and environmental impact. Among the twelve selected industrial research projects, the Digital Technologies for Towers (DTT) project has two different aims, the first being the described multiple remote tower centre development. Second, it intends to validate innovative human-machine interface (HMI) modes and relative technologies in different control towers.
The included solutions were three: solution 35 "Multiple Remote Tower and Remote Tower Centre" (Pj.05-W2-35), solution 97.1 "Virtual/augmented Reality Applications for Tower" (Pj.05-W2-97.1), and solution 97.2 "ASR at the TWR CWP supported by AI and Machine Learning" (Pj.05-W2-97.2). While solution 35 has already been described, a brief insight into solutions 97.1 and 97.2 is provided, with a major focus on solution 97.1 and specifically on exercise 2 which represents the starting point of this thesis project.

Solution 97.1
As described in [32] and [33], this solution tested how enhanced reality techniques, together with tracking labels, attention guidance and air gestures concepts, could be used to increase control tower operators' situational awareness and head-up time[17], exploiting computer-generated visual elements to provide an overlay to the out-of-the-window view of useful information helping to identify and track airport traffic, especially in low-visibility conditions.
In order to fully exploit the possibilities offered by AR, multimodal interactions with holograms through Air Gestures[18] have been considered, to interact with aircraft tracking labels and provide non-time critical clearances. In addition, previous solutions from SESAR have been implemented in simulations to enhance safety and situational awareness: safety nets could be used to trigger the appearance of overlays visually alerting the controller and guiding him toward the conflict event. With the aim of testing all these possibilities, the solution

[17]The time the operator spends looking directly at the out-of-the-window (OTW) visual in head-up position rather than focusing on the workstation terminals to get the needed information.
[18]Air gestures (non-touch) are an innovative way of interaction with an electronic device tracking hand's movements through cameras. In the case of AR, they usually allow the user to interact with holograms through precise movements of their hands.

was organized into three different exercises planned on different simulation platforms and focusing on Tracking Labels, Multimodal Interaction and Attention Guidance. Furthermore, in accordance with the concept of a reality-virtuality continuum (see Section 2.1.3) the possibility was considered to adapt the virtual overlay to environmental conditions, primarily adding useful information and labels with the worsening of visibility.
EXE-05.97.1-VAR-001 validated AR interaction modes at a simulated Schiphol Airport environment (Figure 2.13) with a focus on Attention Guidance. The exercise included the reproduction of the alerting systems existing at the Schiphol airport and the creation of an Attention Capturing & Guidance application (AC&G), which provided information to the controller through an AR device (Microsoft HoloLens 2).
EXE-05.97.1-VAR-002 tested multimodal airport interaction in a simulated reproduction of Bologna Airport. Tower Tools, Tracking Labels, Air Gesture Interaction and Safety Nets/Attention Guidance technologies were all implemented. The exercise exploited previous results from the SESAR exploratory research campaign RETINA (Resilient Synthetic Vision for Advanced Control Tower Air Navigation Service Provision) [34][35], adding important features such as multimodal interaction, through voice commands and air gestures, attention guidance due to safety nets implementation, and adaptive HMI, resulting in the implementation of two different controller positions, the ground controller (TWR GND) and the runway controller (TWR RWY). For these positions, the developed AR interface provided different points of view in terms of visualized information and different interaction modes depending on the tasks to be completed, with the TWR GND operator allowed to interact with hand gestures with the label, giving permission to the pilot for not-time-critical operations, such as push-back and start-up. The validation campaign took place at the UNIBO's real-time Humans-in-the-loop simulation and validation platform located at the Virtual Reality and Simulation Laboratory of the University of Bologna in Forlì. The platform was adapted from the previous RETINA project implementing all the new features (safety nets, multimodal interaction through air gestures and the adaptable HMI). The architecture of the validation platform is presented in figure 2.14 and is composed of multiple modules generating the 4D scenario and sending the simulated information to a pseudo pilot who controls the aircraft in simulation and to the two ATCO positions (GND and RWY).
EXE-05.97.1-VAR-005 was held in shadow mode - the only exercise to do so - at Vitoria-Gasteiz Airport (LEVT), implementing Tracking Labels and Air Gestures in an AR application. The Vitoria-Gasteiz small airport was not equipped with any surface radar, thus ADS-B systems have been used to track the aircraft and a testing ground vehicle. As in the other two projects, a Microsoft HoloLens 2 HMD has been used as an AR system. The developed system implemented tracking labels, while attention guidance was not implementable as safety nets systems were not present in the airport.

The validation campaign demonstrated the effectiveness of the solutions, with real controllers experiencing reduced workload, increased situational awareness and higher performance capability.

Figure 2.13: The NARSIM simulation platform set to reproduce a realistic scenario of the Schiphol Airport.

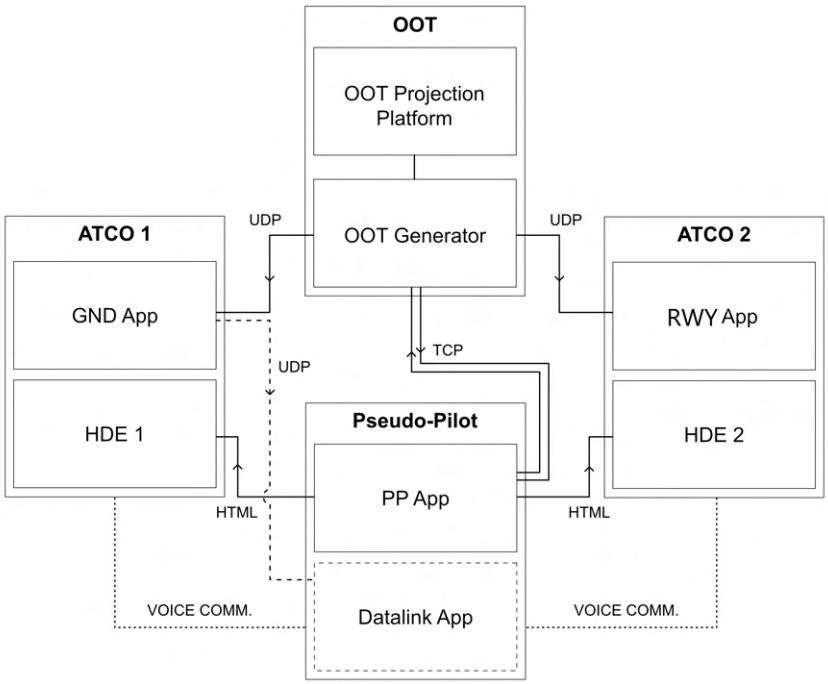


Figure 2.14: The UNIBO validation platform architecture. Data are computed by the OOT (out of the tower view generator) module, which elaborated the visual airport scenario, and sent them to all the other modules.

Solution 97.2

As reported in [36], automatic speech recognition performance for air traffic controllers has been analysed in this solution. 22 ATCOs participated in a validation campaign in a simulated tower and ground environments. Using artificial intelligence, a tool was designed to translate audio signals (voices) in air traffic control concepts, also trying to predict the expected commands from knowledge of the operation context (using data from surveillance, meteorological systems, flight plans,...). The campaign consisted of three different exercises held in Germany, Norway and Italy. The developed Assistant Based Speech Recognition systems helped tower controllers to complete multiple tasks, with a reduced number of misunderstood commands and increased human performance and safety.

EXE-05.97.1-VAR-002: a starting point for real-world applications

Exercise 2 of solution 97.1 was held at the UNIBO Virtual Reality and Simulation Laboratory in Forlì and depicts the state of the art for this thesis project. Whether successful, the exercise was run in a fully simulated scenario, and the logical continuation of the research carried out is the translation of the developed technologies into a real airport environment, which has to be made by steps. Therefore, the next chapters will explain the process followed in implementing a real-world control tower AR system.

Chapter 3

Method

This chapter describes the design methodology used to develop the final application. After that, the methodological passages sustained in design and implementation are clarified.

3.1 The User-Centred Design methodology

ATC is a high-risk and high-responsibility activity in which the controller workload, stress level and situational awareness are of primary importance in assessing not only the efficient and orderly flow of air traffic but especially the continuous keeping of the required levels of safety at every stage of air transport. Thus, specific precautions are requested when developing new instruments to be used in the control room which affect how the controllers/users work (at any level, so in terms of procedures, instrumentation to be used, etc.) and feel (ergonomic, stress levels, complexity of tasks, awareness). In particular, it is mandatory to ensure that developed applications improve the ATCOs service provision without worsening any aspect of their work and without generating new potential hazards, or at least that all noticeable contraindications are well-known and taken into account before deployment.
These requirements can only be satisfied by constantly considering the user needs through direct feedback provision and involvement in the design chain. Similar characteristics are typical of the User-Centred Design (UCD) methodology. UCD is defined by ISO standard 9241-210:2019[1] as "an approach to systems design and development that aims to make interactive systems more usable by focusing on the use of the system and applying human factors/ergonomics and usability knowledge and techniques" [37]. Figure 3.1 makes clear that UCD is a cyclic methodology in which a cycle is composed of four phases:

- **Understand context of use**. The first phase consists of defining the boundaries and activities for which the application is designed (thus the context) taking into account the characteristics of the users, (the tasks to be completed by and the working organization), the technical and physical environment in which the solution has to be exploited, and similar solution already developed, to predict needs, problems and constraints not

[1] In the ISO standard the synonymous term "Human-centred design" (HCD) is actually used, as it emphasizes the existence of relevant stakeholders other than the user. In this case, the terms and the relative definitions are anyway interchangeable.

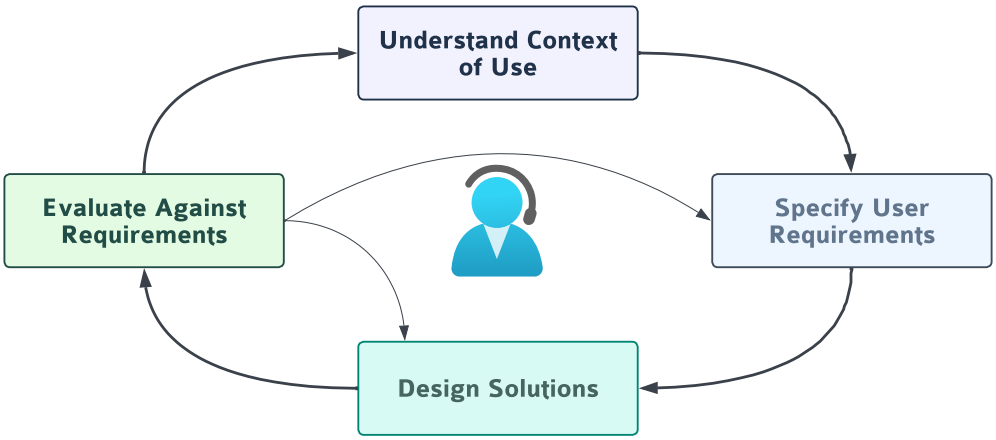


Figure 3.1: User-Centred Design cycle.

obvious before implementation. This phase is characterized also by a brainstorming activity.

- **Specify user Requirements**. After the limits and main characteristics of the project have been defined, designers start defining requirements to achieve the desired characteristic. Requirements can be organized and classified in different ways: they can be quantitative (a certain performance to be reached in numerical terms) rather than qualitative (it could be in terms of user perception or simplicity of use for instance), mandatory or recommended, and, for UCD, they can be application-centred requirements (like electronic hardware or software to be used) or user-centred requirements (in relation to the intended context of use and the business/operative objectives of the application). It could be necessary to define trade-offs between conflicting requirements.

- **Design Solutions**. Once requirements have been defined, the design phase begins in which multiple concept solutions are designed at the beginning and the best one (or multiple ones depending on budget and design validity/opportunities) is brought to a prototype level to be tested.

- **Evaluate against requirements**. The validation phase is that which differs the most in UCD with respect to other design methodologies, not user-centred. In this stage, the user is directly involved in the product testing, with a strong contribution in ensuring that all the user requirements (together with all the other requirements that could not be tested and validated without the user) are satisfied while pointing out any additional need or problem to be faced.

The user should have a central role in each phase of the cycle, being taken as a reference while defining the requirements and designing the solution, and directly involved in the validation phase.
A key aspect of a cyclic design is the iterative nature of the process itself. A review is made at the end of each phase which could lead to returning back to a previous stage of the cycle, for

example rewriting the requirements after new issues arise in the design phase. Once all these inner loops are completed and a validation is submitted whose outcomes satisfy the initial requirements, a new cycle begins in which the outputs results of the validation phase re-enter the loop as inputs of a new cycle. The process continues until the requirements fulfilment and the maturity level are sufficient for deployment/commercialization.

3.1.1 RETINA, DTT and beyond: a human-in-the-loop evaluation strategy

As previously stated, in air traffic management and more generally in the whole aerospace industry, operational safety levels have to be always kept as high as possible. This requires new technologies to be deployed only when they reach higher maturity levels with respect to what usually happens in other industries. As a consequence, much effort is put into the validation phases and usually, multiple development cycles are completed before a product is ready for deployment. This way of developing and maturing new technologies is intrinsic to the SESAR innovation pipeline structure.
Considering the strong reliance this thesis places on the outcomes of the previous RETINA and DTT projects, the work done could be thought of as the beginning of a third cycle which starts from the results of the DTT validation campaign, in particular for the exercise EXE-05.97.1-VAR-002 which was held by a research group from UNIBO. The exercise done in DTT could be identified as the second cycle of this iteration and it took as starting point the outcome of the RETINA validation campaign (which was the first cycle).
The first two cycles were evaluated through a real-time[2] human-in-the-loop simulation [38] (HITL), whose use is unavoidable when simulating for user-centred design.
In fact, both RETINA and EXE-05.97.1-VAR-002 were validated using a simulation rather than a real airport environment and adding AR layers to the scenario presented to real ATCOs through the CAVE environment existing at UNIBO. The exploitation of ATCOs (and a pseudopilot controlling the whole traffic on the pilot side) managing the virtual traffic during the whole validation (thus the term HITL) made them have a direct influence on the simulation outcome in such a way that is difficult if not impossible to replicate otherwise. In addition, HITL allowed for the identification of issues or missing requirements that could not be easily pointed out by other types of simulations. After RETINA, the exercise done inside DTT allowed bringing the best technologies selected from RETINA to a higher maturity level, while implementing and testing new features such as safety nets and air gestures.
Following two effective validations, a further step is proposed in accordance with the UCD methodology to start the implementation of the designed technology in a real environment. Hence, this thesis project aims to develop the framework for the application of AR, particularly the tracking system, to a real airport scenario.

3.1.2 Results from the previous projects

Being part of an iterative process, the passage to the real world highly relates to many useful outcomes from the previous projects, which are converted into a variety of requirements and

[2]It means that the simulation is run at the same speed at which events would occur in the real world.

know-how influencing almost every aspect of the real-world application. The whole application resembles the same structure from RETINA and 97.1-002 solution, and the most essential concepts and tools taken from the previous designs/validations were:

- Augmented reality tools.
- Application development platform.
- AR registration strategy.
- Tracking layers design.
- Runway overlays.
- Weather interface.
- Adaptive Human Machine Interface (HMI).
- Ergonomic aspects.

All these solutions have been useful in the new development cycle as design considerations or requirements and will be described in the next chapters as needed.

3.2 Application development strategy

The application developed for this thesis project aims to test the possibility of using AR technologies for a real control tower, thus tracking real aircraft in their exact world position and ensuring that the AR device capabilities are good enough to ensure a satisfying alignment of the real and virtual world for objects that are not related to the room in which the AR device is operated.
Two aspects are important in considering how the new system has been designed. First, the aim of this first phase of the work is not to fully reproduce what has already been done in simulations. In fact, information like the flight plan and apron parking positions require full access to - and integration of - an airport surveillance and traffic management apparatus, which is worth doing only after the general aircraft tracking system has been validated. Second, the real-world implementation does not aim to fully substitute the simulation apparatus as there are still many advantages in simulation. In fact, HITL real-time simulation allows complete control over the traffic scenario and the weather conditions, making it easier to test new solutions in conditions which are rare or unsafe to make happen in real life (for instance, testing heavy traffic conditions, the safety nets alerting interface for conflicting clearances or the runway overlays in low-visibility conditions), which is one of the reasons why the first two cycles were run in a simulated environment. Thus, the real-world application should be developed in the future together with the simulation, which is a fast lane for the first design and testing of new concepts.
Following these considerations, Fig. 3.2 presents the developing process of the application, in which the results from previous works help in defining the new context of use for the application. Since the actual developed application solves a sub-task of a real-world airport

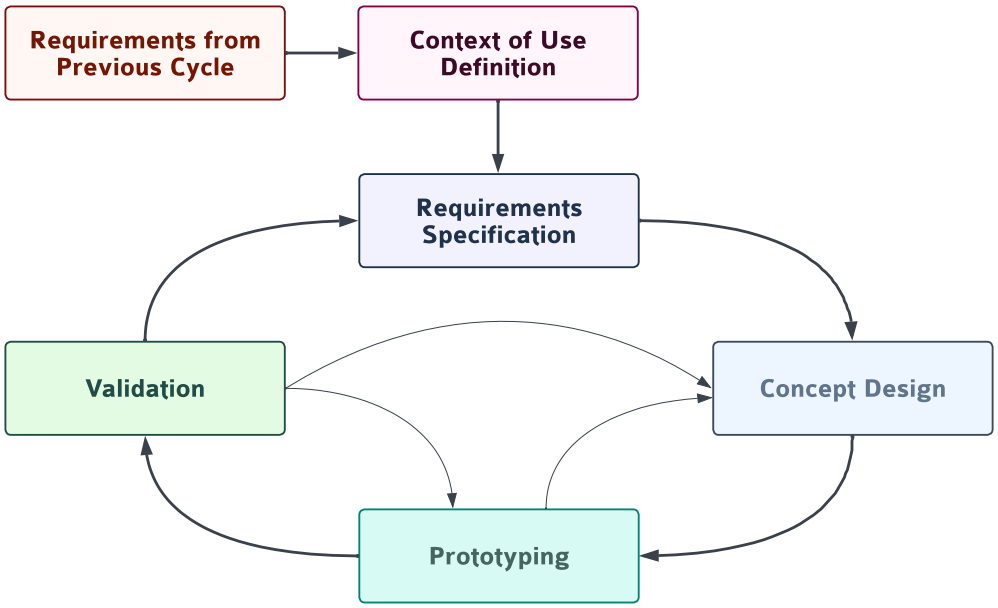


Figure 3.2: Real-time aircraft tracking system development cycle.

AR system implementation problem, the context of use, which largely comes from RETINA and DTT, is not part of the iteration, while a subset of requirements for the completion of the sub-task has been defined and iterated in the development cycle.

In the next sections and chapters, each phase of the development will be described.

3.2.1 Context of use

The prototype context of use has been defined as an early set of functionalities the AR system must provide while ensuring that the implemented design is compatible for future integration with all the systems considered in previous projects. The system should be able to identify the aircraft's position in the space around the user and track it with an AR overlay label correctly identifying the aircraft. In addition, the system should be able to show useful information about the aircraft to reduce the needing for looking at the controller workstation interfaces. For the same reason, also a holographic weather interface should be presented, containing real meteorological information.

The compatibility with some results and feedback from the previous works should also be considered, like the possibility to implement an adaptive HMI or tailor the visualized information according to the current environmental conditions. From the previous works also some limitations to the use of the application arise, especially regarding the holograms' visibility (which are typically visualized by emitted light rays) which dramatically drops in bright environments. This makes all the see-through AR display techniques less effective for example on sunny days. However, this is a minor problem as the main intended use of the application is for low-visibility conditions when no problems have been encountered in the past.

3.2.2 Identification of requirements for a real-world/real-time system

While exploiting the instruments and design choices from previous projects, a whole new set of mandatory requirements was prepared for the new application:

- Acquire the aircraft position with high accuracy.
- Acquire the aircraft position in real-time (minimizing positioning information reception latency).
- Track the aircraft position accurately in the AR environment.
- Acquire and track information for multiple aircraft at the same time.
- Track the aircraft position in real-time.
- Match the AR environment with the real world for accurate aircraft positioning.
- Render the aircraft tracking holograms minimizing computational latency.
- Evaluate possible error sources and counteract them.
- Ensure precision and reliability of tracking and positioning system.
- Develop an appropriate labelling system for aircraft tracking.
- Visualize ATC information usually available on terminals.
- Visualize helping tools for low-visibility conditions.
- Validate all the functionalities of the application.

In addition to these requirements, others result as corrections suggested by controllers during the previous validation phase, precisely the implementation of transparent tracking labels to mitigate the reduction of the view behind them, the positioning of the weather interface and the structure of the taxiway overlay.

3.3 Proposed design

From the old and new requirements, a possible implementation scheme has been developed, including some technical choices which are justified in the subsequent sections. The Microsoft HoloLens 2 was the AR HMD of choice, while the ADS-B data has been chosen to provide positioning and other surveillance data. The meteorological data have been provided through the METAR system. The choice of the HoloLens HMD for the visualization of the holograms comes from a user-centred method based on an integrated Quality Function Deployment (QFD) - Analytic Hierarchy Process (AHP) approach to select the best visualization display technology for the final user in industrial AR applications. The method was developed in the framework of the RETINA project [39]. The QFD-AHP methods were used on the traffic controllers involved in the RETINA validation campaign and HMD were selected as the

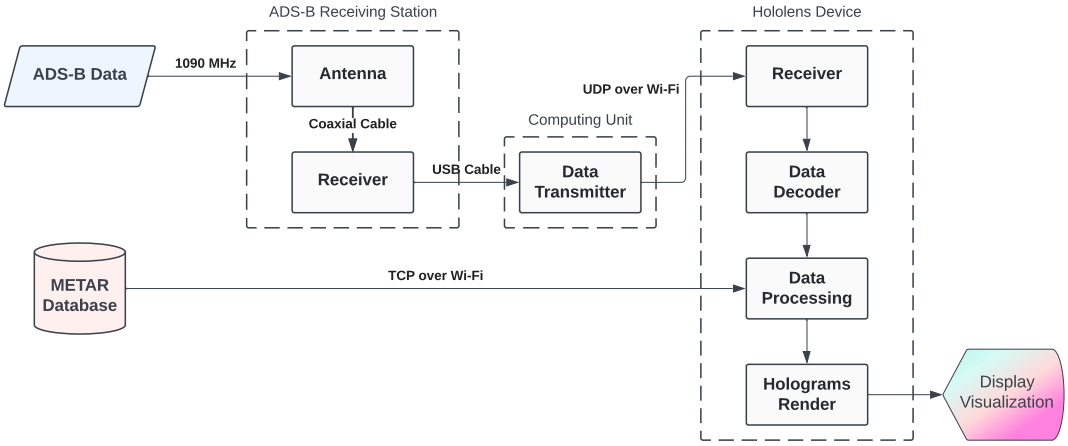


Figure 3.3: Proposed Design Block Diagram.

best technology with respect to see-through conformal displays[3] in terms of both holograms registration and adaptive HMI capabilities. Among the HMD, the Microsoft HoloLens and HoloLens 2 devices are currently the best commercial solution available with their strong computational capability, high hologram positioning accuracy and extended development support (see Section 4.1.1).

For the tracking of aircraft, ADS-B data have been used, as they already contain very precise position information together with much other useful surveillance information. In addition, the simplicity of the receiving station usage and of the data decoding made ADS-B technology remarkably suitable for a research application. Another useful characteristic of ADS-B technology is the indication of the accuracy of the transmitted positions, together with the fact that the position accuracy comes from GNSS and does not depend on the ground weather conditions as with radars, even if the weather can still influence the ADS-B receiving antenna range (which however is not an issue given the small dimensions of the ATZ).

As for the meteorological data, it was needed to provide users with actual present weather. Thus, the METAR data were used as they are the official data which are provided to the pilots and are available on the web for downloading.

The proposed scheme in Figure 3.3 summarizes the main blocks of the final application design. The ADS-B data are automatically transmitted by aircraft and are received by the ground station, composed of an antenna and a receiver unit demodulating the signal. The binary data are then transmitted to a personal computer (computer unit) over a serial port and are forwarded over Wi-Fi to the HoloLens devices.

All the computations on the ADS-B data are directly done by the HoloLens device, which decodes the messages, checks their integrity and uses them to position the labels in the right place. The device has also to render the entire AR layer computing virtual world holograms positions and considering the real/virtual world matching. Together with the aircraft labels, also the METAR data - which are directly downloaded by the HoloLens - are parsed and rendered inside the weather interface.

[3]Conformal Displays are similar to transparent windows on which the holograms are projected.

The implementation accomplished in the prototyping phase will be described in a separate chapter after the central tools employed will have been presented.

Chapter 4

Tools

Prior to explaining the application design, this chapter describes the tools, both hardware and software, which were used for the implementation of the developed system.

4.1 HoloLens development tools

Starting from the HoloLens 2 device, this section describes all the tools needed to develop apps for the HoloLens 2, based on the Universal Windows Platform (UWP) Operating System (OS).

4.1.1 HoloLens 2

Microsoft HoloLens 2 (Figure 4.1) is an augmented reality (AR)/mixed reality (MR) headset developed and manufactured by Microsoft and released in 2019. It runs the Windows Mixed Reality platform using the Windows 10 operating system. Its capabilities have largely enhanced those of the first-generation device HoloLens (figure 4.2), released in 2016. The HoloLens 2 devices can be considered laser-based stereoscopic and full-colour mixed-reality smartglasses.

Figure 4.1: HoloLens 2. Copyright Microsoft.

The device is designed to be used indoors and is able to project well-positioned holograms in the 3D environment, keeping them in place while knowing their own position with respect to

the room. This is possible due to its cameras constantly mapping the environment around the user's head, together with the Inertial Measurement Unit (IMU) continuously tracing the user's movement. The device is able to check the eye's position thus fitting holograms

Figure 4.2: HoloLens 1st generation. Copyright Microsoft.

based on the user's view. The HoloLens 2 updates the hand gestures already existing in the first-generation HoloLens to control the device, allowing them to be used also in developers' applications for an enhanced experience. The HoloLens 2 can be programmed using the Unity Developer platform (and also other platforms), combining the Unity assets with the Microsoft Mixed Reality Toolkit. Applications realized this way on a PC can be later deployed on the device. One HoloLens and three HoloLens 2 devices are available in the Virtual Reality and Simulation Lab of the University of Bologna.
The device specifications are listed in table 4.1.

4.1.2 Unity

Unity Real-Time Development Platform, by Unity Technologies, is a game engine platform used to develop 2D and 3D games and applications. It is written in C++ and supports C# based API Scripting through Visual Studio coding suite, providing the possibility to build applications for many different platforms such as Windows Desktop, Linux, Android or WebGL. Being compatible with Universal Windows Platform applications development, it became an almost mandatory instrument for HoloLens Software Development and deployment, by means of the Microsoft-produced "Mixed-Reality Toolkit".
Unity applications are designed through the Unity Editor. In the editor, a scene is created which corresponds to the scenario within which the game takes place. The scenario is characterized by a three-dimensional left-handed reference frame, with the X axis identifying the right direction, the Y axis referring to the upward direction and the Z axis indicating the depth. The player's point of view of the scene can be set up by means of a camera virtual object, which is positioned and oriented inside the scene. The scene is populated by the Unity Game Objects, which can be positioned inside the scene thanks to their transform property (which contains information about position and orientation). The Canvas Objects instead are used to provide overlays in the scene, such as a score field overlaying the camera view.

HoloLens 2 Technical Specifications	
Display	
Optics	See-through holographic lenses (waveguides)
Resolution	2k 3:2 light engines
Holographic density	>2.5k radiants (light points per radian)
Eye-based rendering	Display optimization for 3D eye position
Sensors	
Head tracking	4 visible light camera
Eye tracking	2 IR cameras
Depth	1-MP time-of-flight (ToF) depth sensor
IMU	Accelerometer, gyroscope, magnetometer
Camera	8-MP stills, 1080p30 video
Audio and speech	
Microphone array	5 channels
Speakers	Built-in spatial sound
Human understanding	
Hand tracking	Two-handed fully articulated model, direct manipulation
Eye tracking	Real-time tracking
Voice	Command and control on-device; natural language with internet connectivity
Environment understanding	
6DoF tracking	World-scale positional tracking
Spatial Mapping	Real-time environment mesh
Mixed Reality Capture	Mixed hologram and physical environment photos and video
Compute and connectivity	
SoC	Qualcomm Snapdragon 850 Compute Platform
HPU	Second-generation custom-built holographic processing unit
Memory	4-GB LPDDR4x system DRAM
Storage	64-GB UFS 2.1
Wi-Fi	Wi-Fi 5 (802.11ac 2x2)
Bluetooth	5
USB	USB Type-C
Software	
Windows Holographic Operating System Microsoft Edge 3D Viewer	

Table 4.1: Summary of HoloLens 2 Technical Specifications

The key asset of the Unity platform is the Unity engine, which is in charge of the behaviour of the applications at runtime. Through the Unity Engine, the Update function contained in scripts is cyclically run and, after each computation, the scene rendering is updated producing a new frame. While Unity applications can be directly run inside the editor during development, the ultimate goal is usually to build standalone applications.

4.1.3 Mixed Reality Toolkit

The Mixed Reality Toolkit (MRTK) is a Microsoft-driven project which provides Microsoft MR developers with a set of components and features to accelerate the creation of AR/MR applications for the Microsoft HoloLens on Unity. According to the project website statements [40], the MRTK delivers the following functions:

- Provides cross-platform input system and building blocks for spatial interactions and UI.
- Enables rapid prototyping via in-editor simulation that allows you to see changes immediately.
- Operates as an extensible framework that allows developers to swap out core components.

The MRTK is the most important tool in HoloLens app development as provides the code for developers to exploit the HoloLens capabilities and the hidden plugins to link the applications to the HoloLens Hardware delivering runtime capabilities. The toolkit is currently updated by the Microsoft MRTK team which directly collaborates with the HoloLens developing team.

4.1.4 C#

C# is a modern, object-oriented, multi-purpose, high-level, programming language. It allows developers to build and run applications exploiting the .NET free, cross-platform, open-source developer platform. C# was designed by Microsoft in 2000 and then recognized as an international standard in 2002. It is part of the C family of programming languages, having many similar features in particular with C++ and Java. C# has been constantly updated over the years with version 11 released in 2022 based on .NET 7.0 framework.
When developing applications in C#, one of the most used platforms is Microsoft Visual Studio.

4.1.5 Visual Studio and Windows Form

Visual Studio is an integrated development environment (IDE) from Microsoft. It holds up software development including computer applications, websites, web apps, and mobile apps. Visual Studio enables the development of all the windows applications' default types: Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. Windows Forms, among all, is the Graphic User Interface (GUI) of the

.NET framework, and it allows the user to interact with an application through graphic widgets (such as text boxes, buttons, dropdown menus, etc.).
In addition, Visual Studio comprises many features to help to code as the IntelliSense service, debugging and integrated API suggestions. Visual Studio is also the IDE of choice for the management of the Unity-integrated C# scripting API. Thus, the code written in Visual Studio can be directly interacted with through the Unity editor and vice versa, with an interaction framework which is already implemented in Unity and Visual Studio.
Furthermore, Visual Studio possesses functionalities for the deployment of applications for many different Windows operating systems, including UWP. In particular, Visual Studio is used for building Microsoft HoloLens Applications developed in Unity and deploying them on the device.

4.2 ADS-B receiving station

The ADS-B signal is automatically sent by aircraft on a 1090MHz frequency. In order to exploit the transmitted information, a receiving station is needed, which is composed of a receiving antenna that catches the waveform signal and amplifies it, and a receiver unit, which receives the signal from the antenna and demodulates it converting the information back into a binary signal making it ready for signal decoding. The two elements are fixed on an extendable tripod in such a way that the system can be quickly moved and set in a different location.
As for the antenna choice, it is a quarter-wave one, meaning that the antenna length equals one-fourth of the 1090MHz signal wavelength[1]. This is the most simple antenna type for ADS-B signals acquisition. The antenna is linked to the receiver through a 50Ω coaxial cable with BNC-type connectors. The ADS-B receiver has been previously developed by the LASIM - Laboratorio di Sistemi per l'InfoMobilità (Systems for InfoMobility Laboratory) of the University of Bologna. The receiver is based on a Silicon Labs USB to UART bridge controller. It is capable to acquire the electromagnetic wave signal from the antenna and demodulate it to digital information which is temporarily stored in the receiver buffer. The data can be transferred to a computer through a serial port communication interface via a USB 2.0 I/O cable for subsequent data processing. Since the receiving unit can only transmit the demodulated signal over a cable connection, a computer unit is needed to read the data from the unit and transmit them over a Wi-Fi connection, which is needed to communicate with HoloLens devices. In figure 4.3 and 4.4 the receiver and its operative setup are shown.

4.3 Matlab

In the preliminary phase of the thesis project, Matlab has been used to test the correct signal acquisition by the receiving unit and the unit's functioning. Matlab is a programming and

[1]The 1090MHz signal is characterized by a wavelength of $\frac{3 \cdot 10^8 m/s}{1090 \cdot 10^6 s^{-1}} \approx 275mm$. Thus, a quarter of the wavelength is approx. $69mm$. The antenna is slightly longer to counteract the material impedance.

Figure 4.3: ADS-B receiver unit developed by the Systems for InfoMobility Lab.

a) Antenna and receiver mounted on the tripod.

b) Whole receiving station setup.

c) Detail of the 1090 MHz quarter wave-length antenna.

Figure 4.4: Multiple points of view on the receiving station setup.

numeric computing platform commercialized by MathWorks® Inc. It is used by engineers and scientists to analyze data, build algorithms, especially with advanced mathematics, and design and simulate models. Due to its very high-level coding language, it allows processing data easier than with traditional programming languages. It was used to acquire and decode ADS-B data.

Chapter 5

System implementation

This chapter describes the implementation of the blocks composing the final platform. With reference to Figure 3.3, the user-centred design methodology has a clear effect on the design of the HoloLens block, whilst the receiving station and the computing unit designs do not have a direct impact on the user experience. The chapter is organized into sections describing the implementation of each block in a logical order rather than chronological. An introductive section describes some work done prior to the final platform implementation. Since ADS-B receiving station block has already been described in Section 4.2 and did not require any further work, it is considered a tool and is thus not described in this chapter.

5.1 ADS-B information selection and decoding

As a preliminary activity, the information contained in the ADS-B messages has been analyzed to decide which messages were valuable for decoding. In the end, all the position, velocity and identification messages were decoded. Since no information was provided about the functioning of the ADS-B receiving station, preliminary receiving tests were run on Matlab, as it provided a much easier interface for serial port communication with respect to traditional programming languages. It was noticed that the UART bridge collected the received messages in an internal buffer prior to serial transmission. The messages were already demodulated and converted into a binary array by the receiver. Among all the different downlink formats of Mode-S messages, the buffer was filled with all the 112-bit extended squitter (ES) messages received. Consequently, a check was needed in the decoding operation for the downlink format of the message (only DF 17 and DF 18[1] messages contain ADS-B information). The decoded information was then organized for each emitting aircraft and the position information was formatted into a KML file and visualized on Google Earth to check for the decoding algorithm's goodness, tracing the route of each detected aircraft. Figure 5.1 shows the aircraft visualization in Google Earth. Since it was not done before, each message type will be described below. A couple of examples of the decoding algorithms in Matlab will also be presented.

[1]The DF 18 is sent by aircraft not provided with a transponder, to let the station know that those aircraft can not be interrogated. However the message types and content is the same than for DF 17

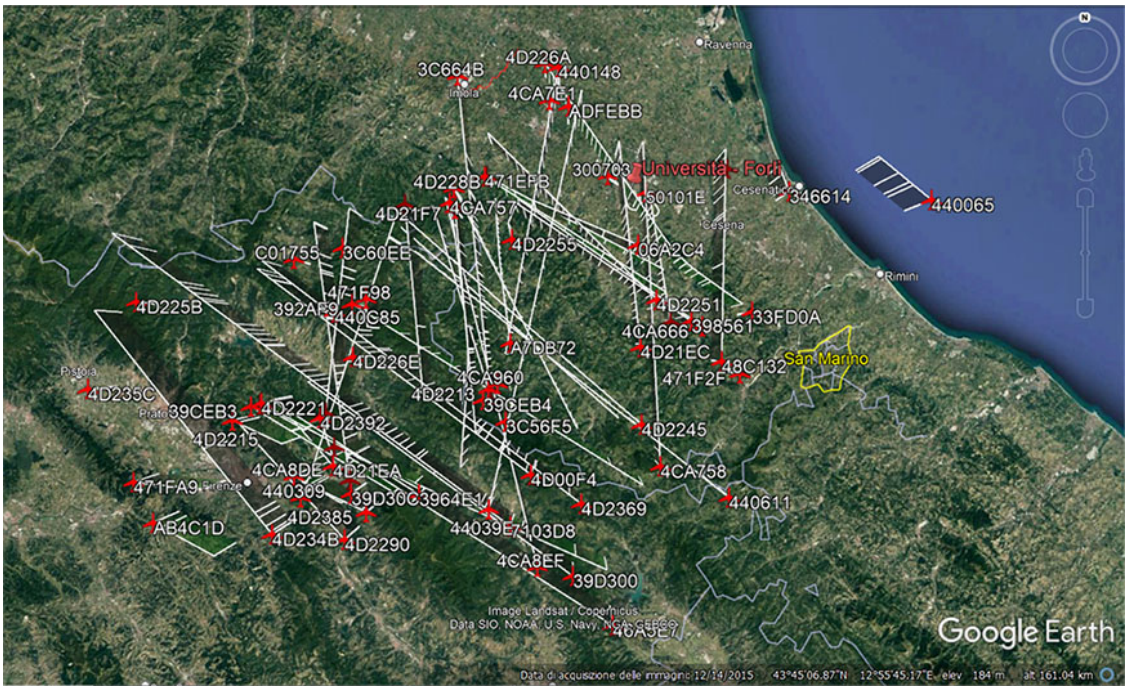


Figure 5.1: Matlab decoded positions visualization on Google Earth.

5.1.1 Message identification and ICAO code determination

Each message type shares the structure of a starting byte which contains information on the downlink format (5 bits which have to be decoded to check for DF 17 and DF 18 messages) and on the message transponder capabilities and a second byte which identifies the message type code (5 bits indicating type codes ranging from 1 to 31) and subtype (only encoded for some type codes). The following three bytes (bytes 3-5) are decoded and converted into a hexadecimal six digits string indicating the aircraft's 24-bit ICAO code, which uniquely identifies every registered aircraft provided with ADS-B worldwide. The following code portions show how the extraction of single bits is done in Matlab and how the ICAO code is retrieved respectively:

```
1       % read first byte and obtain df from first 5 bits
2       temp_1 = raw_data(i,1); % DF(bit 1-5) + (bit 6-8) (byte 1)
3       % right shift bits of 3 positions storing only first 5 bits
4       DF = bitshift(temp_1,-3); %move the bits to the right of 3 positions ...
            eliminating the three LSBs
5
6       if(DF < 17 || DF > 18) % if useless skip while cicle and go to next message
7           continue;
8       end
```

```
1       % To keep right order of bytes --> shift to left of 8 bits and add uint8
2       ICAO = uint32(0);
3       for k = 1:3
```

```
4           ICAO = bitshift(ICAO,8) + raw_data(i,k+1);
5       end %ICAO address bits 9-32 (byte 2-4)
```

After the ICAO code, bytes 5-11 contain the specific information related to the type code.

5.1.2 Aircraft identification

This message type is characterized by TC 2-4 and contains the aircraft callsign (assigned by ATM authority to the flight) and the aircraft wake category. Table 5.1 represents the structure of the 56 information message, where TC is type code, AC aircraft category and C* is a character, and the second number represents the number of bits related to that character, which is 6.

Table 5.1: Aircraft identification message structure

– TC, 5	AC, 3	C1, 6	C2, 6	C3, 6	C4, 6	C5, 6	C6, 6	C7, 6	C8, 6

The callsign is decoded converting the bits representing each character into a number corresponding to a position in the ASCII format characters encoding. The characters array is `#ABCDEFGHIJKLMNOPQRSTUVWXYZ##### ###############0123456789######`, so that the only used values are those encoding capital letters and roman numbers in the ASCII standard. A value for the spaces is provided, which are omitted if at the end of the callsign string. The second information contained in the message is the aircraft wake category, which is obtained by combining the DF and the TC. Whether many combinations are possible, only three categories correspond to those used by ICAO standards and are commonly transmitted: ICAO WTC L (Light - equivalent to TC = 4, CA = 1), ICAO WTC M (Medium - equivalent to TC = 4, CA = 2 or CA = 3), and ICAO WTC H or J (Heavy or Super respectively - equivalent to TC = 4, CA = 5).

5.1.3 Position message and CPR algorithm for position encoding

The 56 bits position message is probably the most important in the ADS-B protocol. Two types of position messages exist (See table 2.4), surface position messages, with TC 5-8 and used when the aircraft is on the ground, and airborne position messages, with TC 9-18 and 20-22, which can also transmit altitude information, from barometer for 9-18 and from GNSS for 20-22. Since altitude is not needed when an aircraft is on the ground, those bits are used for speed and heading information in surface position messages. Tables 5.2 and 5.3 show the position message structure for airborne and the differences with surface type.
To ensure high-precision values of latitude and longitude using a short 56 bits message, a special encoding technique is used, called Compact Position Reporting (CPR). Two different message frames are emitted alternatively, one even (identified by a bit 0 in message position 22) and one odd (with a bit 1 in the same position). The algorithm divides Earth into a different number of zones, 15 for each hemisphere for latitude and up to 59 for longitude, depending on the latitude zone. Each message type contains two 17 bits strings for latitude and longitude respectively, which identify aircraft position inside a CPR zone using slightly

Table 5.2: Aircraft airborne position message structure

Content		Frame	Msg	Bits
Type Code 9–18: with barometric altitude 20–22: with GNSS altitude	TC	33–37	1–5	5
Surveillance status 0: No condition 1: Permanent alert 2: Temporary alert 3: SPI condition	SS	38–39	6–7	2
Single antenna flag	SAF	40	8	1
Encoded altitude	ALT	41–52	9–20	12
Time	T	53	21	1
CPR Format 0: even frame 1: odd frame	F	54	22	1
Encoded latitude	LAT-CPR	55–71	23–39	17
Encoded longitude	LON-CPR	72–88	40–56	17

Table 5.3: Surface position frame differences with respect to airborne

Content		Frame	Msg	Bits
Movement	MOV	38–44	6–12	7
Status for ground track 0: Invalid 1: Valid	S	45	13	1
Ground track	TRK	46–52	14–20	7

different zones division for even and odd messages. By combining an even and odd pair, it is possible to identify the world position zone in which the aircraft is travelling and from each of the messages alone, the relative position inside this zone.
To decode aircraft position, an even/odd (e/o) message pair is needed to first identify the correct world zone. However, once an aircraft has been georeferenced, local decoding could be used, in which the last decoded position is used together with a single message, regardless of whether it is even or odd. For both global and local decoding, messages not older than 10 seconds should be used in order for the algorithms to work well[2]. Thus, in order to properly decode position messages, a time reference for the message transmission or reception time is needed. The decoding of the position messages should be divided into various tasks:

- Extraction of CPR encoded values for latitude and longitude from both types of messages, done in Matlab as in listing 5.1. The CPR values for latitude and longitude positions are obtained by converting the 17 binary strings from the message into a

[2]Global decoding requires the two even/odd messages used to be close enough in time. Local decoding requires the last decoded position to be close enough to the actual one, which results in a time requirement. Ten seconds can be a good trade-off between the frequency at which messages are sent, one pair per second, and the maximum distance between two encoded positions, which ensures the CPR algorithm works well (in the order of tens of nautical miles, corresponding to minutes for a commercial turbofan aircraft at cruise speed).

decimal integer.

- Design of an algorithm for the decoding of altitude from airborne position messages as below in listing 5.2. The altitude in ft is directly calculated by converting its corresponding bits to a number multiplied by 25 and scaled by 1000.
- Design of an algorithm for the implementation of Gray's code. This is used when altitude is greater than 50175 ft, and its use in encoding is indicated by the `q_bit` set to 0. The algorithm used in Matlab and later in C# is adapted from a Python script on the GitHub repository pyModeS by Junzi Sun [41] and is presented in Appendix A.1.
- Design of speed and heading decoding algorithm (Listing 5.3). The speed and heading encoding in the surface message is quite straightforward and the decoding only required a scale conversion and a switch once the right information bits had been isolated.
- Implementation of the local airborne position decoding function. The algorithm used the reference latitude to get the latitude zone and compute the new latitude. Then the same was made for the longitude.
- Implementation of the global airborne position decoding function. This is done first using the CPR value of latitude to compute the global latitude, then calculating the number of longitude zones for that latitude and finally computing the longitude.
- Implementation of the local ground position decoding function. The algorithm implementation is similar to that for local airborne decoding.
- Implementation of the global ground position decoding function. The algorithm implementation is similar to that for the global airborne decoding.
- Adaptation of code for automatic decoding function selection. The `if` condition switch is created to choose the decoding algorithm to use, based on received messages and time stamps.

Here the simple scripts for CPR, altitude, speed and heading decoding are listed:

```
% bit 5 of byte 7 (temp_7) is unused (time field bit)
CPR = bitget(temp_7, 3); % bit 6 of byte 7 (even or odd). Result is uint8, ...
    converted into double (bit 54)
YZ = uint32(bitand(temp_7, 3)); % frame for 2 least significant bits (55-56)
YZ = bitshift(YZ,8) + raw_data(i,8); % bits 57-64
YZ = bitshift(YZ,7);
temp_9 = raw_data(i,9); % bits 65-72 -> byte 9
XZ = uint32(bitget(temp_9, 1)); % 8 (bit 72)
YZ = YZ + bitshift(temp_9, -1); % final value of YZ -> bits 55-71
XZ = bitshift(XZ,8) + raw_data(i,10);  % bits 71-80
XZ = bitshift(XZ,8) + raw_data(i,11); % bits 71-88, final value of XZ
% YZ = integer value for lat CPR, XZ = integer value of lon CPR
%XZ and YZ are divided by 2^17 in functions to get a value between 0 and 1 ...
    identifying the relative aircraft position inside one CPR zone
```

Listing 5.1: CPR latitude and longitude parameters decoding.

```
if (type≥9 && type≤18) % barometric altitude case
    enc_alt = uint32(0) + bitshift(temp_6,-1); % temp_6 stores byte 6 content
    q_bit = bitget(temp_6,1); %get LSB from byte 6
    enc_alt = bitshift(enc_alt,4) + bitshift(temp_7,-4);
    if q_bit == 1
        alt = enc_alt * 25 - 1000; % LSB corresponds to 25 bits and they ...
            are shifted by 1000 feet
    elseif q_bit == 0 % in this case a complex algorithm called grey's ...
        algorithm is needed
        alt = altitude_decoder_qbit0(enc_alt);
    end
elseif (type ≥ 20 && type ≤ 22) % GNSS altitude
    enc_alt = temp_6;
    enc_alt = bitshift(enc_alt,4) + bitshift(temp_7,-4);
    alt = enc_alt*3.28084; % GNSS altitude is given in meters, here it is ...
        converted into ft
end
```

Listing 5.2: Altitude decoding code.

```
movement_encoded = uint8(bitshift(subtype,4) + bitshift(temp_6,-4));
    speed = ground_speed_decoding(movement_encoded);
    % heading in bits 46-52 (if 45 is 1, so valid) so 6 (45-48) plus 7 (49-52)
    if bitget(temp_6, 4) == 1
        track_enc = bitand(temp_6,7);
        track_enc = bitshift(track_enc, 4) + bitshift(temp_7, -4);
        track = 360 / 128 * track_enc;
    else
        track = NaN;
    end

%% Ground Speed Decoding
function speed = ground_speed_decoding(enc_speed)
% maximum ground speed encoded is 170 knots, sufficient for takeoff of ...
    every aircraft
if (enc_speed == 0)
    speed = NaN;
elseif (enc_speed == 1)
    speed = 0;
elseif (enc_speed ≥ 2 && enc_speed ≤ 8)
    speed = 0.125 * (enc_speed - 1);
elseif (enc_speed ≥ 9 && enc_speed ≤ 12)
    speed = 1 + 0.25 * (enc_speed - 9);
elseif (enc_speed ≥ 13 && enc_speed ≤ 38)
    speed = 2 + 0.5 * (enc_speed - 13);
elseif (enc_speed ≥ 39 && enc_speed ≤ 93)
```

```
26      speed = 15 + (enc_speed - 39);
27  elseif (enc_speed ≥ 94 && enc_speed ≤ 108)
28      speed = 70 + 2 * (enc_speed - 94);
29  elseif (enc_speed ≥ 109 && enc_speed ≤ 123)
30      speed = 100 + 5 * (enc_speed - 109);
31  elseif (enc_speed == 124)
32      speed = inf;
33  end
34
35  end
```

Listing 5.3: Speed and Heading decoding functions.

Global message decoding is similar for surface and airborne messages. The same is true for local position decoding. The main difference is that ground decoding is used when an aircraft is in the airport area, and its rough position is already known. For this reason, the algorithm implements a smaller division of the Earth using only a 90 degrees span for latitude encoding instead of 360 degrees. This way the precision is 4 times higher (up to 1.5m instead of $\approx$ 5m). Then, the decoded position has to be translated to the quart of the Earth for which it is the nearest to the airport reference position.
The full code written during the internship for the local airborne position decoding and for the global surface position decoding is provided in Appendix A.2 and A.3. This code is sufficient to understand the difference between global and local, surface and airborne decoding. A schematic of the work done in each algorithm is suggested by the comments lines reported within the code.

5.1.4 Airborne velocity

As airborne position messages encode altitude information, airborne velocity and heading information have to be transmitted on a dedicated message, together with further information related to airborne conditions, precisely vertical rate and the difference between barometric and GNSS altitude. This information is provided by messages with type code 19 and is structured as in Table 5.4.
Four subtype messages exist, with subtypes 1-2 used for surface velocities (those calculated using the GNSS positioning system) in subsonic and supersonic flight respectively and subtypes 3-4 used in the same way but reporting the indicated or true airspeed instead (they are only used when the GNSS positioning system is not working).

5.1.5 Other message type codes

Message type codes 28, 29 and 31 have not been fully decoded since they contain additional surveillance information not relating to the aircraft's dynamic state. However, partial decoding of specific fields in messages 28 and 29 was carried out at some point for debugging purposes. As an example, the onboard reference pressure setting has been decoded when evaluating the altitude information received from the aircraft, to ensure it was consistent with the airport METAR calculated QNH (see Section 5.6.4).

Table 5.4: Aircraft airborne velocity message structure

Content		Frame	Msg	Bits
Type Code TC=19 [Binary: 10011]	TC	33–37	1–5	5
Sub-type	ST	38–40	6–8	3
Intent change flag	IC	41	9	1
IFR capability flag	IFR	42	10	1
Navigation uncertainty category for velocity: ADS-B version 0 ADS-B version 1–2	 NUCr NUCv	43–45	11–13	3
Sub-type specific fields: Refer to Table 5.5 and Table 5.6		46–67	14–35	22
Source bit for vertical rate: 0: GNSS, 1: Barometer	VrSrc	38	36	1
Sign bit for vertical rate: 0: Up, 1: Down	Svr	69	37	1
Vertical rate: All zeros: no information LSB: 64 ft/min VR = 64 x (Decimal value - 1)	VR	70–78	38–46	9
Reserved		79–80	47–48	2
Sign bit for GNSS and Baro altitudes difference: 0: GNSS alt above Baro alt 1: GNSS alt below Baro alt	SDif	81	49	1
Difference between GNSS and Baro altitudes: All zeros: no information	dAlt	82–88	50–56	7

Table 5.5: Subtypes 1 and 2 message structures

Content		Frame	Msg	Bits
Direction for E-W velocity component 0: from West to East 1: from East to West	Dew	46	14	1
East-West velocity component All zeros: no information available Sub-type 1: Speed = Decimal value - 1 Sub-type 2: Speed = 4 x (Decimal value - 1) Unit: Knots	Vew	47–56	15–24	10
Direction for N-S velocity component 0: from South to North 1: from North to South	Dns	57	25	1
North-South velocity component All zeros: No information available Sub-type 1: Speed = Decimal value - 1 Sub-type 2: Speed = 4 x (Decimal value - 1)	Vns	58–67	26–35	10

Table 5.6: Subtypes 3 and 4 message structures

Content		Frame	Msg	Bits
Status bit for magnetic heading 0: not available 1: available	SH	46	14	1
Magnetic heading LSB: 360/1024 degrees Heading = Decimal value	HDG	47–56	15–24	10
Airspeed type 0: Indicated airspeed (IAS) 1: True airspeed (TAS)	T	57	25	1
Airspeed All zeros: no information available Sub-type 1: Speed = Decimal value - 1 Sub-type 2: Speed = 4 x (Decimal value - 1) Unit: knots	AS	58–67	26–35	10

5.1.6 CRC error control

The final three bytes of an ADS-B message contain a string of 24 bits used to check the integrity of the received message, which could be altered by environmental interference, overlapping with other messages or intentional manipulations. Among the different types of error detection algorithms existing for telecommunications, the ADS-B technology implements the cyclic redundancy check (CRC). The CRC algorithm is based on binary polynomial arithmetic and in the case of ADS-B messages, it uses a generator (a fixed divisor) of 24^{th} degree to divide the 88 bits message obtaining a 24-bits reminder, which is then appended to the message. Running the same algorithm on the entire 112-bits received message, it is possible to check if the obtained reminder is null and then the received message is intact. The CRC algorithm pseudo-code is presented in Figure 5.2. The algorithm implementation is further described in Section 5.5.1.

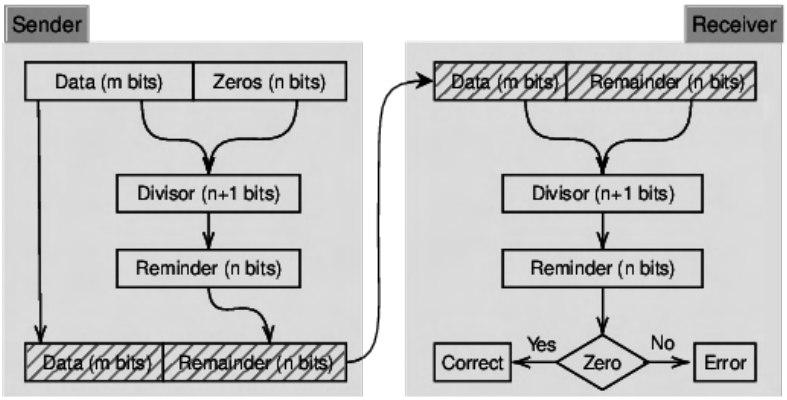


Figure 5.2: CRC algorithm structure. Source: [25].

5.2 ADS-B data transmission to HoloLens

After understanding the ground station setup and the ADS-B messages decoding, the focus has moved to the development of an application to forward the received messages to the HoloLens device. This second block is hosted by a computing unit with both USB serial port and Wi-Fi communication capabilities. A personal computer is well suited to this purpose. The data transmitter application has been developed using the Visual Studio module for Windows Forms design. The final application design is shown in Figure 5.3. A Windows Form application is composed of a functional GUI, which allows the user to configure the application and operate it in an intuitive way, and a background of scripts and complementary files which control the behaviour of the graphical elements as well as the runtime behaviour of the application. In this case, the application is run by one file called "Form1.cs". The application operating flow is described below.

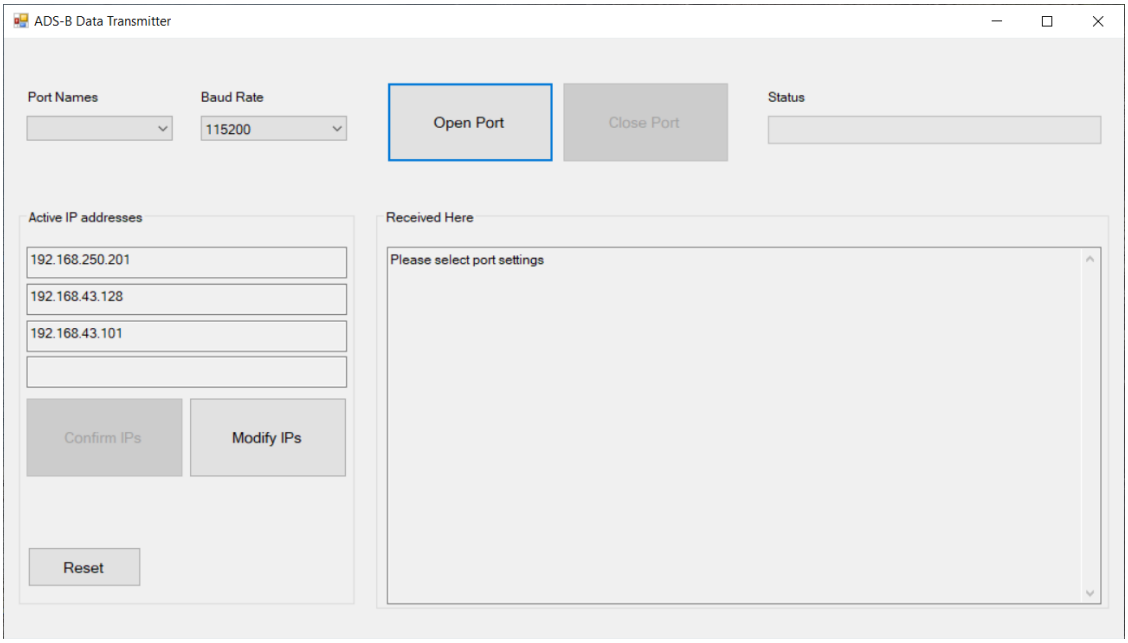


Figure 5.3: Serial.exe application for the transmission of ADS-B messages to HoloLens.

5.2.1 Serial.exe launch and configuration

When the application is launched, it checks for the active serial ports[3] (those to which recognized devices are linked), which can then be selected through the "Port Names" dropdown menu. The data transmission baud rate[4] is selected through the homonym dropdown menu. In order to send the received messages to the HoloLens 2, the two devices must be connected to the same Wi-Fi net, and they have to be set visible to other devices connected to the

[3]Since serial ports are virtual and use different types of interfaces to communicate, when a device connected to a computer uses serial port communication a driver is needed. In the case of the ADS-B receiver, Silicon Labs provides drivers which make the computer aware that the communication with the connected device has to be made through a serial port protocol. If this does not happen, the device can not be recognized as connected to a serial port.

[4]The number of bits which are transmitted through the link for every second, since in the serial port communication protocol data are transmitted one bit at a time.

same net. In addition, the HoloLens connection IP has to be provided to the application. Since more than one HoloLens were used at the same time and the connection link was often changed, a way to directly insert the multiple used IPs through the GUI was implemented. This way, it was possible to configure the application without modifying its scripts.

5.2.2 Messages forwarding

Once the application is configured (valid port, baud rate and at least one IP selected), the receiving unit can be activated by clicking the "Open Port" button. The application opens the serial port communication activating the receiving unit, and creates a listener element which is continuously triggered as soon as some data are available in the receiver buffer. When this happens, the listener launches a callback thread which checks for the number of bytes available[5], and if it is correct reads the bytes from the buffer.
Since an important requirement in the design of the AR system is to embed real-time operations, this intermediate application has as its main objective to forward the messages from the receiver to the HoloLens 2 device in the shortest possible time. For this reason, the forwarding algorithm is directly embedded into the serial port callback function. As soon as a message is read, it is sent to the selected HoloLens IPs and port (in this case all HoloLens were expecting the ADS-B data on port "12346") through a User Data Protocol (UDP). This type of protocol is used as it allows faster data transmission with respect to the Transmission Control Protocol (TCP)[6]. The Serial.exe application acts as a client connecting only when a message has to be sent to HoloLens. The HoloLens devices are instead configured as servers and are always listening for transmitted messages on the configured port.
After the transmission of a message is completed (the time-critical task), the application computes the current time and stores the message and the corresponding time stamp making them visible to the user. This was made for developing purposes, in such a way it was possible to check if data were being received and forwarded and to evaluate the time delay in the data transmission over the Wi-Fi link.
Since not all the received messages were useful - some were not ADS-B messages and some ADS-B messages are not decoded - a choice had to be made between selecting the messages prior to or after transmission. In the first case, some decoding operations have to be done delaying the data transmission, whilst in the second case a lower number of messages has to be sent thus resulting in less connection loading. At the end of the day, the first choice was preferred, as long decoding operations would result in excessive transmission delay while the connection was not noticed to suffer from the volume of transmitted data (which is rather low since the peak received-message rate was largely lower than 30 messages per second, cor-

[5]The messages stored in the buffer were 16 bytes long, as one heading byte with value 255 and one ending byte with value 0 were added to the standard 14-bytes/112-bits message.

[6]UDP is faster as it does not keep track of the order in which the data packets are transmitted over the net. Since data can follow different paths to reach their destination, in UDP it is possible that the sent packets arrive in the wrong order and some can be lost, which makes the protocol not suitable for the transmission of big data blocks which could be broken into pieces for transmission. In the latter case, a TCP is recommended, which adds a control block to the packets allowing data reconstruction (and requests for retransmission if any packet is missing) and rearrangement at the destination address, resulting however in lower transmission rates.

responding to a data volume of $30 \cdot s^{-1} \cdot 128b = 3840b/s = 480B/s < 0.5kB/s$).
Once the data are sent, the most important tasks are completed by the HoloLens application, which is described in the next sections.

5.3 HoloLens application framework

The central core of this thesis work is the AR system based on the HoloLens device, as the other elements of the designed system are just intended to provide the HoloLens application with the needed ADS-B data. Thanks to HoloLens being a commercial device provided with a mainstream operative system (UWP), the developer's work is made easier by the possibility to design applications similar to those working on commercial computers. Apps can be designed and built by means of existing developing platforms with the help of wide documentation and a populated community.
Due to the MRTK, which perfectly integrates with the Unity developing platform, HoloLens apps can be designed starting from a set of AR functionalities already enabled, like spatial awareness, air gesture recognition, hologram rendering and many others. Thus, to build and run an AR application for HoloLens, it is sufficient to import the MRTK Unity package into the Unity editor and configure it, adding the specific features of the application to be designed. The MRTK built-in scripts and functionalities are in charge of the control of the HoloLens Holographic processor and environmental tracking system (Cameras and IMUs), so no drivers have to be written.
The AR application object of this thesis has been implemented as a single Unity project, which achieves all the tasks presented in the third block of Figure 3.3. The application is programmed to receive, elaborate and manage the ADS-B data, using them to recreate a virtual scene with all the elements in the correct position. The holograms are then rendered and collocated inside the real world by means of the MRTK and HoloLens' built-in features. After the App has been designed in Unity, the project can be built and a "solution" package is obtained from which an executable application is built in Visual Studio and then distributed on the HoloLens device. The developed application has been called "test_GPS".
In the next sections, the test_GPS application implementation will be fully described, justifying and explaining all the design steps and the specific tasks accomplished by the application at runtime.

5.3.1 MRTK scene configuration

The first operation done is the creation and configuration of the Unity project for the app development. After creating a basic Unity scenario for 3D app developments, the MRTK has been imported and configured. The configuration adds a series of elements (non-dimensional Game Objects) to the scene, namely the "MixedRealityToolkit" and the "MixedRealitySceneContent" objects. The first in particular allows configuring the app depending on the intended destination of use (type of features to be activated, such as air gestures or vocal commands, dimension of the room, use of spatial awareness etc.). The setting interface is shown in Figure 5.4.

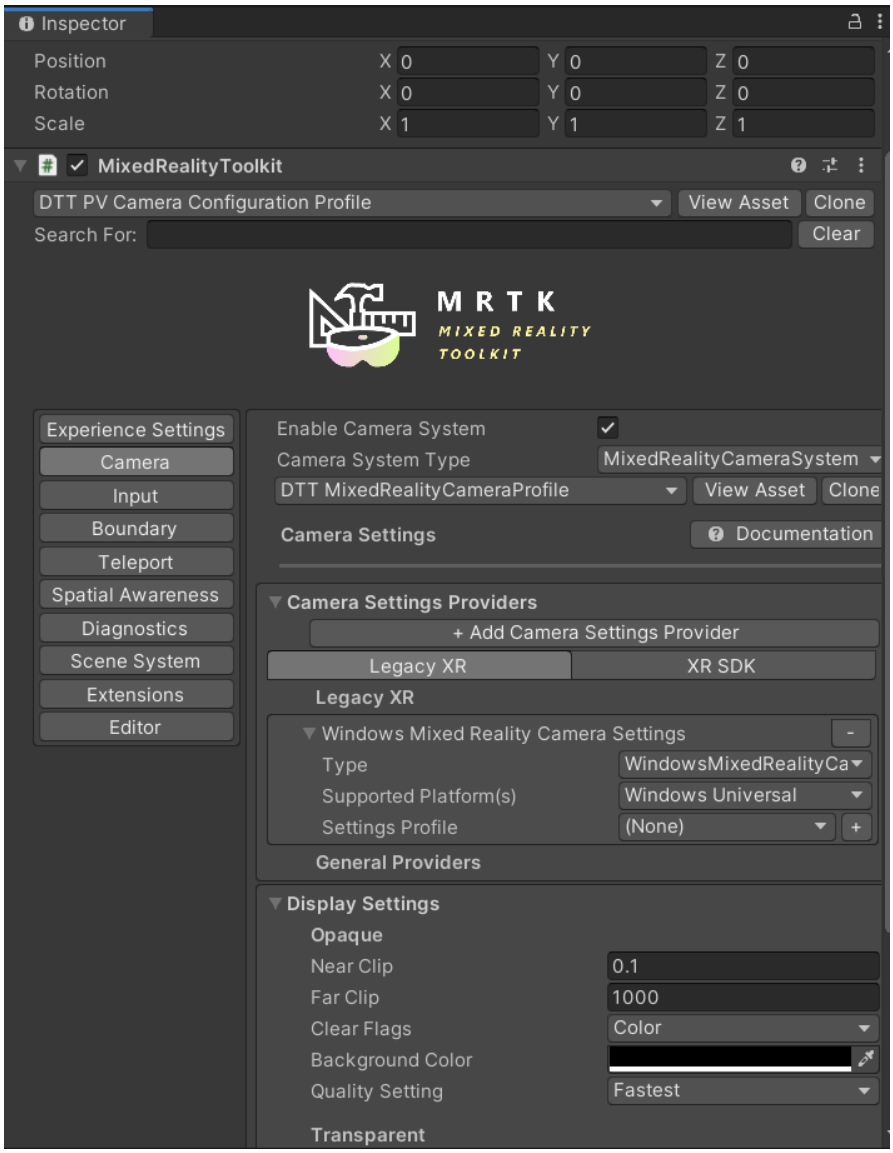


Figure 5.4: MixedRealityToolkit Game Object setting interface in the Unity Inspector.

For the purposes of this work, the following features were activated and set:

- **Experience Setting**, the environmental displacement expected while running the application. In this case, it was set to "room", meaning that HoloLens will be used by a user moving inside a room, rather than sitting, standing, or moving in an unbounded environment. Once the setting is fixed, the HoloLens device automatically adopts consistent tracking strategies.
- **Camera**, used to configure the virtual camera setting (the virtual PoV for the rendering of the holograms inside the scene), and then the quality of the hologram's rendering, the maximum and minimum distance of the rendered holograms and other rendering parameters.

Once the Unity scene has been configured for the MRTK, all the graphic elements added to the scene will be rendered as holograms in the application on HoloLens, with the Main

Camera Game Object in the scene corresponding to the user's point of view at runtime. At this point, the scene is ready for the development of the application.

5.4 HoloLens real and virtual-world matching

The first task in the app development has been the registration process typically needed in AR techniques: in this specific case points whose positions are known in the physical world have to be rendered in the exact same position from the user point of view by HoloLens. It is noticeable that the positioning information about aircraft provided by the ADS-B technology corresponds to geographic coordinates (usually latitude and longitude with respect to the world geodetic system[7] and altitude above the mean sea level[8]). Thus, differently from radar systems, the aircraft positions are provided as global values (with respect to a world-fixed reference system) rather than local values (with respect to a local point, as the radar antenna position).
This aspect makes conflict with the HoloLens tracking system architecture, which is meant to provide only local positioning (for example the position inside a room) by means of cameras mapping the surrounding environment and then tracking the device's position inside it, while the IMU can only help to improve the displacement awareness in large environments, being still considered a local positioning sensor.
Since HoloLens is not provided with a compass sensor or GPS receiver, the calculation of its position in a world reference frame is a complex task, that has been solved through the implementation of a two-step tracking strategy. This consists in joining the HoloLens' local positioning inside the room together with a global positioning of the room itself. Joining the two positions (and orientations), it is possible to know the HoloLens' position and orientation in the world, as with a change of reference frame (Figure 5.5). Since both position and orientation are needed for the reference systems matching, the more practical term "Transform" will be used from now on[9].
This strategy still presents limitations in that the HoloLens is anyway not capable of understanding its position outside of the boundaries of a room. In addition, location changes require repeating the entire procedure, redetermining the HoloLens' position in the new environment and the environment's global position.
The matching of the virtual and real environment is automatically done by HoloLens, which after mapping a room can fix the placement of the holograms keeping them in the same relative position with respect to the surrounding physical scenario. To provide the right positioning of the Holograms inside the environment, a calibration procedure has been employed to be sure that the HoloLens virtual reference system Transform is as expected. This procedure is described in Section 5.4.2.

[7]The world geodetic system is a standard representation model for the Earth as an ellipsoid. It is used for cartography and GNSS positioning. The current standard is the WGS84.

[8]The altitude above the mean sea level (AMSL) is defined as the vertical distance from the surface of the geoid model of the earth, which is the shape that the Earth oceans would take under the exclusive influence of the gravity of Earth, without considering winds and tides.

[9]The term "Transform" is used in the Unity editor to indicate the geographical properties of an object, i.e. its position and orientation inside the unity reference frame.

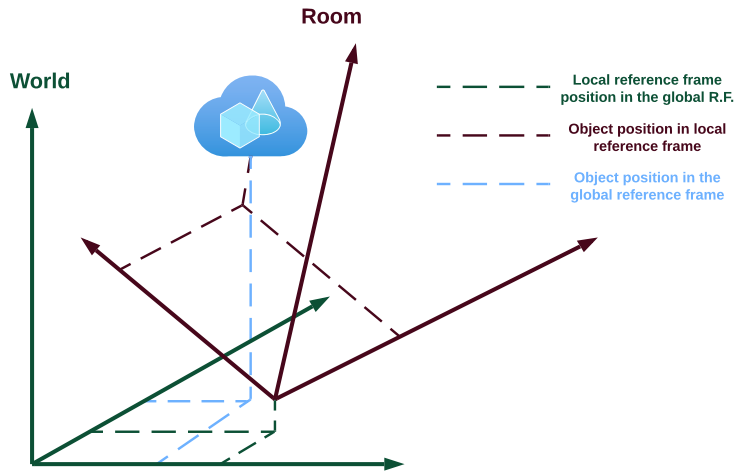


Figure 5.5: Global positioning of a point knowing its position inside a local reference frame and the position of the local frame with respect to the global one.

The MRTK provides two tools for the positioning of virtual objects in relation to the real environment. The older one is the World Anchor, which once activated fixes a hologram position in the space exploiting physical references next to it. However, since the device is always updating its knowledge of the surrounding environment, which is not perfect, the virtual model of the room could be altered over time, and then the holograms could change their virtual position while keeping the same position with respect to the real world. In this way, HoloLens is constantly changing the relative positions of the real and virtual environment with a behaviour which is ill-suited for global positioning.
Nevertheless, recent versions of the MRTK package come with a perfect feature for the purpose: the World Locking Tool (WLT). With this new package, which is easy to configure into Unity, the augmented reality application fixes the relative position of the virtual reference system with respect to the room. This way the reference system remains fixed while the device is displaced around the room. What moves is the virtual camera object (which is virtually in-built with the HoloLens main frontal camera), while the origin of the virtual reference system remains fixed, precisely at the initial position where the HoloLens was during the first deployment of the application. Thanks to this property, it is easier to match the environments as described below.

5.4.1 Matching of HoloLens reference system with a real-world reference system

As a clever feature of the Unity applications for HoloLens, a unit in the Unity scene corresponds to a meter in the reality. If it is made possible to calculate the distance of an object in space, such as the distance between two geographical coordinates in the world, it is then straightforward to convert that distance to a position inside the Unity scene.
Exploiting this aspect, together with the positioning of the virtual reference frame when using the WLT (the reference frame is locked with the same initial Transform - position of the origin

Figure 5.6: HoloLens position determination using Google Earth. By using multiple measures taken by hand inside the laboratory it is possible to measure HoloLens distance with respect to reference points in the map, such as the laboratory walls, thus retrieving the HoloLens position on the map.

and orientation of the axis - of the main frontal camera: the X axis on the right horizontally, the Y axis upward and the Z axis in the direction of the camera depth), the following passages have been observed to match the virtual reference system with the geodetic one.

Computation of the HoloLens global position

Since the ADS-B positions are supplied as geographical coordinates, the best way to calculate the aircraft position with respect to the Unity scenario is to also know the HoloLens position in the same reference frame. This is done in two steps by calculating the HoloLens' position inside the room and the room's position in the geodetic reference frame. For the determination of the room position, the Google Earth desktop application has been used which provides geographical coordinates with a resolution of about 30cm. Since Google Earth is capable of precisely measuring distances on the map, the HoloLens' position inside the room has been measured and then reported on the map using as reference the walls of the laboratory inside which the device was tested. This way, the geographical position of the device has been directly fed to the application as an input. Figure 5.6 shows the procedure as applied to the Virtual Reality and Simulation Laboratory building in Forlì.

Conversion of the Geographical coordinates into an ENU reference frame

The geodetic frame is an ellipsoidal reference system, which is similar to a spherical one in that the points are localized by means of two angles (latitude and longitude) and a radius (a distance from the centre, which can be reduced by the local radius value of the WGS84 ellipsoid giving a height above the ellipsoid). When dealing with non-cartesian coordinates it

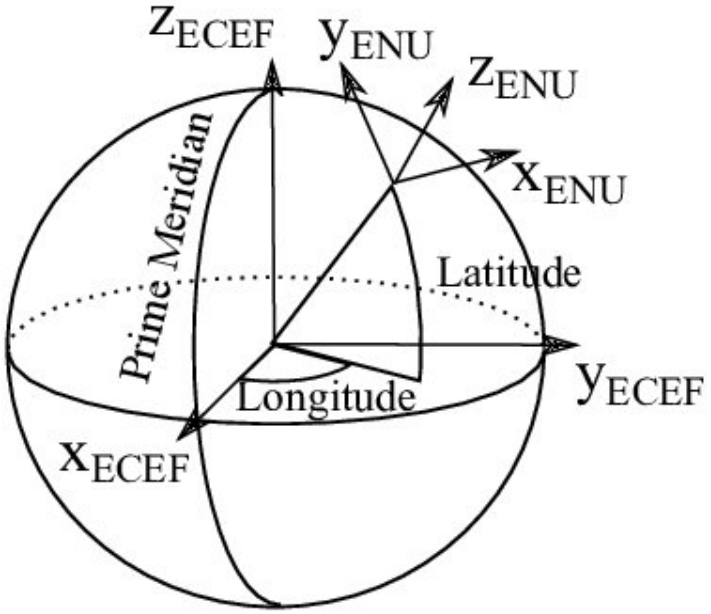


Figure 5.7: Differences among Geodetic, ECEF and ENU reference frames.

is rather difficult to calculate distances between points, and specific conversion tools can be needed. However, by means of an appropriate conversion algorithm, it is possible to evaluate the coordinates of a geographically referenced point with respect to an East-North-Up (ENU) oriented triad located on the surface of the Earth. To do this, geographic coordinates have to be prior converted into an Earth-Centered/Earth-Fixed reference frame. All these reference frames are summarized in Figure 5.7.

The conversion algorithm that is implemented in Unity has been found on the GitHub repository [42]. Having a reference position in geodetic coordinates (the user position on the Earth's surface), the algorithm creates an ENU triad with the origin in that position and calculates the position of another reference point inside the ENU reference frame. The algorithm is approximated, but with high-enough accuracy for small distances (in the order of a few degrees in latitude and longitude, corresponding to hundreds of kilometres).
The advantage of an ENU triad is that the vertical (Up) axis is already aligned with that of a HoloLens device calibrated in a levelled position. If this happens, the two reference frames only differ for a rotation about the vertical axis, which can be easily computed using Google Earth.

Computation of the HoloLens global orientation

When dealing with the global orientation, a similar procedure to that for positions has been exploited. In fact, the device has been initially aligned perpendicularly to one wall of the building, in such a way that it was directly possible to measure the orientation with respect to a global reference frame by measuring that of the building in Google Earth. An example is shown in figure 5.8. By calibrating the HoloLens on a perfectly levelled plane, the vertical axis of the device is aligned to the local vertical, of the ENU reference frame, and then the only rotation needed is about the vertical axis, which is directly measured from Google Earth.

Figure 5.8: HoloLens orientation determination. The HoloLens device is aligned perpendicularly to a wall, thus measuring the direction of that wall or of another wall perpendicular, it is possible to determine the device's direction with respect to the North direction.

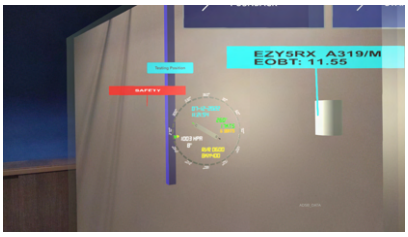
Positioning of physical points in the virtual world

Joining all the previous steps a procedure is defined to consistently align the virtual reference frame to a real object whose coordinates are provided in the geodetic reference frame.
Using the WLT suite and properly calibrating the HoloLens device, it is possible to fix the virtual reference frame Transform in a known position and orientation with respect to the room in which the device is operated. Using Google Earth it is then possible to retrieve the geodetic coordinates of this calibration point and the Z axis (the camera depth direction) rotation with respect to the North direction of an ENU frame with origin in that same point, provided that the HoloLens has been calibrated in a perfectly levelled position. At this point, given the geodetic coordinates of any object, it is possible to obtain its position with respect to the ENU reference frame through the conversion algorithm. Applying a rotation about the Y axis equal to that between the North and the virtual triad Z axis, the position computed in the virtual reference frame is made coinciding with the real position of the point.
In order to test the matching procedure, different objects have been tracked during the development. Some of these objects are shown in Figure 5.9.

5.4.2 Calibration procedure

As already suggested, a calibration procedure is needed to ensure that the HoloLens reference system is aligned as expected with respect to the real world. As part of the calibration, different objects whose position is known are tracked to ensure that the correct alignment is

a) Alignment of a hologram with a vertical trellis outside the laboratory.



b) Transparent hologram aligned to the panels of the CAVE environment in the Lab.

c) Another visual of the hologram aligned with the CAVE's panels.

Figure 5.9: Different objects tracked in the real world using geodetic coordinates. All the holograms have been aligned only given their geodetic coordinates and orientation.

reached. The calibration procedure is implemented as follows:

- The HoloLens device is placed in the calibration point (on a levelled table) and the application is run by means of vocal commands while not wearing the device. Markers are used to ensure that the device is aligned as desired.
- After ten seconds from the launch of the application an acoustic signal informs that the calibration procedure is completed. This fictitious time is needed in particular during the first deployment of the application as the device needs a few seconds to build a model of the room and fix the origin of the reference system using the WLT.
- Since the application is set to save the environment tracking from one session to another, if the calibration is not good (which can be determined by looking at the reference calibration objects), the application has to be deleted from the device and redeployed. If the calibration is successful, in the next sessions there is no needing for calibrating the device from a fixed point, since the device recognizes its position inside the room and keeps the reference frame in a fixed position.

After solving the matching problem and defining a valid calibration procedure, the implementation moved on to the real-time tracking of aircraft. First, ADS-B data had to be processed, and after that, a system for the real-time visualization of the tracked position was designed.

5.5 ADS-B data management

This section describes the acquisition and decoding of the ADS-B data and the subsequent organization of the decoded information prior to using it for the implementation of the tracking labels. All the operations on the ADS-B data are performed inside the main script of the Unity application (AndroidUDPListener.cs), which also implements the real-time positioning of the tracking labels and the decoding of the METAR data.

5.5.1 ADS-B data acquisition

The ADS-B data acquisition is directly linked to the transmission of the data from the Serial.exe application. In AndroidUDPListener.cs, a server is created through the UWP Data-

gramSocket() method. The socket is bound to receive data on port "12346" with a listener which is triggered when ADS-B data are received. A callback function is then launched which reads the message data and collects them into a buffer array which is then processed in the Update() cyclic function (the main function of Unity scripts, which is executed at runtime between frames rendering to update the scene content). Prior to adding the messages to the buffer, in the callback function, the current time at the moment of the reception is saved and the CRC algorithm is run to check for message integrity. The server code is listed in Appendix B.1, whilst the CRC code has been taken from the GitHub repository [43].

5.5.2 ADS-B data decoding

The ADS-B data decoding operation corresponds to the decoding block in Figure 3.3. The decoding algorithms are the same exploited on the Matlab software, and all the code has been converted from Matlab language to C#. The decoding is controlled by the Update() function. When a new message is received, the `gotMessage` boolean variable is set to true. In this case, in the Update, a `while` loop empties the message buffer calling the decoding function on each message. Listing 5.4 shows the structure of the while loop inside the update function. In the listing, the code for creating a string containing the decoded information is shown. This feature was used for debugging purposes and was then eliminated in the final release.

```
1  if (gotMessage)
2      {
3          gotMessage = false;
4
5          // At each update decode all the messages still not solved
6          while (counter_decoder < msg_stream.Count())
7          {
8              //For decoding purposes, a string is created which shows the ...
                   decoded information
9              var cons_txt = new StringBuilder("");
10
11             //Decoding function call
12             decode_ADSB(msg_stream[counter_decoder], ...
                   time_stamp_stream[counter_decoder], cons_txt);
13
14             var sb = new StringBuilder("");
15             foreach (var b in msg_stream[counter_decoder])
16             {
17                 sb.Append(b + ", ");
18             }
19
20             string messageCopia_ac;
21             if (cons_txt.ToString() != "")
22             {
23                 DateTime date_dec = new ...
                       DateTime(time_stamp_stream[counter_decoder]);
```

```
24                  messageCopia_ac = htmlMETAR.Split(new string[] { "\n", ...
                        "\r", "\r\n" }, StringSplitOptions.None)[1] + "\n" + ...
                        date_dec.ToString("hh:mm:ss.fff") + " " + sb.ToString() ...
                        + "\n" + cons_txt.ToString();
25                  //adsb_text.text = messageCopia_ac;
26                  adsb_text.text = "";
27              }
28
29              //Count the number of decoded messages
30              counter_decoder++;
31
32          }
33      }
```

Listing 5.4: While loop for ADS-B messages decoding.

The decoding function, called "decode_ADSB" has been previously tested on a desktop application. It takes as input the current message to be decoded and the corresponding time stamp. Inside the function, all the relevant message types are decoded as was done before in Matlab. As an example, the decoding algorithm for the airborne velocity message is reported in Appendix B.2. In the first part of the function, the aircraft is identified by means of the ICAO code and the type code of the message is obtained. The second part of the function is organized as a series of `if` statements that select the decoding algorithm based on the type code of the message.
When dealing with position messages, an algorithm has been implemented to decide which type of decoding algorithm has to be used. The first time a global position decoding algorithm is used to get a valid position. Therefore, an even/odd couple of messages received in a time interval lower than 10 seconds are needed to decode the position the first time. After that, local position decoding is used until ten seconds pass from the last global decoding. The structure of this if statement is presented in the Listing 5.5

```
1  if (type ≥ 5 && type ≤ 8) // ground position decoding
2  {
3      // Check if both even and odd messages have been received, if the time ...
           distance between the two messages is less than 10 seconds and if ...
           the last local position decoding has happened more than 10 seconds ...
           before
4      // In this case, global position decoding is used
5      if ((ac_list[ac_count].coord_eo[1 - CPR, 0] != -1) && (time_stamp - ...
           ac_list[ac_count].time_last_eo[1 - CPR]) ≤ 100000000 && (time_stamp ...
           - ac_list[ac_count].time_glb) ≥ 100000000)
6      {
7          // Execute Global position decoding algorithm
8          (lat, lon, inv_mex) = ...
               Globally_unambiguous_ground_position_decoding( ...
               ac_list[ac_count].coord_eo, CPR);
9
10         // check if valid result(two messages from the same zone for global)
11         if (inv_mex != 1)
```

```
12          {
13              // save new lon/ lat
14              ac_list[ac_count].lat_dec.Add(lat);
15              ac_list[ac_count].lon_dec.Add(lon);
16
17              // save last decoded altitude
18              ac_list[ac_count].alt.Add(alt);
19              // time_stamp of the last update to see if not too large time ...
                    passed for the local update.
20              ac_list[ac_count].time_glb = time_stamp;
21          }
22      }
23      // Check if a global position decoded less than 10 seconds before exists
24      else if ((ac_list[ac_count].coord_eo[1 - CPR, 0] != -1) && (time_stamp ...
            - ac_list[ac_count].time_glb) ≤ 100000000)
25      {
26          // Solve locally
27          float lat_ref = ac_list[ac_count].lat_dec.Last();
28          float lon_ref = ac_list[ac_count].lon_dec.Last();
29          (lat, lon, inv_mex) = ...
                Locally_unambiguous_ground_position_decoding(YZ, XZ, CPR, ...
                lat_ref, lon_ref);
30          if (inv_mex != 1)
31          {
32              ac_list[ac_count].lat_dec.Add(lat);
33              ac_list[ac_count].lon_dec.Add(lon);
34              ac_list[ac_count].alt.Add(alt);
35          }
36      }
37  }
```

Listing 5.5: Local or global decoding algorithm selection.

5.5.3 Decoded information management

A key aspect in the implementation of a real-time application is the constant update of aircraft information based on the received messages. For this reason, a data processing block has been designed that constantly updates the information about every tracked aircraft. Since the message decoding is done at runtime, the decoded information is directly organized inside the decoding function as soon as it is obtained.
In order to separate the information relative to different aircraft, a template class has been created (the "AC_list_struct" class), which is provided with fields for every type of information (Listing 5.6).

```
1  public class AC_list_struct
2  {
3      // all the needed variables;
4      public uint icao;                    // icao 24-bits unique code
5      public string call_sign;             // aircraft callsign
```

```
6       public string wake_category;        // aircraft wake category
7       public List<float> lat_dec;         // list of decoded latitude positions
8       public List<float> lon_dec;         // list of decoded longitude positions
9       public int[,] coord_eo;             //
10      public long time_glb;               // time of last global position ...
            decoding
11      public List<float> alt;             // list of decoded altitudes
12      public float speed;                 // aircraft speed
13      public float heading;               // aircraft heading
14      public int vr;                      // vertical rate
15      public long time_msg;               // time_stamp of last message
16      public int tot_msg;                 // number of messages received for ...
            that aircraft
17      public long[] time_last_eo;         //
18      public byte adsb_type;              //
19      public byte[] nic;                  //
20      public bool id_msg_reg;             //
21      public bool aircraft_created;       // becomes true when we instantiate ...
            the game object for a new aircraft
22  };
```

Listing 5.6: AC_list template.

Since the aircraft are identified by the unique ICAO code, which is comprised in every message, it is possible to evaluate whether the aircraft that sent a message is being identified for the first time or other messages from the same aircraft have already been received. The ICAO code is stored in the AC_list `icao` field. Inside the ADSB_decode function, a new object of the AC_list_struct class is added to the array "AC_list" every time a message from a new aircraft is received (if in the list of tracked aircraft, no one is assigned the decoded ICAO code). The AC_list element fields are initialized the first time and then are constantly updated every time a message is received. This way, the information related to each aircraft is constantly updated and organized to be visualized. The AC_list array can be then used to implement the Tracking Label of each aircraft.

5.6 Tracking Labels implementation

The Tracking Label system forms the graphic core of the application together with the weather interface. Starting from the aircraft information array, in AndroidUDPListener.cs these data are used for the live tracking of aircraft even in low-visibility conditions and for the visualization of important surveillance information in the head-up position. In the script, the graphical Game Objects are created and adequately positioned with respect to the real world. The chosen surveillance information is then visualized in the labels. According to the work done in previous projects, some corrections are applied regarding the graphical aspects of the labels. When dealing with the real-time tracking of the aircraft, different aspects were considered:

- **Design of the Label graphic interface**. Tracking Labels not only have to identify the position of an aircraft, but they also have to be clearly visible and possibly boost the comprehension the controller has of aircraft intentions. This is done by colour coding,

which can trigger the controller's attention at first impact, and by a proper choice of the surveillance information visualized, presented within the label. Since tracking labels are extended objects, the point of the label corresponding to the aircraft position has to be clearly identifiable. When designing the Tracking Labels, the dimension of the labels has to be considered, as they must be big enough to make the surveillance information readable without saturating the controller's view.

- **Visualization of a Tracking Label**. Following the same techniques from Section 5.4.1, the aircraft's geographical coordinates have to be converted to a position inside the Unity scene. The Tracking Label for that specific aircraft has then to be instantiated as a Unity Game Object and to be adequately positioned in the scene. Since aircraft are usually at great distances with respect to those contemplated by HoloLens for the rendering of the Holograms, additional considerations are needed in the choice of the label position, as will be explained later.

- **Management of aircraft labels**. As obvious in dealing with surveillance tasks, aircraft are constantly moving. Thus, a manner to constantly update their scene position and the position of their Tracking Label accordingly is required, i.e. the scenario has to be updated in real-time. In addition, multiple aircraft are usually detected at an airport and in the surrounding area, thus requiring multiple tracking labels to be visualized at the same time. For each tracked aircraft a label is needed, which has to be constantly updated. These tasks are accomplished through an aircraft labels positioning manager function, that has been implemented inside the AndroidUDPListener script.

- **Visualization of surveillance information**. The labels are used to provide the ATCOs with surveillance information directly in the proximity of the aircraft position. The information (textual) has to be clearly readable and must be updated accordingly to what is provided by the ADS-B messages. It is important to choose which information is provided to the ATCO at any time depending on the aircraft's intention and the environmental conditions.

5.6.1 Tracking Labels graphic

The tracking labels concept has been developed in the context of the RETINA project, with the selected graphic further refined during the implementation of Solution 97.1, exercise 002 of the DTT project. As the design of the labels had already been validated twice directly by ATCOs, it has been taken as a reference also for this work. Figure 5.10 shows how these labels appear in HoloLens. The cyan colour was assigned to departing aircraft, while the yellow colour was implemented for arrivals. The labels are constituted of a rectangle containing surveillance information and a vertical line linking the rectangle to the right aircraft. In this way, the aircraft is not covered by the label.

As an outcome of the ATCOs' feedback for the DTT EXE-05.97.1-VAR-002 validation campaign, the needing for improvements on this specific design resulted. In particular, it was suggested to make the labels transparent to avoid covering too much of the background.

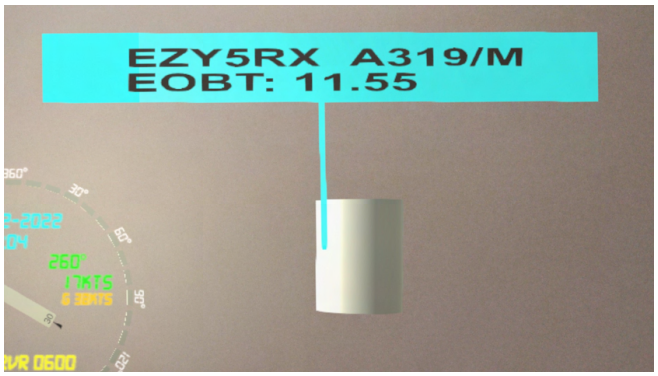


Figure 5.10: Label design for Solution 97.1-EXE002. The shown label refers to a departing aircraft.

Different solutions have then been proposed as shown in Figure 5.11, varying the transparency and the text contour and testing the impact on different backgrounds.
Among the two different text visualization possibilities, the first one has been selected (upper row of the previous figure). After direct testing on HoloLens, the label transparency was set to 50%. Figure 5.12 shows the result as later implemented in Unity.

Figure 5.11: Proposes for labels design after EXE002 validation outcomes.

5.6.2 Tracking Label Game Object design

Tracking Labels can be designed using the Unity Canvas UI (user interfaces). This toolkit allows the implementation of all the components typical of a user interface, including buttons and text boxes, into the Unity scene. Three different types of Canvas are available: Overlay Canvas, Camera Canvas and World Canvas. The Canvas term indicates in all cases the reference layer to which the graphic elements are related. The main difference among the canvas types is how they are positioned with respect to the scene. Overlay Canvas act as if it were directly superimposed on the display, and the related graphic is not interactable by

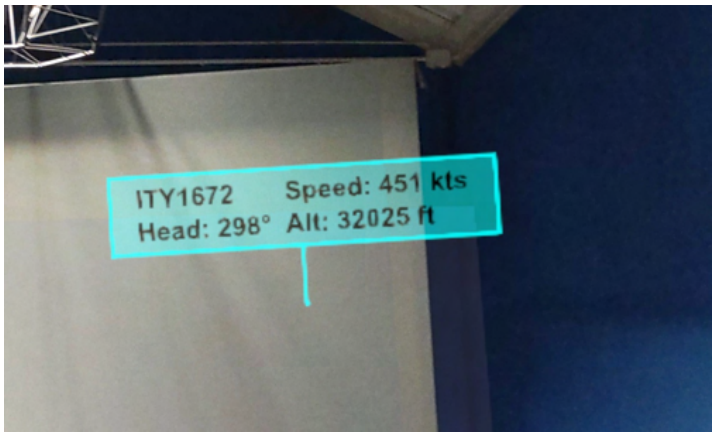


Figure 5.12: Label design implemented in Unity as visualized in the HoloLens device.

the user. The Camera Canvas layer is positioned parallel to the camera field and integral to it. Thus, objects which are built on this type of Canvas move with the camera and are fixed with respect to the User PoV. Conversely, the graphic elements which are rendered with respect to World Canvas have a fixed position inside the scene. When a Game Object is assigned to a canvas, functions exist to place it with respect to the canvas position, as in a local reference frame. However, it can be observed that apart from the Canvas Overlay type, the canvas position does not limit the position of the related objects. Panels, buttons and other UI elements can be rendered to be offset from the canvas, and as with every other Game Object, their position can be directly assigned in the global scene reference frame.
The Canvas selection process is described in Section 5.6.6. Here instead the Label design in Unity is described, which is independent of the chosen Canvas type. The first element of the label is a spherical Game Object which identifies the position of the aircraft. Above this element, a rectangular Panel Object is created using the selected transparent design. The mask for the Panel, with an opaque contour and a semi-transparent internal portion, was rendered using third-party imaging software and then imported into Unity as a source image. Exploiting the Unity Game Object hierarchical structure, the panel has been created as a child object of the sphere, so that it moves consistently with it. In the same way, a text object is created inside the panel object to host the surveillance information. Finally, a script is created to draw a line connecting the sphere and the panel. The line is not visible in Figure 5.13 as the script is executed only at runtime. After the first sketch of the label, its dimension and proportions have been tailored to ensure that all the needed text could be hosted inside the label and that the pointer and the panel were at the right distance to allow seeing the aircraft behind in reality.

5.6.3 Aircraft Game Objects position evaluation

Prior to visualizing the implemented label in Unity, a mandatory task is the creation of the aircraft Game Object in the right position in the scene. The utilised procedure is the one described in Section 5.4.1. Given an aircraft position, the coordinates (latitude, longitude and altitude of the aircraft) are passed to a function called GivePosition(). The function

Figure 5.13: Label object visualized in the Unity editor. On the left, in the hierarchy panel, the nested structure of the different elements is visible.

implements the coordinate system converter returning the aircraft position.
The converter uses the HoloLens initial position (again latitude, longitude and altitude), to calculate the differential coordinate components with respect to it. These coordinates are those of an aircraft in an ENU reference frame with the origin coinciding with the HoloLens' initial position. Since the ENU triad is right-handed, while Unity uses a left triad with a different orientation, the axes have to be rearranged. The ENU Up axis is the second axis (Y) in Unity, and the East axis (the first) points to the right and corresponds to the Unity first axis (X). The ENU North axis (pointing toward) corresponds to the Unity Z axis. After the coordinate components have correctly been assigned, they are given in a Unity reference frame aligned as the ENU frame. Since the HoloLens virtual reference frame is aligned with the HoloLens' initial orientation, a rotation about the virtual Y axis is needed to get the values of the components in the virtual reference frame. Since the converter returns the position as a Vector3 variable, the alignment is made by rotating this vector, called `aircraft_position` of a negative angle equal to `-holo_calib_dir` (the direction of the virtual Z axis with respect to the North direction) about the reference frame's vertical Y axis.

5.6.4 Altitude correction for labels positioning

As soon as the testing phase began, a significant error was found in the vertical positioning of the aircraft labels, with aircraft labels being often placed far lower than the expected position, in particular during the landing phase, and rarely above the expected position. After excluding any error in the ADS-B decoded altitude - which has been compared with that provided by commercial ADS-B-based aircraft monitoring websites -, three main causes were found which could contribute to the error:

- **Decoded altitude resolution**. The first problem is related to the ADS-B altitude information, which is provided with a resolution of 25 feet (7,62m), resulting in an uncertainty of ± 12,5 ft.
- **Altitude type**. Since the ADS-B system is derived from Mode S and shares some of the transmitted information, it usually transmits the uncorrected altitude (QNE),

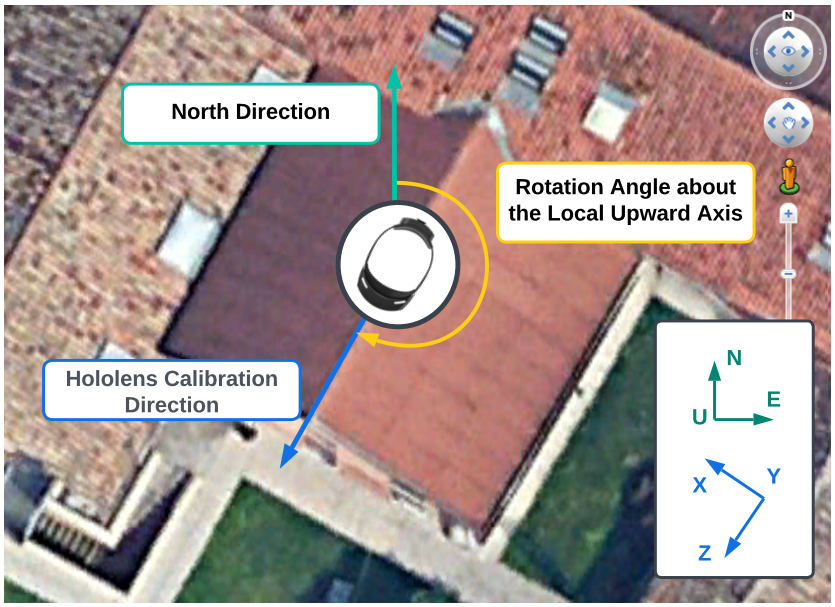


Figure 5.14: Orientation of ENU and HoloLens' virtual reference frames.

which is the altitude calculated from the pressure difference with the standard pressure at sea level (1013.25 millibars). Uncorrected altitude is used in Mode-S as it allows the evaluation of vertical separation when using TCAS systems. However, since the atmospheric pressure value fluctuates in time and space, the QNE altitude does not correspond to a fixed distance from the ground. This is instead provided by the QNH altitude, which corrects the QNE considering the pressure measured on the ground at a certain location, for example at an airport, and the elevation of the location, and corresponds to the pressure that there would be at the seal level in that location. Thus, a pilot which uses the QNH setting in his altimeter would measure at landing an altitude equal to the airport elevation, while the QNE altitude would variate depending on the atmospheric pressure. Since great variation can happen in the atmospheric pressure values, this error can reach unacceptable levels - in the order of hundreds of feet - in the worst cases.

- **Near ground effects**. The last error source derives from a physical condition known as ground effect. In simplicity, when an aircraft is landing and approaching the ground, the flow moving below the aircraft is compressed by the presence of the runway and a pressure increase is generated below the aircraft. This increase can influence the measured value of the static pressure giving a lowered value of the altitude [44]. This error can be worsened in the case of a deteriorated static port.

The biggest error source is the one given by the use of the QNE altitude. Anyway, having the QNH pressure value for the near Forlì airport as provided by the METAR data (Section 5.7), it was possible to correct the altitude. Then, considering the local elevation of the airport it was possible to further correct the altitude so that an aircraft would measure a value of 0 ft at ground level (i.e. the QFE altitude). Since the QNH pressure is provided in the METAR with a precision of 1 millibar, an error of $\pm$ 0.5 millibars can exist, which results in an altitude

uncertainty of 12-15 ft.
Summing the uncertainty on the altitude and QNH values, it can arrive at approximately ± 25 ft. In the worst case, this uncertainty can sum to the value given by the ground effect error, with divergences up to 50 ft measured in the final altitude value (aircraft measuring -50 ft at touching down).
Since the altitude was more frequently lower than the expected one and a label positioned slightly above the aircraft was not a big concern - as was also confirmed during validation - it was decided to increase the altitude value by default of 20 ft, corresponding to the average height of an aircraft fuselage above the ground. In addition, the altitude was set to 0 when becoming negative, avoiding unreal values and saturating the label to never go under the runway level. In Figure 5.15, two examples of the labels positioning before and after the corrections are compared.

Figure 5.15: Aircraft altitude errors correction. On the left, the aircraft altitude has not been corrected. On the right, the label is well positioned after altitude corrections.

As altitude determination issues were faced, the obtained position was good enough for the purposes of this thesis work.

5.6.5 Aircraft Game Object creation

Once the correct position for the aircraft has been found, this position can be used to create a Game Object with that Transform. An empty Game Object is created (which has the Transform property but is not rendered by Unity) as it is not convenient to directly place the Tracking Labels at the aircraft position. Since tracked aircraft can be at great distances from the user position, also their position inside the unity scene would be recognized as far by the HoloLens device. This has two main drawbacks. The first results from HoloLens being a stereoscopic-vision device. Thus, two different images are rendered for each eye to provide a three-dimensional perception of the holograms. HoloLens is limited in the capacity to render holograms at far or very small distances since the two displays are set for a focal distance of 2 meters: the suggested distance for holograms' positioning is between 1.25 and 5 metres [45]. In addition, as a Game Object is moved away from the user, its dimension has to be increased to maintain the same angular size, as shown in Figure 5.16. However, if the object dimension is increased, it requires a higher computational effort to be generated in the Unity space, even if the final render dimension is the same. For this reason, two different Game

Objects have been implemented. One is the empty Game Object previously described, which is dimensionless and can then be placed everywhere in the scene without consuming excessive computational resources. The Tracking Label is then implemented in a second Game Object which is kept at an adequate distance from the user.

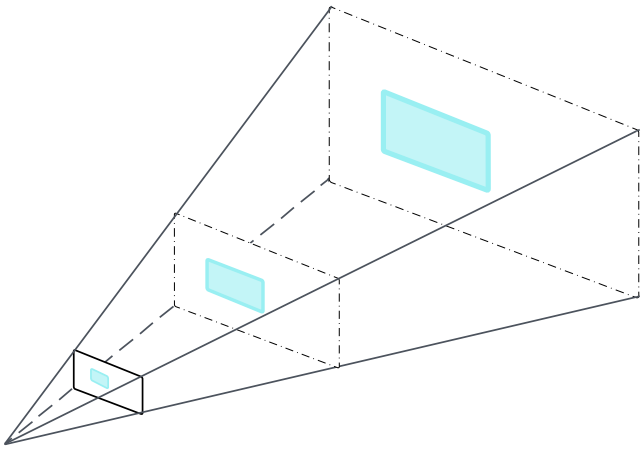


Figure 5.16: As an object is moved away from a point of view, it has to be scaled accordingly to maintain the same angular size.

5.6.6 Tracking Labels positioning

In implementing the Game Object for the tracking label, two different approaches have been tried. Since the ADS-B receiver is capable of tracking many different aircraft at once, many holograms could populate the application scene. Moreover, every user movement, even small head shaking - which continuously happens -, results in the device recalculating the position of every hologram and rendering again it, whether the holograms are moving in space or not. This happens at a rate of 60fps, with the HoloLens device also making some adjustments between frames to keep the holograms steady. Summing up the high computational effort of hologram rendering and the need to render a high number of holograms at once, the computational effort could be very high.
For this reason, at first glance, an efficient approach was tried for the rendering of the tracking labels. The labels were rendered using the Camera Canvas, fixing the distance of the label at three meters from the user. In order to calculate the positioning of the tracking labels, a Unity C# "MonoBehaviour" class built-in function has been used, called `WorldToScreenPoint`. This function returns the world position of an object Transform as the pixel position on the screen at which this object appears. Since the Camera Canvas is consistent with the camera field of view, it is straightforward to position the Game Object on it. In this way, the computation of holograms' positions is easier as the canvas element is not moved with respect to the device. However, due to an intrinsic limitation in the use of Camera Canvas by HoloLens, the resulting labels were not stable as they used to shake and jitter[10], as visible in Figure 5.17.

[10]Following Microsoft definition "users observe jitter as high-frequency shaking of a hologram".

Figure 5.17: Hologram's shaking visualized by merging two consecutive frames of a video recorded on HoloLens while using Camera Canvas rendering.

The Camera Canvas was then left for a more stable approach using World Canvas. With this second approach, a World Canvas Game Object is created to host the labels. In this case, in order to correctly visualize the label, a point has to be found on the gaze line projected from the user's point of view to the aircraft position. The label can then be positioned at the found point.
An MRTK package was found which was able to solve this task while ensuring the stabilization of the Holograms. The system is called Solver and comprises a series of components which can be added to a hologram Game Object to place and stabilize it in different ways with respect to another Game Object's position. The solver system stabilized the holograms by computing their position in the field of view during the period within the rendering of a frame and the computation of the next frame. Among the Solver system components, the "InBetween" is capable of suspending a hologram at an intermediate position between the camera and another Game Object. Using the aircraft empty Game Object together with the virtual camera position it is then possible to suspend the label on the gaze ray directed toward the aircraft, with a perfect alignment and keeping the hologram at a fixed distance from the user. Since the InBetween control script determines the suspended object's position as a percentage of the line joining the other two elements, the built-in script was modified to keep this distance fixed to 4 meters from the user. In this way, the Tracking Label Game Object has not to be modified if the tracked aircraft distance is varying, and different labels present the same dimension. Figure 5.18 shows how aircraft labels maintain the same dimensions for aircraft at different distances. In Figure 5.19 instead, the stability of the holograms for a moving aircraft is shown. In addition to the positioning of the labels, another important task is the orientation of the same. Since this is a common task in Unity, it is simply accomplished by adding the billboard script component to the label Game Object. In this way, the label constantly faces the user while keeping its vertical orientation.

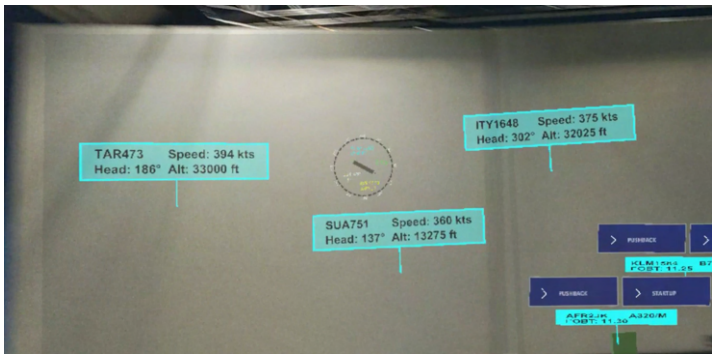


Figure 5.18: Hologram's labels consistent dimensions.

Figure 5.19: Aircraft label remains stable in different consecutive frames as aircraft moves.

5.6.7 Tracking Labels content

As a further step in the Tracking Labels implementation, surveillance information had to be added to the labels. This is easily done by formatting the text UI elements which are contained in the label Game Object using the ADS-B information. During the project development, the information used was the aircraft callsign, which is used by controllers to identify the flight, the speed (whether it was surface speed or airspeed), the heading, and the QNE altitude.
Whether only this information was used during the thesis, mainly to test the application, the information visualized can be easily tailored to any specific needing. The vertical rate and the aircraft wake category - which is used by ATCOs for time separation - are already available, the aircraft distance from the tower is simply the modulus of its position vector in Unity, while that of a landing aircraft from the runway can be computed as the modulus of the vector difference between the aircraft Game Object position vector and the runway threshold position vector, which is also available in Unity. In addition, the QFE altitude, which would show the aircraft at a 0 altitude at landing and take-off, is already used to solve the altitude positioning issues and is not visualized only for debugging purposes.
Furthermore, the visual information could be also tailored to specific conditions, like the flight phase or the visibility level, which is already made available to the application from the METAR data (see Section 5.7).

5.6.8 Tracking Labels management system

The last task in the Tracking Labels implementation is the management of the tracked aircraft to provide the continuous creation, positioning and update of the Tracking Labels. For a constant update, the manager algorithm has been implemented inside the Update() cycle of

the AndroidUDPListener script, right after the while loop for the ADS-B decoding, in such a way that the Labels are updated with the most recent information available. Since the managing algorithm is quite compact, it is entirely listed below in three blocks and subsequently explained. The majority of the algorithm is well explained by the comments directly inside the listing, and only a few comments will be added.

```
1  // At each update verify all the aircraft existing in the list
2  for (int acac = 0; acac < ac_list.Count(); acac++)
3  {
4      // Check if the position is available for the aircraft (altitude saved)
5      if (ac_list[acac].alt.Count() > 0)
6      {
7          // Calculate the new position of the aircraft empty object
8          // The function takes as input lat, lon and corrected altitude and ...
               returns a vector three containing the position information in ...
               the Unity reference frame
9          aircraft_list_manager.GetComponent<AC_Pos_Manager>().GivePosition( ...
               (double)ac_list[acac].lat_dec.Last(), ...
               (double)ac_list[acac].lon_dec.Last(), ...
               Math.Max(((ac_list[acac].alt.Last() + (metar_QNH-(1013.25 - ...
               landing_gear_pressure)) * 27) * 0.3048 - ...
               aerodrome_elevation),landing_gear_pressure * 27), out Vector3 ...
               aircraft_position);
```

Listing 5.7: Tracking Labels management algorithm. Computation of aircraft position.

In Listing 5.7, the first part of the code is presented. The algorithm opens with a for loop and checks all the aircraft which are contained in the AC_list array created by the ADS-B decoding function. If the position information is available for an aircraft (if position messages have been received for an aircraft with that ICAO code), then the position is computed in the Unity coordinates. The next listing blocks show the code for the creation or the update of the Tracking Labels.

```
1          // If the aircraft gameObject has not been created yet, an aircraft ...
               empty gameObject and the corresponding Tracking Label ...
               gameObject are created.
2          if (ac_list[acac].aircraft_created == false)
3          {
4              // Create and set the correct position for the new aircraft ...
                   world object using the ENU coordinates.
5              GameObject my_aircraft = Instantiate(aircraft_world_label, ...
                   aircraft_position, Quaternion.identity);
6              // Rotate the gameObject position vector to align with the ...
                   application reference frame
7              my_aircraft.transform.RotateAround(new Vector3(0f, 0f, 0f), ...
                   Vector3.up, -holo_calib_dir);
8              ac_list[acac].aircraft_created = true;
9              // Set the aircraft gameObject name so that it coincides with ...
                   the aircraft ICAO
```

```
10              my_aircraft.name = ac_list[acac].icao.ToString();
11
12              // Create and set the correct position for the tracking label
13              //The label is created as a child of the reference world canvas
14              GameObject my_aircraft_label = ...
                    Instantiate(aircraft_canvas_label, new Vector3(0, 0, 0), ...
                    Quaternion.identity, canvas_aircraft_positioning.transform);
15              //The label name is set as <ICAO>_label
16              my_aircraft_label.name = ac_list[acac].icao.ToString() + "_label";
17
18              //Assign the aircraft empty object as the one to use to compute ...
                    the Label position with the InBetween component
19              refsolver = ...
                    my_aircraft_label.GetComponent<Microsoft.MixedReality. ...
                    Toolkit.Utilities.Solvers.InBetween>();
20              //Activate the InBetween component to automatically position ...
                    the label
21              refsolver.SecondTransformOverride = my_aircraft.transform;
22              my_aircraft_label.GetComponent<Microsoft.MixedReality.Toolkit. ...
                    Utilities.Solvers.InBetween>().enabled = true;
23
24              //Orientate the label to always face the camera
25              Vector3 directionToTarget = my_aircraft.transform.position - ...
                    Camera.main.transform.position;
26              my_aircraft_label.transform.rotation = ...
                    Quaternion.LookRotation(directionToTarget);
27
28              // Create the label surveillance content
29              Text my_aircraft_text = ...
                    my_aircraft_label.GetComponentInChildren<Text>();
30              my_aircraft_text.text = string.Format("{0,-13}", ...
                    ac_list[acac].call_sign) + "Speed: " + ...
                    string.Format("{0,3}", Math.Round(ac_list[acac].speed)) + " ...
                    kts\nHead: " + string.Format("{0,3}", ...
                    Math.Round(ac_list[acac].heading)) + "°  Alt: " + ...
                    ac_list[acac].alt.Last() + " ft";
31          }
```

Listing 5.8: Tracking Labels management algorithm. Creation of a new Tracking Label.

Listing 5.8 describes the code related to the creation of a new Tracking Label. In order to create multiple labels, the empty Game Object for the aircraft position and the model of the Tracking Label are saved as Prefabs in Unity, which means they are considered reference blocks which can be copied and added to the scene by scripts. Using the new computed position, a new aircraft GameObject is instantiated and its position vector is rotated to convert it for the Unity reference frame. The instantiated Game Object is then named after its ICAO code so that it can be retrieved when its position has to be updated. In a similar manner, the tracking label is instantiated as a child of the World Canvas. Then it InBetween component is set with the corresponding empty aircraft GameObject as the second point for the positioning other than the user's point of view. After that, the label is renamed in a similar manner as

the empty object and is oriented toward the camera position. Finally, the text component of the label is filled with the chosen surveillance elements.

```
1           else
2           {
3               // Same procedure as before, just finding the already existing ...
                    game object instead of instantiating a new one
4               GameObject my_aircraft = ...
                    GameObject.Find(ac_list[acac].icao.ToString());
5
6               //Position is assigned as for a new aircraft
7               my_aircraft.transform.position = aircraft_position;
8               my_aircraft.transform.RotateAround(new Vector3(0f, 0f, 0f), ...
                    Vector3.up, -holo_calib_dir);
9
10              //Find the label gameObject to align right rotation and text ...
                    content
11              GameObject my_aircraft_label = ...
                    GameObject.Find(ac_list[acac].icao.ToString() + "_label");
12
13              my_aircraft_label.transform.rotation = ...
                    Quaternion.LookRotation(directionToTarget);
14
15              Text my_aircraft_text = ...
                    my_aircraft_label.GetComponentInChildren<Text>();
16              my_aircraft_text.text = string.Format("{0,-13}", ...
                    ac_list[acac].call_sign) + "Speed: " + ...
                    string.Format("{0,3}", Math.Round(ac_list[acac].speed)) + " ...
                    kts\nHead: " + string.Format("{0,3}", ...
                    Math.Round(ac_list[acac].heading)) + "°  Alt: " + ...
                    ac_list[acac].alt.Last() + " ft";
17
18          }
19      }
20  }
```

Listing 5.9: Tracking Labels management algorithm. Update of an existing Tracking Label.

If the Tracking Label of an aircraft has already been created, the code in 5.9 is used. This code is quite similar to the previous block. It is interesting to notice that since the Label is positioned automatically with respect to the aircraft position, only the latter has to be calculated by the script. Here the `Find` command is used to select the right Game Objects which have to be modified.
With the previous algorithm, the main block of the application was completed. The system was then able to receive the ADS-B data and the HoloLens device could correctly position the labels coherently with the aircraft position in the real world, updating them as soon as possible.

5.7 Weather interface implementation

As a parallel task to the tracking of aircraft, the possibility to implement in a head-up position some tower workstation's systems has been evaluated. In particular, exploiting the existence of a weather interface from the previous projects, the design has been exploited to implement a new weather system with real data. As with the tracking labels, the work has been divided between the graphic interface implementation and the data acquisition and processing.
Since the easiest way to obtain airport-related weather data is using the METAR, which is openly distributed on the net, this solution has been chosen. The METAR (MEteorological Terminal Air Report) is a routine message of airport weather observation which is hourly emitted at a pre-established time usually between minutes 50 and 59 of the hour. The METAR message contains some standard weather parameters which are useful for air navigation at the airport location.

5.7.1 METAR messages

The METAR message is a standard to report the weather information that is usually collected at any operative airport by the local weather station apparatus. The message is encoded in such a way that it can be directly read by any user knowing the encoding rules, independent of the message source location. A message is organized into eight standard groups:

1. **Identification group**. Can be preceded by the writing "METAR". Contains the ICAO identification code for the airport and the day and time of the report followed by the letter "Z" indicating that the one reported is the "Zulu" time. An additional element can indicate if the weather information is automatically computed and if the message is empty (NIL).

2. **Surface wind**. This block contains the airport surface wind indications, i.e. the wind direction and intensity. If gusts are detected they are reported. High variations in wind direction are also indicated in the message.

3. **Visibility**. This block reports the prevailing visibility at the airport location, in metres. If the visibility is highly variable with the considered direction, a second piece of information can be reported indicating the minimum visibility and its direction.

4. **Runway Visual Range**. This block is usually an alternative to the visibility one. It indicates the distance from which the runway marks can be seen by a pilot, preceded by the runway to which that information is applied.

5. **Present Weather**. Reported as one or more blocks containing the weather phenomena in place at the airport location, such as rain, fog or dust, completed by qualifiers for the intensity and trend of the event.

6. **Clouds or Vertical Visibility**. The Clouds block contains up to 4 sub-blocks reporting the principal cloud groups present at the airport location. It indicates the altitude, cloud amount and type of each cloud group. As an alternative, vertical visibility can

be provided which indicates the amount of vertical space which is free from clouds. If no clouds or weather phenomena are in place and the visibility is greater than 10000 metres, the CAVOK indication is given instead of the previous four blocks.

7. **Air and dew point temperature**. A block indicating the two temperatures in Celsius degrees. If an M precedes the number, it indicates that the temperature is lower than 0 ("Minus").

8. **Pressure QNH**. Indicated the equivalent sea-level pressure at the airport location. Can be provided in hPa (hectopascals, equivalent to millibars) or in inches of mercury.

After the standard blocks, a METAR message could be followed by additional information such as the future weather forecast, which anyway was not useful for the purposes of this thesis work. A hypothetical METAR message is shown for the sake of completeness:

```
METAR LIPK 211025Z 31015G27KT 280V350 4000 1400SW R24/P2000 +SHRA FEW005 FEW010CB ...
SCT018 BKN025 10/03 Q0995
```

The METAR refers to the Forlì airport (LIPK) and was emitted on the 21st of the current month, at 10:25 UTC (211025Z). The wind is heading at 310° with a velocity of 15 kts, and gusts up to 27 kts have been measured in the last 10 minutes (31015G27). The wind direction has variated between 280° and 350° (280V350). The prevailing visibility is 4000 metres (4000) with reduced visibility of 1400 metres in the South-West direction (1400SW). The RVR for runway 24 is greater than 2000 metres (R24/P2000). Present weather phenomena are shower rain (SHRA) with heavy intensity (+). Four clouds blocks are indicated: a few clouds (1/8 to 2/8 of the sky covered) at 50 ft (FEW005), a few clouds at 1000ft with a cumulonimbus structure (FEW010CB), scattered clouds (3 to 4 oktas) at 1800ft (SCT018), and broken clouds (5/8 to 7/8) at 2500ft (BKN025). Finally, the air and dew point temperatures are 10°C and 3°C respectively and the QNH pressure is 995hPa.

5.7.2 METAR decoding

Due to their highly standardized structure, METAR data can be easily processed by a machine. At the same time, the message blocks are clearly comprehensible for a human being. Thus, when decoding METAR messages, it has been sufficient to separate the composing message blocks such that they could be organized inside the graphic interface.
The decoding algorithm was first implemented on a desktop application prior to transferring it into the Unity app. Since the algorithm is fairly easy, only a summary of the operations done for METAR processing is presented:

- A class is created with fields corresponding to each piece of weather information contained in the METAR.

- The METAR is processed starting from the beginning. The message is first split into the composing blocks, which are then processed in the same order as they come.

- The location and time blocks are stored.

- The wind block is processed and heading, velocity and eventually gusts are stored in different variables. If the wind waving block is reported, the waving extremes are stored.
- in case the CAVOK indication is reported, the information is stored and the algorithm jumps to the temperature decoding.
- If no CAVOK is reported, the visibility blocks are checked. If lower visibility is indicated, the blocks are stored as maximum and minimum visibility.
- The algorithm checks for the number of RVR blocks reported (if any) and saves them in an array.
- The present weather phenomena are directly saved as blocks inside their special variable after they have been recognized.
- The cloud blocks are recognized and stored as an array without processing their content. If no cloud block is reported, the algorithm checks for vertical visibility, otherwise moves on to temperature decoding.
- The air and dew point temperatures are saved in two different variables.
- The QNH string block is converted into a number and stored to be used in altitude correction.

Once the decoding algorithm was ready, it was implemented and tested in Unity. Then a procedure to automatically obtain weather data was developed.

5.7.3 Weather data acquisition

Since the HoloLens can communicate via Wi-Fi, an easy way to receive METAR data is to download them from any suited website. The selected website is a database of METAR stations' data from the National Oceanographic and Atmospheric Administration of the United States. Each station's METAR is uploaded as a one-line webpage and can be read from a script with a few lines of code. The website URL for the LIPK airport is `https://tgftp.nws.noaa.gov/data/observations/metar//stations/LIPK.TXT`. The code for the METAR acquisition is reported in Appendix C.1.
Since the QNH pressure value is used in the labels management system, the METAR has to be processed prior to the start of the Update() cycle in the main script. Then, the METAR string is synchronously downloaded and decoded in the AndroidUDPListener script before initializing the ADS-B listener server. Once the METAR data have been stored, they can be visualized in the weather graphic interface.

5.7.4 Graphic interface implementation

The weather interface was first designed for the RETINA project. Since only a few significant METAR groups were used in the simulations, the interface has been updated to host a larger number of weather elements, as the present weather phenomena and multiple cloud groups,

as well as both the visibility and the runway visual range. Figure 5.20 and 5.21 report the new and old METAR interfaces respectively.

Figure 5.20: Old METAR interface.

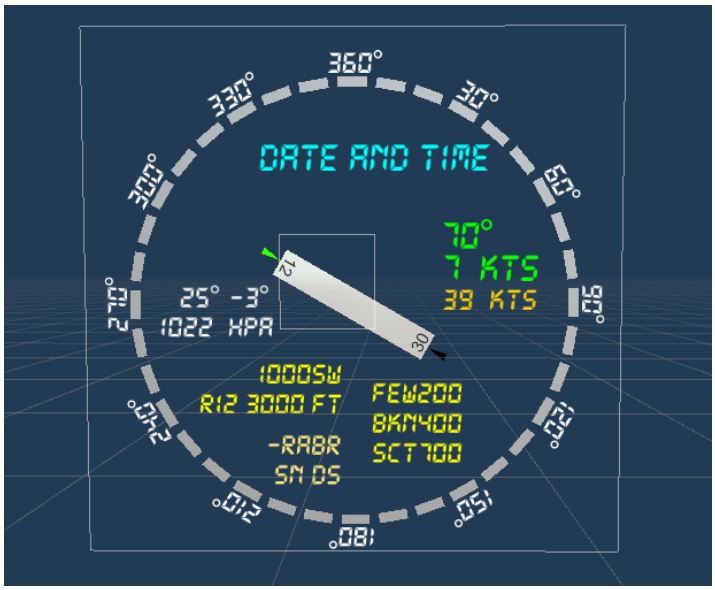


Figure 5.21: Updated METAR interface.

A number of text UIs have been implemented to show the different METAR groups. The interface has been programmed to tailor its appearance to weather conditions, in particular improving the graphical representation of the wind direction. A series of Figures is proposed which describes the different possible conditions which are visualized in the weather interface. The first pictures were taken before the update of the interface graphic and only implement the

new wind visualization system. The last two pictures also integrate the weather phenomena and a higher number of cloud groups. The last task in the weather interface design has

Figure 5.22: METAR visualization examples I. In the first case, the wind is waving as indicated by the smaller green arrows. In the second case, wind gusts are detected (maximum speed 38kts) and are indicated by the yellow arrow. In the third case, the wind has low intensity and variable direction.

been the positioning of the interface itself. As a result of the EXE-05.97.1-VAR-002, different opinions arose among ATCOs about the best position for the weather interface. Two options were considered: placing the weather interface in a fixed position above the runway, in a place that is often encountered by the ATCO's gaze as he looks over airport traffic, or keeping the interface permanently inside the ATCO field of view. The first option was chosen as it ensured better visibility to the ATCOs, otherwise always partially limited by the presence of the interface in their field of view. A hypothetical position for the interface has been chosen

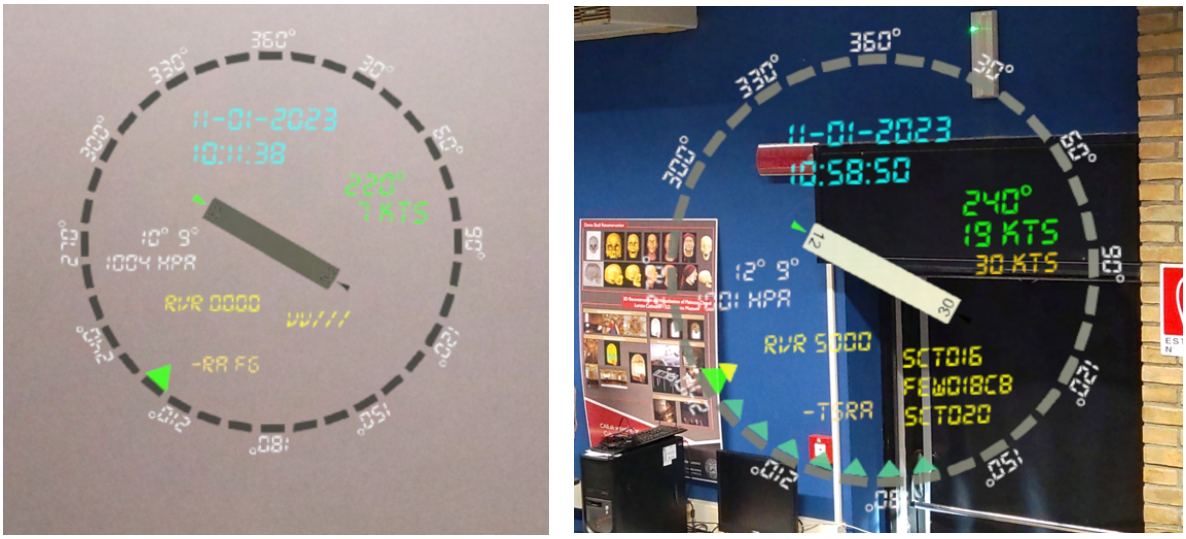


Figure 5.23: METAR visualization examples II. These interfaces implement present weather information. In the first case, the visibility is under 50 meters and vertical visibility is not available. Light rain and fog are present. In the second case, much information is provided. With wind gusts, waving wind, weather phenomena, RVR, and multiple cloud groups are reported.

5.8 Runway overlay

As the last graphical element, the runway overlay was added for operations in low-visibility conditions. The overlay consisted of two simple lines delimiting the runway sides, which were drawn in Unity using the geographic coordinates of the runway extremes with the same conversion method exploited for the other elements. As the runway is not completely flat, six reference points were used to better follow its contour. In addition, the altitude of the reference points was obtained from the official airport cartographic maps. The overlay is as in Figure 5.24.

Figure 5.24: Aircraft at take-off with runway overlay visualized.

As an additional feature of the runway overlay, it can be used to better evaluate the goodness of the vertical calibration of the device.

5.9 App deployment procedure

After having described all the elements composing the application, this section explains how the application is prepared to be distributed on the HoloLens device and how the entire implemented system is used at runtime.

5.9.1 App deployment on HoloLens 2

Two passages are required for the distribution of the test_GPS application on HoloLens. First, the application must be built in Unity with the correct setting, in particular, the application name (in this case "text_GPS") has to be assigned and the Unity builder has to be configured for a Universal Windows Platform application with ARM64 system architecture, with HoloLens as the target device and Visual Studio as the intended distribution platform. Once the application has been built, a folder is created which contains the solution ready for deployment through Visual Studio. Opening the solution with the latter, it is possible to compile the solution and obtain the executable which is then distributed on the HoloLens

device through a USB cable. At this point, the application is installed on HoloLens and can be deployed by opening it from the HoloLens Start Menu interface.

5.9.2 Application execution

When running the system, two operations are needed. The application on the HoloLens device has to be launched and the ADS-B data transmission has to be started by launching the Serial.exe application from a PC connected to the receiver unit via USB and to the same Wi-Fi connection as the HoloLens. The order in which the two applications are launched is not important, provided that both are correctly launched and configured. The main passages of the system activation are reported here. For simplicity, it is considered the case in which the Serial.exe is started first.

1. The receiving station is positioned outside as far as possible from any near obstruction, like walls and trees. The receiver unit is connected to the pc.

2. The Serial.exe application is launched on the pc. The serial port is selected and the IP address of the HoloLens in the current net is configured. The application is activated. As soon as the serial port communication is activated, the receiver starts demodulating the messages and the app starts reading and forwarding them to the HoloLens.

3. The HoloLens device is positioned at the calibration point with the right orientation and the tilting display locked in its end position.

4. The test_GPS application is launched using vocal commands and the device is not moved until the calibration is completed.

5. The device is worn and the calibration is visually verified. In the meantime, if messages are being received, the Tracking Labels are already visualized.

6. The application is used.

Chapter 6

Validation

The final phase of the design cycle is the validation, which aims to test end evaluate the application to objectively assess its functionality and the fulfilment of the initial design requirements. At this stage of the design and given the subset of tasks that the application had to accomplish, the proposed human-in-the-loop evaluation of the application design (Chapter 3) has been done directly by the developer, who verified all the technical requirements and the repeatability of the proposed configuration procedures.
Prior to focusing on a schematic one-by-one evaluation of the requirements, the primaeval application had to be cleaned from a variety of holograms that progressively populated it during the development for debugging and testing purposes. What's more, during the prototyping phase, the application had been tested in an environment which was far from anything resembling a control tower. In fact, the Virtual Reality and Simulation Laboratory does not provide an out-of-the-window view at all, and during the platform implementation, the airport traffic had been observed directly exiting the room from a door overlooking the runway of the Forlì airport. In addition, during the development of the application, the know-how on some operative procedures, such as the calibration, has naturally come together in time, resulting in sub-optimal and fragmented operations. For these reasons a series of tasks to be performed prior to the technical validation was identified, aiming also to make the deployment and configuration of the application repeatable and reproducible[1]. In fact, validation would be partially meaningless if these two characteristics are not featured. The pre-validation work has been divided as follows:

- **Cleaning of debugging elements**. During development, a series of debugging elements have been used, like an overlay canvas constantly reporting the decoded aircraft position and the METAR string, some holograms used as a reference for the calibration and others used just for testing how HoloLens works. Only a few of these elements were needed for the validation and have been kept.

- **Development of a new repeatable calibration procedure**. During the implementation, the HoloLens calibration set-up was progressively obtained by the addition of

[1] Repeatability refers to the possibility to repeat an experiment with the same team and the same know-how. Reproducibility means that the experiment can be successfully conducted even after changing the setup and/or the team.

multiple holograms and refining the HoloLens' initial positioning. The altitude calibration was not really considered, as it was just verified by visual estimation without a clear reference. In addition, some peculiar characteristics of the laboratory were exploited, which could hardly be reproduced in a different environment. Thus, a more consistent calibration procedure had to be implemented. This procedure should consider both horizontal and vertical calibration references. As part of the new procedure, the HoloLens position and configuration have been better fixed. Then, calibration holograms have been selected for both horizontal and vertical calibration, defining a methodology which could be valid in any airport environment. Finally, a way of hiding the reference elements after the calibration has been implemented.

- **Transfer to a new environment**. This passage was needed to have a suited reference environment for the validation and to test the reproducibility of the work done for implementation. In order to change the testing room, all the elements in the application directly related to the environment characteristics (HoloLens' initial position and orientation, runway characteristics,...) had to be identified and grouped. Based on these parameters and on the fragmented work done in the first environment, a more consistent and ordered procedure has been studied which could be followed to perfectly configure a completely new environment. The app has then been modified to be well-structured for efficient reconfiguration. Only at this point, a suitable environment has been chosen. After that, the needed information about the new room has been collected and both the room and the application have been prepared. Finally, the app has been deployed and calibrated. As it was the first time the procedure was implemented, some minor corrections have been considered.

The scene cleaning was easily done once the application was completed. Therefore, the next sessions will only focus on the calibration and change of environment procedures. After that, the last section will focus on app validation.

6.1 Development of a calibration procedure

In this section, a deeper insight is given into the passages followed to define a new calibration procedure, which comprehends both the calibration effectively done when launching the app and the design of a suitable system of reference of holograms used to verify the goodness of the calibration.

6.1.1 HoloLens positioning

At first, HoloLens was calibrated simply by fixing its frontal position on a table by means of a reference marker positioned on the table. The reference only fixed the lateral positioning of the HoloLens, which still had some degree of freedom in the horizontal orientation. In addition, the vertical calibration was corrected by adjusting the initial rotation of the HoloLens frontal enclosure, giving a different orientation to the frontal camera when calibrating, with a trial-and-error approach. However, this procedure is obviously not suitable if repeatability

is required.
A better practice has been derived by exploiting the HoloLens symmetry. The HoloLens' frontal position is fixed by means of a marker. Other markers are used to trace a line corresponding to the HoloLens symmetry plane from an upper view. Finally, two additional markers give an additional indication of the rear lateral position. In this way, the HoloLens can be calibrated always with the same position and orientation. As for the vertical orientation of the camera, the visor tilting mechanism is brought to its end stop. If the table is perfectly levelled, the camera is oriented horizontally, and the virtual Y axis will perfectly coincide with the local vertical.

6.1.2 Design of the new calibration system

The first part of the calibration procedure is given by the positioning of the device. In addition to that, a frame of holograms is needed which the user can exploit to verify the goodness of the calibration itself, by checking the holograms' alignment with chosen reference points. Since the calibration was previously performed thanks to the experience matured by the developer in knowing when different holograms should appear in the environment, a more reproducible method was needed. A new series of holograms has been identified, which ensures the correct matching of the reference frames. These holograms are organized into two groups, the first determining the horizontal calibration (the rotation of the HoloLens reference frame about the vertical axis, and the second the vertical calibration (rotation about the virtual X-axis). The adopted strategy uses three holograms for horizontal calibration. In fact, from geometrical considerations on the user's field of view, it appears that two points are not sufficient to determine the user's position with respect to them, as the holograms are projections and the user can only be sure of their angular position and not of their virtual distance. Since infinite possibilities exist for which the user sees the holograms being at the same reciprocal distance (same angular size) and aligned with the right reference points, a third point is needed which does not belong to the circumference passing through the HoloLens calibration position and the first two reference points. Figure 6.1 schematize this geometrical condition. To best evaluate the horizontal position of the virtual objects with respect to their reference points, some vertically shaped holograms have been used.

With regard to the vertical calibration, if the HoloLens position and orientation in the horizontal plane are already fixed, only one degree of freedom is kept free, corresponding to the rotation about the virtual X-axis. Thus, the runway overlay, which appears as two narrow lines usually positioned horizontally or diagonally with respect to the tower controller, can be used to determine if the virtual reference plane is inclined with respect to the real world.

6.1.3 Calibration procedure definition

When preparing the app for a new environment, the following calibration procedure can be used. Three distant reference points which are visible and observable from the operative environment are chosen, better if well distanced from each other. The coordinates of the three points are retrieved. Then, three reference holograms, already implemented in Unity, are positioned using the real coordinates of the three reference points. For the vertical calibration,

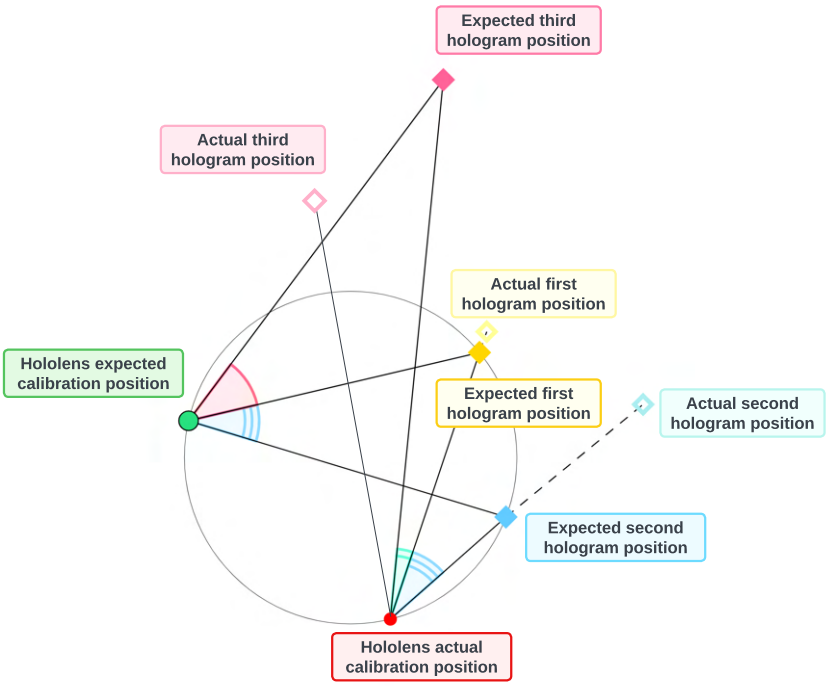


Figure 6.1: Three points calibration geometric considerations. If only two points are used for the calibration in the horizontal plane, it could be impossible to determine if the calibration is successful. A third point ensures that an error in the HoloLens' initial position and orientation can be detected.

it is sufficient to implement the runway overlay with the correct coordinates. Finally, once the HoloLens calibration position has been defined and georeferenced, the markers are positioned in the correct place on the table to fix the HoloLens calibration placement.
The developed procedure can be consistently repeated in every chosen room, deciding the HoloLens starting point and orientation and identifying suitable reference points in the outdoor background visible from the room (usually the airport facility).

6.1.4 Hiding of the calibration frame

Once the calibration is completed, the horizontal calibration holograms represent a visual disturbance for the operator. Thus, a virtual button has been implemented which can be pressed to hide the horizontal calibration holographic frame.

6.2 Change of environment

The calibration procedure is part of a more general procedure defined for changing the testing environment. This section describes the standardized procedure which has been defined to deploy the application in a new environment. In doing this, some changes have been made to the application, recovering all the aspects of the application which were directly related to the reference environment. A procedure has been defined that presents in logical order all the

configuration operations needed for deployment.

6.2.1 Organization of relevant variables and app adaptation

Since in the original application, different features and elements were added at different moments, many variables were repeated in different scripts containing for instance the same values of latitude and longitude. Therefore, the application has been checked to find all these variables which have been merged into a compact amount of variables which have to be set when preparing the app for a new environment. These variables are the HoloLens calibration coordinates (latitude, longitude, altitude and orientation) and the airport runway elevation (the one reported in the maps as a reference). In addition, a list of latitude and longitude positions for the runway overlay is needed (also different altitudes if the runway is not levelled to have a better matching). Finally, also the three reference holograms' coordinates are needed. All these variables have been grouped in the AndroidUDPListener script, which is the only one to be configured for the change of environment.

6.2.2 Definition of the environment configuration procedure

The repeatable configuration procedure is straightforward at this point:

- Once a new environment is selected, a reference point is selected where to place the HoloLens. A table or another flat and levelled surface is used for the calibration.
- The HoloLens direction is fixed toward a window from which suitable reference points - which are now selected - are visible. The HoloLens calibration position is visualized using markers.
- The coordinates are retrieved. For the HoloLens, the height of the room from the outdoor ground is measured together with the position of the HoloLens with respect to any point which is visible on Google Earth, as an external wall of the room. Then on Google Earth, the horizontal distances are reproduced and the position of the HoloLens is found. As for the altitude, that of the ground is measured on Google Earth or using airport maps if the room is placed within the airport facilities. Then the HoloLens altitude is obtained by summing up the height of the room and the height of the table. Regarding the calibration holograms and the runway overlays, all the coordinates are retrieved using Google Earth apart from the runway reference points' altitude which comes from the more precise airport maps. Since the horizontal frame's holograms are only used for horizontal calibration, not much precision is needed in the vertical positioning. All these values are assigned to the relative variables in the application.
- The weather interface and the calibration deactivation virtual button are placed in the desired position in the scene, calculating the offset with respect to the HoloLens calibration points (which in Unity has coordinates (0,0,0)).
- The application is built and distributed.

- The antenna is positioned outside and connected to the pc. The pc and the HoloLens are connected to the same Wi-Fi net and made visible to other devices.
- The Serial.exe app is launched, and the data transmission is started.
- The HoloLens device is positioned in the calibration point for the first calibration, and the app is launched using vocal commands.
- The device is worn by the user. The calibration is checked and if it is successful, the calibration frame is deactivated through the virtual button.
- The application is used by the operator.

6.2.3 Implementation of the configuration procedure on a new environment

Once the procedure was ready, a room has been selected which was suitable for validation purposes. The chosen room was located on the second floor of the University facility and had a large window directly facing the airport runway with good visibility (Figures 6.2). Moreover, the room allowed direct access to a terrace, which was helpful again for the validation and in which the antenna could be placed optimally (Figure 6.3).

Then, the procedure from the previous section has been implemented, step by step, and the

Figure 6.2: Room visibility on the runway, with active runway overlays and two far aircraft detected.

configured app has been deployed. Three buildings have been chosen as calibration reference points. The calibration has been successful and only a few adjustments had to be made, precisely the positioning of the weather interface and of the virtual button. Anyway, these two aspects were not related to the implemented procedure, which was well-designed. The overall time needed for the change of environment was measured in about three hours.

As a final remark on the goodness of the environment change procedure, a couple of images are presented (Figures 6.4 and 6.5). Once the app was configured and deployed through a standard procedure, the validation could be performed, being sure that the possible satisfaction of the requirements would be reliable and consistent even if performed in a different environment.

Figure 6.3: Terrace visibility over the runway.

6.3 Technical Validation

The technical validation was run at the conclusion of the design cycle, to evaluate the satisfaction of each requirement and highlight any improvement or correction to be implemented. Recalling the concept of a user-centred design, special importance is given to the last three requirements. In the following lines, requirements are evaluated one by one, explaining how they have been met and whether something has to be done to better satisfy them.

- **Acquire the aircraft position with high accuracy**. This requirement has been accomplished thanks to the use of the ADS-B data. Since this technology is based on high-accuracy onboard GNSS systems, the provided position information is much higher than with traditional radars.

- **Acquire the aircraft position in real-time**. ADS-B data intrinsically include the concept of real-time acquisition, as they are continuously and automatically transmitted. For the data transmission to be real-time, the data had to be transmitted as soon as they were received, which was made by a callback function inside the Serial.exe app. Thanks to the fast UDP data protocol, the data transmission delay between the Serial app and the HoloLens application was measured to always be far less than a second. Thus, the requirement was considered to be satisfied.

- **Match the AR environment with the real world for accurate aircraft positioning**. This requirement was satisfied by exploiting the reference frame conversion algorithm and with proper app implementation, using the WLT feature and the calibration procedure. Different virtual objects, comprising the aircraft, have been visualized in the right position in the real world.

- **Track the aircraft position accurately in the AR environment**. This requirement

Figure 6.4: Room visibility on the runway, with a Ryanair aircraft taking off.

has been fulfilled by means of a combination of the matched virtual and real world and the aircraft position coming from the ADS-B receiver.

- **Acquire and track information for multiple aircraft at the same time**. The requirement was met by implementing a series of features. First, the proper choice of the tracking label design and of the visualized surveillance information was implemented. Second, the aircraft positioning manager could position the label based on the detected aircraft's position information. This way, multiple aircraft were tracked at the same time.

- **Track the aircraft position in real-time**. The ADS-B data were transmitted in real-time and decoded as soon as received by the HoloLens. At the same time, the aircraft positioning manager was continuously updating the holograms with the most recent decoded information. The main delay was then due to the ADS-B data transmission chain. In this case, the minimum possible latency had to be searched for. With a proper design of the transmission chain, the final latency was contained, with the tracking labels being always positioned in the proximity of the tracked aircraft. The only distortion in the label position was due to the time interval between position message transmission, which is 2 seconds for ground messages and 0.5 seconds for airborne messages. The overall requirement is then satisfied.

- **Render the aircraft tracking holograms, minimizing computational latency**. Since hologram rendering is a time-consuming task, good practices should be adopted to reduce the computational effort. There is no general rule in doing this, or a measure to evaluate how good the computation is. However, some considerations can be done on how the app is implemented. As good practice, it was chosen to always keep the dimension and the distance of the tracking labels constant, to avoid a continuous scaling of the same. After testing the app, it was found that the target HoloLens frame rate of

Figure 6.5: Room visibility on the runway, with an Aeroitalia aircraft taking off.

60fps was maintained, even when several aircraft were tracked simultaneously. On the contrary, the chosen positioning method, using the Solver component, may be improved in the future. For this reason, the requirement has been considered satisfied, but with a provision.

- **Evaluate possible error sources and counteract them**. It is almost impossible to counteract every possible error source, especially if unknown. Nevertheless, a proper testing campaign can highlight the main issues of an application. In this case, several corrections were made, starting from many initial bugs which were fixed in the ADS-B decoding algorithm after proper debugging and passing through the altitude errors correction. As after extended testing, no big issues were found on the final deployed application, the requirement can somehow be considered fulfilled.

- **Ensure precision and reliability of tracking and positioning system**. The precision and reliability of the HoloLens labels positioning system come from the satisfaction of the previous requirements. As for the tracking system on its own (meaning the ADS-B data), adequate information is contained in the message telling how accurate the transmitted position information is. Thus, in the rare case in which a position message contains imprecise information, it is possible to discharge it. In the case of an aircraft transmitting with a deteriorated GNSS system, there is no way to obtain and accurately track it. In this case, other surveillance systems should be used together with ADS-B, but this goes beyond the purposes of this thesis work. Thus, for what concerns this project, the requirement has been considered satisfied.

- **Develop an appropriate labelling system for aircraft tracking**. For this requirement, together with the next two, the user experience is fundamental for an adequate

evaluation. The term "appropriate" is rather qualitative, hence only the final user can provide an opinion on how he considers the design. In this case, the tracking labels system was shown to a traffic controller, which confirmed the goodness of the implemented transparency in the labels and positively evaluated the positioning of the tracking elements. In particular, he confirmed that the choice of having the label always over the aircraft, even if not in direct contact with it was adequate for tower control purposes. As suggested by the traffic controller, the labelling system has been completed by the differentiation of the labels for departing and landing aircraft, by using cyan and yellow labels respectively as shown in Figures 6.2 and 6.4.

- **Visualize ATC information usually available on terminals**. This requirement was satisfied under two different aspects. As for the weather interface, it had already been validated during previous projects' validation campaigns. From the feedback over those validations, the design and the positioning have been taken. As an additional task, the implementation of the interface with real and current METAR information has been accomplished.

- **Visualize helping tools for low-visibility conditions**. This requirement was met. In fact, the tracking labels are on their own proper instruments which help ATCOs, especially in low-visibility conditions. In addition, the implementation of the runway overlay, which could be tailored to visibility range, is an additional element which participates in enabling high-performance low-visibility airport operations.

With an optimal completion of the initial requirements, the design cycle could be considered complete.

Chapter 7

Conclusion

Starting from the results of two previous projects, RETINA and DTT EXE-05.97.1-VAR-002, this thesis work aimed to transfer the previously developed technology to a real-world environment, developing a platform to validate the use of Augmented Reality (AR) technologies for surveillance purposes in the airport control tower.
The design was carried out following the user-centred design methodology, and joining a series of outcomes from the previous projects - such as the use of tracking labels and of a weather interface -, with the specific requirements of a real-world implementation of augmented reality and airport surveillance, including the capacity of tracking the aircraft in real-time and designing suitable labels to point out their exact position while providing additional surveillance information.
As a result of all the collected requirements, a conceptual design has been proposed, in which an ADS-B data stream is used to feed the tracking position information to an HoloLens Head Mounted Display. On HoloLens, an application reads the ADS-B data and transforms them in coordinates for the holograms, visualizing the Aircraft Tracking Labels in correspondence to the real aircraft positions in the physical world.
In the first part of the implementation, the ADS-B technology - used for high-precision tracking of aircraft, has been exploited by means of a ground station used to receive the data, and all the surveillance information useful for the subsequent work has been decoded, in particular, the aircraft position information and the identification parameters.
Once it was ensured that the ADS-B technology was suitable for the purposes of the work, the second part began, in which the AR application has been developed exploiting the HoloLens. In doing so, the major aspect of AR techniques implementation has been considered: the registration of the virtual world with the physical is obtained by means of the HoloLens tracking system together with the use of a real-time application in Unity. The application was prepared to continuously be aware of the aircraft coordinates and reposition the holograms in real-time with respect to the user moving inside the operative environment. An operative calibration procedure has been considered to provide the initial matching of the two worlds - virtual and real -, from that moment automatically maintained by means of the HoloLens world-locking tools. In designing and implementing the Tracking Labels, some additional problems were solved. In particular, the Label's opacity required some corrections to reduce the Label's

invasiveness and the vertical positioning of the Labels had to counteract some errors in the aircraft's vertical position determination. For the first technical test on the tracking labels implementation, the application was seen by an ATC operator which gave positive feedback on both these corrections and in general on the real-time tracking system functioning.
As a complementary part, a preliminary study for the implementation of surveillance information in head-up positions and the environmental condition's adaptiveness of the application was carried out. In particular, a weather interface with real airport weather data was implemented together with a system of airport runway overlays to be used in low-visibility conditions.
Once the implementation was concluded, the work done was transferred to a repeatable and reproducible procedure to be followed when implementing the application, which is environment-dependent, in a new location. Finally, this procedure was tested, and the resulting application was validated with respect to the initial requirements. The application demonstrated to successfully implement the features required initially: the tracking labels were tracking the aircraft accurately and in real-time, the weather interface automatically implemented real and current METAR data and the runway overlays were traced.
For the sake of conciseness, other passages which allowed the completion of the thesis work are not presented in this document, in particular the extensive testing and some algorithm structure for both the ADS-B decoding and HoloLens registration strategies, as well as the specific design of the calibration holograms in Unity.

7.1 Future development

Room for improvement is still available for the application. The platform could be completed by improving the surveillance information visualized in the Tracking Labels, in particular bringing the application to the airport control tower and testing it in shadow mode with the integration of additional surveillance technology which was not available in the structure at which this thesis work was carried out. In addition, the whole tracking system dynamics could be smoothed by interpolating the aircraft position in the time that passes within ADS-B messages transmission by means of the velocity, heading and vertical rate information. In the same way, also the delay in the data transmission and processing could be counteracted by means of the same information, forecasting the aircraft position.
Furthermore, it can not be thought of the future development of this work without referring to the solutions already developed in the previous projects run in the simulated environment. In fact, many concepts developed in the context of the SESAR JU and implemented in DTT EXE-05.97.1-VAR-002 should be transferred and tested in the real world, with particular reference to the safety nets implementation and the full implementation of the application adaptivity for different work positions (Ground and Runway).
As a final remark, it is important to notice that in the context of the SESAR research campaign, the spread and sharing of the developed concept has a central role in the design of the future of European airspace. As an example, many of the concepts developed from RETINA

onwards, such as the tracking labels and the runway overlays, could be applied, with due technological considerations, also to the Remote Tower concept, which presents the same operative limitations for heavy traffic and low-visibility of an on-site control tower. Thus, future development, aiming for an always higher maturity level, should be considered as part of a larger research campaign in which every solution and advancement contributes to a bigger and shared research for the future of European air transport.

Bibliography

[1] Pinska, E., "An investigation of the head-up time at tower and ground control positions", Eurocontrol (30 December 2006).

[2] Report on "Effects of Novel Coronavirus (COVID-19) on Civil Aviation: Economic Impact Analysis", ICAO. Montréal, Canada (27th of January, 2023).

[3] *EUROCONTROL Forecast Update 2022-2028. European Flight Movements and Service Units*, EUROCONTROL (October 2022).

[4] Shparberg, S., Lange, B., *Global Market Forecast 2022*, Airbus, Toulouse (8th of July, 2022).

[5] *Air Traffic Management: Freeing Europe's Airspace*, Commission of the European Communities, Brussels (6th of March, 1996).

[6] SESAR Joint Undertaking website: `https://www.sesarju.eu/`, consulted on the 14th of January, 2023.

[7] `https://www.sesarju.eu/sesar-solutions/enhanced-visual-operations`, consulted on the 15th of January, 2023.

[8] Reisman, R., Brown, D., "Design of Augmented Reality Tools for Air Traffic Control Towers", 6th AIAA Aviation Technology, Integration and Operations Conference - ATIO (2006). `doi.org/10.2514/6.2006-7713`

[9] Robertson, G. G., Card, S. K., and Mackinlay, J. D., "Three views of virtual reality: nonimmersive virtual reality," in *Computer*, vol. 26, no. 2, pp. 81-86 (Feb. 1993). `doi.org/10.1109/2.192002`

[10] Immersive Virtual Reality, in *Furht, B. (eds) Encyclopedia of Multimedia*. Springer, Boston, MA (2008). `doi.org/10.1007/978-0-387-78414-4_85`

[11] Freina, L., Ott, M., "A Literature Review on Immersive Virtual Reality in Education: State Of The Art and Perspectives" (2015).

[12] Cipresso, P., Giglioli Chicchi, I. A., Alcañiz Raya, M., Riva, G., "The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature", in *Frontiers in Psychology*, vol. 9 (2018). `doi.org/10.3389/fpsyg.2018.02086`

[13] Milgram, P., Takemura, H., Utsumi, A., Kishino, F., "Augmented reality: A class of displays on the reality-virtuality continuum" in *Telemanipulator and Telepresence Technologies*, vol. 2351 (Jan. 1994). doi.org/10.1117/12.197321

[14] Skarbez, R., Smith, M., Whitton, M. C., "Revisiting Milgram and Kishino's Reality-Virtuality Continuum" in *Frontiers in Virtual Reality*, vol. 2 (2021). doi.org/10.3389/frvir.2021.647997

[15] Bimber, O., Raskar, R., *Spatial Augmented Reality: Merging Real and Virtual Worlds*, A K Peters, Ltd (2005).

[16] https://www.skybrary.aero/, consulted on the 23th of January, 2023.

[17] Doc4444, Procedures for Air Navigation Services Air Traffic Management (PANS-ATM), 16th Ed. International Civil Aviation Organization (ICAO), pp. 19-47 (2016).

[18] Regulation (EC) No 549/2004 of the European Parliament and of the Council of 10 March 2004 laying down the framework for the creation of the single European sky, *Official Journal of the European Union*, L 096/1 (31st of March 2004), http://data.europa.eu/eli/reg/2004/549/2009-12-04

[19] *Annex 11 to the Convention of the International Civil Aviation - ATS Service*, 15th ed. International Civil Aviation Organization (ICAO), pp. 22-32, 34-38, 53-61 (2018).

[20] *Annex 2 to the Convention of the International Civil Aviation - Rules of the Air*, 10th ed. International Civil Aviation Organization (ICAO), pp. 16-21, 34, 36-37 (2005).

[21] https://www.skybrary.aero/articles/primary-surveillance-radar-psr, consulted on the 30th of January, 2023.

[22] https://www.skybrary.aero/articles/secondary-surveillance-radar-ssr, consulted on the 30th of January, 2023.

[23] https://skybrary.aero/articles/multilateration, consulted on the 30th of January, 2023.

[24] " EUROCONTROL-SPEC-171: EUROCONTROL Specification for Advanced-Surface Movement Guidance and Control System (A-SMGCS) Services", 2nd ed. EUROCONTROL (2020).

[25] Sun, Junzi. *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*, TU Delft OPEN Publishing, 2nd Edition (2021). doi.org/10.34641/mg.11.

[26] RTCA DO-260. Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance – Broadcast (ADS-B) and Traffic Information Services – Broadcast (TIS-B) Change 1, RTCA Inc., 1828 L Street, NW, Suite 805 Washington, DC 20036-5133, USA (17 December 2020).

[27] "Single European Sky", https://www.europarl.europa.eu/factsheets/en/sheet/133/air-transport-single-european-sky, consulted on the 3rd of February, 2023.

[28] High Level Group on aviation regulation, https://transport.ec.europa.eu/transport-modes/air/high-level-groups/high-level-group-aviation-regulation_en, consulted on the 9th of February, 2022.

[29] *SESAR Solutions Catalogue 2021*, 4th ed. Publications Office of the European Union (2021). doi.org/10.2829/998701. https://www.sesarju.eu/catalogue

[30] *Annex 17 to the Convention of the International Civil Aviation - Aviation Security*, 12th ed. International Civil Aviation Organization (ICAO), p. 11 (2022).

[31] *Annex 19 to the Convention of the International Civil Aviation - Safety Management*, 2nd ed. International Civil Aviation Organization (ICAO), p. 16 (2016).

[32] Santarelli, R., Bagassi, S., Corsi, M., Teutsch, J., Garcia Lasheras, R., Amaro Carmona, M. A., Groskreutz, A. R., "Towards a digital control tower: the use of augmented reality tools to innovate interaction modes", 12th SESAR Innovation Days, Budapest, Hungary, 5-8 December 2022. https://www.sesarju.eu/sesarinnovationdays

[33] https://www.remote-tower.eu/wp/project-pj05-w2/solution-97-1/, consulted on the 20th of November, 2022.

[34] Retina project website, http://www.retina-atm.eu/, consulted on the 9th of February, 2023.

[35] Masotti, N., Bagassi, S., De Crescenzio, F., "Augmented Reality for the Control Tower: The RETINA Concept", *in: De Paolis, L., Mongelli, A. (eds) Augmented Reality, Virtual Reality, and Computer Graphics. AVR 2016. Lecture Notes in Computer Science*, vol 9768. Springer, Cham (2016). doi.org/10.1007/978-3-319-40621-3_32

[36] https://www.remote-tower.eu/wp/project-pj05-w2/solution-97-2/, consulted on the 9th of February 2022.

[37] International Organization for Standardization, *Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems, (ISO Standard No. 9241-210:2019)* (2018).

[38] Bagassi, S., De Crescenzio, F., Piastra, S., Persiani, C. A., Ellejmi, M., Groskreutz, A. R., Higuera, J., "Human-in-the-loop evaluation of an augmented reality based interface for the airport control tower", in *Computers in Industry*, Volume 123, article number 103291 (2020). https://doi.org/10.1016/j.compind.2020.103291.

[39] Bagassi, S., De Crescenzio, F., Piastra, S., "Augmented reality technology selection based on integrated QFD-AHP model", in *International Journal on Interactive Design and Manufacturing* 14, 285–294 (2020). https://doi.org/10.1007/s12008-019-00583-6

[40] https://github.com/microsoft/MixedRealityToolkit-Unity, consulted on the 14th of February, 2023.

[41] https://github.com/junzis/pyModeS/tree/master/pyModeS/decoder, consulted on the 25th of September 2022.

[42] https://gist.github.com/yossioo/2862f58f1847baaeb9282c863fe86538, consulted on the 16th of February, 2023.

[43] https://github.com/cgommel/adsbsharp/commits?author=cgommel, consulted on the 16th of February, 2022.

[44] https://www.faatest.com/books/FLT/Chapter17/GroundEffect.htm, consulted on the 16th of February, 2023.

[45] https://learn.microsoft.com/en-us/windows/mixed-reality/design/comfort, consulted on the 16th of February, 2023.

Appendices

Appendix A

Matlab Position Decoding

A.1 Gray's algorithm for altitude decoding

```
function alt = altitude_decoder_qbit0(enc_alt)

%[0 1 2 3 4 5] M [7] Q [9 10 11 12]
%[    :6     ] M [7] Q [    9:    ]

%[0 1 2 3 4 5 6] Q [8 9 10 11]
%in enc alt:
%[1 2 3 4 5 6 7]   [8 9 10 11]

%case q_bit = 0
%use a string not containing q_bit --> enc_alt
%then we assign each bit to a value:
c1 = bitget(enc_alt,11);
a1 = bitget(enc_alt,10);
c2 = bitget(enc_alt,9);
a2 = bitget(enc_alt,8);
c4 = bitget(enc_alt,7);
a4 = bitget(enc_alt,6);
b1 = bitget(enc_alt,5);
b2 = bitget(enc_alt,4);
d2 = bitget(enc_alt,3);
b4 = bitget(enc_alt,2);
d4 = bitget(enc_alt,1);

% graystr = D2 + D4 + A1 + A2 + A4 + B1 + B2 + B4 + C1 + C2 + C4
graystr = ...
    uint32(0)+bitshift(d2,10)+bitshift(d4,9)+bitshift(a1,8)+bitshift(a2,7);
graystr = graystr+bitshift(a4,6)+bitshift(b1,5)+bitshift(b2,4)+bitshift(b4,3);
graystr = graystr+bitshift(c1,2)+bitshift(c2,1)+c4;
alt = gray2alt(graystr);

function alt = gray2alt(graystr)
    gc500 = bitshift(graystr,-3);
```

```
33      n500 = gray2int(gc500);
34
35      gc100 = bitand(graystr,7);
36      n100 = gray2int(gc100);
37      if any([0, 5, 6]== n100)
38          alt = NaN;
39          return
40      end
41
42      if n100 == 7
43          n100 = 5;
44      end
45
46      if mod(n500,2)
47          n100 = 6 - n100;
48      end
49
50      alt = (n500 * 500 + n100 * 100) - 1300;
51
52  end
53  end
54
55  % gray2int
56  function num = gray2int(num)
57      num = bitxor(num,bitshift(num,-8));
58      num = bitxor(num,bitshift(num,-4));
59      num = bitxor(num,bitshift(num,-2));
60      num = bitxor(num,bitshift(num,-1));
61  end
```

A.2 Local Airborne position decoding

```
1  %% Local Airborne Position
2  function [lat,lon,invalid_mex] = ...
       Locally_unambiguous_airborne_position_decoding(YZ,XZ,CPR,lat_ref,lon_ref)
3  invalid_mex = 0;
4  % can be applied to both odd and even messages
5
6  % constants
7  NZ = 15; % number of latitude zone between equator and a pole --> 15
8  Nb = 17; % number of bits in YZ, XZ
9  i = CPR; % 0 even, 1 odd
10
11 % values between 0 and 1 representing position inside one specific CPR ...
       Earth zone
12 lat_CPR = YZ / (2^Nb);
13 lon_CPR = XZ / (2^Nb);
14
15 % latitude zone span in degrees - depends on type of message -> 90 for surface
```

```
16 dLat = 360/(4*NZ-i); % extension of one lat zone in degrees
17
18 % latitude zone index (reverse of latitude zone calculation for global)
19 j = floor(lat_ref / dLat) + floor(0.5 + (mod(lat_ref, dLat) / dLat) - lat_CPR);
20
21 % calculate new latitude positioning
22 lat = dLat * (j + lat_CPR);
23
24 % then we need to reapply the time zones division for longitude basing on ...
       new latitude
25 NL = NL_function(lat)-i; % gives the number of longitude zones for that ...
       latitude
26 % we rearrange in case NL = 0
27 if (NL > 0)
28     dLon = 360 / NL;
29 elseif NL == 0
30     dLon = 360;
31 else
32     % if NL < 0 the decoded position is not valid
33     invalid_mex = 1;
34     return
35 end
36
37 % longitude zone index
38 m = floor(lon_ref / dLon) + floor(0.5 + (mod(lon_ref, dLon) / dLon) - lon_CPR);
39
40 % longitude
41 lon = dLon*(m + (XZ/(2^Nb)));
42
43 end
```

A.3 Global Surface position decoding

```
1 %% Global Ground Position
2 function [lat,lon, invalid_mex] = ...
       Globally_unambiguous_ground_position_decoding(YZe,XZe,YZo,XZo,CPR)
3 % takes as input the lat and lon bit arrays for even odd and the CPR value ...
       of the most recent message among the 2
4
5 % we have to consider ref, now we consider Forli Airport (44,19972N, 12,06361E)
6 lat_ref = 44.1997;
7 lon_ref = 12.0636;
8
9 % constants
10 NZ=15;
11 Nb=17;
12
13 % calculate position inside a zone
14 lat_CPRe = YZe / (2^Nb);
```

```
lon_CPRe = XZe / (2^Nb);
lat_CPRo = YZo / (2^Nb);
lon_CPRo = XZo / (2^Nb);

% latitude zone span in degrees. now 90 degrees as reference, so 1.5 ...
    degrees span of lat zone
dLate = 90/(4*NZ);
dLato = 90/(4*NZ-1);

% latitude zone index
j = floor(59 * lat_CPRe - 60 * lat_CPRo + 0.5);

% calculating new latitude positioning for both even and odd
late = dLate * (mod(j,60) + lat_CPRe);
lato = dLato * (mod(j,60) + lat_CPRo);

% check if same lat-zone - so valid couple of messages:
NLe = NL_function(late);
NLo = NL_function(lato);

% returning an error if the two messages belong to different time zones
invalid_mex = 0;
if NLe ≠ NLo
    invalid_mex = 1;
    lat = [];
    lon = [];
    return
end

% now check if it is the right quadrant, otherwise move to south emisphere
if abs(late-90 - lat_ref) < abs(late - lat_ref)
    late = late - 90;
end

% assign current latitude for newest mex and compute only needed lon
if CPR == 0
    lat = late;
    % longitude index
    m = floor(lon_CPRe * (NLe - 1) - lon_CPRo * NLe + 0.5);
    n_e = max(NLe,1);
    dLone = 90 / n_e;
    lon = dLone*(mod(m,n_e) + lon_CPRe);
elseif CPR == 1
    lat = lato;
    m = floor(lon_CPRe * (NLo - 1) - lon_CPRo * NLo + 0.5);
    n_o = max(NLo-1,1);
    dLono = 90 / n_o;
    lon = dLono*(mod(m,n_o) + lon_CPRo);
end

% bring to right range looking at the minimum distance with lon_ref
lon_others = [lon, lon + 90, lon + 180, lon + 270];
```

```
66 diff_lon = (abs(lon_others-lon_ref));
67 lon = lon_others(diff_lon == min(diff_lon));
68
69 end
```

Appendix B

ADS-B data managing codes

B.1 ADS-B data acquisition server code

```
using Windows.Networking.Sockets;
using Windows.Web.Http;

DatagramSocket socket;

// Start() is run once when the application is launched
async void Start()
{
    //...
    // Async callback function is placed at the end of the script
    Debug.Log("Waiting for a connection...");
    // create a new socket and add a listener
    socket = new DatagramSocket();
    socket.MessageReceived += Socket_MessageReceived;

    // bind the socket to the endpoint
    try
    {
        await socket.BindEndpointAsync(null, "12346");
    }
    catch (Exception e)
    {
        Debug.Log(e.ToString());
        Debug.Log(SocketError.GetStatus(e.HResult).ToString());
        return;
    }

    Debug.Log("exit start");
}

//...

// Code to receive ADS-B messages and save in vector asyncronously
```

```
34 private async void ...
       Socket_MessageReceived(Windows.Networking.Sockets.DatagramSocket sender,
35     Windows.Networking.Sockets.DatagramSocketMessageReceivedEventArgs args)
36     {
37         // Immediately save the current time
38         DateTime localDate = DateTime.Now;
39         long time_stamp = localDate.Ticks;
40         time_stamp_stream.Add(time_stamp);
41
42         //Read the message that was received from the UDP echo client.
43         Stream streamIn = args.GetDataStream().AsStreamForRead();
44         byte[] message = new byte[16];
45         await streamIn.ReadAsync(message,0,16);
46
47         // Check for the message validity
48         if (message.Length == 16)
49         {
50             byte[] buffer = message.Skip(1).Take(14).ToArray();
51             if (Message_CRC_check(buffer))
52             {
53                 // If message is intact, add to buffer for processing ...
                       inside the update loop
54                 msg_stream.Add(buffer);
55                 gotMessage2 = true;
56             }
57         }
58     }
```

B.2 Airborne Velocity Message decoding in C#

```
1  // Airborne Velocity Decoding
2  static (float speed, float heading, int vr, int alt_diff) ...
       airborne_velocity_decoding(uint subtype, byte[] temp_air)
3  {
4      //Initialize the returned variables
5      float speed;
6      float heading;
7      int vr;
8      int alt_diff;
9
10     // Format the velocity message bytes
11     byte temp_6 = temp_air[0];
12     byte temp_7 = temp_air[1];
13     byte temp_8 = temp_air[2];
14     byte temp_9 = temp_air[3];
15     byte temp_10 = temp_air[4];
16     byte temp_11 = temp_air[5];
17
18     if (subtype == 1 || subtype == 2)
```

```
19      {
20          // W/E velocity
21          byte s_we = (byte)((temp_6 >> 2) & 1);
22          int v_we = (temp_6 & 3);
23          v_we = (v_we << 8) + temp_7;
24
25          // N/S velocity
26          byte s_ns = (byte)(temp_8 >> 7);
27          int v_ns = (temp_8 & 127);
28          v_ns = (v_ns << 3) + (temp_9 >> 5);
29
30          // Velocity
31          float v_x = (float)Math.Pow(-1, s_we) * (v_we - 1);
32          float v_y = (float)Math.Pow(-1, s_ns) * (v_ns - 1);
33          speed = (float)Math.Sqrt(Math.Pow(v_x, 2) + Math.Pow(v_y, 2));
34          heading = (float)(Math.Atan2(v_x, v_y) * 360 / (2 * Math.PI)) % 360;
35          if (heading < 0)
36              heading += 360;
37      }
38      else if (subtype == 3 || subtype == 4)
39      {
40          // subtype 2 or 4
41          // heading availability
42          int av_head = (temp_6 >> 2) & 1;
43          // magn heading
44          int heading_temp = (temp_6 & 3);
45          heading_temp = (heading_temp << 8) + temp_7;
46          heading = (float)heading_temp * 360 / 1024;
47
48          // airspeed velocity
49          byte speed_type = (byte)(temp_8 >> 7);
50          uint speed_temp = (uint)(temp_8 & 127);
51          speed = (speed_temp << 3) + (uint)(temp_9 >> 5);
52          speed = speed - 1;
53
54      }
55      else
56      {
57          speed = -1;
58          heading = -1;
59      }
60
61      if (subtype == 2 || subtype == 4)
62          speed *= 4;
63
64      // now last 2 bytes for VR and altitude difference
65      // Bits from 68-- > temp_9, temp_10, temp_11
66
67      // source bit 0 GNSS, 1 BARO
68      int vr_source = (temp_9 >> 4) & 1;
69      int s_vr = (temp_9 >> 3) & 1;
70      vr = temp_9 & 7;
```

```
71      vr = vr << 6 + temp_10 >> 2;
72      int s_alt_diff = temp_11 >> 8;
73      alt_diff = temp_11 & 127;
74
75      vr = (int)Math.Pow(-1, s_vr) * 64 * (vr - 1);
76      // 0 gnss above, 1 gnss below
77      alt_diff = (int)Math.Pow(-1, s_alt_diff) * 25 * (alt_diff - 1);
78
79      return (speed, heading, vr, alt_diff);
80  }
```

Appendix C

METAR data managing codes

C.1 METAR acquisition and management

```
1  using Windows.Web.Http;
2
3  public class dec_metar
4  {
5      public string location;
6      public DateTime date;
7      public string windheading;
8      public string windspeed;
9      public string windgust;
10     public string windwaving;
11     public string[] windrange;
12     public bool cavok;
13     public string vis_max;
14     public string vis_min;
15     public string[] rvr_runway;
16     public string[,] rvr_vis;
17     public string perturbations;
18     public string[] clouds;
19     public string temp;
20     public string dew_point;
21     public float pressure;
22 }
23
24 public class AndroidUDPListener : MonoBehaviour
25 {
26     // Variables for metar decoding
27     string htmlMETAR = "";
28     float metar_QNH;
29     float landing_gear_pressure;
30     public bool metar_received = false;
31     public dec_metar dec_metar_msg;
32
33     async void Start()
```

```
34      {
35
36          //...
37
38          //First thing try downloading metar, before receiving messages as ...
                it is needed to decode
39          Uri url = new Uri ...
                ("https://tgftp.nws.noaa.gov/data/observations/metar// ...
                stations/LIPK.TXT");
40          htmlMETAR = await MakeWebRequest(url);
41
42          // Split METAR message and decode using function at the end of the ...
                script
43          string raw_metar = htmlMETAR.Split(new string[] { "\n", "\r", ...
                "\r\n" }, StringSplitOptions.None)[1];
44          metar_decoder_function(raw_metar, out dec_metar_msg);
45
46          metar_QNH = dec_metar_msg.pressure;
47          landing_gear_pressure = 0.5f;
48          metar_received = true;
49      }
50
51      // Code to download METAR string
52      private async Task<string> MakeWebRequest(Uri url)
53      {
54          HttpClient http = new HttpClient();
55          http.DefaultRequestHeaders.Add("User-Agent", "Mozilla/5.0 (Windows ...
                NT 6.1; WOW64) AppleWebKit/537.17 (KHTML, like Gecko) ...
                Chrome/24.0.1312.57 Safari/537.17");
56
57          HttpResponseMessage response = await http.GetAsync(url);
58          return await response.Content.ReadAsStringAsync();
59      }
60  }
```